TECHNISCHE
UNIVERSITÄT
DARMSTADT

Towards Practical Privacy-Preserving Clustering and Health Care Data Analyses

at the Department of Computer Science
of the Technical University of Darmstadt

Doctoral Thesis

submitted in fulfillment of the requirements for the degree of
Doctor of Engineering (Dr.-Ing.)

by

Helen Möllering, M.Sc.
born in Steinfurt, Germany

Advisors: Prof. Dr.-Ing. Thomas Schneider
Prof. Dr. Melek Önen

Date of submission: 07.08.2023
Date of defense: 29.09.2023

D 17
Darmstadt, 2023

;

Helen Möllering:
Towards Practical Privacy-Preserving Clustering and Health Care Data Analyses
Darmstadt, Technische Universität Darmstadt, 2023
Tag der mündlichen Prüfung: 29.09.2023

This document was published by tuprints, an e-publishing service of the Technical University of Darmstadt.

http://tuprints.ulb.tu-darmstadt.de
tuprints@ulb.tu-darmstadt.de

Please cite this document as:
URN: urn:nbn:de:tuda-tuprints-247085
URL: https://tuprints.ulb.tu-darmstadt.de/24708

Urheberrechtlich geschützt / In Copyright (https://rightsstatements.org/page/InC/1.0/).

Erklärung

Hiermit versichere ich, Helen Möllering, M.Sc., die vorliegende Doctoral Thesis ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Bei der abgegebenen Thesis stimmen die schriftliche und die zur Archivierung eingereichte elektronische Fassung überein.

Darmstadt, 07.08.2023

Helen Möllering, M.Sc.

Abstract

With the rapid growth in the adoption and democratization of advanced data analysis techniques such as Machine Learning (ML) and Artificial Intelligence (AI), it is imperative to address the privacy concerns arising from the inherent pervasive collection and utilization of sensitive information. In this context, cryptographic secure computation techniques play a crucial role as they enable data analysis while upholding the confidentiality of the input data. These techniques however come with a significant performance overhead compared to plaintext computation. Moreover, some phases of the analysis process, e.g., data preparation, parameter determination, and quality evaluation, have not been adequately considered by the cryptographic community. As a consequence, state-of-the-art secure computation protocols for privacy-preserving data analysis and machine learning nowadays are often not yet ready for being deployed in real-world applications.

In this thesis, we start by investigating the deficiencies of existing secure computation-based solutions for data analyses. Based on our analysis, we take an end-to-end approach to design, implement, and evaluate practical privacy-preserving protocols tailored for specific use cases. We thereby focus on two directions: The first part investigates privacy-preserving protocols for a general class of ML algorithms, namely clustering. The clustering protocols we design combine cryptographic techniques and protocol optimizations with the real-world requirements of plaintext clustering. The second part introduces efficient secure computation protocols for two concrete data analysis applications from the medical domain: The matching of compatible donors and patients for kidney donations and epidemiological modeling. Our solutions critically incorporate interdisciplinary insights.

Practical Privacy-Preserving Clustering. Clustering, an unsupervised ML technique that enables the identification of underlying patterns and structures within data, has a wide range of applications in diverse fields such as health care, marketing, and finance. To ensure the privacy of sensitive input data, numerous secure computation protocols have been proposed for privacy-preserving clustering in recent years. Unfortunately, evaluating their suitability for specific applications and comparing them in terms of privacy guarantees, efficiency, and quality is often complex due to variations in underlying plaintext clustering algorithms, cryptographic techniques, security models, as well as intended participant scenarios and use cases. We systematize the state-of-the-art in privacy-preserving clustering, introduce criteria on how to assess the suitability of a protocol for a specific use case, and lay out open research questions that need to be solved to make private clustering practical for real-world applications.

Based on the results of our systematization, we introduce the first practical and fully privacy-preserving density-based clustering protocol. In contrast to the K-means clustering algorithm, which is commonly used as a baseline in previous private clustering protocols, our private density-based clustering protocol flexibly determines the number of clusters that suits the input data well and is insensitive to outliers. This makes it much more attractive for real-world applications than a private K-means protocol. One application is the distributed ML training

concept, Federated Learning (FL), which is susceptible to backdoor attacks manipulating the model in addition to inference attacks extracting information about the training data used. We devise a novel defense mechanism for FL based on our privacy-preserving density-based clustering protocol.

This part of the thesis is based on the following three publications:

[HMSY21] A. HEGDE, H. MÖLLERING, T. SCHNEIDER, H. YALAME. **"SoK: Efficient privacy-preserving clustering"**. In: *Proceedings on Privacy Enhancing Technologies (PoPETs)* 2021.4 (2021). Online: https://ia.cr/2021/809. Code: https://encrypto.de/code/SoK_ppClustering, S. 225–248. CORE Rank A. Appendix A.

[BCE+21] B. BOZDEMIR, S. CANARD, O. ERMIS, H. MÖLLERING, M. ÖNEN, T. SCHNEIDER. **"Privacy-preserving density-based clustering"**. In: *ASIA Conference on Computer and Communications Security (ASIACCS)*. Online: https://ia.cr/2021/612. Code: https://encrypto.de/code/ppDBSCAN. ACM, 2021, S. 658–671. CORE Rank A. Appendix B.

[NRC+22] T. D. NGUYEN, P. RIEGER, H. CHEN, H. YALAME, H. MÖLLERING, H. FEREIDOONI, S. MARCHAL, M. MIETTINEN, A. MIRHOSEINI, S. ZEITOUNI, F. KOUSHANFAR, A.-R. SADEGHI, T. SCHNEIDER. **"FLAME: Taming backdoors in federated learning"**. In: *USENIX Security Symposium (USENIX Security)*. Online: https://ia.cr/2021/025. USENIX, 2022, S. 1415–1432. CORE Rank A*. Appendix C.

Privacy-Preserving Health Care Data Analysis. A particularly important area for privacy research is health care data analysis due to the highly sensitive nature of medical information and the strong regulatory requirements for data privacy. Secure computation alleviates the trade-off between data usability and privacy by enabling insightful data analyses while still maintaining the privacy and trust of patients, ultimately advancing the quality of care and medical outcomes. We address two important health care data analysis applications in this work. Firstly, we design a novel secure computation protocol addressing the Kidney Exchange Problem (KEP). Secondly, we introduce the problem and first solutions for privacy-preserving epidemiological modeling.

The goal of the KEP is to arrange a series of mutual exchanges between donor-patient pairs. These patients are in need of a kidney donation and unfortunately only have incompatible donors in their own social network. Our secure computation protocol SPIKE finds a locally optimal set of exchange cycles of compatible donors and patients and improves privacy compared to currently deployed centralized approaches. Our approach can reduce legal burdens by keeping patient data locally private and significantly improves efficiency and medical robustness compared to that of the previous state-of-the-art by Breuer et al. (CODASPY'22).

The second highly relevant health care application that we address in this thesis is epidemiological modeling. It predicts the spread of an infectious disease and facilitates the discovery of effective containment strategies. So far, however, epidemiological models often suffer from the lack of precise information about physical contacts, hindering an accurate prediction. We

present the RIPPLE framework, in which we formally define privacy-preserving epidemiological modeling. It enables running epidemiological simulations on the up-to-date contact graph of participants without affecting their individual privacy. Additionally, we present two practical instantations with different security and efficiency trade-offs that can run distributed simulations with half a million people in just a few minutes.

This part of the thesis is based on the following publication and technical report:

[BHK+22] T. BIRKA, K. HAMACHER, T. KUSSEL, H. MÖLLERING, T. SCHNEIDER. **"SPIKE: Secure and private investigation of the kidney exchange problem"**. In: *BMC Medical Informatics and Decision Making* 22.1 (2022). Online: https://arxiv.org/abs/2204.09937. Code: https://github.com/encryptogroup/ppke, S. 253. CORE Rank B. Appendix D.

[GHJ+23] D. GÜNTHER, M. HOLZ, B. JUDKEWITZ, H. MÖLLERING, B. PINKAS, T. SCHNEIDER, A. SURESH. **"Privacy-preserving epidemiological modeling on mobile graphs"**. https://ia.cr/2020/1546. Code: https://zenodo.org/record/6599225. 2023. Appendix E.

Overall, this thesis contributes to make privacy-preserving clustering and secure computation protocols for medical analyses more practical for real-world usage.

Zusammenfassung

Die rasante Zunahme der Akzeptanz und die Demokratisierung von fortschrittlichen Datenanalysetechniken wie Maschinellem Lernen (ML) und Künstlicher Intelligenz (KI) macht es zwingend erforderlich, sich mit den damit einhergehenden Datenschutzbedenken zu befassen, die sich aus der allgegenwärtigen Erfassung und Nutzung sensibler Daten ergeben. In diesem Zusammenhang spielen kryptographische Techniken eine entscheidende Rolle, da sie Datenanalysen unter Wahrung der Privatheit von Daten ermöglichen. Diese Techniken sind jedoch im Vergleich zu Klartextberechnungen mit erheblichen Effizienzeinbußen verbunden. Darüber hinaus wurden einige Phasen von Datenanalyseprozessen von der aktuellen Forschung zu sicheren kryptografischen Protokollen nicht angemessen berücksichtigt. Es fehlen zum Beispiel typischerweise die Datenaufbereitung, die Festlegung der Parameterwerte und die Auswertung der Analyseergebnisse mit Hinblick auf ihre Qualität. Infolgedessen sind existierende Ansätze in diesem Bereich, der im Englischen *privacy-preserving data analysis* oder *privacy-preserving machine learning* genannt wird, noch nicht für den Einsatz in realen Anwendungen geeignet.

In dieser Dissertation untersuchen wir zunächst bestehende kryptographische Protokolle für die sichere Datenanalyse auf mögliche Einschränkungen in ihrer praktischen Einsetzbarkeit in realen Anwendungen. Auf Grundlage unserer Analyse entwerfen, implementieren und evaluieren wir maßgeschneiderte sichere Protokolle zur Wahrung der Privatheit von Daten. Dabei betrachten wir alle Phasen des betrachteten Anwendungsfalls, um eine praktische Einsetzbarkeit zu forcieren. Wir konzentrieren uns auf zwei thematische Schwerpunkte: Der erste Teil dieser Arbeit untersucht sichere Protokolle für eine allgemeine Klasse von ML-Algorithmen, die Clusteranalyse. Dabei werden kryptographische Techniken und Protokolloptimierungen mit den praktischen Anforderungen der Clusteranalyse von Klartextdaten kombiniert. Im zweiten Teil dieser Arbeit werden effiziente Protokolle für die sichere Mehrparteienberechnung in zwei konkreten Datenanalyseanwendungen aus dem medizinischen Bereich vorgestellt: Das Matching von kompatiblen SpenderInnen und PatientInnen für Nierenspenden und die epidemiologische Modellierung. Unsere Lösungen beziehen essenzielle Erkenntnisse aus dem medizinischen Bereich mit ein.

Effiziente Clusteranalyse unter Wahrung der Privatheit. Clustering, eine sogenannte nicht überwachte ML-Technik, ermöglicht die Identifizierung von Mustern und Strukturen in Daten. Die Technik findet Anwendung in vielen verschiedenen Bereichen wie dem Gesundheitswesen, dem Marketing und dem Finanzsektor. Um die Vertraulichkeit sensibler Eingabedaten zu gewährleisten, wurden in den letzten Jahren zahlreiche sichere kryptographische Protokolle für die Durchführung von Clusteranalysen unter Wahrung der Privatheit der Eingabedaten vorgestellt. Diese Protokolle basieren jedoch auf unterschiedlichen Klartext-Clusteranalyse Algorithmen und Sicherheitsmodellen, verwenden verschiedene kryptographische Techniken und wurden für unterschiedliche Szenarien oder spezifische Anwendungsfälle entwickelt. Diese Faktoren erschweren die Vergleichbarkeit der Ergebnisse und damit auch die Beurteilung ihrer Eignung für bestimmte Anwendungen. Wir systematisieren den Stand der Forschung

im Bereich der sicheren Protokolle für Clusteranalysen unter Wahrung der Privatheit der Eingabedaten. Dabei stellen wir Kriterien vor, anhand derer die Eignung eines Protokolls für einen bestimmten Anwendungsfall beurteilt werden kann, und erfassen offene Forschungsprobleme, die gelöst werden müssen, um private Clusterbildung für reale Anwendungen praktikabel zu machen.

Basierend auf den Ergebnissen unserer Systematisierung stellen wir das erste effiziente kryptographische Protokoll für eine dichtebasierte Clusteranalyse vor, dass die Privatheit der Eingabedaten vollständig wahrt. Im Gegensatz zum K-means-Algorithmus, der anderen kryptographischen Protokollen zur sicheren Clusteranalyse oft zugrunde liegt, bestimmt unser sicheres dichtebasiertes Clusteranalyse-Protokoll flexibel die Anzahl der Cluster, die zu den Eingabedaten passt, und ist unempfindlich gegenüber Ausreißern in den Eingabedaten. Dies macht es für viele reale Anwendungen attraktiver und praktisch einsetzbarer als ein K-means-basiertes Protokoll. Ein Anwendungsbeispiel ist das verteilte ML-Trainingskonzept Federated Learning (FL), das anfällig für sogenannte Backdoor- und Inferenzangriffe ist. Backdoor-Angriffe versuchen das Modell zu beschädigen oder zu manipulieren, während Inferenzangriffe versuchen, Informationen über die verwendeten Trainingsdaten zu extrahieren. Wir entwickeln ein neuartiges Verteidigungssystem für FL, das auf unserem dichtebasierten Clustering-Protokoll basiert, und FL gegen beide Arten von Angriffen robuster macht.

Dieser Teil der Dissertation basiert auf den folgenden drei Publikationen:

[HMSY21] A. HEGDE, H. MÖLLERING, T. SCHNEIDER, H. YALAME. **"SoK: Efficient privacy-preserving clustering"**. In: *Proceedings on Privacy Enhancing Technologies (PoPETs)* 2021.4 (2021). Online: https://ia.cr/2021/809. Code: https://encrypto.de/code/SoK_ppClustering, S. 225–248. CORE Rank A. Appendix A.

[BCE+21] B. BOZDEMIR, S. CANARD, O. ERMIS, H. MÖLLERING, M. ÖNEN, T. SCHNEIDER. **"Privacy-preserving density-based clustering"**. In: *ASIA Conference on Computer and Communications Security (ASIACCS)*. Online: https://ia.cr/2021/612. Code: https://encrypto.de/code/ppDBSCAN. ACM, 2021, S. 658–671. CORE Rank A. Appendix B.

[NRC+22] T. D. NGUYEN, P. RIEGER, H. CHEN, H. YALAME, H. MÖLLERING, H. FEREIDOONI, S. MARCHAL, M. MIETTINEN, A. MIRHOSEINI, S. ZEITOUNI, F. KOUSHANFAR, A.-R. SADEGHI, T. SCHNEIDER. **"FLAME: Taming backdoors in federated learning"**. In: *USENIX Security Symposium (USENIX Security)*. Online: https://ia.cr/2021/025. USENIX, 2022, S. 1415–1432. CORE Rank A*. Appendix C.

Privatheit von Daten in medizinische Anwendungen. Ein besonders wichtiger Bereich für die Datenschutzforschung sind Datenanalysen im medizinischen Kontext, da diese Informationen hochsensibler Natur sind und sie zudem strengen gesetzlichen Datenschutzanforderungen unterliegen. Kryptographische Protokolle ermöglichen in diesem Zusammenhang eine überzeugende Vereinbarkeit von Datennutzbarkeit und Datenschutz, indem sie aufschlussreiche Datenanalysen ermöglichen und gleichzeitig die Privatheit der medizinischen Daten und somit das Vertrauen der Patienten wahren. Dies kann letztlich zu einer verbesserten Qualität der Pflege und medizinischen Versorgung führen. In dieser Arbeit befassen wir uns mit zwei wichtigen medizinischen Anwendungen. Zum einen entwerfen wir ein neuartiges sicheres

Protokoll für die Berechnung von Nierenspenderaustauschen. Zusätzlich definieren wir zum ersten Mal das Problem der epidemiologischen Modellierung unter Wahrung der Privatheit der Kontaktdaten und präsentieren erste Lösungsansätze für dieses Problem.

Das Ziel des Nierenspenderaustauschs ist es, Nierenspenden zwischen Spender-Patienten-Paaren zu arrangieren. Diese Patienten benötigen eine Nierenspende und haben nur inkompatible Spender in ihrem eigenen sozialen Netzwerk. Unser kryptographisches Protokoll SPIKE findet eine lokal optimale Lösungsmenge von Austauschzyklen zwischen kompatiblen Spendern und Patienten und verbessert den Schutz der Privatheit der Daten im Vergleich zu den derzeit eingesetzten zentralisierten Ansätzen. Zudem verringert es möglicherweise die rechtlichen Hürden für medizinische Institutionen einem solchen Programm beizutreten, da die Patientendaten lokal beim Datenhalter verbleiben können. Zusätzlich verbessert SPIKE die Effizienz und die medizinische Robustheit im Vergleich zum bisherigen Stand der Forschung von Breuer et al. (CODASPY'22) erheblich.

Die zweite wichtige medizinische Anwendung, mit der wir uns in dieser Arbeit befassen, ist die epidemiologische Modellierung. Sie ermöglicht es, die Ausbreitung einer Infektionskrankheit vorherzusagen und erleichtert die Entwicklung wirksamer Eindämmungsstrategien. Bislang leiden epidemiologische Modelle jedoch häufig unter dem Mangel an präzisen Informationen über physische Kontakte in einer Population, was eine genaue Vorhersage erschwert. Wir stellen das RIPPLE-Framework vor, in dem wir die Anforderungen an eine verteilte epidemiologische Modellierung definieren. Es ermöglicht die Durchführung epidemiologischer Simulationen auf dem aktuellen Kontaktgraphen der Teilnehmer, ohne deren individuelle Privatsphäre zu beeinträchtigen. Darüber hinaus stellen wir zwei konkrete Instanziierungen von RIPPLE vor, die unterschiedliche Kompromisse in Bezug auf Sicherheit und Effizienz bieten. Mit ihnen können verteilte Simulationen mit einer halben Million Menschen in nur wenigen Minuten durchgeführt werden.

Dieser Teil der Dissertation basiert auf einer Publikation und einem technischen Bericht:

[BHK+22] T. Birka, K. Hamacher, T. Kussel, H. Möllering, T. Schneider. **"SPIKE: Secure and private investigation of the kidney exchange problem"**. In: *BMC Medical Informatics and Decision Making* 22.1 (2022). Online: https://arxiv.org/abs/2204.09937. Code: https://github.com/encryptogroup/ppke, S. 253. CORE Rank B. Appendix D.

[GHJ+23] D. Günther, M. Holz, B. Judkewitz, H. Möllering, B. Pinkas, T. Schneider, A. Suresh. **"Privacy-preserving epidemiological modeling on mobile graphs"**. https://ia.cr/2020/1546. Code: https://zenodo.org/record/6599225. 2023. Appendix E.

Insgesamt trägt diese Arbeit dazu bei, kryptographische Protokolle für Cluster- und medizinische Datenanalysen praktikabler für den Einsatz in realen Anwendungen zu machen.

My Contributions

This thesis is based on four peer-reviewed publications and one technical report currently under submission. All five works are results of fruitful collaborations with my supervisor Prof. Dr.-Ing. Thomas Schneider (Technical University of Darmstadt) and a large set of excellent researchers: Beyza Bozdemir (EURECOM/Gen Digital), Timm Birka (Technical University of Darmstadt), Sébastien Canard (Orange Labs/Télécom Paris), Huili Chen (University of California San Diego), Orhan Ermis (EURECOM/Luxembourg Institute of Science and Technology), Hossein Fereidooni (Technical University of Darmstadt), Daniel Günther (Technical University of Darmstadt), Aditya Hegde (IIIT Bangalore/Johns Hopkins University), Marco Holz (Technical University of Darmstadt/FITKO), Kay Hamacher (Technical University of Darmstadt), Benjamin Judkewitz (Charité-Universitätsmedizin), Tobias Kussel (Technical University of Darmstadt/DKFZ German Cancer Research Center), Farinaz Koushanfar (University of California San Diego), Samuel Marchal (Aalto University/F-Secure/WithSecure), Azalia Mirhoseini (Google/Anthropic), Markus Miettinen (Technical University of Darmstadt), Thien Duc Nguyen (Technical University of Darmstadt), Melek Önen (EURECOM), Benny Pinkas (Bar-Ilan University/Aptos Labs), Phillip Rieger (Technical University of Darmstadt), Ahmad-Reza Sadeghi (Technical University of Darmstadt), Ajith Suresh (Technical University of Darmstadt/Technology Innovation Institute), Hossein Yalame (TU Darmstadt), and Shaza Zeitouni (Technical University of Darmstadt). I am immensely grateful to all my collaborators for their significant contributions as well as invaluable advice and continuous inspiration that has fueled my motivation throughout my PhD journey. In the following, I will detail my contributions in our papers.

Chapter 2 is based on three publications: [**BCE+21**; **HMSY21b**; **NRC+22**]. [**HMSY21b**] is joint work with Aditya Hegde, Thomas Schneider, and Hossein Yalame. The publication is the result of Aditya's internship in the ENCRYPTO group in 2020 which I jointly supervised with Hossein. Hossein had the initial idea for the project, while I analyzed, categorized, and systematized more than 50 papers introducing secure computation protocols for privacy-preserving clustering. Moreover, I identified the requirements for private clustering from a security, efficiency, and functional perspective. Additionally, I developed a guideline for choosing suitable private clustering protocols based on the respective application and its requirements. Hossein identified the four relevant private clustering protocols for benchmarking. He and Aditya analyzed their security and privacy properties as well as their efficiency with respect to asymptotic communication and computation complexity. I contributed to the design and setup of the experimental evaluation, while Aditya was in charge of the implementation and benchmarking. Moreover, Aditya and I jointly identified open challenges for the real-world deployment of private clustering schemes based on our insights from the literature and experiments. Thomas guided our research and provided continuous feedback during the full scope of the project and for the paper write-up. [**BCE+21**] is the result of a collaboration with Beyza Bozdemir, Sébastien Canard, Orhan Ermis, Melek Önen, and Thomas Schneider. I contributed significantly to this publication by designing and implementing the secure two-party computation protocol for ppDBSCAN. Beyza designed ppTRACLUS by suggesting

an approximation of the distance measure. I also led and executed the benchmarking of the implementation with respect to clustering quality and efficiency and did the experimental and theoretical comparison to related work. Beyza and Orhan tested the approximation's effect on the clustering quality. Further, I reviewed related work on private clustering and Beyza surveyed related literature on privacy-preserving trajectory analysis. Orhan, Sébastien, Melek, and Thomas were general advisors to the project and provided continuous feedback in all phases of the project. [**NRC+22**] is joint work with Thien Duc Nguyen, Phillip Rieger, Huili Chen, Hossein Yalame, Hossein Fereidooni, Samuel Marchal, Markus Miettinen, Azalia Mirhoseini, Shaza Zeitouni, Farinaz Koushanfar, Ahmad-Reza Sadeghi, and Thomas Schneider. While Thien led the design of FLAME and proposed the high-level design of Private FLAME, I led the Secure Multi-Party Computation (SMPC) protocol design for the HDBSCAN approximation and Hossein Yalame led the design of the mixed MPC protocols for secure aggregation and clipping. Huili worked on the formalization of the backdoor defense strategies and their security guarantees. Phillip was in charge of the complete implementation and benchmarking (except for parts stated and referenced in the paper taken from existing works), while Thien, Hossein Yalame, and I provided advice for the respective parts that we designed. I also contributed to systematizing related work on backdoor and inference attacks as well as secure aggregation on federated learning and its shortcomings. Further, Thien was the main coordinator/lead for the write-up to which also Phillip, Hossein Yalame, Markus, and I contributed. Hossein Fereidooni, Samuel Marchal, Markus Miettinen, Azalia Mirhoseini, Shaza Zeitouni, Farinaz Koushanfar, Ahmad-Reza Sadeghi, and Thomas Schneider were general advisors to this project and provided continuous feedback in all phases.

Chapter 3 is based on [**BHK+22**; **GHJ+23**]. [**BHK+22**] is joint work with Timm Birka, Kay Hamacher, Tobias Kussel, and Thomas Schneider. This publication is the result of the Bachelor thesis of Timm that I co-supervised with my other co-authors. Based on my initial idea, Tobias and I outlined the project structure and guided the protocol design. Tobias and Timm specified the medical requirements for the compatibility assessment of donors and patients. All three of us contributed to the design, conduction, and experimental evaluation of our SMPC protocol addressing the KEP based on Timm's implementation of the protocol. Kay and Thomas led the research project. [**GHJ+23**] is joint work with Daniel Günther, Marco Holz, Benjamin Judkewitz, Benny Pinkas, Thomas Schneider, and Ajith Suresh. Benjamin had the initial idea for the project, while I led the definition of the requirements for a privacy-preserving epidemiological modeling system as well as the design of all aspects of the concrete instantiations that trade-off between different security assumptions and efficiency requirements. Morevover, I investigated different attack scenarios that guided the design of the instantiations. Marco and Daniel defined the details for the private information retrieval instantiation. Ajith optimized the communication overhead of the $\mathsf{PIR}_{\mathsf{sum}}$ protocol. Together with Ajith and Daniel, I defined the experimental evaluation setup and parameters, while Oliver Schick and Nora Khayata implemented and ran the micro-benchmarks. Benjamin, Benny, Thomas, and Ajith provided continuous feedback and advise during the different phases of the project and paper write-up.

Acknowledgments

Over the past four years culminating in this doctoral thesis, I have experienced an incredible (and occasionally challenging) journey fueled by exciting experiences. Along this path, I have gained extensive knowledge and grown, not only with respect to my research but also personally. None of this would have been achievable without the significant individuals in my life to whom I am deeply thankful. I want to take the opportunity here to express my heartful gratitude.

First and foremost, I am profoundly indebted to my advisor, Thomas Schneider. His invaluable guidance, mentorship, and endless patience with my numerous (sometimes crazy) ideas and plans have left an indelible mark. The expertise he shared, along with insightful feedback and unwavering dedication, greatly influenced both my research and this thesis. I am deeply appreciative of the trust he placed in me, entrusting responsibilities for teaching, projects, and student supervision. He played a pivotal role by offering opportunities to work internationally and learn from many esteemed researchers. Yet, what I admire most is his sense for the important things in life and his understanding of matters beyond work.

Before I came to TU Darmstadt during my master studies at EURECOM, I worked on a research project in the area of privacy-preserving machine learning with Melek Önen and wrote my Master thesis under her co-supervision on security aspects of Federated Learning. These experiences fueled my wish to dive deeper into this research field. During my thesis I also worked closely with Giorgia Azzurra Marson. In this time, Ágnes Kiss, who also did her PhD at ENCRYPTO, was mentoring me in the scope of the EIT-Digital Alumni Mentorship Program. All three of them, Melek, Giorgia, and Ágnes, are great female role models to me and guided my decision to start my PhD journey at ENCRYPTO.

A huge expression of gratitude goes out to the ENCRYPTO PhDs and Postdocs, including Andreas Brüggemann, Gowri R. Chandran, Daniel Günther, Marco Holz, Ágnes Kiss, Nora Khayata, Raine Nieminen, Ajith Suresh, Oleksandr Tkachenko, Amos Treiber, Christian Weinert, and Hossein Yalame. They have not only been exceptional colleagues but have also evolved into cherished friends over the past years. A special acknowledgment is owed to Christian and Alex, whose guidance steered me when I was struggling and helped me to make crucial decisions.

Additionally, I extend my thanks to Jens Adler and Petra Fuhrmann for their unwavering assistance with IT and organizational matters. Without them, many critical aspects could have easily veered off course.

A significant appreciation goes out to the CROSSING team, specifically Stefanie Kettler and Jacqueline Brendel. Their efforts paved the way for various learning opportunities, including my research visit at CMU, enriching conference experiences, and fostering my professional development projects.

I wish to express my deep gratitude to the individuals who made my research visit at CMU so enriching. A huge thanks goes to Wenting Zheng for generously hosting and providing

valuable mentorship, guidance, and support, all of which played a vital role in enhancing my academic journey. Her influence significantly shaped my learning trajectory, and I am immensely grateful. I also extend my gratitude to Qi, Shreya, and Jinhao for the incredible collaboration and shared experiences. Moreover, I'd like to thank Paul for being the most amazing office mate one could wish for. His friendship added an unforgettable dimension to my time at CMU and beyond.

I was also lucky to have the honor to work with a large group of students who deeply impressed me with their dedication and excitement about research. Thanks to Aditya, Oliver, Timm, Hannah, Liang, Anne, Max, Thanh, and Klaus for making me enjoy every moment of supervision and collaboration.

Next, I express my gratitude towards my PhD committee members — Melek Önen, Thomas Schneider, Carsten Binnig, Marc Fischlin, and Sebastian Faust — for granting me the opportunity to defend my work. I value the stimulating discussions we had and the subsequent ideas they sparked.

Last but not least, I want to thank my family and friends. Without my parents, siblings, and grandparents I would never be the person I am today. They gave me the possibilities and taught me the mindset to pursue this journey. A special thanks goes out to my sister Lea and my parents, who stood by me during some of the most challenging moments on this path.

I wish to express my deepest gratitude to Lena for her enduring friendship throughout the last four years, helping me out countless times. I will always cherish this. My gratitude also extends to Marie, Manu, and Alex — their friendship has added happiness and balance to my life, always helping me to remember what is normal. I also want to thank Steffi for her many hours of attentive listening, even when I found myself reiterating thoughts twentyfold. And Rabea, for her unwavering support and her willingness to share her home which will never be forgotten. Thanks to Lennart, the Ridestall, and the USZ road cycling community for introducing me to a new passion and making my last summer in Darmstadt incredibly memorable.

Thank you all.

Contents

Abstract III

Zusammenfassung VI

My Contributions IX

Acknowledgments XI

Contents XIII

1 Introduction 1
- 1.1 Encrypted and Distributed Data Processing Techniques . . . 4
- 1.2 Thesis Outline . . . 8
- 1.3 Open Access . . . 11

2 Practical Privacy-Preserving Clustering 12
- 2.1 Our Contributions . . . 13
- 2.2 Related Work . . . 24

3 Privacy-Preserving Health Care Data Analysis 29
- 3.1 Our Contributions . . . 30
- 3.2 Related Work . . . 40

4 Conclusion 46
- 4.1 Summary . . . 46
- 4.2 Future Work . . . 48

Bibliography 52

Lists 72

List of Own Publications 74

Curriculum Vitae 75

Appendices 81

A SoK: Efficient Privacy-preserving Clustering (PoPETS'21) 81

B Privacy-preserving Density-based Clustering (ASIACCS'21) 106

C FLAME: Taming Backdoors in Federated Learning (USENIX Security'22) 121

D SPIKE: Secure and Private Investigation of the Kidney Exchange problem (BMC'22) 141

E Privacy-Preserving Epidemiological Modeling on Mobile Graphs (In Submission to TOPS'23) 173

1 Introduction

In recent years, the constantly growing adoption of Machine Learning (ML) and Artificial Intelligence (AI) across industries as well as many aspects of individuals' everyday life has brought forth a pressing need to address the thereby inherently arising privacy concerns [CSR+20; De 21; LDS+21]. ML models require vast amounts of data to be trained for deriving meaningful insights and support decision-making processes [ZPWV17; RHW19]. This automatically conflicts with the privacy requirements of individuals' sensitive information. Privacy, in the context of data analysis, encompasses the protection of personal data from unauthorized access, the preservation of confidentiality, and the mitigation of biases or discrimination [Ric21].

By safeguarding privacy, individuals can be shielded from potential harm, e.g., arising from data breaches or misuse. Moreover, having appropriate privacy protection mechanisms in place enables individuals to maintain control over their personal information, empowering them to make informed decisions regarding data usage and sharing. This aspect plays a crucial role in fostering trust. When individuals are confident that their personal data is handled securely, they are more likely to engage in data sharing and collaborative efforts [Acq14, p. 79; MKA04; DH06]. This, in turn, leads to the creation of more comprehensive and diverse data analysis, ultimately improving the accuracy, fairness, and generalizability of ML models [BG18; Tre18; YKX22].

Beyond these crucial societal and ethical considerations, even pure financial business interests call for the development of effective privacy-sensitive approaches towards data analyses: Various regulations such as the General Data Protection Regulation (GDPR)[1] in the European Union, the California Consumer Privacy Act (CCPA)[2] in California, and the Personal Information Protection and Electronic Documents Act (PIPEDA)[3] in Canada define strict standards for the collection, storage, and use of personal data. These regulations include provisions for obtaining explicit consent, ensuring data minimization, and guaranteeing the right to erasure. They threaten businesses with severe financial penalization. For example, in 2019, the Federal Trade Commission (FTC) filed a case against Facebook Inc. for mishandling of user data and deceptive privacy practices which resulted in a settlement requiring Facebook to pay a record fine of $5 Billion [FTC19]. Despite comprehensive adjustments to its privacy policies and practices that were also included in this settlement, the company's subsidiary Meta Platforms Ireland Limited was again convicted by the European Data Protection Board (EDPB)

[1] https://eur-lex.europa.eu/eli/reg/2016/679/oj

[2] https://oag.ca.gov/privacy/ccpa

[3] https://www.priv.gc.ca/en/privacy-topics/privacy-laws-in-canada/the-personal-information-protection-and-electronic-documents-act-pipeda/

to pay €1.2 Billion for non-GDPR compliant data transfers of user data from the EU to the US in 2023 [DPC23].

Having those aspects in mind, utilizing Privacy-Enhancing Technologies (PETs) [HZNF15] in ML is a promising avenue as they enable to gain valuable insights from the large amount of data available nowadays while simultaneously also ensuring provably secure privacy protection [TCBK20; CP21; SKKT22; NC23]. Examples are anonymization [Swe02], anonymous communication [Cha81], encryption [DR99; Gen09], access control mechanisms [SV00], and differential privacy [DR+14]. In the scope of this thesis, we focus on protecting data privacy during computation, i.e., extracting insights from data without revealing the underlying raw input data. Thinking a step further, secure computation can also reduce bias and protect against discrimination by hiding individual's sensitive attribute values and limiting access to only aggregated results [Ric21]. Especially when legal regulations explicitly address PETs in the future, the utilization of ML without falling under data privacy regulations – because no access to personal data is required – may significantly simplify the usage of ML-driven applications for companies [HR22; TMSS22].

The most prominent cryptographic techniques for secure computation can be categorized in two types: Homomorphic Encryption (HE) [Gen09] and Secure Multi-Party Computation (SMPC) [Yao82]. Both offer provably secure computation guarantees based on either computationally hard problems or information theoretic guarantees and non-collision assumptions removing the need for a trusted third party to have access to the data and run the computation. HE enables to compute directly on encrypted data without the need for decryption. SMPC enables multiple parties to jointly compute arbitrary functions on their private data while leaking nothing beyond what can be inferred from the output.

Despite these promising properties for privacy-preserving data analysis, using HE and SMPC comes with multiple challenges which is why Privacy-Preserving Machine Learning (PPML) is far from being usable in practice yet:

- *Challenge C1:* Cryptography incurs a significant performance overhead with respect to computation and/or communication. The required computational resources, specialized hardware, and infrastructure constrains the practical deployment.
- *Challenge C2:* Knowledge about those techniques is still a niche, while implementing such mechanisms and protocols requires a high level of expertise as faulty implementations can lead to a complete break of the security guarantees. Implementations and frameworks provided by researchers are only prototypes that are not suited to be used for industrial purposes, but they can provide guidance for industry-scale implementations given they are well-documented and explained.
- *Challenge C3:* PPML systems developed by the privacy research community often lack an interdisciplinary approach, resulting in a limited understanding of the complete scope of ML training or inference processes and state-of-the-art results from ML research. As a consequence, some of the proposed systems may not be practical as they overlook

important aspects such as pre- and post-processing, which are essential for making the systems deployable in real-world applications.

- *Challenge C4:* Organizations must comply with strict data protection regulations, which is often designed to be technology-neutral to accommodate new developments [TMSS22]. However, this approach also creates legal uncertainty in the context of PPML as existing privacy legislation has not yet provided clear guidelines and certifications for the data privacy regulations compliant usage of secure computation [HR22].

An orthogonal approach to using cryptographic techniques for PPML is Federated Learning (FL), a prominent distributed ML paradigm introduced by Google's ML research [MMR+17]. FL enables multiple data owners to jointly train a model while keeping their data locally private. Compared to PPML training protocols based on HE or SMPC such as [XJL19; KVH+21; TKTW21], FL is significantly more efficient. Several advanced open-source frameworks have been developed for FL[4], making it a promising solution for avoiding the challenges of HE- and SMPC-based systems discussed above. But, FL has been found to be vulnerable to manipulation and data leakage [NSH19; BVH+20; LYZY20; WSR+20; XHCL20]. These attacks can destroy model performance [BVH+20; LYZY20], insert backdoors leading to attacker-steered predictions [BVH+20; WSR+20; XHCL20], or leak private information about the training data [NSH19]. As a result, additional protective measures are necessary to fully guarantee privacy when using FL.

In this dissertation, we seek to address parts of the aforementioned technical challenges C1, C2, and C3 for PPML with HE and SMPC as well as the vulnerabilities of FL. Due to the non-technical focus of challenge C4, we do not address it here, but discuss it in the scope of future research in Section 4.2.2. Our focus lies on clustering, a prominent class of ML algorithms that organizes data into groups based on similarities or patterns within the data. Clustering finds applications in diverse areas, such as marketing for customer segmentation and behavior analysis [TPH16; KBSC18], health care for patient and treatment profiling and disease diagnosis [GKK+02; XWRJ13], and finance for fraud detection [Sab12; SHA+19]. To safeguard the highly sensitive information involved, we analyze and design fully privacy-preserving, efficient, and *practical* clustering protocols. Moreover, we look at two specific data analysis applications from the health care domain that require exceptionally high privacy protection as they deal with highly sensitive health care data.

Through our efforts, we aspire to make meaningful contributions to the advancement of PPML with the ultimate goal of enabling its widespread deployment on a large scale. To do so, we first analyze the shortcoming of existing solutions for private clustering and the aforementioned health care use cases with respect to efficiency and usability. Then, we design, implement, and benchmark improved solutions that address some of those limitations. Our concrete use case related protocols from the health care domain incorporate domain knowledge from the medical field, while the clustering protocols build upon the requirements of plaintext clustering. By leveraging this expertise, we aim to develop protocols that not only prioritize privacy but also provide practical solutions that integrate real-world requirements of the ML

[4]For example, https://fedml.ai/ and https://www.tensorflow.org/federated.

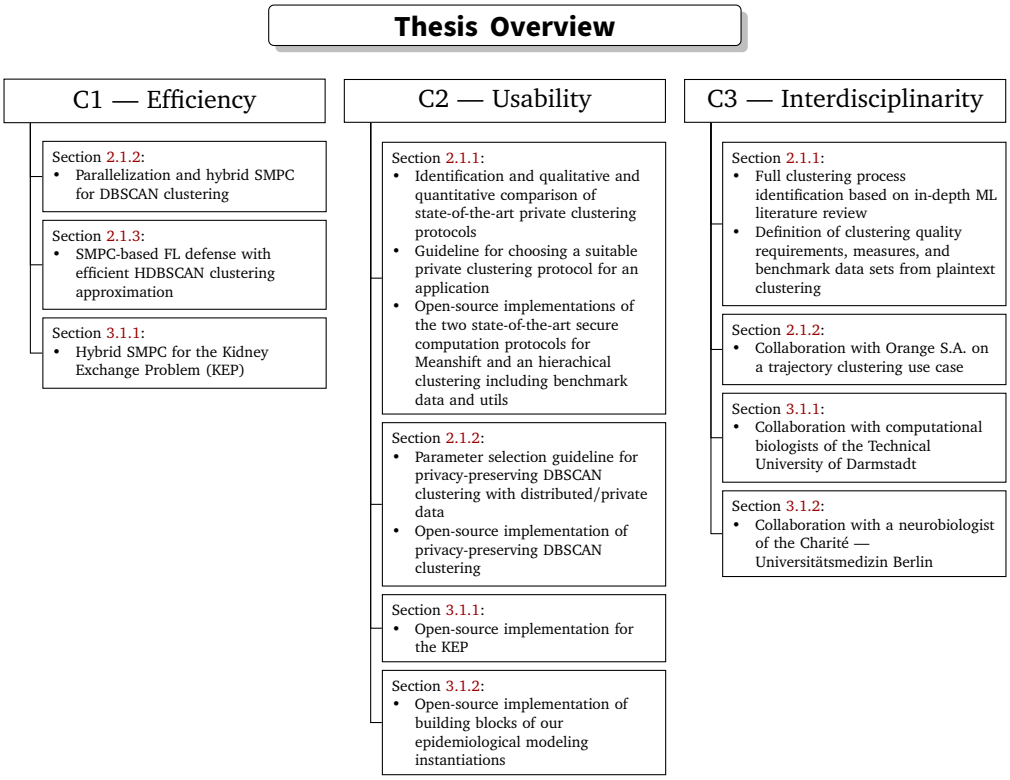


Figure 1.1: Overview of our results for the three technical challenges towards practical privacy-preserving clustering and health care data analyses: efficiency (C1), understandability/usability for practitioners (C2), and interdisciplinarity (C3).

community and medical experts. Figure 1.1 provides an overview of how we address the three identified technical challenges for privacy-preserving data analysis throughout this thesis.

In the following, we first present the PETs that are used in this dissertation in Section 1.1. In Section 1.2, we outline the structure of this thesis and summarize our contributions. We conclude this chapter by a statement about open access to our work in Section 1.3.

1.1 Encrypted and Distributed Data Processing Techniques

Privacy-Enhancing Technologies (PETs) are a class of techniques that use cryptographic primitives, statistical concepts, or hardware components and aim to support data confidentiality, as well as integrity and availability while allowing for meaningful data utilization [HZNF15]. The Organisation for Economic Co-operation and Development (OECD) lists 14 PETs in its report about emerging PETs in 2023 [OEC23]. Those techniques are used to design secure computation protocols that securely realize the envisioned use case without having plaintext access to the input data [Yao82]. In an ideal world, such a functionality could be realized by

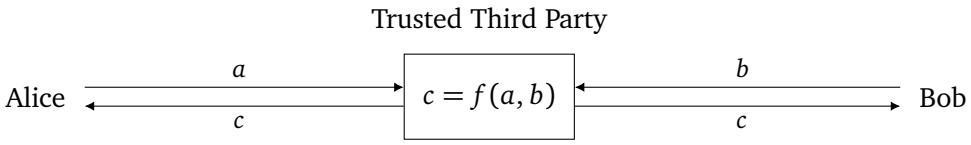


Figure 1.2: Ideal functionality of a secure computation protocol: The two parties Alice and Bob send their respective private inputs a and b to a trusted third party that computes the function f on the inputs and returns the result c to both parties. Alternatively, the result can also be returned to either Alice, Bob, or another additional party.[5]

sharing the input data with a trusted entity which then securely computes the function on behalf of the data owners and only the results are output to the intended parties. However, in reality external third parties are rarely trusted. Thus, secure computation protocols emulate the *ideal functionality* as shown in Figure 1.2 without relying on a trusted party. A protocol is considered to be secure when an adversary cannot gain any additional information beyond what can be learned in the ideal world [Can00; Lin17].

In this section, we summarize three cryptographic and one hardware-based technique that are relevant for this work and can process data "under encryption": SMPC, HE, Trusted Execution Environments (TEEs), and Private Information Retrieval (PIR). Additionally, we explain FL, a state-of-the-art paradigm for privacy-aware distributed ML training.

Secure Multi-Party Computation (SMPC) is a class of cryptographic techniques that enable two or more mutually distrusting parties to jointly compute an arbitrary function f on their private inputs without revealing any information related to individual inputs to other parties. Starting as theoretical concept about 45 years ago [Sha79; Yao82; GMW87], SMPC research has made significant progress, e.g., [BMR90; KS08; ZRE15], leading to the first practical deployments of SMPC-protocols these days: Meta uses SMPC for lift measures of advertisement campaigns [Rey22], J.P. Morgan Chase uses SMPC for inventory matching [PAD+23], and Google and Apple use SMPC for their COVID-19 exposure notification systems [AG21].

Prominent examples of SMPC techniques are Yao's Garbled Circuits (GCs) [Yao86] and arithmetic or Boolean secret-sharing based on the Goldreich-Micali-Wigderson (GMW) protocol [GMW87]. GCs enable to securely evaluate Boolean circuits among two parties with their respective inputs in a constant number of rounds. In contrast, two-party linear secret sharing computation based on the GMW protocol has a round complexity linear in the multiplicative depth of the circuit. While there are many more SMPC protocols, e.g., [Sha79; BGW88; BMR90; DKL+13; AFL+16; CCPS19], those two examples already demonstrate that depending on the nature and complexity of the required computations, efficiency can greatly be affected based on the chosen SMPC technique. It has actually been shown that for some application and scenarios it is most efficient to combine multiple SMPC techniques [DSZ15; BDST22].

[5] For simplicity, we present the two-party case, but the ideal functionality trivially extends to more parties.

For this reason it is far from trivial to design practically efficient SMPC-based applications and often requires detailed domain knowledge.

Another aspect that has to be considered when designing an SMPC protocol is the adversarial behavior. A semi-honest (aka passive or honest-but-curious) adversary is assumed to correctly follow the protocol specifications while trying to maximize its information gain [Lin20]. Although being relatively weak, this security model is still the de facto standard used in many works on PPML [SS08; JVC18; RWT+18; SAA19; BCD+20; TMW+20; PSSY21; HLC+22; HLHD22] as it offers a favorable trade-off between efficiency and privacy. Moreover, it often forms the first step towards stronger security models [Kol06; Lin17; EKR+18]. Additionally, it ensures protection against curious administrators, accidental data leakage, and reduces the impact of a potential data breach [Lin17]. Legal requirements as well as financial interest can also be sufficient to enforce semi-honest behavior by service providers or other data owners [BDST22]. In contrast, a malicious adversary may arbitrarily deviate from the protocol [Lin20]. Covert security is slightly weaker than malicious security as it only guarantees that malicious behavior is detected with certain probability [Lin20]. A few works establish hybrid security models, assuming some parties to behave semi-honestly while others might act maliciously [LMSP21; CGOS22; DWT+23; XHZ+23]. In this work, we focus on semi-honest security.

Next to running SMPC-protocols among multiple-parties, another approach is to consider an outsourcing scenario. In this setup, the input data is secret-shared among two or more computing parties that run the computation. Outsourcing can be beneficial in scenarios where data owners lack sufficient computation capacity or bandwidth for SMPC, leading them to use cloud providers instead [KR11]. However, the computing parties have to be trusted to be non-colluding.

Homomorphic Encryption (HE) schemes enable to perform computations on encrypted data without the need for decryption. With HE, data owners can encrypt their data and outsource computations to untrusted entities, e.g., cloud servers, who perform calculations on the encrypted data without access to the plaintext input data. The encrypted results are sent back to the data owners who decrypt and obtain the final result.

An HE scheme consists of the following four algorithms [Gen09; GHS12]:

- $\text{KeyGen}(\lambda) \longrightarrow (pk, sk)$: Given the security parameter λ, the key generation algorithm KeyGen generates a public key pk and a secret key sk.
- $\text{Enc}(pk, m) \longrightarrow ct$: Given the public key pk and a plaintext message m, the encryption algorithm Enc encrypts m and returns a ciphertext ct.
- $\text{Dec}(sk, ct) \longrightarrow m$: Given the secret key sk and a ciphertext ct, the decryption algorithm Dec decrypts the ciphertext to obtain the original plaintext message m.

- Eval$(pk, C, ct_1, \ldots, ct_k) \rightarrow ct'$: The evaluation algorithm Eval takes as input the public key pk, a circuit C, and a set of encrypted inputs $ct_1, \ldots, ct_k$, where each ciphertext ct_i encrypts a plaintext message m_i, $i \in [k]$. It performs computation on the encrypted values and returns $ct' = \text{Enc}(pk, C(m_1, \ldots, m_k))$.

In 2009, Gentry [Gen09] proposed the first Fully Homomorphic Encryption (FHE) scheme, which supports an unlimited number of both multiplications and additions thanks to a novel bootstrapping technique. This technique involves re-scaling the noise level in ciphertexts which increases with the number of possible computations. This is in theory sufficient to express any computable function, but the significant computational overhead of state-of-the-art HE schemes remains a bottleneck for its wide-spread usage [AAUC18].

Trusted Execution Environments (TEEs) such as Intel Software Guard Extensions [MAB+13; CD16; MAA+16] and ARM TrustZone-based Trusted Execution Environments (TEE) solutions [HGX+17; PS19] are isolated data processing environments. They are designed to protect the confidentiality, authenticity, and integrity of code and data even in the presence of potentially compromised operating systems [SAB15; SMS+22]. TEEs are constructed with dedicated hardware components and/or software-based mechanisms [SAB15; CSFP20]. Trustworthiness and correct behavior of a TEE environment can be remotely verified using attestation mechanisms [SAB15]. In recent years, the security guarantees of TEEs have come under scrutiny due to the discovery and publication of various vulnerabilities, such as fault-based [QWLQ19], side-channel [BMD+17; WCP+17], and (micro-)architectural attacks [GESM17; CZK+18]. These weaknesses have raised concerns about the overall security of TEEs calling for a critical examination of their protection mechanisms [CSFP20; MRRL23].

Private Information Retrieval (PIR) refers to a cryptographic technique enabling clients to privately retrieve elements from a database, without disclosing their specific queries to the database owner. The ideal functionality of PIR is illustrated in Figure 1.3. While a straightforward solution would be to download the entire database, this approach incurs high communication costs. Therefore, efficient PIR schemes aim at minimizing the communication cost. Computational single-server PIR (cPIR), introduced by Kushilevitz and Ostrovsky [KO97], assumes a computationally bounded adversary and often employs HE in recent cPIR schemes, such as [ALP+21; CHK22; MW22; HHC+23]. However, achieving obliviousness requires computationally expensive HE operations on each database block, leading to substantial computational overhead. Alternatively, multi-server PIR approaches, like [DHS14; GI14; DHS17; KOR19; GSW21; GHPS22; MZRA22], rely on multiple non-colluding servers that hold (partial) copies of the database. These schemes are generally more computationally efficient, thanks to utilizing only XOR operations and query compression techniques such as Function Secret Sharing (FSS) [BGI15], but they require communication between the client and all servers.

Figure 1.3: Ideal functionality for Private Information Retrieval (PIR) where a client privately requests the i-th database block $\mathsf{D}[i]$ from a public database D with N blocks without revealing the index i and the retrieved block $\mathsf{D}[i]$ to the database owner. The client receives the requested block $\mathsf{D}[i]$ and might also obtain some additional information about the database.

Federated Learning (FL) is a distributed ML paradigm that enables collaborative model training without requiring to centrally store the training data. In FL, a central aggregator coordinates the training process among a typically large set of heterogeneous clients each possessing their own private dataset. The data held by these clients is typically non-independent and identically (non-IID) distributed. They utilize their data to train local model updates which are then aggregated by the central aggregator into an improved global model. The aggregation can, for example, be done by (weighted) averaging [MMR+17].

From a security and privacy perspective, FL has however been found to be vulnerable to two types of attacks [JFK20]:

- *Manipulation, e.g., [BBG19; BCMC19; BVH+20; JFK20; TTGL20; WSR+20; XHCL20]:* Clients involved in the FL process might try to manipulate the model to either reduce accuracy or even steer the model into specific directions. For example, backdoor attacks inject a hidden pattern or trigger, known as a backdoor or Trojan, into the global model. These backdoors can be designed to become active under specific conditions or input patterns, compromising the integrity and functionality of the trained model. When the manipulated model is deployed, an adversary can exploit the backdoor to achieve unauthorized access or perform targeted misclassifications.
- *Training Data Inference, e.g., [MSDS19; NSH19; JFK20; ZZCY20]:* Inference attacks on FL aim at extracting sensitive information about individual client data by analyzing the individual local updates (potentially in combination with outputs or responses of the global model). Adversaries can exploit statistical information, such as model probabilities or confidence scores, to infer details about the training data or even reconstruct individual data points.

Bagdasaryan et al. [BVH+20] claimed that successfully defending against backdoor attacks is in inherent conflict with training data privacy.

1.2 Thesis Outline

The following chapters summarize our contributions for the design of efficient cryptographic protocols that enable practical privacy-preserving clustering and medical data analyses.

Additionally, we give context information and provide an overview about related work, as well as recent advancements that followed our work.

The remainder of this thesis consists of the following two main chapters. Figure 1.1 summarizes how our work addresses the technical challenges for practical privacy-preserving clustering and health care data analysis.

Chapter 2: In Section 2.1.1, we present our Systematization of Knowledge (SoK) [**HMSY21b**] of the state-of-the-art secure computation protocols for privacy-preserving clustering. We provide the first exhaustive comparative evaluation of 59 secure protocols for eight plaintext clustering algorithms with respect to their privacy guarantees, computation and communication efficiency, as well as clustering quality. Our work also features open-source implementations of two of the most advanced private clustering schemes by Cheon et al. [CKP19] and Meng et al. [MPOT21] that were previously not publicly available. Our repository at `https://encrypto.de/code/SoK_ppClustering` includes nine clustering benchmark datasets from [Ult05; GP10] representing various typical clustering challenges for future extensions of our work to new private clustering schemes. Additionally, we define the requirements for the practical usability of privacy-preserving clustering protocols in real-world applications with private/distributed input data based on an in-depth ML literature research for plaintext clustering. Based on these results, we identify limitations in the existing privacy-preserving clustering protocols and outline three major areas for improvement: The need for secure protocols for additional clustering algorithms, the missing coverage of the full clustering process beyond the core clustering, and enhanced efficiency. Moreover, we introduce a guideline enabling ML practitioners to select a suitable privacy-preserving clustering protocol for their respective application in three steps without requiring in-depth cryptographic expertise.

Building upon the results of our SoK, we present a privacy-preserving SMPC protocol for Density-based Spatial Clustering of Applications with Noise (DBSCAN) [**BCE+21**] in Section 2.1.2. This protocol addresses aspects of all three of the aforementioned major improvement areas: It is the first fully-privacy preserving solution for the DBSCAN clustering algorithm, offers a high level of flexibility as can detect clusters of any shape, and it is insensitive to outliers [EKSX+96]. Moreover, we lay out how to set the two input parameters of privacy-preserving DBSCAN for applications with distributed data: The first parameter defines the maximal distance between two data records to be considered as neighbors belonging to the same cluster, while the second parameter determines the minimal cluster size. Meaningful parameter selection is a crucial part of the pre-processing as it significantly influences the clustering result's quality. With respect to efficiency, we optimize our protocol by combining different Secure Two-Party Computation (2PC) techniques [DSZ15] based on the operation type and conversion cost in each part of the computation. Additionally, we re-design the original plaintext clustering algorithm by leveraging the inherent obliviousness requirement of secure computation protocols for effectively parallelizing distance computation and the neighborhood determination in a Single Instruction, Multiple Data (SIMD) fashion. We

demonstrate our protocol's usability on a trajectory clustering use case in the context of a collaboration with Orange S.A.

In Section 2.1.3, we further demonstrate the practicality of our privacy-preserving DBSCAN protocol by using it for approximating the computationally expensive Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) algorithm [CMS13] for the FL [MMR+17] defense system FLAME [**NRC+22**]. Our benchmarking demonstrates that our approximation is able to defend manipulations by backdoor attacks [BVH+20; XHCL20] equally well as the original plaintext variant while additionally providing protection against data privacy breaches caused by inference attacks [MSDS19; NSH19].

Chapter 3: This part of our thesis focuses on privacy-preserving analyses in the health care domain. Interestingly, we found that these protocols encounter comparable challenges as private clustering, specifically, practicality issues arising from insufficient efficiency and a lack of understanding regarding requirements of the medical side. Hence, we transferred some of our insights from Chapter 2 for the design of practical SMPC protocols for two concrete health care applications.

SPIKE [**BHK+22**] in Section 3.1.1 addresses the Kidney Exchange Problem (KEP) which computes exchange cycles of compatible pairs of patients and donors for living kidney donations. By carefully combining three 2PC techniques [DSZ15] based on cycle depth, the dominance of (non-)linear operations, and conversion costs, we significantly optimize communication and computation overhead. This optimization leads to a three to five orders of magnitude improvement in runtime efficiency compared to previous works [BMWM20; BMW22], depending on the exchange cycle size. Our collaboration with computational biology research facilitated the extension of the compatibility assessment by four additional factors, enhancing the medical robustness of the output solution compared to [BMWM20; BMW22]. These enhancements enable to scale the matching process to significantly more donor-patient pairs within reasonable time. Moreover, SPIKE offers a technical solution to comply with legal data protection requirements, potentially enabling also small medical institutions to participate in kidney exchange programs that were previously excluded due to complicated policy-based data protection measures.

In Section 3.1.2, we introduce the problem of privacy-preserving epidemiological modeling [**GHJ+23**], which extends beyond the scope of contact tracing [AMX+20; TPH+20] on which privacy research has mainly focused so far. Unlike contact tracing, which alerts individuals after encountering a person who tested positive for COVID-19, privacy-preserving epidemiological modeling adopts an *ex ante* perspective, predicting the spread of an infectious disease. This proactive approach enables policy makers and governments to make informed decisions regarding public health measures for containing infectious diseases. Our RIPPLE framework addresses the challenge of limited recent contact information faced by epidemiological models and potentially leading to imprecise simulations. By leveraging data collected from mobile devices

about close physical proximity, RIPPLE enables the privacy-preserving simulation of disease spread on the most up-to-date contact graph. To demonstrate practicality, we provide two instantiations of RIPPLE using either TEE or PIR, including open-source prototype implementations and microbenchmarks. Similar to Section 3.1.1, we again collaborated with medical experts, working closely with a neurobiologist during the design phase, to ensure compatibility with commonly used compartment-based epidemiological models [Bra08; ŠBC+21].

Chapter 4 concludes the thesis and provides an outlook on potential future work.

1.3 Open Access

The all papers in this dissertation are available on the IACR Cryptology ePrint Archive or arXiv.org. Additionally, we open-sourced the code of three of the peer-reviewed publications under the liberal MIT license and will open-source the code of RIPPLE [**GHJ+23**] upon acceptance. The code of our SoK can be found at `https://github.com/encryptogroup/SoK_ppClustering`, the code of our privacy-preserving density-based clustering is available at `https://github.com/encryptogroup/ppdbscan`, and the code of SPIKE is accessible at `https://github.com/encryptogroup/ppke`. By ensuring that our research results are accessible to everyone, we want to encourage replication, verification, and further improvements not only by academic research but also for real-world deployments (cf. Challenge C2 — Usability in Figure 1.1). Thereby, we hope to contribute to the practicality and widespread adoption of secure computation for data analysis in the near future.

2 Practical Privacy-Preserving Clustering

Clustering is a prominent unsupervised Machine Learning (ML) technique aiming at grouping similar items into groups while different items should end up in different clusters [XW05; XT15]. It allows to identify inherent patterns, relationships, anomalies within datasets, and also can also be used for data organisation, summarization, and segmentation [PJ89; RF07; DXLL09; ZL09].

Definition 2.0.1 (Clustering)**.** Let's consider a dataset of N data records, denoted by $\{x_1, x_2, ..., x_N\}$, where each element $x_i, i \in [N]$ belongs to a d-dimensional space. The goal of a clustering is to find a partition of this dataset into K clusters, denoted by $\{C_1, C_2, ..., C_K\}$, such that the partition maximizes *homogeneity* in the clusters and *external separation* between different clusters using a distance metric m.

Clustering offers manifold applications in various domains of everyday life involving highly sensitive information: Health care providers can use clustering to personalize treatment plans for individual patients or to identify high-risk patient groups and develop targeted interventions for those that share specific medical conditions such as cardiovascular diseases or mental health disorders [MMW+10; CMC+11; TPK15; TKRM22]. However, even health record data of a single patient is often not centrally stored but distributed among different health institutions and specialists [BTS+15] who cannot simply share the data due to legal requirements such as the Health Insurance Portability and Accountability Act (HIPAA)[1] in California. Generally, running analyses over a broader database, i.e., from different sources, improves generalizability and robustness. Similar issues can also be observed when employing clustering for detecting fraudulent activities in financial transactions that can only be detected when monitoring activities across multiple accounts at different financial institutions [Sab12; SHA+19]. Other examples are competing business interest for the analyses of social networks, customer segmentation, and recommendation systems for marketing purposes [SKKR02; MSST07; KBSC18]. For such use cases, secure computation protocols can be a promising solution.

59 papers present various ideas how to privately realize clustering using either Homomorphic Encryption (HE) or Secure Multi-Party Computation (SMPC) (cf. Section 1.1). However, those works often compare to only very few other works or even not at all. Additionally, they are based on different plaintext clustering algorithms (which are differently well suited for different clustering challenges), have different input parameters, consider different security

[1] https://www.hhs.gov/hipaa/index.html

models, accept some leakage that is claimed to be unproblematic, etc. All these factors make those protocols very hard to compare and to understand — even with expertise on secure computation — which protocol is best for which use case. It becomes an impossible challenge for ML practitioners that would like to use secure computation in a black box manner to get both data utility and privacy.

Outline. In this chapter, we first present our contributions for systematizing the state-of-the-art of privacy-preserving clustering and for the first efficient, fully private, and practical density-based clustering protocol including an application to robust Federated Learning (FL) in Section 2.1. We then put our efforts into context to related research in Section 2.2.

2.1 Our Contributions

In private clustering, we make two major contributions: In Section 2.1.1, we present our Systematization of Knowledge (SoK) of the state-of-the-art in private clustering. This specifically includes that we outline the first systematic approach for the selection of a well suited secure computation-based clustering protocol based on specific application requirements. Moreover, we also point out research gaps that often hinder the so far existing solutions to be deployable in real-world systems and, thus, should be addressed by future works. We use the insights gained in our SoK in Section 2.1.1 for the design of the first fully privacy-preserving density-based clustering scheme ppDBSCAN which we present in Section 2.1.2. Our protocol is not only practically efficient, but also tolerant to outliers and can automatically determine the number of required clusters based on the inherent structure of the input data. Additionally, we adapted our generic ppDBSCAN protocol such that it can be used to effectively defend against both backdoor and inference attacks in FL in Section 2.1.3.

2.1.1 Systematization of Knowledge for Privacy-Preserving Clustering

This thesis offers a significantly improved understanding of private clustering protocols based on secure computation techniques (cf. Section 1.1). It provides the first systematic theoretical and empirical evaluation of previous works conducted within the following publication:

[HMSY21] A. Hegde, H. Möllering, T. Schneider, H. Yalame. **"SoK: Efficient privacy-preserving clustering"**. In: *Proceedings on Privacy Enhancing Technologies (PoPETs)* 2021.4 (2021). Online: https://ia.cr/2021/809. Code: https://encrypto.de/code/SoK_ppClustering, S. 225–248. CORE Rank A. Appendix A.

Subsequently, we will begin by presenting our systematization factors, followed by an introduction to our guideline outlining a selection process consisting of three steps to identifying

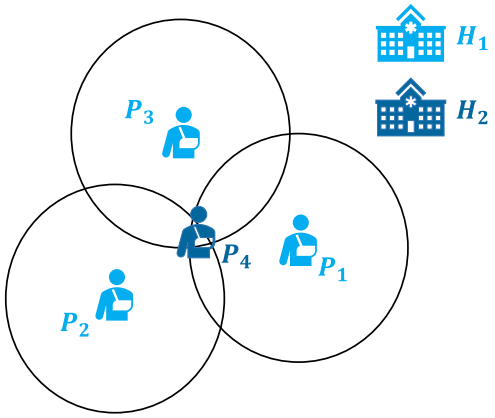


Figure 2.1: Approximating a patient's data from leaked information in [KR07].

appropriate private clustering protocols for specific applications. Finally, we will discuss the open research questions that have emerged from our comprehensive analysis.

A State-of-the-Art Assessment was urgently needed for the area of privacy-preserving clustering considering the 59 secure computation-based protocols that have been published in the last decades. As all secure computation protocols, an ideal private clustering scheme should offer three properties: (1) full privacy, (2) efficiency, and (2) high clustering quality. We will discuss those aspects in the context of this work in the following.

By full privacy we mean the fulfillment of an ideal functionality as shown in Figure 1.2, where the function f is the respective clustering algorithm. Secure clustering protocols emulate the trusted third party and perform the computation "under encryption" without revealing any information beyond what can be inferred from the output. However, several existing works fail to fulfill these requirements, leaking intermediate values in pursuit of improved efficiency. This can have severe implications, including the potential disclosure of specific data records [LHLX12].

Let's consider an example discussed by Liu et al. [LHLX12] and also explored in our work [**BCE+21**] (cf. Section 2.1.2): Certain Density-based Spatial Clustering of Applications with Noise (DBSCAN)-based protocols like [KR07] intentionally disclose which elements are similar (i.e., identified as neighbors due to being below a specific distance threshold) to improve efficiency. In this scenario, we have two hospitals, H_1 and H_2, aiming to cluster patients based on their medical parameters. A subset of their data is depicted in Figure 2.1, showing the data records belonging to three patients P_1, P_2, and P_3 of hospital H_1, and the data record of patient P_4 of H_2. The data records of the first hospital are too dissimilar to be directly assigned to a cluster, whereas patient P_4's data from the second hospital H_2 is sufficiently similar to the data of all three patients from the first hospital H_1 to be considered as their neighbors. When the secure computation protocol leaks information about which elements are neighbors, H_1 can deduce the medical parameters of patient

P_4. This example illustrates that privacy preservation for input data cannot be guaranteed if any information is leaked.

Efficiency is not solely a concern limited to private clustering; it is a general challenge faced by secure computation protocols. These protocols introduce substantial computation and communication overheads due to the utilization of cryptographic techniques and interactive nature of some techniques that require communication between the parties to realize some operations. Hence, it is essential to carefully design and optimize such protocols with regards to their communication and computation complexity. Additionally, for a concrete assessment of efficiency, benchmarking a reference implementation against previous works on the same or at least similar hardware and identical setup configurations can provide a valuable insights about practical usability.

The evaluation of clustering quality is based on the proximity within clusters and the separation between different clusters, which cannot be measured using accuracy scores commonly employed in supervised ML algorithms like neural networks. This distinction arises from the fundamental difference between supervised and unsupervised ML approaches. In supervised ML such as neural networks, labeled training datasets are used to train models for inference on new data, enabling classifications, predictions, etc. On the other hand, unsupervised ML, such as clustering, does not involve model training or labeled data; instead, it seeks inherent patterns within the data. Consequently, quality assessment in clustering relies on the chosen definition of proximity, and different algorithms may produce significantly varied clustering outputs without being inherently incorrect since there is no universally "correct" solution (also known as ground truth in ML terminology). To enable as similar quality assessment as the accuracy measure for supervised ML, synthetic datasets have been created that are used to benchmark and compare clustering algorithms for particular clustering challenges (varying density, outliers, specific cluster shapes, etc.).

Taking these aspects into account, we identified the following seven key factors along which we systematized the existing works (cf. Tabelle 3 in [**HMSY21b**] in Appendix A):

1. *Secure Computation Technique:* Existing private clustering protocols fall into two categories: Those exclusively based on HE and those solely based on SMPC. Additionally, a few works have adopted a hybrid approach, combining an SMPC-based part with an HE-based component in the protocol. Typically, HE protocols introduce a higher computational overhead compared to SMPC protocols, whereas SMPC protocols necessitate more communication due to the inherent interactivity among the participating parties.

2. *Security Model:* The majority of works in the field of private clustering operate in the semi-honest security model and only a small number addresses the malicious security model. Additionally, a significant portion of works fail to explicitly define their chosen security model, thereby leaving it unspecified.

3. *Participant Scenarios:* Regarding the participants involved, privacy-preserving clustering schemes can be categorized into two main scenarios. The first scenario involves classical outsourcing, where two computing cloud servers undertake the computation on behalf

of the data owners. The second scenario corresponds to the classical multi-party computation setup. Many existing works in private clustering focus on the case of two data owners conducting the computation directly between themselves, while other protocols go with an arbitrary number of data owners. Additionally, certain schemes rely on the assistance of a semi-trusted third party.

4. *Privacy:* As mentioned earlier, an ideal privacy-preserving clustering scheme should ensure full privacy. However, our analysis showed that only a limited number of protocols currently provide this guarantee, while others inadvertently leak intermediate states, such as centroids or intermediate cluster assignments.

5. *Data Partitioning:* Many private clustering scenarios involve that data is distributed among multiple data owners. Considering the perspective of the entire dataset, this distributed data can be categorized as horizontally, vertically, or arbitrarily distributed. Horizontal distribution means that each data owner possesses complete data records, while vertical distribution means that data owners have values for specific parameters across all data records. Arbitrary data distribution combines aspects of both. Certain private clustering protocols are tailored for either horizontal or vertical data distribution, while others can accommodate a mix of horizontal and vertical, i.e., arbitrary, data partitioning.

6. *Efficiency:* As previously mentioned, cryptographic protocols introduce significant performance overhead in terms of communication and computation. Therefore, it is crucial to optimize these protocols to ensure their practicality and feasibility. It is important to note that the efficiency of protocols that leak intermediate information cannot be meaningfully compared to those that provide full privacy guarantees. Our analysis reveals that, on average, four out of the ten fully private protocols, namely, [CKP19; MRT20; **BCE+21**; MPOT21], offer the highest efficiency. The other six protocols either utilize outdated additively HE or have been experimentally shown to be slower than the four other works. Among the fully private protocols, K-means and Mean-shift exhibit superior efficiency in terms of communication, with K-means also outperforming others in terms of computation.

7. *Plaintext Clustering Algorithm:* The field of privacy research has covered only eight plaintext clustering algorithms: K-means [Ste56], K-medoids [Zha07], Gaussian Mixture Model (GMM)-based clustering [BSC+97], Mean-shift [FH75], DBSCAN [EKSX+96], Hierarchical Clustering (HC) [ELLS11, S. 71–110], BIRCH [ZRL96], and Affinity Propagation [FD07]. It is important to note that no clustering algorithm can be universally considered superior to others (cf. Table 2.1). In our experimental evaluation of the four most efficient clustering protocols, we also conducted a quality benchmark analysis. The results revealed that the ppDBSCAN protocol [**BCE+21**] offers the highest clustering quality on average, followed by the hierarchical clustering schemes. K-means and Mean-shift, on the other hand, exhibited comparatively lower quality results with significant variation across different datasets.

Guideline — The Protocol Selection for Applications can be done using the systematization criteria listed above. They formed the basis for our guideline given the requirement of a specific application.

- *Step 1 — Application Requirements Analysis:* Our guideline begins by specifying the requirements of the application that we aim to realize through the use of a privacy-preserving clustering protocol. It outlines the desired levels of (a) computational efficiency, (b) communication efficiency, (c) security, (d) participant scenario including the (e) number of involved parties, (f) privacy guarantees, and (g) characteristics of the clustering process. Additionally, it is beneficial to prioritize these aspects in order of importance.
- *Step 2 — Incompatibility Elimination:* With the outcomes of step 1, we can eliminate protocols that do not meet the application's specific requirements. For instance, if the application requires a multi-party computation scenario but cannot guarantee a non-collusion assumption due to competitive relationships among the parties, all SMPC-based protocols must be disregarded. Similarly, in scenarios where high clustering quality is vital and non-convex clusters are anticipated, density-based clustering protocols can be a favorable option.
- *Step 3 — Final Selection:* In this step, the most appropriate private clustering scheme is chosen from the remaining compatible options. The selection process considers the key factors that are most relevant to the application at hand and identifies the protocol that demonstrates the best performance in those aspects.

We refer to [**HMSY21b**, Section 3.3] for more details. There, we also discuss example applications to showcase the selection process.

Three Major Open Research Directions were identified by us as part of our systematization efforts. They require future exploration to enhance the practicality of private clustering.

More Clustering Algorithms: Expanding the typical scope of secure computation protocols, we go a step further by considering practicality from a ML standpoint. In alignment with existing literature on plaintext clustering, an ideal clustering algorithm should possess the following capabilities: 1) the ability to handle arbitrarily shaped clusters, 2) scalability for large datasets, 3) efficient integration of new data records without re-clustering previous ones, 4) compatibility with numerical (discrete and continuous) and nominal variables, and 5) effective handling of outliers. Furthermore, the algorithm should exhibit: 6) insensitivity to the order of input data records, 7) reasonable storage requirements, 8) minimal input parameters, and 9) the ability to handle high-dimensional data records. The assessment of a clustering protocol's performance with respect to these properties is determined by the underlying plaintext algorithm, as demonstrated in Table 2.1, which showcases the strengths and weakness of the eight algorithms privacy research focused on so far. No clustering algorithm is generally superior to others which is why we need privacy-preserving solutions for more clustering algorithms. Concretely, it would be beneficial to develop protocols that

	K-means	K-medoids	GMM	Mean-shift	DBSCAN → Sect. 2.1.2	HC[a]	BIRCH	Aff. Prop.
Cluster Shapes	−	−	−	+	+	∘	−	−
Large Datasets	∘	−	∘	−	−	−	+	−
Update Input Data	+	+	+	∘	+	−	+	−
Nominal Variables	−	+	−	−	+	+	−	+
Outliers	−	∘	∘	−	+	∘	+	+
Input Order	+	+	+	+	∘	+	−	+
Storage	+	−	+	+	+	−	+	−
# Parameters	−	−	−	∘	∘	−	∘	∘

[a] Single/Complete Linkage.

Table 2.1: Comparison of plaintext clustering algorithms. + indicates that the clustering algorithm performs well with respect to the indicated aspect, ∘ indicates an average performance and − indicates that it has some weaknesses. The DBSCAN algorithm [EKSX+96] underlying our ppDBSCAN protocol [**BCE+21**] is highlighted in gray. Please refer to [**HMSY21b**, Section 2.1] in Appendix A for more details.

have mostly parameters independent of the input data, are resilient to noisy input, and can effectively handle data originating from diverse distributions.

E2E Perspective: Beyond executing the clustering process itself, clustering involves multiple other steps: a data preparation (e.g., cleaning, normalization, and scaling), the selection of relevant features, the best suited clustering algorithm, its input parameter values, and distance measures, as well as a quality evaluation. So far, privacy research has merely focused on "translating" an algorithm into a secure variant which neglects aforementioned aspects. To make privacy-preserving clustering practical, we however need to enable usability for non-cryptographic experts. Thus, we need to develop easily assessable solutions that can simply be plugged in into applications without having ML experts to deal with extending/designing cryptographic protocols themselves.

Memory Efficiency: While research on SMPC and HE often centers around communication and computation overhead optimization, it is important to recognize that these techniques also impose significant memory requirements due to their inherent operations. For example, the storage of intermediate values as well as ciphertexts (which are typically larger than their plaintext counterparts) contributes to the overall memory usage. Our experimental results have revealed that memory usage can become a bottleneck for commodity server hardware, highlighting the necessity for efforts in optimizing memory utilization to further enhance the efficiency and practicality of private clustering. For example, the two HE-based hierarchical clustering protocols by Meng et al. [MPOT21] already need about 13 and 60 GBs of RAM for a small dataset with $N = 200$ data records, each of dimension $d = 8$, and $K = 10$ clusters and more than 128 GBs memory for a dataset with $N = 65536$ data records of dimension $d = 4$, and $K = 20$ clusters.

Impact. Our systematization efforts address two significant gaps in advancing the practicality of privacy-preserving clustering research. Firstly, we identified and theoretically

and experimentally evaluated the most promising protocols, providing insights into their respective trade-offs. This enables a better understanding of the strengths and limitations of each protocol. Secondly, we have developed a practical guideline that can be utilized by ML practitioners to select an appropriate protocol for their specific clustering application without requiring secure computation expertise. Beyond, we have highlighted the existing research challenges that need to be addressed to facilitate the transition of privacy-preserving clustering from the research domain to real-world applications. Our work was also presented to the ML community as a contributed talk [**HMSY21a**] at the Privacy in Machine Learning Workshop of the Conference on Neural Information Processing Systems (NeurIPS).

2.1.2 Privacy-Preserving Density-based Clustering

This thesis presents the first fully privacy-preserving and efficient density-based clustering protocol within the following publication:

[BCE+21] B. Bozdemir, S. Canard, O. Ermis, H. Möllering, M. Önen, T. Schneider. **"Privacy-preserving density-based clustering"**. In: *ASIA Conference on Computer and Communications Security (ASIACCS)*. Online: https://ia.cr/2021/612. Code: https://encrypto.de/code/ppDBSCAN. ACM, 2021, S. 658–671. CORE Rank A. Appendix B.

Compared to previous fully privacy-preserving clustering schemes, e.g., [CKP19; MRT20; MPOT21], our ppDBSCAN offers a higher level of flexibility. It dynamically determines the appropriate number of clusters based on the input data, accommodates clusters of arbitrary shapes, and remains robust in the presence of outliers. These properties make our protocol highly practical for real-world deployments, particularly in scenarios where data is distributed among multiple data owners. In such cases, it becomes challenging to pre-define the number of clusters or predict the data distribution, making our protocol a valuable solution. Next, we summarize the underlying plaintext algorithm and then delve into two optimizations that facilitate parallelized computations of the clustering process, leading to improved computational efficiency.

DBSCAN is a density-based clustering algorithm that aims to identify clusters of data records based on their density distribution in the feature space. It operates by grouping together data records in dense areas while separating them from regions of lower density [EKSX+96].

The clustering process of the DBSCAN clustering algorithm [EKSX+96] is sketched in Figure 2.2. In step ①, the algorithm iterates through all input records and checks if they have at least minPts neighbors within distance ϵ. Such input records are called core elements and form a cluster with their neighbors shown in step ②. In a recursive step ③, the neighborhood of neighbors of a core element are analyzed as well. Points that are reachable from a core point, either directly or through a chain of core points, are considered part of the same cluster.

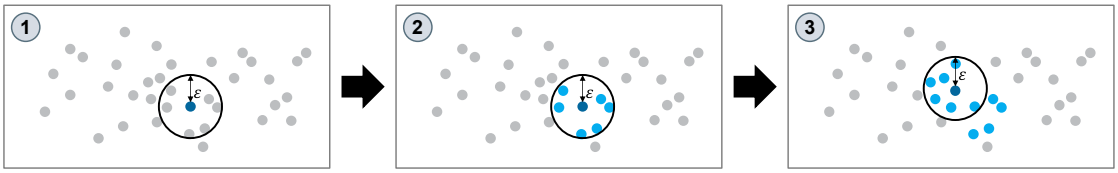


Figure 2.2: Clustering process of DBSCAN [EKSX+96] based on the maximum distance ϵ between two elements to be considered as neighbors. An element with at least minPts neighbors forms a cluster. We set minPts = 4 in this example.

Points that do not satisfy the criteria of a core element and are not reachable from any core element are marked as noise.

The algorithm does not require specifying the number of clusters in advance and is capable of detecting clusters of arbitrary shape. It can handle datasets with varying densities and is robust to outliers.

We introduce an optimized hybrid SMPC-based protocol for DBSCAN that combines Yao's Garbled Circuits [Yao82] for non-linear operations with and Arithmetic secret sharing based on the Goldreich-Micali-Wigderson (GMW) protocol [GMW87] protocol for linear operations (cf. Section 1.1). Additionally, we optimize computation by two parallelizations that we summarize in the following. Moreover, we outline how DBSCAN's parameters can be set when data is not centrally stored but distributed among multiple data owners.

Our Parallelized Distance Computation improves the runtime efficiency in ppDBSCAN. In plaintext DBSCAN, an iteration over the dataset is performed to identify the neighborhood for each data record. Let's denote a specific data record by P_d. To determine its neighbors, we need to compute and compare the pairwise distance d between P_d and all other input data records with respect to a threshold ϵ. In order to optimize this step, we decouple it in our privacy-preserving version ppDBSCAN from the clustering process and compute the pairwise distances beforehand, i.e., we first compute all pair-wise distances and continue then with the actual clustering process. In this manner, we reduce the overhead by half since the distance computation is performed only once per input record pair instead of running it twice, which would be necessary if the results were not stored. We further enhance the computational efficiency by performing the distance computation in parallel using a Single Instruction, Multiple Data (SIMD) approach, where the distance between P_d and other data records is calculated concurrently. Specific to the Euclidean Distance, a typically chosen distance measure, we propose to replace it by its squared variant and use the squared radius ϵ^2 for the neighborhood determination [ETLP13]. This effectively avoids the expensive square root computation without affecting the clustering result.

Our Parallelized Clustering results from the obliviousness requirement of secure computation protocols. The original DBSCAN algorithm employs a queue structure in plaintext to

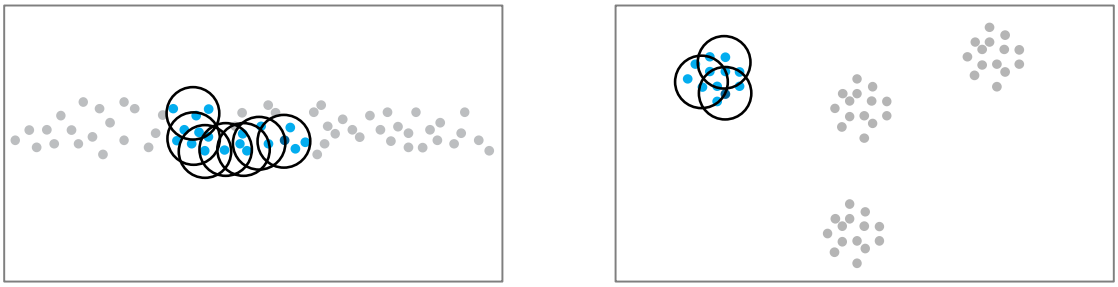
expand the neighborhood of a data record which already forms a cluster by itself. The objective is to investigate and incorporate the neighborhoods of neighboring data records into the cluster if they also meet the minimum density requirement. However, in secure computation, it is crucial for all computations to be oblivious, ensuring they are independent of the input data. Naively implementing a queue structure would inadvertently require to expose information about the distances between inputs and the density of regions in the input space, which is not acceptable from a privacy standpoint. Alternatively, a few constructions, e.g., in [ZE13; KS14], have been proposed that realize data structures like queues in a fully privacy-preserving manner, but they come with impractical performance overheads for many real-world applications due the needed expensive primitives such as Oblivious RAM (ORAM) [TJS19; Shi20].

To address this problem, we proposed a solution that avoids using a queue. Instead, we iterate through all input data records, ensuring obliviousness in the secure computation. However, this approach results in a cubic complexity of the SMPC protocol in the input dataset size N, which is computationally expensive. As a point of comparison, plaintext DBSCAN has a runtime complexity of $\mathcal{O}(N \log N)$ when spatial indexing method is used for the neighborhood queries [EKSX+96]. To mitigate the runtime overhead of our SMPC protocol, we introduce two optimizations:

1. The obliviousness requirement requires to iterate over the full input data set instead of using a queue for recursive neighborhood checks. This allows us to perform those checks in a SIMD-fashion, leveraging parallel computation techniques that reduce storage requirements and improve runtime.

2. We limit the number of iterations for the recursive neighborhood search of cluster elements to a constant value, denoted as $\mathsf{maxIterations}$. This constant represents the maximum depth or length of consecutive neighborhood chains that are investigated. Figure 2.3 illustrates the influence of a predetermined number of iterations on the detection of very elongated clusters (Figure a) in contrast to circular clusters (Figure b). Based on our benchmark results using five datasets (including four datasets selected based on previous work and one synthetic trajectory dataset based on location data from mobile phones provided by Orange S.A.), we found that terminating the process after only $\mathsf{maxIterations} = 4$ iterations is safe, as it does not result in any missed cluster records. Furthermore, for datasets where elongated clusters are expected, we propose an alternative approach. After each neighborhood expansion, a privacy-preserving check can be implemented to verify if the cluster has been updated. If no update has occurred, the expansion is terminated, and the clustering process proceeds to the next unvisited input data record. Although this approach leaks one bit of information per iteration/check, it can be acceptable in certain applications considering the improved efficiency it offers.

Meaningful Parameter Estimation is often overlooked in papers on private clustering, which poses a significant challenge as those values directly impacts the quality of the clustering output (cf. Section 2.1.1). In the case of ppDBSCAN, there are two parameters to consider:



(a) (b)

Figure 2.3: Effect of a fixed number of iterations in ppDBSCAN. If the value is set too low, elongated clusters (as depicted on the left side) cannot be detected, while dense circular clusters (as shown on the right side) can still be identified effectively.

the squared radius ϵ^2 and the minimal cluster size minPts. Even in scenarios where data is distributed among multiple data owners, ppDBSCAN's parameters can still be effectively set: In many cases, they can be derived from the specific use case itself. We discuss two concrete examples in our paper (cf. [**BCE+21**, Section 4.3.3]). As fallback, Ester et al. [EKSX+96] recommend to set $\mathsf{minPts} = \frac{N}{100}$, where N is the dataset size. Similarly, when the neighborhood radius is not given by the application, it can be approximated by a sorted k-distance graph [EKSX+96]. We propose two options to do so: If a data owner holds a sufficiently representative dataset, they can compute $k = \mathsf{minPts}/\frac{N_i}{n}$, where N_i is the data owner i's input data. Alternatively, the sorted k-distance graph can also be computed under SMPC or a secure aggregation protocol can be employed. More details are given in [**BCE+21**, Section 4.3.3].

Impact. With our optimizations, we achieve a clustering time of approximately 420 seconds for a dataset of 400 data records using ppDBSCAN in a semi-honest two-party computation setup and a LAN network with 10 Gbit/s and 0.2 ms Round-Trip Time (RTT) (cf. [**BCE+21**, Sections 5.3 and 5.4] for more details). In our comprehensive survey [**HMSY21b**] paper that is also part of this thesis (cf. Section 2.1.1), we demonstrate that ppDBSCAN surpasses all other state-of-the-art private clustering protocols in terms of clustering quality. Only one protocol proposed by Mohassel et al. [MRT20] outperforms ppDBSCAN in terms of runtime, but it is based on K-means, a clustering algorithm that is in contrast to DBSCAN highly sensitive to outliers, limited to handling convex clusters, and requires prior knowledge of the number of clusters. Thus, ppDBSCAN offers the best trade-off between clustering quality and computational overhead. Moreover, considering that ppDBSCAN's input parameters can be appropriately chosen in a distributed setting, our work is highly suitable for real world applications. We demonstrate this by studying a concrete use case in collaboration with researchers from Orange S.A., in which we apply ppDBSCAN for clustering trajectories to identify typical travelling routes (cf. [**BCE+21**, Sections 4.4 and 5]).

2.1.3 Robust FL with Private Clustering

This thesis uses the private clustering protocol in Section 2.1.2 for our FL backdoor defense called FLAME such that it also impedes inference attacks within the following publication:

[NRC+22] T. D. NGUYEN, P. RIEGER, H. CHEN, H. YALAME, H. MÖLLERING, H. FEREIDOONI, S. MARCHAL, M. MIETTINEN, A. MIRHOSEINI, S. ZEITOUNI, F. KOUSHANFAR, A.-R. SADEGHI, T. SCHNEIDER. **"FLAME: Taming backdoors in federated learning"**. In: *USENIX Security Symposium (USENIX Security)*. Online: https://ia.cr/2021/025. USENIX, 2022, S. 1415–1432. CORE Rank A*. Appendix C.

According to Bagdasaryan et al. [BVH+20], there exists an inherent conflict between privacy and security in FL. The authors argue that detecting malicious clients attempting to manipulate the model requires analyzing and comparing local updates, which however might compromise data privacy by revealing information about the contributing client's training data. In the subsequent paragraph, we describe how we transformed the Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) [CMS13]-based component of our backdoor defense FLAME into an efficient SMPC protocol. This adaptation illustrates that FL can be strengthened to withstand both types of attacks effectively.

FLAME's Plaintext Backdoor Defense focuses on filtering local updates for "poisoned" contributions, i.e., updates that attempt to manipulate the global model, intentionally causing it to misclassify attacker-selected inputs in a manner predetermined by the attacker. This manipulation strategy is also called a backdoor injection [BVH+20]. FLAME's baseline idea is to combine two defense strategies to get the best of both worlds: Large impact manipulations will be easy to detect when comparing multiple local updates in one update iterations assuming that the majority of updates is benign. Manipulations that have a low impact and are meant to move the model slowly into the envisioned direction can be de-oriented by a small amount of normally distributed noise. FLAME first filters the first type of attacks by combining HDBSCAN [CMS13] and clipping. Then, it adds differentially private noise for making the second types of attacks meaningless.

Realizing HDBSCAN [CMS13] for private FLAME in an SMPC protocol requires an approximation for efficiency reasons. Plaintext FLAME uses HDBSCAN as it does not require to specify the maximal distance between two data records to be considered as neighbors in advance. Instead, HDBSCAN generates a Minimum Spanning Tree (MST) that encompasses all possible distances. This MST is then utilized to detect dense regions, forming clusters, while disregarding noisy data records as outliers. This approach enables HDBSCAN to form clusters with different densities and irregular shapes. In contrast, DBSCAN as discussed in Section 2.1.2, requires to pre-define the maximum distance among two data records to be

considered as neighbors for forming a cluster and, therefore, has a fixed density across all clusters making it slightly less flexible as HDBSCAN.

However, building a MST has quadratic complexity (e.g., using Prim's algorithm [KV72]) and is already very expensive in the plaintext domain. Hence, an SMPC protocol results in an infeasible overhead especially considering that known plaintext optimizations cannot be used as computation must be oblivious. Instead, we approximate HDBSCAN by using our ppDBSCAN protocol from Section 2.1.2. In ppDBSCAN, the squared maximal distance ϵ^2 [EFG+09] is uniformly fixed for all clusters in contrast to HDBSCAN. We select the value using a binary search.

Impact. Our benchmarks showed that private FLAME with the ppDBSCAN-based approximation not only matches plaintext FLAME's defense performance but in some cases even outperforms the plaintext version with respect to the detection rate of manipulated updates. For example, it detects all manipulations in our benchmarks with the CIFAR-10 dataset while plaintext FLAME has a True Negative Rate (TNR) of 86.2% (cf. [**NRC+22**] in Appendix C for more details). Moreover, our implementation demonstrated acceptable runtimes considering the large dimensions that are typical for FL in combination with the inherent overhead of SMPC. For example, the largest model with about 20 Million parameters and 100 clients needs about 6 hours for one iteration in a semi-honest Secure Two-Party Computation (2PC) setup and a LAN network with 10 Gbit/s and 0.2 ms RTT. Our protocol is agnostic to the used SMPC technique and inherits its security guarantees. To summarize, our DBSCAN-based approximation enables an efficient and effective backdoor defense while strengthening data privacy on client level. Our findings invalidate the claim made by Bagdasaryan et al. [BVH+20] that privacy and security are fundamentally incompatible in FL and cannot be achieved simultaneously. Notably, as companies, including Google with their keyboard application [ZRX+23], are starting to actively deploy FL systems, ensuring the security and privacy of such systems has become critical. Our private FLAME offers a significant step towards enhancing the trustworthiness of these existing and future FL systems, without compromising the privacy of training data across distributed sources.

2.2 Related Work

In this section, we begin by giving an overview of prior work in the field of Privacy-Preserving Machine Learning (PPML) SoKs and privacy-preserving clustering in Section 2.2.1. Then, in Section 2.2.2, we delve into the subsequent research that followed the publications presented in this chapter.

2.2.1 Previous Work

In the following, we put our works in context to earlier systematizations of knowledge in the context of PPML as well as privacy-preserving secure clustering.

SoKs for PPML. To the best of our knowledge, no previous work has surveyed or systematized secure computation protocols for privacy-preserving clustering. After a few very early works on privacy-preserving data mining [VBF+04; YBDN11], the advent of ML and its inherent privacy concerns have fostered a plethora of secure protocols for different ML algorithms that have subsequently been reviewed and systematized: Especially deep learning gained a lot of attention and resulted in many systematization efforts, e.g., in [CL18; ZCWS18; RRK19; ABB+20; BDC20; LYW+20; MTV+20; TCBK20; AZL+21; KT21; PTC+21]. Kiss et al. [KNL+19] provide a systematic review of private decision tree evaluation, while Haralampieva et al. [HRP20] focus on private image classification tasks. The privacy issues of FL led to the design of privacy-enhancing measures like secure aggregation protocols tailored to its requirements, which were then also systematized by multiple researchers [**FMM+21**; YZH21]. Other works, such as [PMSW18; CSR+20; Yan20; KMA+21; LDS+21; XBJ21], take a broader perspective by systematizing security and privacy issues, attacks, and defenses in ML or FL in general.

Privacy-Preserving Clustering. In our SoK in [**HMSY21b**] in Appendix A, we have conducted a thorough analysis of previous research on secure computation-based private clustering. Hence, we only provide a brief summary here and encourage interested readers to refer to our paper for more comprehensive information.

In total, we have identified 59 publications and technical reports that primarily focus on secure variants of eight plaintext clustering algorithms: K-means [Ste56], K-medoids [Zha07], Gaussian Mixture Model (GMM)-based clustering [BSC+97], Mean-shift [FH75], DBSCAN [EKSX+96], HC [ELLS11, S. 71–110], BIRCH [ZRL96], and Affinity Propagation [FD07]. These works propose cryptographic protocols to ensure privacy, utilizing either HE, SMPC techniques, or a combination of both.

However, among these publications, ten works stand out for achieving full privacy preservation without leaking intermediate information. These ten works are: [BO07; ZE13; RSB+15; JA18; KC18; CKP19; MRT20; **BCE+21**; **KMSY21**; MPOT21].

Among the ten fully privacy-preserving works, our ppDBSCAN protocol [**BCE+21**], the SMPC-based K-means protocol by Mohassel et al. [MRT20], the HE-based Mean-shift protocol by Cheon et al. [CKP19], and the hierarchical clustering protocol combining both HE and SMPC by Meng et al. [MPOT21][2] demonstrate the best performance in terms of trade-offs

[2]Note that subsequent to the publication of our SoK paper [**HMSY21b**], Meng et al. [MPOT21] published their work on SMPC-based hierarchical clustering. It is important to mention that their work was already available as a pre-print which served as the basis for our own research.

between clustering quality and communication and computation efficiency. While ppDBSCAN, [MRT20], and [MPOT21] are two-party computation protocols, [CKP19] operates in an outsourcing setting where a single data owner encrypts their data and sends it to an untrusted party for computation in the encrypted domain.

[MRT20] exhibits the best asymptotic communication and computation complexity considering dataset size, number of clusters, and data dimension. It also concretely outperforms all other schemes in terms of runtime. For instance, on larger datasets ($N > 8192$ data records) in a LAN network with 1Gbps and a roundtrip time (RTT) of 1 ms, it is up to two orders of magnitude faster than the second fastest protocol by Cheon et al. [CKP19]. Our ppDBSCAN protocol [**BCE+21**] offers the second best concrete runtimes on small datasets ($N < 200$ data records), being on average 14× slower than [MRT20] over LAN and 2.5× slower over WAN (100 Mbps, RTT 100 ms). The performance gap narrows with an increasing number of clusters since the communication complexity of ppDBSCAN is independent of the number of clusters, whereas [MRT20] and [CKP19] are affected by it.

Regarding communication, [CKP19] performs best for datasets with more than 150 data records and at least 10 clusters, as the data can be encrypted in a single ciphertext with efficient packing. For smaller datasets, [MRT20] has about half the communication cost of [CKP19], as the latter cannot effectively utilize ciphertext packing for those.

Our ppDBSCAN protocol [**BCE+21**] achieves the best clustering quality across different datasets, demonstrating good performance for clusters with arbitrary shapes, noisy datasets, and high cluster variance. While [MRT20] excels in terms of efficiency, it exhibits weaknesses in clustering quality due to the underlying sensitivity of the K-means algorithm to noise and its ability to only detect convex-shaped clusters. Cheon et al.'s Mean-shift protocol [CKP19] achieves similar clustering quality as the K-means protocol by Mohassel et al. [MRT20]. The private HC-based approach proposed by Meng et al. [MPOT21] encounters difficulties when dealing with high cluster variance, resulting in the possibility of incorrect cluster merges.

Overall, there is no superior private clustering protocol, highlighting the need for future work, as identified in our comprehensive review (cf. Table 2.1 and [**HMSY21b**] in Appendix A), to design privacy-preserving variants for additional state-of-the-art plaintext clustering algorithms. These variants should consider not only performance measures but also practical deployment issues, such as data preparation and parameter value settings in a multi-party setup.

2.2.2 Subsequent Work

Next, we present an overview of recent SoK papers and surveys in the field of PPML and novel secure computation protocols for privacy-preserving clustering that have been published after our works. Furthermore, we provide a contextualization of these new works in relation to our own results.

Algorithm	Paper	PETs	Participant Scenario	Data	Output	Efficiency
K-means	[BO07, CCS'07]	HE+ASS	2PC	*a*	final centroids	−−
	[RSB+15, CIC'15]	HE	Outsourcing, 2 Servers	*h*	final centroids	−−
	[JA18, SAC'18]	HE	Outsourcing, 1 Server	*x*	final centroids	−
	[KC18, CLOUD'18]	HE	Outsourcing, 2 Servers	*x*	cluster sizes	−−
	[MRT20, PETS'20]	GC	Outsourcing, 2 Servers *or* 2PC	*h*	final centroids	++
Mean-shift	[CKP19, SAC'19]	HE	Outsourcing, 1 Server	*x*	final centroids	−
Affinity Propagation	[**KMSY21**, SECRYPT'21]	ASS	Outsourcing or MPC	*a*	final clusters	−
DBSCAN	[ZE13, S&P'13]	GC	2PC	*h*	cluster labels, centroids/size possible	−
	[BCE+21, ASIACCS'21]	**GC+ASS**	**Outsourcing, 2 Servers *or* 2PC**	***a***	**cluster labels, centroids/size possible**	**+**
HC	[MPOT21, CCSW'21]	HE+GC	2PC	*h*	final dendogram	-

Table 2.2: Overview of fully privacy-preserving clustering protocols in comparison to our ppDBSCAN (highlighted in gray and bold, cf. Section 2.1.2). HE is homomorphic encryption [Gen09], ASS is arithmetic secret sharing [GMW87], RSS is replicated secret sharing [AFL+16], and GC is garbled circuits [Yao86]. *v* indicates vertically partitioned data, i.e., the data owners hold the values for a subset of parameters from all data records. *h* indicates horizontally partitioned data, where the data owners hold complete data records with all parameters, *a* is arbitrarily partitioned data, and *x* indicates the scheme has only one data owner. +/− rates the computational efficiency in comparison to each other. For more details please refer to [**HMSY21b**] in Appendix A.

SoKs for PPML. Cabrero-Holgueras and Pastrana [CP21] investigated the latest secure computation protocols for deep learning especially focusing on exploring efficiency and practicality from an ML practitioners' point of view, similar to our own work. Mann et al. [MWCB22] present an exhaustive investigation of private neural network inference systems and outline limitations for large scale deployments. Then just recently, Ng et al. [NC23] expanded on the research conducted by [CP21], presenting a comprehensive overview of the most recent advancements. [NC23] also includes detailed analyses of results that addressed individual linear and non-linear components of neural networks. In an extension of Kiss et al.'s work [KNL+19] on private decision tree inference, Chatel et al. [CPTH21] expanded the scope to include training as well.

Additional recent SoK papers have focused on specific Privacy-Enhancing Technologies (PETs) for PPML. For instance, Panzade et al. [PT22] explored functional encryption, while [PTH21; PTC+21; Lau22] delved into HE. Sagar et al. [SK21] analyzed PPML solutions in the outsourcing scenario, and Kuzniewski et al. [KMS22] examined PPML open-source implementations. Zalonis et al. [ZAGK22] focused on private ML in medical applications.

To cover the growing body of research on improved privacy in FL, Mansouri et al. [MOJC23] and Liu et al. [LGY+22] presented detailed reviews of recent developments in secure aggregation for FL. Song et al. [SWRH23] systematically placed SMPC-based distributed training in comparison to FL with a focus on privacy.

Privacy-Preserving Clustering. Wei et al. [WTC22] introduce an honest-majority Secure Three-Party Computation (3PC) protocol for K-means clustering using replicated secret-sharing [AFL+16] in the semi-honest security model. Although the protocol claims to be fully

privacy-preserving, the division building block used, inspired by Wagh et al. [WTB+21], leaks the range of the denominator. In an experimental comparison using FALCON [WTB+21] as the implementation framework, Wei et al.'s protocol outperforms Mohassel et al.'s K-means protocol [MRT20] by approximately 20× in terms of computation time and an average factor of 65× for datasets of size $N = \{10^4, 10^5\}$ in a localhost setup. A performance improvement is expected considering that Mohassel et al.'s protocol is a 2PC protocol, while Wei et al.'s protocol is a 3PC protocol. In comparison, our ppDBSCAN [**BCE+21**] is slower than Mohassel et al.'s protocol [MRT20], and consequently, also slower than Wei et al.'s protocol [WTC22]. However, in terms of clustering quality, ppDBSCAN remains superior due to the underlying K-means algorithm of Wei et al. [WTC22].

Zhang et al. [ZHS+22] propose a private K-means protocol utilizing multi-key homomorphic encryption (HE) with two variations: A multi-party computation scenario involving multiple data owners conducting the computation alongside a helper server, and an outsourcing scenario where the full computation is done among two servers. However, their comparison protocol discloses the distances between the compared data records to either data owners or one of the servers and reveals the comparison result to the other server. Similarly, their secure minimum protocol exposes the comparison result to one of the servers. The authors neither provide theoretical nor experimental comparisons against previous state-of-the-art approaches. Moreover, the accuracy results demonstrate a significant decline of over 10% when compared to plaintext K-means. In summary, our ppDBSCAN [**BCE+21**] offers stronger privacy guarantees and superior clustering quality compared to Zhang et al.'s protocol.

Recently, Shriram et al. [SKK+23] introduced an interesting new direction in private clustering. They developed the first 3PC protocol for approximative local clustering of distributed graphs using a heat-kernel PageRank vector [CS18]. In contrast to previous works like ppDBSCAN [**BCE+21**], their focus is on local clustering, which involves finding similar data records to a specific input element in order to form a cluster. This approach is distinct from global clustering methods that aim to cluster the entire dataset and which we looked at in the scope of this thesis.

We summarize key aspects our ppDBSCAN in comparison to other fully privacy-preserving global clustering protocols in Table 2.2. None of the subsequent works are included as they either leak information [WTC22; ZHS+22] or are designed for local clustering [SKK+23].

3 Privacy-Preserving Health Care Data Analysis

Artificial Intelligence (AI) and Machine Learning (ML) hold great promise for various industries and aspects of life. For example, in the health care sector, the analysis of medical data has emerged as a strong driver for advancements in the field of medicine [KTS+17; LCL+18; DK19; MP19; NA20]. This is primarily due to the increase of available medical data thanks to digitisation efforts [Esp98; BTS+15; KTS+17], advancements in computational power [DBH18; MP19], and the development of sophisticated algorithms [ZJYC02; KTS+17; NA20]. We believe that the integration and analysis of diverse datasets distributed among multiple sources [BTS+15] can open up new possibilities for extracting valuable insights and improving health care outcomes:

P1 - *Comprehensive Patient Profile:* Combining data from different health care providers, medical institutions, and research studies creates a comprehensive view of patients' medical histories, treatments, and outcomes. This supports health care professionals to make better-informed decisions and develop personalized treatment plans based on a broader range of information.

P2 - *Fairness and Bias Reduction:* Including diverse data from various sources allows health care professionals and researchers to identify and address biases related to factors such as ethnicity, socioeconomic status, or geographic location promoting a deeper understanding of health care disparities and supports the development of interventions and policies aimed at reducing bias and improving health care equity for all individuals.

P3 - *Identification and Analysis of New/Rare Diseases:* New or rare diseases or conditions may not be well-documented in a single health care facility. By pooling data from multiple sources, it becomes easier to identify and study these rare cases, leading to better understanding, early diagnosis, and treatment strategies.

P4 - *Macro-perspective on Health Situation:* Aggregated data from various sources can provide insights into population health trends, prevalence rates of diseases, and patterns of health behaviors. This information is valuable for public health agencies and policymakers to develop targeted interventions and allocate resources efficiently. Moreover, with the aggregated information it becomes easier to detect anomalies in single regions early on that might indicate an outbreak of a new disease.

P5 - *Realistic Evidence:* Sufficiently large and diverse real-world evidence of patient experiences and treatment outcomes complements controlled clinical trials and provides valuable insights into how treatments perform in real-life settings, helping to bridge the gap between research and practice [SAD+16].

Due to its highly sensitive nature, it is clear that health care data must not simply be shared among different data holding entities. This is also reflected in multiple strong privacy regulations for health care data like Health Insurance Portability and Accountability Act (HIPAA) or the European Health Data Space (EHDS)[1] (which is expected to be introduced as regulation in 2025).

Outline. In this chapter, we start by presenting a novel secure computation protocol for the Kidney Exchange Problem (KEP) and a new privacy research problem in the context of epidemiological modeling in Section 3.1. Our Secure Multi-Party Computation (SMPC) protocol for the KEP in Section 3.1.1 addresses P3 by enabling the participation of different medical institutions in kidney exchange programs. This allows them to merge their data while maintaining the privacy of patient and donor information locally at each data holder. Additionally, it contributes to enhancing fairness (P2) by increasing the chances for patients in need of a kidney donation to find compatible donors. Moving on to our privacy-preserving epidemiological modeling framework in Section 3.1.2, it effectively tackles P3 and P4. This framework empowers epidemiologists to conduct experiments that model various potential outcome scenarios for new or raw diseases, considering different parameter values. By incorporating recent contact graph information, our framework significantly improves the accuracy of these simulations, empowering decision-makers to make better informed choices regarding resource allocation and potential containment measures. Furthermore, our framework may also serve P5 by identifying erroneous assumptions when comparing the simulations' predictions with the actual outcomes retrospectively. This aspect contributes to refining and improving future epidemiological modeling efforts. In Section 3.2, we provide a comprehensive contextualization of our results by examining prior related work and reviewing subsequent studies that followed upon our work.

3.1 Our Contributions

Within the domain of privacy-preserving medical analysis of distributed data, our work makes two key contributions: Firstly, in Section 3.1.1, we introduce our novel SMPC-based protocol for the KEP. This protocol offers improved robustness and efficiency compared to existing expensive state-of-the-art solutions in the field [BMWM20; BMW22]. Secondly, in Section 3.1.2, we outline our contributions to privacy-preserving epidemiological modeling. We defined and introduced this novel research problem, and provided two concrete instantiations as proposed solutions. Both our contributions provide specific and pragmatic privacy-preserving approaches in the field of health care data analysis that effectively safeguard the confidentiality of sensitive data while facilitating the utilization of distributed medical data.

[1] https://health.ec.europa.eu/ehealth-digital-health-and-care/european-health-data-space_en

3.1.1 SPIKE: A Secure and Private Investigation of the Kidney Exchange problem

This thesis improves upon previous work [BMWM20; BMW22] on secure computation protocols that address the KEP: We introduce Secure and Private Investigation of the Kidney Exchange Problem (SPIKE), a more efficient and robust hybrid SMPC-protocol, that was presented in the following publication:

[BHK+22] T. BIRKA, K. HAMACHER, T. KUSSEL, H. MÖLLERING, T. SCHNEIDER. **"SPIKE: Secure and private investigation of the kidney exchange problem"**. In: *BMC Medical Informatics and Decision Making* 22.1 (2022). Online: https://arxiv.org/abs/2204.09937. Code: https://github.com/encryptogroup/ppke, S. 253. CORE Rank B. Appendix D.

In the following, we summarize the Kidney Exchange Problem (KEP) and present our privacy-preserving Secure Two-Party Computation (2PC) protocol, SPIKE, which offers a solution to the KEP while ensuring data privacy. We highlight the enhancements we have made to improve the medical robustness in terms of likelihood of a successful transplant of SPIKE compared to previous approaches [BMWM20; BMW22]. Additionally, we present our hybrid 2PC sub-protocols that combine Garbled Circuits (GCs) [Yao86] with Goldreich-Micali-Wigderson (GMW)-style Boolean and arithmetic secret sharing [GMW87] (cf. Section 1.1). For the design of these, we successfully used the insights we gained in the design phase of ppDBSCAN (cf. Section 2.1.2). The result, SPIKE, offers an efficient and private solution tailored specifically for the KEP.

The KEP as defined by Abraham et al. [ABS07] and Roth et al. [RSÜ04] is typically modelled as an optimization problem on a directed graph structure. It refers to the situation where multiple patients in need of kidney transplants have a willing but incompatible living donors. In kidney exchange programs, this situation is solved by forming a cycle — as shown in Figure 3.1 for a cycle size of 2 — among those pairs of incompatible patients and donors such that in the end each patient receives a kidney from a compatible donor. To solve the KEP and form such a cycle, a prominent approach is to use integer programming (e.g., [ABS07; CKVR13]) in the plaintext. Thereby, the goal is typically to maximize the number of transplants, but also other aspects such as medical constraints, cycle size, waiting time can be taken into account [ALR+17; BVM+21].

As patients are located in different hospitals, their data is not centrally stored and legal requirements enforce a high burden. Especially smaller medical facilities might therefore not be able to join the process reducing the chances for patients to find a compatible donor. Secure computation offers a solution to compute a KEP matching suggestion without compromising data privacy and, thus, can enable more institutions and patients to join KEP programs.

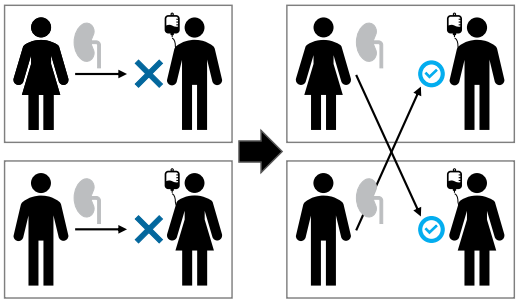


Figure 3.1: Illustration of a kidney exchange cycle involving two donor-patient pairs, which is also called crossover exchange. On the left side, there are two incompatible pairs, and on the right side, they exchange donors and patients to form a compatible matching.

Improved Medical Robustness refers to the likelihood that an output kidney exchange cycle results in successful transplants. Cancellations pose a challenge in living kidney donations through "exchange cycles" as patients and donors may drop out for various reasons, ultimately causing the entire cycle to fail [PCCS18]. Factors contributing to these dropouts include medical considerations where health complications or changes in health status can make individuals ineligible for donation or transplants. Psychological factors, such as anxiety, fear, or uncertainty, can lead to withdrawal due to emotional apprehensions regarding the procedure. Additionally, personal circumstances, like financial constraints or family commitments, can impact individuals' ability to participate in the exchange cycle. These factors collectively contribute to the cancellation issues observed in living kidney donation exchange programs [PCCS18].

An algorithmic solution to the KEP evaluates compatibility based on quantifiable (often medical) factors, but those are often not exhaustive for efficiency reasons or even infeasible to compute since the KEP is NP-complete [BMW23]. While a final evaluation by medical experts is crucial to ensure that no important detail is overlooked due to the vital nature of the matter, the absence of essential factors in the algorithmic pre-evaluation can result in increased rejection rates during the manual check of exchange cycles by the experts. Thus, to enhance the robustness of SPIKE's results, we add four additional medical parameters to the ABO blood type and Human Leukocyte Antigens (HLA) crossmatch used in previous work [BMWM20; BMW22]: HLA match, age, weight, and sex [Ope97; WSB+00; EHB+03; ALR+17; MKA+17]. Additionally, we restrict the size of exchange cycles to 2 or 3 in our benchmarks which reduces the impact of dropouts. This aligns well with practical feasibility aspects such as the availability of medical personnel and surgery capacity, since all transplants within one cycle should ideally be performed simultaneously at the same or nearby hospitals [ABS07; PCCS18]. With this in mind, we now shift our focus to discussing our core contributions, specifically the secure computation protocols designed for these parameter checks. We refer the reader to our paper [**BHK+22**] in Appendix D for more information about the biological background.

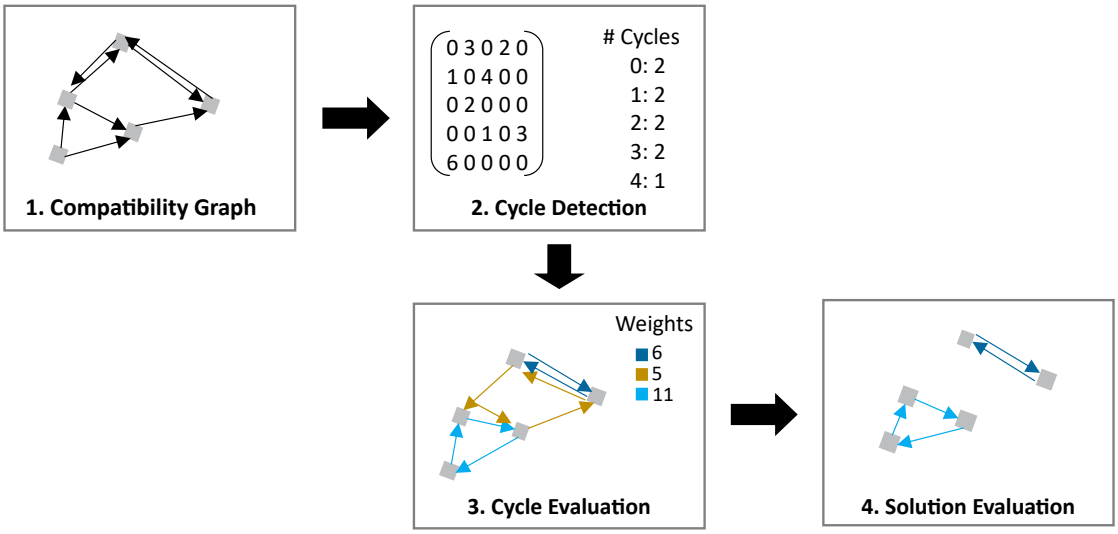


Figure 3.2: Overview of SPIKE's four phases [BHK+22]. It outputs a set of mutually exclusive exchanges cycles with the highest compatibility scores while donors' and patients' data remains private.

Tailored Combinations of SMPC Techniques enhance the efficiency of SPIKE, a flexible secure computation-based approach to the KEP. It comprises four distinct phases shown in Figure 3.2: (1) Compatibility Graph Computation, (2) Cycle Detection, (3) Cycle Evaluation, and (4) Solution Evaluation. For each of these phases, we have designed efficient 2PC protocols, envisioning a scenario where two semi-honest servers carry out the computation on behalf of multiple data owners (i.e., medical institutions) within a classical outsourcing setup (cf. Section 1.1). Similar as for ppDBSCAN (cf. Section 2.1.2), we hereby effectively combine multiple different SMPC techniques: GCs [Yao86] and arithmetic and Boolean secret sharing. Note that SPIKE follows a greedy strategy for efficiency reasons that might output a "locally" optimal set of exchange cycles instead of the global optimum of all possible set of exchange cycles. We argue that this is sufficient because, in practice, donors and patients are expected to have sparse compatibility. Therefore, SPIKE is likely to output the true global optimum or a solution very close to it, even when using the greedy approach. Our intuition was recently validated by Breuer et al. [BMW23] who tested SPIKE on real-world data from the United Network for Organ Sharing (UNOS)[2]. Their results indicate that SPIKE's output contains on average about 75% to 80% of the globally optimal matches of donors and patients for a dataset with 50 to 200 pairs in total. In the following, we provide an overview of our 2PC sub-protocols developed for each phase. The modular structure of SPIKE allows for easy modification to incorporate additional medical factors or cater to specific expert requirements.

[2] https://unos.org/

1. *Compatibility Graph Computation* encompasses the process of first checking for all combinations of each patient P with each donor D for general incompatibility due to a potential HLA mismatch (cf. Table 4 in [**BHK+22**]). Subsequently, the quality of the match is evaluated based on HLA matching (cf. Table S1 in [**BHK+22**]), ABO blood group (cf. Table S1 in [**BHK+22**]), age (cf. Table S4 in [**BHK+22**]), sex (cf. Table S5 in [**BHK+22**]), and weight (cf. Table S6 in [**BHK+22**]). To quantify the quality of the combination of P and D in one score, all factors are combined using a weighted summation (cf. Table 5 in [**BHK+22**]).

   For each of the six factors, we employ categorization labels such as *A* for optimal fit, *B* for good fit, or *Eq* for identical age classes, among others. This process primarily involves performing comparisons or equality checks and subsequently categorizing the results using multiplexers. Our micro-benchmarks indicate that the evaluation of this phase is most efficient with a Boolean GMW instantiation [GMW87] (cf. Section 1.1), which avoids any conversion costs. Only the weighted summation of quality scores is performed using Arithmetic sharing, as it is the most efficient method for linear operations. We additionally optimize this phase by computing the comparisons in a Single Instruction, Multiple Data (SIMD) fashion, effectively reducing both memory and runtime overhead.

2. *Cycle Detection* involves calculating the number of potential exchange cycles based on the input of the desired cycle length (typically 2 or 3 for practical reasons). To accomplish this, an unweighted adjacency matrix is generated from the compatibility graph obtained in Phase 1 (cf. Tables 6 and S7 in [**BHK+22**]). The matrix entries indicate the number of paths of the desired length that start at vertex i and end at vertex j. For cycles, the entries lie on the diagonal since the start and end vertices are the same. The total number of potential exchange cycles is obtained by summing the entries on the diagonal and correcting it for duplicated ("congruent") cycles (cf. Table 6 in [**BHK+22**]).

   To optimize this process, the unweighted adjacency matrix is efficiently computed using the Boolean GMW protocol [GMW87] (cf. Section 1.1). Then, to perform the necessary linear operations for computing the number of cycles, we convert the representation to Arithmetic GMW, ensuring efficient execution.

3. *Cycle Evaluation* returns a descending list of unique exchange cycles based on their compatibility score, which represents the likelihood of their success according to the evaluated criteria. The process begins by computing the relevant circuits (cf. Table 7 in [**BHK+22**]). These circuits must have the correct length, be mutually exclusive (i.e., not containing the same patient/donor pairs), and consist only of compatible donor and patient pairs. Following this, the sorting is performed using a k-Nearest Neighbors (kNN) protocol inspired by Järvinen et al. [JLL+19] (cf. Table S8 in [**BHK+22**]).

   To optimize efficiency, we disclose the output of phase 2, which is the number of existing exchange cycles. This number is then used as the value of k for the subsequent sorting with kNN, narrowing down the sorting space to the required scope. We argue that

this approach is acceptable as the number of exchange cycles is an aggregated value and not directly linked to individual identities. This interpretation aligns with the legal perspective: For example, Article 4 of the General Data Protection Regulation (GDPR) considers personal information as "any information relating to an identified or identifiable natural person".[3]

The computational complexity of identifying exchange cycles depends on three factors: The number of donor-patient pairs, their compatibility as they define the number of potential exchange cycles, and the desired length of the cycles. Considering the non-linear nature of many operations (such as comparisons and multiplexers) that dominate this phase, as well as the depth of the circuits, we have chosen a GC-based instantiation [Yao86] (cf. Section 1.1). This decision is driven by the constant number of communication rounds of GCs.

4. *Solution Evaluation* determines the set of exchange cycles with the highest likelihood of success (cf. Tables 9 and S12 in [**BHK+22**]). It is crucial that these cycles are disjoint, as each donor can only be matched with one patient (cf. Table S11 in [**BHK+22**]).

   Considering the circuit depth involved in finding the set with the maximum likelihoods, we have instantiated the former using GCs [Yao86] (cf. Section 1.1). On the other hand, our micro-benchmarks have shown that achieving disjointness is most efficient using Boolean GMW [GMW87] (cf. Section 1.1).

Impact. As of July 16, 2023, the demand for kidney donations in the US remains substantial, with more than 90 000 individuals on the waiting list, according to the Organ Procurement and Transplantation Network of the U.S. Department of Health & Human Services [PN23]. The average waiting time for a kidney donation is between 3 to 5 years [Fun23]. However, the number of transplants performed annually is relatively low, with less than 20 000 transplants from deceased donors and less than 7 000 transplants from living donors conducted each year [PN23]. The considerable gap between the number of individuals requiring a kidney transplant and the limited availability of organs emphasizes the pressing need for an expansion in living kidney donations. One potential solution to address this challenge is the wider implementation of kidney exchange programs. By facilitating the exchange of kidneys between compatible donors and recipients, these programs have the potential to increase the pool of available organs and improve the chances of finding suitable matches. However, stringent privacy regulations for medical data can pose challenges for smaller health care institutions hindering them to participate in those programs due to limited resources to address the associated bureaucratic burden. Secure computation-based solutions like SPIKE can serve as a gateway for these entities by offering provable data privacy. By potentially enabling more institutions to participate, they have the potential to increase the number of kidney donations and democratize health care by enabling a wider access.

[3] https://gdpr-info.eu/art-4-gdpr/.

SPIKE itself represents an important step towards making private KEP practical for real-world deployment. This is due to its significantly improved efficiency compared to the previous state-of-the-art [BMWM20; BMW22]. It outperforms the runtime of the Threshold Paillier-based KEP protocol [BMWM20] by a factor of about 30 000× with 9 pairs and a cycle length of 3. In addition, when compared to the SMPC-based approach [BMW22], it demonstrates an average runtime improvement of approximately 400× for 40 pairs and a cycle length of 3. Furthermore, it is more robust thanks to the inclusion of additional medical factors.

3.1.2 Privacy-Preserving Epidemiological Modeling

Within the context of this thesis, we addressed the issue of privacy-preserving epidemiological modeling. The initial results were presented as a poster [**GHJ+22**] at the ACM Conference on Computer and Communications Security (CCS) and our final results can be found in the following technical report:

[GHJ+23] D. Günther, M. Holz, B. Judkewitz, H. Möllering, B. Pinkas, T. Schneider, A. Suresh. **"Privacy-preserving epidemiological modeling on mobile graphs"**. https://ia.cr/2020/1546. Code: https://zenodo.org/record/6599225. 2023. Appendix E.

We first motivate the problem of privacy-preserving epidemiological modeling. Then, we present our framework RIPPLE which formalizes the requirements of privacy-preserving epidemiological modeling. Subsequently, we summarize our two practical instantiations: One is based on Trusted Execution Environments (TEE) while the other relies on cryptographic techniques. In this scope, we also introduced PIR-SUM, a novel Private Information Retrieval (PIR) construction (cf. Section 1.1), returning the sum of the results of multiple PIR queries that might be of independent interest.

Privacy-Preserving Epidemiological Modeling goes beyond contact tracing for which the privacy research community has shown great interest during the COVID-19 pandemic [AMX+20; CFG+20; CG20; HMA+20; IF20; TPH+20; Vau20; ABIV21; HMM+21; PR21; RBS21b]. While contact tracing takes a backward perspective when identifying individuals *after* they have come into contact with infected persons to mitigate the spread of the disease, epidemiological modeling simulates the spread of the disease in the future. This modeling has proven to be a valuable tool for governments and decision-makers in determining effective containment measures, particularly during the COVID-19 pandemic [MMA+20]. However, the accuracy of predictions in this research field is reduced due to the limited availability of precise contact information for the population [MHJ+08; REE08; SKL+10]. Privacy-Enhancing Technologies (PETs) can be a valuable mean for the creation of decentralized systems providing access to *accurate recent* contact information for epidemiological modeling without compromising the privacy of individual contacts.

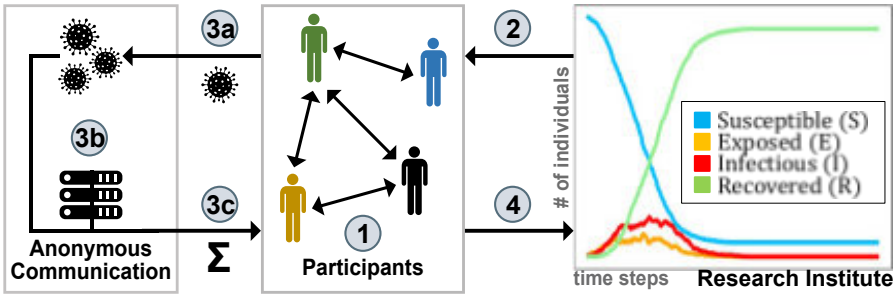


Figure 3.3: Overview of the RIPPLE Framework in [GHJ+23]. In step ①, the mobile devices of the participants collect anonymous encounter tokens during interactions. In step ②, the research institute begins a simulation by broadcasting the initialization parameters. In step ③a , the participants securely upload their infection likelihood to the servers. In step ③b, the servers securely compute the cumulative infection likelihood per participant. In step ③c, the participants retrieve their cumulative infection likelihood. In step ④, the aggregate results (#S,#E,#I,#R) are sent to the research institute.

RIPPLE realizes epidemiological modeling on recent contact information, running in a distributed fashion across mobile devices. These devices record encounters in the physical world similar to contact tracing apps like Germany's Corona-Warn-App[4]. The system encompasses three types of entities: A research institute, participants with mobile devices that contribute their contact information, and a set of one or more communication servers. The research institutes orchestrates the simulation, sets its parameters, and receives the simulation output. The participants are responsible for recording physical encounters and running the distributed simulation on their individual mobile devices. To facilitate communication among the participants for simulating infections, the communication servers establish anonymous channels through which messages can be exchanged.

We envision a hybrid security model: While it is realistic to assume that the research institute and communication server(s) behave semi-honestly as they are instantiated by governments or public health care institutes, we cannot assume this for all participants. For this reason, we assume in our work that some participants might deviate from the protocol to gain additional information, but do not try to manipulate or torpedo the simulation results. We recognize that it might be an interesting direction for future work to address stronger security models.

RIPPLE consists of four phases: Token Generation, Simulation Initialization, Simulation Execution, and Result Aggregation. The system overview is given in Figure 3.3, while Figure 2 in [GHJ+23] provides a detailed protocol description. The Token Generation means the encounter information collection phase executed by mobile devices similar to contact tracing apps. During the Simulation Initialization phase, the research institute defines the parameters of a simulation and broadcasts the necessary information for running a simulation to the

[4] https://www.coronawarn.app/en/

participants. Result Aggregation is a simple aggregation step of binary vectors which can be instantiated with a generic state-of-the-art secure aggregation protocol [ETLP13; **FMM+21**; KÖB21]. In the third phase, the Simulation Execution, the participants exchange anonymous messages indicating the likelihood of infections between each other. This is the critical step from a privacy perspective as it contains several pitfalls that we summarize next.

Linking Identities and Sybil Attacks are two attacks by participants trying to extract information about the contact graph that led the design of our instantiations. Let's imagine we do the simulation phase in a straightforward fashion: Each participant computes a likelihood taking its collected encounter information as input to a formula provided by the research institute. Then, it anonymously sends the likelihood to the other participant it has met. This second participant receives such likelihoods from all participants it encountered during the simulated time period, aggregates them, and updates its infection status if the aggregated likelihood surpasses a certain threshold defined by the research institute. The main challenge is ensuring anonymous message exchanges between participants. But, if two participants meet multiple times and their encounter information is used in multiple simulations, the infection likelihoods between those encounters and simulations may correlate which is observable by the receiving participant. We call this information leakage a *linking identities attack*. To defend against this attack, we suggest that the receiver is only permitted to access the aggregated likelihood of all messages it receives.

To circumvent the aggregation and be able to access and analyse individual infection likelihoods, malicious participants might use multiple mobile devices to collect different encounters one-by-one. We call this a *sybil attack*. A registration process linked to individual tokens issued by governments can be an effective countermeasure that prevents an adversary from creating dummy identities.

Our Instantiations of the RIPPLE framework are called $\text{RIPPLE}_{\text{TEE}}$ *and* $\text{RIPPLE}_{\text{PIR}}$. They offer different trade-offs in terms of security assumptions and efficiency.

$\text{RIPPLE}_{\text{TEE}}$'s (cf. Figure 6 in [**GHJ+23**] in Appendix E) security relies on a TEE (cf. Section 1.1) at the mobile devices of each participant. The infection likelihood is securely stored, processed, and encrypted within the TEE of the sender. The TEE also guarantees are secure transmission through an anonymous communication channel to the intended recipient. The recipient runs the decryption and aggregation process within their own TEE. This approach ensures that participants in the communication do not have knowledge of the identities of other participants and can only access aggregated infection likelihoods. By implementing these measures, the system effectively safeguards against linking identity attacks, preserving the privacy and anonymity of the participants involved.

$\text{RIPPLE}_{\text{PIR}}$ (cf. Figure 8 in [**GHJ+23**] in Appendix E) is constructed based on cryptographic protocols and their respective security guarantees. The foundation of $\text{RIPPLE}_{\text{PIR}}$ is a generic multi-server PIR scheme such as [KO97; DHS14; DHS17; ACLS18; GH19; CK20; GHPS22],

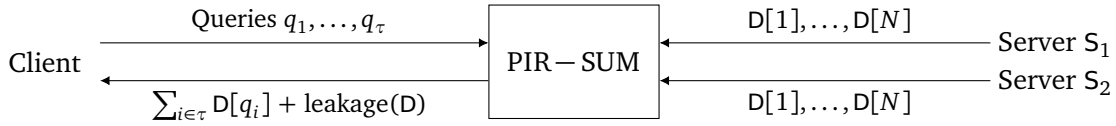


Figure 3.4: Ideal functionality of PIR − SUM where a client privately requests τ distinct database blocks by queries $q_1, \ldots, q_\tau$ from a public database D with N blocks without revealing the queries $q_i, i \in [\tau]$ to the PIR-servers. The client receives the sum of the requested database blocks $\sum_{i \in \tau} \mathsf{D}[q_i]$ and might also obtain some additional information about the database. Depending on the underlying PIR scheme more than two non-colluding PIR servers can be involved and they might not hold the entire database.

which allows a client to retrieve a specific element from a database without revealing which element was requested to the non-colluding servers holding the database (cf. Section 1.1 and Figure 1.3). Within RIPPLE$_{\text{PIR}}$, a novel construction called PIR − SUM is utilized. The ideal functionality is shown in Figure 3.4. This construction extends the functionality of multi-server PIR by restricting the client to retrieve only the aggregated sum of multiple queried blocks from the database, without disclosing the individual blocks to the participant or revealing which specific blocks were queried to the server(s). By leveraging PIR − SUM, RIPPLE$_{\text{PIR}}$ achieves the desired privacy protection against linking identity attacks. PIR − SUM is defined as follows:

Definition 3.1.1 (PIR − SUM)**.** Client C_i has a set of τ distinct indices denoted by $\mathcal{Q} = \{q_1, \ldots, q_\tau\}$ and wants to retrieve $\mathsf{res} = \sum_{i \in \tau} \mathsf{D}[q_i]$, where D is a database with N elements of ℓ-bits. D is held in the clear by two or more servers that do not learn any information about $\mathcal{Q}$.

Our PIR − SUM construction ensures security against linking identity attacks in RIPPLE$_{\text{PIR}}$. The details of our instantiation of PIR − SUM in [**GHJ+23**] are omitted here as the final optimized protocol is not part of our contributions. However, the concept, definition, and earlier versions of PIR − SUM are part of our contributions within this thesis.

Impact. The availability of recent and accurate contact information would significantly enhance the precision of epidemiologists' predictions regarding the trajectory of infectious diseases. This, in turn, would greatly influence the decision-making process of public health authorities when devising strategies for managing public health care. The significance of up-to-date contact data extends beyond extraordinary situations like the COVID-19 pandemic and encompasses various regular procedures, including the development of vaccination strategies and resource allocation for hospitals, personnel, and financial budgets.

Our microbenchmarks for the two instantiations serve as compelling evidence that the implementation of our protocols on an industry-scale can result in practical and usable systems. For example, our prototype implementation shows that RIPPLE$_{\text{TEE}}$ runs a simulation of 14 days with half a million participants in less than 4.5 minutes, while RIPPLE$_{\text{PIR}}$ also takes

only about 7 minutes. We assume a LAN setup with 10 Gbit/s and 0.1 s Round-Trip Time (RTT) between the PIR servers. Those numbers do not include the upload and download by the clients as those times can significantly vary between mobile devices. Both systems can be seamlessly integrated into existing contact tracing applications, leveraging the infrastructure that is already in place. The successful demonstration of our proof-of-concept highlights the feasibility of deploying our privacy-preserving epidemiological modeling in real-world scenarios, ultimately hopefully contributing to the enhancement of public health measures.

Our PIR — SUM building block might be of independent interest for usage in other contexts. It goes beyond secure aggregation protocols that were, for example, explored in the context of smart metering [KDK11; ETLP13] and Federated Learning (FL) [**FMM+21**; KÖB21; MOJC23] as it hides not only the individual responses but also the queries. This might be interesting in tax auditing processes, where government authorities need to verify the total income or expenditure of individuals or entities. Currently, such verification requires revealing detailed evidence of financial transaction information, which compromises privacy. With the utilization of PIR — SUM, a privacy-preserving audit process can be enabled. The protocol computes the sum of different transactions without exposing individual financial information or which queries were made. This approach serves dual purposes: On one hand, it allows tax authorities to verify overall income or expenditure without harming the privacy of individual taxpayers' financial data. On the other hand, it empowers authorities to conduct audits without alerting potential tax evaders. A real-world study by Bogdanov et al. [BKK+16] illustrates the practical usefulness of secure computation for investigating financial data. In their research, the authors privately analyze students' tax payments and study data to explore the relationship between graduation time and students working during their studies.

3.2 Related Work

In this subsection, we put SPIKE [**BHK+22**] and RIPPLE [**GHJ+23**] in context to previous work in Section 3.2.1. Then, we discuss the latest research findings that followed upon our work in Section 3.2.2.

3.2.1 Previous Work

We start by presenting two earlier works on privacy-preserving approaches to the KEP, before summarizing related work on contact tracing and other privacy research results in the context of epidemiology.

Secure Computation Protocols for Solving the KEP. Before SPIKE [**BHK+22**], the problem of privacy-preserving kidney exchange protocols was addressed by two works by Breuer et al. [BMWM20; BMW22]. In their initial work [BMWM20], the authors introduced the problem to the privacy community and proposed a privacy-preserving protocol based on

a Threshold variant of the Paillier Homomorphic Encryption (HE) scheme [FPS01]. This protocol is designed for a multi-party computation scenario involving an arbitrary number of incompatible donor-patient pairs assumed to behave semi-honestly. Concurrently to SPIKE, Breuer et al. presented another secure multi-party computation (SMPC) protocol for the KEP in their subsequent work [BMW22]. This protocol utilizes the Ben-Or-Goldwasser-Wigderson (BGW) [BGW88] based on Shamir's Secret Sharing [Sha79] in an outsourcing scenario. Both protocols by Breuer et al. construct an adjacency matrix which is then used to find an exchange cycle set that maximizes the number of compatible patient-donor pairs. However, the exhaustive "search" strategy employed in their first protocol [BMWM20] has an exponential computation overhead, leading to a runtime of 13 hours for only 9 donor-patient pairs with a LAN network with 1Gb/s bandwidth and 1ms latency. In their second protocol [BMW22], Breuer et al. significantly improved efficiency by reformulating the problem to a maximum weight matching for finding "cross-matches" of size two, i.e., exchange cycles are limited to exactly two donor-patient pairs. With an implementation that uses three computing parties in the outsourcing scenario, the second protocol successfully finds cross-matches for 13 pairs in 16 minutes.

In both of their studies [BMWM20; BMW22], Breuer et al. focus solely on the HLA crossmatch and the ABO blood group (cf. [**BHK+22**, Section Background] in Appendix D) when evaluating compatibility. However, SPIKE expands on this approach by incorporating four additional factors (HLA match, age, weight, sex). This can enhance the medical robustness of the output matching, thereby increasing the likelihood for a positive decision of the medical experts to proceed with the transplants as suggested by the SPIKE's output. Similar to the second work [BMW22], SPIKE's modular structure also allows to dynamically include or exclude pairs of donors and patients as they may drop out or newly join over time without having to re-run all four phases for all inputs. Moreover, SPIKE offers significantly improved efficiency compared to both prior works. This is thanks to our fine-tuned SMPC protocols that efficiently combine GCs and Boolean and Arithmetic GMW (cf. Section 1.1) based on cycle depth and operation (non-)linearity such that communication and computation overhead is optimized. Concretely, SPIKE requires 1.59 seconds for the matching with all six medical factors of 9 donor-patient pairs and a cycle length of 3 instead of the 13 hours of [BMWM20] with only two medical parameters. With 40 donor-patient pairs and a cycle length of two, SPIKE completes in 3.6 minutes, while the approach by Breuer et al. [BMW22] takes 25 hours in a LAN setup with 1Gb/s bandwidth and 1ms RTT.

Contact Tracing and Privacy-Preserving Medical Data Analyses. To the best of our knowledge, our work introduces privacy-preserving distributed epidemiological modeling on real contact graphs to the privacy community for the first time, with no previous solutions to this problem. Araki et al. [AFO+21] present a honest majority Secure Three-Party Computation (3PC) protocol for privacy-preserving graph analysis in an *outsourcing scenario* and name epidemiological modeling as potential application. However, collusion between two of the computing parties can result in the complete reconstruction of the contact graph. In contrast, our work allows to run such simulations in a fully distributed manner among a large

set of mobile devices without the need to rely on the non-collusion between servers. This can foster trust motivating more people to join the simulation.

During the COVID-19 pandemic in 2020, the privacy and security challenges of contact tracing gained significant attention [AMX+20; CIY20; Fra20; TPH+20; Vau20; NMD+22], which served as a source of inspiration for our work. In the context of contact tracing applications, two main design approaches emerged: Decentralized and centralized design architectures, each with its own advantages and disadvantages, leading to intense discussions especially on the respective privacy guarantees [Vau20]. Decentralized systems do not collect or process sensitive information centrally, so that contacts of infected persons are determined locally on users' mobile devices [Con20; Vau20; NMD+22]. Examples of such decentralized systems include [CKL+20; CFG+20; TPH+20; AG21; ABIV21; PR21]. On the other hand, centralized systems involve a central authority in the computation and data storage process, such as the generation of tokens in physical encounters [Vau20], or the determination and information of contacts [NMD+22]. Examples of centralized contact tracing systems are [IF20; SH20]. There are also hybrid designs and alternative approaches that have been explored, like [DPT20; TSS+20; BDH+21; RBS21b]. Ahmed et al. [AMX+20] and Wen et al. [WZL+20] have comprehensively surveyed and compared multiple contact tracing apps, and Sun et al. [SWX+21] have analyzed the security and privacy aspects of contact tracing apps on Android operating systems using their COVIDGUARDIAN tool accessing potential risks by static program and data flow analyses.

Beyond contact tracing, cryptographic research has also investigated other aspects of privacy-preserving data analysis in the context of infectious diseases. For instance, during the COVID-19 pandemic, Lueks et al. [LGV+21] introduced CrowdNotifier, a presence tracing system that notifies users in a privacy-preserving manner if they have come into contact with someone who tested positive. Lighthouses [RBS21a] is an extension of the decentralized contact tracing architecture by Apple and Google [AG21]. It provides warnings to users about super-spreader locations. PRISC [FZWX23] presents an idea for computing a heatmap indicating the infection likelihood for different locations. Similarly, Bampoulidis et al. [BBH+22] give a private set intersection protocol that combines location data of mobile devices provided by mobile network operators with information about infected individuals input by health authorities for infection hotspot detection. Barsocchi et al. [BCC+21] introduce an indoor localization system incorporating a GDPR-based access control system to safeguard personal data. The system's purpose is to guide users through public spaces, such as supermarkets, while ensuring they maintain maximum distance from others, thus encouraging social distancing.

In the broader context of health care and bioinformatics, Barni et al. [BFK+09a; BFK+09b; BFK+09c; BFL+11] and Mansouri et al. [MBÖE20] have presented multiple secure evaluation protocols specifically focused on the classification of ElectroCardioGram (ECG) signals. Moreover, other works address privacy-preserving solutions for medical imaging [KMRB20; TÖHL22], genomic data analysis [DHSS17; TS18; TWSH18; ST19; ICÖ21; HKST22], and biometric feature analysis, including biometric identification [EFG+09; SSW09; SZ13; BCF+14; DSZ15; TNK+19]. Ciceri et al. [CMÖE19] discuss the development of HE- and SMPC-based

classification systems for arrhythmia detection and stress management. Additionally, Cremones et al. [CVC+23] provide a comprehensive exploration of various aspects related to the practical usability of FL in real-world health care applications.

3.2.2 Subsequent Work

In this subsection, we compare SPIKE to three publications that were released in parallel or subsequently. Then, we also discuss two follow-up works of RIPPLE.

Secure Computation Protocols for Solving the KEP. Concurrently to our work, Brüggemann et al. [BBK+22] presented an SMPC protocol for approximative maximum weight matching and mentioned kidney exchange as one of its potential applications. It offers high efficiency and the protocol guarantees to return half of the maximally possible weight in the worst case. The authors do not evaluate the effect of their approximation with respect to kidney exchange. Their protocol's functionality corresponds to SPIKE's procedure when the cycle length is set to two. Brüggemann et al.'s reported benchmarks cannot be directly compared to ours as they instantiate their protocol in a honest-majority 3PC setting with MP-SDPZ [Kel20] while we use 2PC.

Later, Breuer et al. [BHP+22] presented a follow-up work addressing the KEP with an integer programming approach. This new protocol builds on BGW [BGW88] with Shamir's Secret Sharing [Sha79] and is designed for an outsourcing scenario (cf. Section 1.1), involving three or more computing parties holding the secret-shared medical information of donor-patient pairs. The authors build up upon the plaintext Branch-and-bound algorithm introduced in [LD10], which divides the global optimization problem into multiple sub-problems in a tree structure for improved efficiency. Each sub-problem is then addressed using the privacy-preserving Simplex algorithm protocol proposed by Toft [Tof09]. Similar as their first work in [BMWM20], this new approach can handle arbitrary cycle lengths, but offers significantly faster runtimes for cycle sizes larger than 12 donor-patient pairs. For instance, the protocol only takes 6 minutes for a execution with a cycle size of 3 and 12 donor-patient pairs, while the re-implementation of [BMWM20] (with the original HE scheme replaced by Shamir's Secret Sharing) requires 4 hours in a LAN setting with a bandwidth of 1GB/s and a RTT of 1ms. However, this enhanced efficiency comes with a trade-off as the protocol leaks the structure of the Branch-and-Bound tree with respect to the pruning information and the number of Simplex iterations of each sub-problem.

Recently, Breuer et al. [BMW23] presented a follow-up work that directly builds upon SPIKE's greedy strategy and proposes a new SMPC protocol for an approximate solution to the KEP. Unlike SPIKE, their protocol does not reveal the number of existing exchange cycles, making it more privacy-preserving in this aspect. Instead of using the set of all possible exchange cycles in each iteration, [BMW23] optimize complexity by evaluating the maximum weight for each subset first. The authors implement their protocol using MP-SPDZ [Kel20] and replicated secret-sharing [AFL+16] with three computing parties assuming an honest majority.

Benchmark results show that [BMW23] is significantly faster than the 2PC-instantiation of SPIKE for matching more than 10 donor-patient pairs. For example, they report that they outperform SPIKE by a factor of 12.8× for 15 pairs in a LAN setting with 1 GB/s bandwidth and 1 ms RTT. Breuer et al. [BMW23] also conduct a quality evaluation using a self-implemented plaintext version of SPIKE and report slightly better performance for their approach, which they attribute to considering cycles of length 2 and 3, while SPIKE focuses on cycles of either length 2 or 3.

Regarding robustness, both [BBK+22] and [BHP+22] focus solely on the HLA crossmatch and the ABO blood group as compatibility criteria between donors and patients. In contrast, [BMW23] follows SPIKE's approach and enhances the robustness of the matching process by incorporating additional factors such as sex, age, HLA match, but also the geographic closeness and the calculated panel reactive antibody (CPRA) score of the patient.

We summarize key aspects of previous and subsequent related work in comparison to SPIKE in Table 3.1.

Paper	PETs	Participant Scenario	Cycle Length	Exact	Privacy	Efficiency
Breuer et al. [BMWM20, WPES'20]	Paillier	MPC	$\mathcal{A}$	✓	●	−−
Breuer et al. [BMW22, CODASPY'22]	SSS/BGW	MPC/Outsourcing	2	✓	●	−
SPIKE [BHK+22, BMC'22]	**GCs & GMW**	**2PC Outsourcing**	$\mathcal{A}^a$	✗	◐	+
Brüggemann et al. [BBK+22, WPES'22]	ASS	MPC	2	✗	●	++
Breuer et al. [BHP+22, PST'22]	SSS/BGW	MPC/Outsourcing	$\mathcal{A}$	✓	○	++
Breuer et al. [BMW23]	ASS	MPC	2 + 3	✗	●	++

[a] Cycle size is an input to the protocol and fixed during a matching.

Table 3.1: Comparative overview of related work on secure computation protocols addressing the KEP. Paillier is a threshold variant of the Paillier encryption scheme [FPS01], SSS/BGW is the BGW [BGW88] based on Shamir's secret sharing [Sha79], GCs are Yao's Garbled Circuits [Yao86], GMW is arithmetic and Boolean secret sharing based on the GMW protocol [GMW87], and ASS is an arbitrary arithmetic linear secret sharing scheme. The protocols have either a multi-party computation (MPC) or a two-party computation (2PC) scenario with or without outsourcing. $\mathcal{A}$ are exchanges cycles of arbitrary length. ✓ indicates whether the protocol finds a globally optimal solution to the KEP, ✗ is a greedy strategy. ● is full privacy, while ◐ reveals the number of existing exchange cycles, and ○ leaks pruning information and number of Simplex iterations for each sub-problem. +/− rates the computational efficiency in comparison to each other. Note that [BHP+22; BBK+22; BMW23] have not been experimentally compared which is why we cannot directly compare their concrete computation efficiency. Our SPIKE (cf. Section 3.1.1) is highlighted in gray and bold.

Privacy-Preserving Epidemiological Data Analysis. To go beyond epidemiological modeling and contact tracing, Martinico et al. [MAZW22] propose Glass-Vault an extension to centralized contact tracing applications that enables general-purpose privacy-preserving analysis of data from infected users. Going into a similar direction, Cheng et al. [CLK23] present a theoretical concept for a data collection platform based on SMPC for privacy-preserving

epidemiological data analysis. The authors concentrate on analyzing the relationship between the number of users, colluding and non-colluding servers, and the desired privacy guarantees. Although Glass-Vault [MAZW22] and Cheng et al. [CLK23] both aim at enabling privacy-preserving epidemiological data analysis, the two works do not reference or compare each other.

4 Conclusion

In this chapter, we conclude this thesis by summarizing our main results in Section 4.1. We also look ahead to possible future research in Section 4.2, suggesting areas that can be explored further based on the findings of this thesis.

4.1 Summary

This thesis focused on advancing the field of privacy-preserving data analysis to make it more practical for real-world applications. We accomplished this through several key contributions that address the three of the four challenges towards practical Privacy-Preserving Machine Learning (PPML) discussed in Chapter 1 that are solvable with technical means: C1 (Efficiency), C2 (Usability), and C3 (Interdisciplinarity). Firstly, we addressed challenges C2 and C3 with a comprehensive survey and systematization of the current state-of-the-art in privacy-preserving clustering. Building upon these insights, we addressed C1, C2, and C3 with our efficient and fully privacy-preserving density-based clustering protocol capable of producing high quality results for various clustering problems including trajectory clustering and robust Federated Learning (FL) (cf. Chapter 2). Moreover, we introduced privacy-preserving solutions for two highly relevant data analysis challenges on distributed data in the health care domain for which we closely collaborated with computational biologists and a neurobiologist (C3) (cf. Chapter 3). Both results offer optimized efficiency (C1) and open-source implementations (C2). To conclude, our work identified and solved multiple open research questions that impeded real-world usability: We significantly enhanced privacy guarantees and efficiency in terms of communication and computation, addressed previously overlooked pre- and post-processing tasks in distributed data settings, and provided guidance on selecting and adapting secure computation protocols to suit specific application requirements. Our results represent a significant step towards making privacy-preserving data analysis more practical and applicable in diverse real-world scenarios.

Practical Clustering. In [HMSY21b], we conducted a thorough assessment of the state-of-the-art secure computation protocols for privacy-preserving clustering. This involved analyzing 59 publications and technical reports, evaluating aspects such as privacy guarantees, efficiency, participant scenarios, data partitioning, and clustering quality. We also provided a guideline for selecting the most suitable private clustering protocol for specific

applications without the need for in-depth cryptographic expertise. Based on our findings, we identified three key areas for future research in the area of secure computation protocols for privacy-preserving clustering: Additional secure protocols for state-of-the-art plaintext clustering methods, for pre- and post-processing, and overall efficiency improvement for communication, computation, and memory consumption. Our contributions lead to better usability by Machine Learning (ML) practitioners (C2) and are based on the requirements of the ML community (C3).

Drawing from these insights, we then developed the first fully privacy-preserving and practical Secure Multi-Party Computation (SMPC) protocol for Density-based Spatial Clustering of Applications with Noise (DBSCAN) in [**BCE+21**], referred to as ppDBSCAN. This protocol offers good clustering quality, as it can detect clusters of arbitrary shapes, handle outliers effectively, and flexibly determine the number of clusters based on input data. Notably, ppDBSCAN only requires two input parameters: The maximal distance between two input records to be considered as neighbors and the minimal cluster size. Both can often be set based on specific applications or approximated one of the data owners are jointly using an SMPC protocol. This makes our protocol highly adaptable to a wide range of real-world applications. Multiple optimizations lead to practical efficiency (C1) that can be verified with our open-source implementation (C2). Additionally, we showcased the practical usability of ppDBSCAN in a trajectory clustering use case (C3) and in a defense system in [**NRC+22**] that safeguards against privacy attacks and manipulation attempts in the context of FL.

In summary, our research in the scope of this thesis significantly enhanced the state-of-the-art in privacy-preserving clustering and the results mark an important step towards making privacy-preserving clustering truly practical for real-world applications.

Health Care Applications. In [**BHK+22**], we proposed the SMPC protocol SPIKE addressing the Kidney Exchange Problem (KEP) in close collaboration with computational biologists (C3). It realizes the matching between donors and patients through weighted summation of six compatibility scores. SPIKE's open-source prototype implementation outperforms previous privacy-preserving KEP protocols in computation, with gains of two to five orders of magnitude in a LAN setting with cycle sizes of 2 or 3 (C1 + C2). The expanded compatibility evaluation enhances the robustness of matchings, while improved efficiency enables its use in large matching pools, potentially benefiting more patients in need of a donation (C2).

Additionally, we introduced the concept of privacy-preserving epidemiological modeling in [**GHJ+23**] and defined its requirements in our RIPPLE framework. Furthermore, we presented, implemented, and evaluated two specific open-source instantiations based on Trusted Execution Environments (TEE) or Private Information Retrieval (PIR) (C2). With our work on epidemiological modeling, we aim to contribute to the field of epidemiological research by enabling access to recent contact information in a privacy-preserving manner for more accurate simulations of infectious disease spread and the potential effects of interventions. Our work is the result of a collaboration with a neurobiologist of the Charité — Universitätsmedizin Berlin (C3).

4.2 Future Work

In the following, we outline directions for potential follow-up work for privacy-preserving clustering in Section 4.2.1 and for health care data analyses in Section 4.2.2: We recommend to realize additional clustering techniques with Homomorphic Encryption (HE) or SMPC, adopting an end-to-end approach that encompasses pre- and post-processing steps alongside the core clustering process, and focusing on developing solutions that are practical and accessible for ML practitioners.

4.2.1 Real-World Privacy-Preserving Clustering

In Section 2.1.1, we already summarized the limitations of existing secure computation protocols for privacy-preserving clustering which we identified in our Systematization of Knowledge (SoK) in [**HMSY21b**]. Here, we provide more details and introduce additional ideas.

Latest Advancements in Plaintext Clustering Research. Until now, privacy research has adapted merely eight clustering algorithms into privacy-preserving versions. However, no single clustering algorithm excels in all aspects, such as efficiency and clustering quality, across various clustering problems. Furthermore, these algorithms vary in the input parameters they demand, and determining suitable values when data is distributed among multiple owners can be more challenging for one than the other. To address this, a broader range of plaintext clustering algorithms should be transformed into privacy-preserving variants using HE or SMPC, ensuring compatibility with diverse application needs.

In addition to designing plaintext clustering algorithms optimized for different clustering challenges, the field of ML research has explored various aspects that have not yet received attention in the cryptographic community. For instance, online/incremental clustering methods have been developed to process streaming data or sequentially arriving data, enabling continuous updates to the clustering model as new data points are received [CCFM97; BH06]. This idea is highly relevant for secure computation-based clustering protocols, where efficiency is crucial. Moreover, multi-view clustering deals with datasets that have multiple representations or "views" of the same underlying data, combining information from these views to produce a more accurate and robust clustering result [CLL+22]. Similarly, multi-modal clustering [BJ07] involves clustering data from different modalities, such as text, images, and numerical data, which aligns well with the concept of distributed data in the context of private clustering. These examples demonstrate the need for interdisciplinary efforts with ML researchers to identify the required privacy-preserving solutions for different real-world clustering problems and settings, and to prioritize privacy, efficiency, and quality requirements accordingly.

Pre- and Post-processing. Pre- and post-processing are essential steps in the clustering process that aim to enhance the quality and effectiveness of the clustering results. Pre-processing involves various data preparation tasks, such as data cleaning, normalization, and feature selection, to ensure the data is suitable for clustering algorithms. It also includes the selection of an appropriate clustering algorithm and determining the values for its input parameters.

On the other hand, post-processing focuses on refining the clustering output to improve its overall quality. This can involve handling noise, merging small clusters to avoid overfitting, or improving cluster boundaries for better interpretability. Additionally, a crucial aspect of post-processing is evaluating the clustering quality, which can be challenging in unsupervised settings without labeled data. In such cases, internal clustering quality measures are often employed, which assess the similarity of elements assigned to the same cluster and the separateness of different clusters.

In the context of privacy-preserving clustering, addressing these pre- and post-processing steps becomes even more complex due to the constraints imposed by distributed/privately held data. Most of the aforementioned aspects have never been considered in the existing private clustering literature. Instead, the existing secure clustering protocols focus only on the core clustering process. Hence, they are mostly not yet ready to be deployed for real-world applications due to the missing steps. Therefore, privacy research should take an end-to-end perspective, considering the whole life-cycle of a clustering process when designing secure protocols, to ensure the obtained clusters are meaningful, interpretable, and of high quality for the specific application's requirements.

Comparability and Usability. In the future, ML practitioners are likely to decide upon the adoption of privacy-preserving clustering, making it essential to ensure maximum usability. This involves several crucial aspects: Firstly, the secure computation protocols should be easily comparable, e.g., based on the seven key factors identified in our SoK in [**HMSY21b**] (cf. Section 2.1.1), which include the used secure computation technique, the security model, the participant scenario, privacy guarantees, data partitioning, efficiency, and the plaintext clustering algorithm. Secondly, protocols must be user-friendly and adaptable to specific application requirements without requiring cryptographic expertise. Thirdly, while the cryptographic community often aims at improving communication and computation overhead, our research highlights that memory consumption can also become a bottleneck. Therefore, future private clustering protocols should consider memory optimization as well. All aforementioned factors emphasize the importance of open-source implementations and standardized benchmarks to facilitate and accelerate the adoption of privacy-preserving clustering solutions in practice.

4.2.2 PETs for Medical Data Analyses

In this section, we outline potential directions for future research in both secure protocols for the KEP and privacy-preserving epidemiological modeling. Moreover, we also highlight general avenues for future work on privacy-preserving data analyses in the health care domain.

Stronger Security Models. Until now, our SPIKE protocol [**BHK+22**] and related works have focused on the semi-honest security model, assuming adversaries honestly follow the protocol while attempting to gain additional information. While this may suffice for many scenarios, such as joint computation among generally trusted large, intra-national, or intra-European medical institutions that cannot centralize data due to legal requirements, there are situations that demand stronger security models. An example could be computations involving health care data of EU citizens across different jurisdictions [Eur21]. Brüggemann et al. [BBK+22] and Breuer et al. [BMW23]'s works implemented their secure computation protocols with replicated secret sharing by Araki et al. [AFL+16] within the MP-SPDZ framework [Kel20]. As MP-SPDZ also implements multiple maliciously secure SMPC techniques, these existing implementations can be readily adapted to the malicious security model with minimal implementation overhead. While SPIKE's protocol is agnostic to the underlying SMPC technique, a full re-implementation would be required to achieve malicious security since the ABY framework [DSZ15] used in our work provides only semi-honest security. However, implementing the existing protocols as they are straightforwardly with a maliciously secure SMPC techniques might result in sub-optimal efficiency. Similarly, our instantiations of RIPPLE [**GHJ+23**] did not account for potential manipulations by clients, which may be crucial when running simulations on a country scale with millions of users. Therefore, we advocate for further research efforts to design efficient solutions with stronger security guarantees for both applications.

KEP Optimization Goal, Approximation Effect, and Extended Functionalities. To address the KEP, two previous works by Breuer et al. [BMWM20; BMW22] focused on maximizing the number of kidney transplants. SPIKE [**BHK+22**] and the recent work by Breuer et al. [BMW23] introduced additional compatibility evaluation criteria to enhance the expected medical robustness of the output exchange cycles. SPIKE achieves this by maximizing the weighted sum of the criteria, allowing medical experts to fine-tune the weights accordingly. However, there are numerous other aspects that could be integrated into the optimization process. For example, Biró et al. [BVM+21] highlighted that many European countries prioritize the number of transplants in their kidney exchange programs as the primary optimization goal. Additionally, they also use criteria such as the minimal length of selected exchange cycles or waiting time of patients in need for donation as supplementary objectives or constraints in living donation kidney exchange programs.

Going beyond the combination of multiple optimization goals, there also are national differences in allowed or preferred exchange cycle sizes, policies regarding altruistic donations (donors without a patient in need), or rules about multiple (half-)compatible donors registering for one patient [BVM+21]. Advanced privacy-preserving approaches can be designed to incorporate these various optimization goals and tailor the protocols to meet specific national requirements, including the development of interfaces to facilitate interactions across national borders.

Collaborations with Medical and Legal Experts. In this paragraph, we touch upon a general aspect for privacy-preserving data analysis in health care domain, but also in other fields. Medical research plays a crucial role in society, with significant potential for advancements through data analysis and machine learning. Utilizing these tools can lead to valuable insights in diagnostics and treatments, ultimately improving patient outcomes. However, strict privacy regulations pose challenges by imposing stringent requirements to protect patients' personal information. While these regulations are essential for safeguarding patient privacy, they can limit the size and diversity of datasets available for medical research. Finding a balance between protecting patient privacy and enabling valuable medical research is therefore critical. Privacy-Enhancing Technologies (PETs) can offer promising solutions to address this issue as they allow to analyze and derive insights from distributed/private medical datasets while preserving patient privacy. But only with a collaborative approach it is possible to create actually useful systems: Firstly, joined efforts can effectively identify the most relevant and urgent medical use cases where privacy requirements currently hinder the optimal usage of available distributed medical data. Secondly, leveraging the combined expertise fosters the creation of tailored systems that offer provably secure privacy guarantees and, importantly, incorporate practical requirements based on the input of medical experts. This ensures that the resulting systems are usable and useful for the medical experts who will be utilizing the systems for their work and research at the end.

Secure computation techniques present a promising solution to preserve privacy while enabling data analysis, but they have not yet been explicitly addressed by legislation [HR22], cf. challenge C4 in Chapter 1. As a result, their deployment in health care data analysis currently faces legal uncertainties. To support the practical usage of new privacy-preserving technologies like secure computation, legislation should provide certifications and guidelines that define their legal requirements and ensure their lawful implementation in health care and other domains [HR22].

4.2.3 Recent Results in SMPC Research

In the last years, the field of SMPC has seen multiple fundamental improvements affecting the design of hybrid SMPC protocols. For example, function-dependent preprocessing, as in [CCPS19; PSSY21; BHS+23], offers efficiency benefits if all preprocessing is linked to a specific functionality. Similarly, Silent OT Extension [BCG+19] significantly reduces communication cost of Oblivious Transfers [ALSZ17] a cryptographic primitive used in many SMPC techniques [DSZ15]. Those advancements have an effect on how to design efficient hybrid protocols mixing those SMPC techniques. The Secure Two-Party Computation (2PC) framework ABY [DSZ15] that we use for ppDBSCAN (cf. Section 2.1.2), FLAME (cf. Section 2.1.3), and SPIKE (cf. Section 3.1.1) does not implement those latest developments. Thus, the efficiency of our protocols can be further optimized by accounting for those recent results in their design.

Bibliography

[AAUC18] A. ACAR, H. AKSU, A. S. ULUAGAC, M. CONTI. **"A survey on homomorphic encryption schemes: Theory and implementation"**. In: *Computing Surveys (CSUR)*. ACM, 2018.

[ABB+20] M. AZRAOUI, M. BAHRAM, B. BOZDEMIR, S. CANARD, E. CICERI, O. ERMIS, R. MASALHA, M. MOSCONI, M. ÖNEN, M. PAINDAVOINE. **"SoK: Cryptography for neural networks"**. In: *Privacy and Identity Management*. Springer, 2020.

[ABIV21] G. AVITABILE, V. BOTTA, V. IOVINO, I. VISCONTI. **"Towards defeating mass surveillance and SARS-CoV-2: The Pronto-C2 fully decentralized automatic contact tracing system"**. In: *Workshop on Secure IT Technologies against COVID-19 (CoronaDef)*. Internet Society, 2021.

[ABS07] D. J. ABRAHAM, A. BLUM, T. SANDHOLM. **"Clearing algorithms for barter exchange markets: Enabling nationwide kidney exchanges"**. In: *International Conference on Electronic Commerce (ICEC)*. ACM, 2007.

[ACLS18] S. ANGEL, H. CHEN, K. LAINE, S. SETTY. **"PIR with compressed queries and amortized query processing"**. In: *Symposium on Security and Privacy (SP)*. IEEE, 2018.

[Acq14] A. ACQUISTI. **"The economics and behavioral economics of privacy"**. In: *Privacy, big data, and the public good: Frameworks for engagement*. Cambridge University Press Cambridge, 2014.

[AFL+16] T. ARAKI, J. FURUKAWA, Y. LINDELL, A. NOF, K. OHARA. **"High-throughput semi-honest secure three-party computation with an honest majority"**. In: *Conference on Computer and Communications Security (CCS)*. ACM, 2016.

[AFO+21] T. ARAKI, J. FURUKAWA, K. OHARA, B. PINKAS, H. ROSEMARIN, H. TSUCHIDA. **"Secure graph analysis at scale"**. In: *Conference on Computer and Communications Security (CCS)*. ACM, 2021.

[AG21] APPLE, GOOGLE. **"Exposure notification privacy-preserving analytics (ENPA) white paper"**. In: https://covid19-static.cdn-apple.com/applications/covid19/current/static/contact-tracing/pdf/ENPA_White_Paper.pdf. Accessed: 06/29/23. 2021.

[ALP+21] A. ALI, T. LEPOINT, S. PATEL, M. RAYKOVA, P. SCHOPPMANN, K. SETH, K. YEO. **"Communication–computation trade-offs in PIR"**. In: *USENIX Security Symposium*. USENIX, 2021.

[ALR+17] V. B. ASHBY, A. B. LEICHTMAN, M. A. REES, P. X.-K. SONG, M. BRAY, W. WANG, J. D. KALBFLEISCH. **"A kidney graft survival calculator that accounts for mismatches in age, sex, HLA, and body size"**. In: *Clinical Journal of the American Society of Nephrology (CJASN)*. American Society of Nephrology, 2017.

[ALSZ17] G. ASHAROV, Y. LINDELL, T. SCHNEIDER, M. ZOHNER. **"More efficient oblivious transfer extensions"**. In: *Journal of Cryptology (JoC)*. Springer, 2017.

[AMX+20] N. AHMED, R. A. MICHELIN, W. XUE, S. RUJ, R. MALANEY, S. S. KANHERE, A. SENEVIRATNE, W. HU, H. JANICKE, S. K. JHA. **"A survey of COVID-19 contact tracing apps"**. In: *Access*. IEEE, 2020.

[AZL+21] E. ANTWI-BOASIAKO, S. ZHOU, Y. LIAO, Q. LIU, Y. WANG, K. OWUSU-AGYEMANG. **"Privacy preservation in distributed deep learning: A survey on distributed deep learning, privacy preservation techniques used and interesting research directions"**. In: *Journal of Information Security and Applications*. Elsevier, 2021.

[BBG19] G. BARUCH, M. BARUCH, Y. GOLDBERG. **"A little is enough: Circumventing defenses for distributed learning"**. In: *Advances in Neural Information Processing Systems (NeurIPS)*. Curran Associates, 2019.

[BBH+22] A. BAMPOULIDIS, A. BRUNI, L. HELMINGER, D. KALES, C. RECHBERGER, R. WALCH. **"Privately connecting mobility to infectious diseases via applied cryptography"**. In: *Proceedings on Privacy Enhancing Technologies (PoPETs)*. De Gruyter, 2022.

[BBK+22] A. BRÜGGEMANN, M. BREUER, A. KLINGER, T. SCHNEIDER, U. MEYER. **"Secure maximum weight matching approximation on general graphs"**. In: *Workshop on Privacy in the Electronic Society (WPES)*. ACM, 2022.

[BCC+21] P. BARSOCCHI, A. CALABRÒ, A. CRIVELLO, S. DAOUDAGH, F. FURFARI, M. GIROLAMI, E. MARCHETTI. **"COVID-19 & privacy: Enhancing of indoor localization architectures towards effective social distancing"**. In: *Array*. Elsevier, 2021.

[BCD+20] F. BOEMER, R. CAMMAROTA, D. DEMMLER, T. SCHNEIDER, H. YALAME. **"MP2ML: A mixed-protocol machine learning framework for private inference"**. In: *Conference on Availability, Reliability and Security (ARES)*. ACM, 2020.

[BCE+21] B. BOZDEMIR, S. CANARD, O. ERMIS, H. MÖLLERING, M. ÖNEN, T. SCHNEIDER. **"Privacy-preserving density-based clustering"**. In: *ASIA Conference on Computer and Communications Security (ASIACCS)*. Online: https://ia.cr/2021/612. Code: https://encrypto.de/code/ppDBSCAN. ACM, 2021, S. 658–671. CORE Rank A. Appendix B.

[BCF+14] J. BRINGER, H. CHABANNE, M. FAVRE, A. PATEY, T. SCHNEIDER, M. ZOHNER. **"GSHADE: Faster privacy-preserving distance computation and biometric identification"**. In: *Workshop on Information Hiding and Multimedia Security (IH&MMSEC'14)*. ACM, 2014.

[BCG+19] E. BOYLE, G. COUTEAU, N. GILBOA, Y. ISHAI, L. KOHL, P. SCHOLL. **"Efficient pseudorandom correlation generators: Silent OT extension and more"**. In: *Annual International Cryptology Conference (CRYPTO)*. Springer. 2019.

[BCMC19] A. N. BHAGOJI, S. CHAKRABORTY, P. MITTAL, S. CALO. **"Analyzing federated learning through an adversarial lens"**. In: *International Conference on Machine Learning (ICML)*. PMLR, 2019.

[BDC20] A. BOULEMTAFES, A. DERHAB, Y. CHALLAL. **"A review of privacy-preserving techniques for deep learning"**. In: *Neurocomputing*. Elsevier, 2020.

[BDH+21] W. BESKOROVAJNOV, F. DÖRRE, G. HARTUNG, A. KOCH, J. MÜLLER-QUADE, T. STRUFE. **"Contra corona: Contact tracing against the coronavirus by bridging the centralized–decentralized divide for stronger privacy"**. In: *International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT)*. Springer, 2021.

[BDST22] L. BRAUN, D. DEMMLER, T. SCHNEIDER, O. TKACHENKO. **"Motion–A framework for mixed-protocol multi-party computation"**. In: *Transactions on Privacy and Security (TOPS)*. ACM, 2022.

[BFK+09a] M. BARNI, P. FAILLA, V. KOLESNIKOV, R. LAZZERETTI, A. PAUS, A.-R. SADEGHI, T. SCHNEIDER. **"Efficient privacy-preserving classification of ECG signals"**. In: *International Workshop on Information Forensics and Security (WIFS)*. IEEE, 2009.

[BFK+09b] M. BARNI, P. FAILLA, V. KOLESNIKOV, R. LAZZERETTI, A.-R. SADEGHI, T. SCHNEIDER. **"Combining signal processing and cryptographic protocol design for efficient ECG classification"**. In: *Workshop on Signal Processing in the EncryptEd Domain (SPEED)*. 2009.

[BFK+09c] M. BARNI, P. FAILLA, V. KOLESNIKOV, R. LAZZERETTI, A.-R. SADEGHI, T. SCHNEIDER. **"Secure evaluation of private linear branching programs with medical applications"**. In: *European Symposium on Research in Computer Security (ESORICS)*. Springer, 2009.

[BFL+11] M. BARNI, P. FAILLA, R. LAZZERETTI, A.-R. SADEGHI, T. SCHNEIDER. **"Privacy-preserving ECG classification with branching programs and neural networks"**. In: *Transactions on Information Forensics and Security (TIFS)*. IEEE, 2011.

[BG18] J. BUOLAMWINI, T. GEBRU. **"Gender shades: Intersectional accuracy disparities in commercial gender classification"**. In: *Conference on Fairness, Accountability and Transparency (FAccT)*. PMLR, 2018.

[BGI15] E. BOYLE, N. GILBOA, Y. ISHAI. **"Function secret sharing"**. In: *International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*. Springer, 2015.

[BGW88] M. BEN-OR, S. GOLDWASSER, A. WIGDERSON. **"Completeness theorems for non-cryptographic fault-tolerant distributed computation"**. In: *Symposium on Theory of Computing (STOC)*. ACM, 1988.

[BH06] J. BERINGER, E. HÜLLERMEIER. **"Online clustering of parallel data streams"**. In: *Data & Knowledge Engineering*. Elsevier, 2006.

[BHK+22] T. BIRKA, K. HAMACHER, T. KUSSEL, H. MÖLLERING, T. SCHNEIDER. **"SPIKE: Secure and private investigation of the kidney exchange problem"**. In: *BMC Medical Informatics and Decision Making* 22.1 (2022). Online: https://arxiv.org/abs/2204.09937. Code: https://github.com/encryptogroup/ppke, S. 253. CORE Rank B. Appendix D.

[BHP+22] M. BREUER, P. HEIN, L. POMPE, B. TEMME, U. MEYER, S. WETZEL. **"Solving the kidney exchange problem using privacy-preserving integer programming"**. In: *International Conference on Privacy, Security & Trust (PST)*. IEEE, 2022.

[BHS+23] A. BRÜGGEMANN, R. HUNDT, T. SCHNEIDER, A. SURESH, H. YALAME. **"FLUTE: Fast and Secure Lookup Table Evaluations"**. In: *Symposium on Security and Privacy (SP)*. IEEE, 2023.

[BJ07] R. BEKKERMAN, J. JEON. **"Multi-modal clustering for multimedia collections"**. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2007.

[BKK+16] D. BOGDANOV, L. KAMM, B. KUBO, R. REBANE, V. SOKK, R. TALVISTE. **"Students and taxes: A privacy-preserving study using secure computation"**. In: *Proceedings on Privacy Enhancing Technologies (PoPETs)*. De Gruyter, 2016.

[BMD+17] F. Brasser, U. Müller, A. Dmitrienko, K. Kostiainen, S. Capkun, A.-R. Sadeghi. **"Software grand exposure: SGX cache attacks are practical"**. In: *USENIX Workshop on Offensive Technologies (WOOT)*. USENIX, 2017.

[BMR90] D. Beaver, S. Micali, P. Rogaway. **"The round complexity of secure protocols"**. In: *Symposium on Theory of Computing (STOC)*. ACM, 1990.

[BMW22] M. Breuer, U. Meyer, S. Wetzel. **"Privacy-preserving maximum matching on general graphs and its application to enable privacy-preserving kidney exchange"**. In: *Conference on Data and Application Security and Privacy (CODASPY)*. Elsevier, 2022.

[BMW23] M. Breuer, U. Meyer, S. Wetzel. **"Efficient privacy-preserving approximation of the kidney exchange problem"**. https://arxiv.org/pdf/2302.13880.pdf. 2023.

[BMWM20] M. Breuer, U. Meyer, S. Wetzel, A. Mühlfeld. **"A privacy-preserving protocol for the kidney exchange problem"**. In: *Workshop on Privacy in the Electronic Society (WPES)*. ACM, 2020.

[BO07] P. Bunn, R. Ostrovsky. **"Secure two-party K-means clustering"**. In: *Conference on Computer and Communications Security (CCS)*. ACM, 2007.

[Bra08] F. Brauer. **"Compartmental models in epidemiology"**. In: *Mathematical Epidemiology*. 2008.

[BSC+97] C. A. Bouman, M. Shapiro, G. Cook, C. B. Atkins, H. Cheng. **"Cluster: An unsupervised algorithm for modeling Gaussian mixtures"**. https://engineering.purdue.edu/~bouman/software/cluster/manual.pdf. 1997.

[BTS+15] A. Belle, R. Thiagarajan, S. Soroushmehr, F. Navidi, D. A. Beard, K. Najarian. **"Big data analytics in healthcare"**. In: *BioMed Research International*. Hindawi, 2015.

[BVH+20] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, V. Shmatikov. **"How to backdoor federated learning"**. In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*. PMLR, 2020.

[BVM+21] P. Biró, J. Van de Klundert, D. Manlove, W. Pettersson, T. Andersson, L. Burnapp, P. Chromy, P. Delgado, P. Dworczak, B. Haase. **"Modelling and optimisation in European kidney exchange programmes"**. In: *European Journal of Operational Research*. Elsevier, 2021.

[Can00] R. Canetti. **"Security and composition of multiparty cryptographic protocols"**. In: *Journal of Cryptology*. Springer, 2000.

[CCFM97] M. Charikar, C. Chekuri, T. Feder, R. Motwani. **"Incremental clustering and dynamic information retrieval"**. In: *Symposium on Theory of Computing (STOC)*. ACM, 1997.

[CCPS19] H. Chaudhari, A. Choudhury, A. Patra, A. Suresh. **"Astra: High throughput 3PC over rings with application to secure prediction"**. In: *Cloud Computing Security Workshop (CCSW)*. ACM, 2019.

[CD16] V. Costan, S. Devadas. **"Intel SGX explained"**. https://eprint.iacr.org/2016/086.pdf. 2016.

[CFG+20] J. Chan, D. Foster, S. Gollakota, E. Horvitz, J. Jaeger, S. Kakade, T. Kohno, J. Langford, J. Larson, P. Sharma, S. Singanamalla, J. Sunshine, S. Tessaro. **"PACT: Privacy sensitive protocols and mechanisms for mobile contact tracing"**. https://arxiv.org/pdf/2004.03544.pdf. 2020.

[CG20] M. CIUCCI, F. GOUARDÈRES. **"National COVID-19 contact tracing apps"**. In: *EPRS: European Parliamentary Research Service*. 2020.

[CGOS22] N. CHANDRAN, D. GUPTA, S. L. B. OBBATTU, A. SHAH. **"SIMC: ML inference secure against malicious clients at semi-honest cost"**. In: *USENIX Security Symposium*. USENIX, 2022.

[Cha81] D. L. CHAUM. **"Untraceable electronic mail, return addresses, and digital pseudonyms"**. In: *Communications of the ACM*. ACM, 1981.

[CHK22] H. CORRIGAN-GIBBS, A. HENZINGER, D. KOGAN. **"Single-server private information retrieval with sublinear amortized time"**. In: *International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*. Springer, 2022.

[CIY20] H. CHO, D. IPPOLITO, Y. W. YU. **"Contact tracing mobile apps for COVID-19: Privacy considerations and related trade-offs"**. https://arxiv.org/pdf/2003.11511.pdf. 2020.

[CK20] H. CORRIGAN-GIBBS, D. KOGAN. **"Private information retrieval with sublinear online time"**. In: *International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*. Springer, 2020.

[CKL+20] R. CANETTI, Y. T. KALAI, A. LYSYANSKAYA, R. L. RIVEST, A. SHAMIR, E. SHEN, A. TRACHTENBERG, M. VARIA, D. J. WEITZNER. **"Privacy-preserving automated exposure notification"**. https://eprint.iacr.org/2020/863.pdf. 2020.

[CKP19] J. H. CHEON, D. KIM, J. H. PARK. **"Towards a practical cluster analysis over encrypted data"**. In: *International Conference on Selected Areas in Cryptography (SAC)*. Springer, 2019.

[CKVR13] M. CONSTANTINO, X. KLIMENTOVA, A. VIANA, A. RAIS. **"New insights on integer-programming models for the kidney exchange problem"**. In: *European Journal of Operational Research*. Elsevier, 2013.

[CL18] S. CHANG, C. LI. **"Privacy in neural network learning: Threats and countermeasures"**. In: *Network*. IEEE, 2018.

[CLK23] J. CHENG, N. LIU, W. KANG. **"On the asymptotic capacity of information-theoretic privacy-preserving epidemiological data collection"**. In: *Entropy*. MDPI, 2023.

[CLL+22] M.-S. CHEN, J.-Q. LIN, X.-L. LI, B.-Y. LIU, C.-D. WANG, D. HUANG, J.-H. LAI. **"Representation learning in multi-view clustering: A literature review"**. In: *Data Science and Engineering*. Springer, 2022.

[CMC+11] M. C. CONRY, K. MORGAN, P. CURRY, H. MCGEE, J. HARRINGTON, M. WARD, E. SHELLEY. **"The clustering of health behaviours in Ireland and their relationship with mental health, self-rated health and quality of life"**. In: *BMC Public Health*. Springer, 2011.

[CMÖE19] E. CICERI, M. MOSCONI, M. ÖNEN, O. ERMIS. **"PAPAYA: A platform for privacy preserving data analytics"**. In: *ERCIM News*. ERCIM EEIG, 2019.

[CMS13] R. J. CAMPELLO, D. MOULAVI, J. SANDER. **"Density-based clustering based on hierarchical density estimates"**. In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*. Springer, 2013.

[Con20] P.-P. CONSORTIUM. **"PEPP-PT Documentation"**. https://github.com/pepp-pt/pepp-pt-documentation. Accessed: 07/23/23. 2020.

[CP21] J. CABRERO-HOLGUERAS, S. PASTRANA. **"SoK: Privacy-preserving computation techniques for deep learning"**. In: *Proceedings on Privacy Enhancing Technologies (PoPETs)*. De Gruyter, 2021.

[CPTH21] S. CHATEL, A. PYRGELIS, J. R. TRONCOSO-PASTORIZA, J.-P. HUBAUX. **"SoK: Privacy-preserving collaborative tree-based model learning"**. In: *Proceedings on Privacy Enhancing Technologies (PoPETs)*. De Gruyter, 2021.

[CS18] F. CHUNG, O. SIMPSON. **"Computing heat kernel PageRank and a local clustering algorithm"**. In: *European Journal of Combinatorics*. Elsevier, 2018.

[CSFP20] D. CERDEIRA, N. SANTOS, P. FONSECA, S. PINTO. **"SoK: Understanding the prevailing security vulnerabilities in TrustZone-assisted TEE systems"**. In: *Symposium on Security and Privacy (SP)*. IEEE, 2020.

[CSR+20] R. CAMMAROTA, M. SCHUNTER, A. RAJAN, F. BOEMER, Á. KISS, A. TREIBER, C. WEINERT, T. SCHNEIDER, E. STAPF, A.-R. SADEGHI, D. DEMMLER, H. CHEN, S. U. HUSSAIN, S. RIAZI, F. KOUSHANFAR, S. GUPTA, T. S. ROSING, K. CHAUDHURI, H. NEJATOLLAHI, N. DUTT, M. IMANI, K. LAINE, A. DUBEY, A. AYSU, F. S. HOSSEINI, C. YANG, E. WALLACE, P. NORTON. **"Trustworthy AI inference systems: An industry research view"**. https://arxiv.org/abs/2008.04449. 2020.

[CVC+23] F. CREMONESI, M. VESIN, S. CANSIZ, Y. BOUILLARD, I. BALELLI, L. INNOCENTI, S. SILVA, S.-S. AYED, R. TAIELLO, L. KAMENI, R. VIDAL, F. ORLHAC, C. NIOCHE, N. LAPEL, B. HOUIS, R. MODZELEWSKI, O. HUMBERT, M. ÖNEN, M. LORENZI. **"Fed-BioMed: Open, transparent and trusted federated learning for real-world healthcare applications"**. https://arxiv.org/pdf/2204.05136.pdf. 2023.

[CZK+18] H. CHO, P. ZHANG, D. KIM, J. PARK, C.-H. LEE, Z. ZHAO, A. DOUPÉ, G.-J. AHN. **"Prime+count: Novel cross-world covert channels on ARM TrustZone"**. In: *Annual Computer Security Applications Conference (ACSAC)*. ACM, 2018.

[DBH18] F. K. DOŠILOVIĆ, M. BRČIĆ, N. HLUPIĆ. **"Explainable artificial intelligence: A survey"**. In: *International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. IEEE, 2018.

[De 21] E. DE CRISTOFARO. **"A critical overview of privacy in machine learning"**. In: *Security & Privacy*. IEEE, 2021.

[DH06] T. DINEV, P. HART. **"An extended privacy calculus model for e-commerce transactions"**. In: *Information Systems Research*. Informs, 2006.

[DHS14] D. DEMMLER, A. HERZBERG, T. SCHNEIDER. **"RAID-PIR: Practical multi-server PIR"**. In: *Cloud Computing Security Workshop (CCSW)*. ACM, 2014.

[DHS17] D. DEMMLER, M. HOLZ, T. SCHNEIDER. **"OnionPIR: Effective protection of sensitive metadata in online communication networks"**. In: *Applied Cryptography and Network Security (ACNS)*. Springer, 2017.

[DHSS17] D. DEMMLER, K. HAMACHER, T. SCHNEIDER, S. STAMMLER. **"Privacy-preserving whole-genome variant queries"**. In: *Conference on Cryptology And Network Security (CANS)*. Springer, 2017.

[DK19] T. DAVENPORT, R. KALAKOTA. **"The potential for artificial intelligence in healthcare"**. In: *Future Healthcare Journal*. Royal College of Physicians, 2019.

[DKL+13] I. DAMGÅRD, M. KELLER, E. LARRAIA, V. PASTRO, P. SCHOLL, N. P. SMART. **"Practical covertly secure MPC for dishonest majority–or: breaking the SPDZ limits"**. In: *European Symposium on Research in Computer Security (ESORICS)*. Springer, 2013.

[DPC23] DPC-IRELAND. **"Data Protection Commission announces conclusion of inquiry into Meta Ireland"**. In: https://dataprotection.ie/en/news-media/press-releases/Data-Protection-Commission-announces-conclusion-of-inquiry-into-Meta-Ireland. Accessed: 06/29/23. 2023.

[DPT20] T. DUONG, D. H. PHAN, N. TRIEU. **"Catalic: Delegated PSI cardinality with applications to contact tracing"**. In: *International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT)*. Springer. 2020.

[DR+14] C. DWORK, A. ROTH. **"The algorithmic foundations of differential privacy"**. In: *Foundations and Trends® in Theoretical Computer Science*. Now Publishers, Inc., 2014.

[DR99] J. DAEMEN, V. RIJMEN. **"AES proposal: Rijndael"**. https://www.cs.miami.edu/home/burt/learning/Csc688.012/rijndael/rijndael_doc_V2.pdf. 1999.

[DSZ15] D. DEMMLER, T. SCHNEIDER, M. ZOHNER. **"ABY - A framework for efficient mixed-protocol secure two-party computation"**. In: *Network and Distributed System Security Symposium (NDSS)*. Internet Society, 2015.

[DWT+23] C. DONG, J. WENG, Y. TONG, J.-N. LIU, A. YANG, Y. CHENG, S. HU. **"Fusion: Efficient and secure inference resilient to malicious server and curious clients"**. In: *Network and Distributed System Security Symposium (NDSS)*. The Internet Society, 2023.

[DXLL09] L. DUAN, L. XU, Y. LIU, J. LEE. **"Cluster-based outlier detection"**. In: *Annals of Operations Research*. Springer, 2009.

[EFG+09] Z. ERKIN, M. FRANZ, J. GUAJARDO, S. KATZENBEISSER, I. LAGENDIJK, T. TOFT. **"Privacy-preserving face recognition"**. In: *Proceedings on Privacy Enhancing Technologies (PoPETs)*. Springer, 2009.

[EHB+03] A. E. EL-AGROUDY, N. A. HASSAN, M. A. BAKR, M. A. FODA, A. A. SHOKEIR, A. B. S. EL-DEIN, M. A. GHONEIM. **"Effect of donor/recipient body weight mismatch on patient and graft outcome in living-donor kidney transplantation"**. In: *American Journal of Nephrology*. S. Karger AG Basel, Switzerland, 2003.

[EKR+18] D. EVANS, V. KOLESNIKOV, M. ROSULEK. **"A pragmatic introduction to secure multi-party computation"**. In: *Foundations and Trends® in Privacy and Security*. Now Publishers, Inc., 2018.

[EKSX+96] M. ESTER, H.-P. KRIEGEL, J. SANDER, X. XU. **"A density-based algorithm for discovering clusters in large spatial databases with noise"**. In: *International Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, 1996.

[ELLS11] B. S. EVERITT, S. LANDAU, M. LEESE, D. STAHL. **"Cluster analysis"**. John Wiley & Sons, 2011.

[Esp98] A. L. ESPINOSA. **"Availability of health data: Requirements and solutions"**. In: *International Journal of Medical Informatics*. Elsevier, 1998.

[ETLP13] Z. ERKIN, J. R. TRONCOSO-PASTORIZA, R. L. LAGENDIJK, F. PÉREZ-GONZÁLEZ. **"Privacy-preserving data aggregation in smart metering systems: An overview"**. In: *Signal Processing Magazine*. IEEE, 2013.

[Eur21] EUROPEAN DATA PROTECTION BOARD. **"Recommendations 01/2020 on measures that supplement transfer tools to ensure compliance with the EU level of protection of personal data"**. In: https://edpb.europa.eu/our-work-tools/our-documents/recommendations/recommendations-012020-measures-supplement-transfer_en. 2021.

[FD07] B. J. FREY, D. DUECK. **"Clustering by passing messages between data points"**. In: *Science*. American Association for the Advancement of Science, 2007.

[FH75] K. FUKUNAGA, L. HOSTETLER. **"The estimation of the gradient of a density function, with applications in pattern recognition"**. In: *Transactions on Information Theory*. IEEE, 1975.

[FMM+21] H. FEREIDOONI, S. MARCHAL, M. MIETTINEN, A. MIRHOSEINI, H. MÖLLERING, T. D. NGUYEN, P. RIEGER, A.-R. SADEGHI, T. SCHNEIDER, H. YALAME, S. ZEITOUNI. **"SAFELearn: Secure aggregation for private federated learning"**. In: *Deep Learning and Security Workshop (DLS@SP)*. Full version: https://ia.cr/2021/386. IEEE, 2021, S. 56–62.

[FPS01] P.-A. FOUQUE, G. POUPARD, J. STERN. **"Sharing decryption in the context of voting or lotteries"**. In: *Financial Cryptography (FC)*. Springer, 2001.

[Fra20] FRAUNHOFER-AISEC. **"Pandemic contact tracing apps: DP-3T, PEPP-PT NTK, and ROBERT from a privacy perspective"**. https://eprint.iacr.org/2020/489.pdf. 2020.

[FTC19] FTC. **"FTC imposes $5 Billion penalty and sweeping new privacy restrictions on Facebook"**. In: https://www.ftc.gov/news-events/news/press-releases/2019/07/ftc-imposes-5-billion-penalty-sweeping-new-privacy-restrictions-facebook. Accessed: 06/29/23. 2019.

[Fun23] A. K. FUNC. **"Transplant waiting list"**. In: https://www.kidneyfund.org/kidney-donation-and-transplant/transplant-waiting-list. Accessed: 07/16/23. 2023.

[FZWX23] Y. FENG, Q. ZHANG, H. WU, C. XIN. **"PRISC: Privacy-preserved pandemic infection risk computation through cellular enabled IoT devices"**. In: *Internet of Things Journal*. IEEE, 2023.

[Gen09] C. GENTRY. **"Fully homomorphic encryption using ideal lattices"**. In: *Symposium on Theory of Computing (STOC)*. ACM, 2009.

[GESM17] J. GÖTZFRIED, M. ECKERT, S. SCHINZEL, T. MÜLLER. **"Cache attacks on Intel SGX"**. In: *European Workshop on Systems Security (EuroSec)*. ACM, 2017.

[GH19] C. GENTRY, S. HALEVI. **"Compressible FHE with applications to PIR"**. In: *Theory of Cryptography Conference (TCC)*. Springer, 2019.

[GHJ+22] D. GÜNTHER, M. HOLZ, B. JUDKEWITZ, H. MÖLLERING, B. PINKAS, T. SCHNEIDER, A. SURESH. **"POSTER: Privacy-preserving epidemiological modeling on mobile graphs"**. In: *Conference on Computer and Communications Security (CCS) Posters/Demos*. ACM, 2022, S. 3351–3353. CORE Rank A*.

[GHJ+23] D. GÜNTHER, M. HOLZ, B. JUDKEWITZ, H. MÖLLERING, B. PINKAS, T. SCHNEIDER, A. SURESH. **"Privacy-preserving epidemiological modeling on mobile graphs"**. https://ia.cr/2020/1546. Code: https://zenodo.org/record/6599225. 2023. Appendix E.

[GHPS22] D. GÜNTHER, M. HEYMANN, B. PINKAS, T. SCHNEIDER. **"GPU-accelerated PIR with client-independent preprocessing for large-scale applications"**. In: *USENIX Security Symposium*. USENIX, 2022.

[GHS12] C. GENTRY, S. HALEVI, N. P. SMART. **"Fully homomorphic encryption with polylog overhead"**. In: *Annual International Conference on the Theory and Applications of Cryptographic Technique (EUROCRYPT)*. Springer, 2012.

[GI14] N. GILBOA, Y. ISHAI. **"Distributed point functions and their applications"**. In: *International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*. Springer, 2014.

[GKK+02] S. GREENFIELD, S. H. KAPLAN, R. KAHN, J. NINOMIYA, J. L. GRIFFITH. **"Profiling care provided by different groups of physicians: Effects of patient case-mix (bias) and physician-level clustering on quality assessment results"**. In: *Annals of Internal Medicine*. American College of Physicians, 2002.

[GMW87] O. GOLDREICH, S. MICALI, A. WIGDERSON. **"How to play any mental game"**. In: *Symposium on Theory of Computing (STOC)*. ACM, 1987.

[GP10] D. GRAVES, W. PEDRYCZ. **"Kernel-based fuzzy clustering and fuzzy clustering: A comparative experimental study"**. In: *Fuzzy sets and systems*. Elsevier, 2010.

[GSW21] D. GÜNTHER, T. SCHNEIDER, F. WIEGAND. **"POSTER: Revisiting hybrid private information retrieval"**. In: *Conference on Computer and Communications Security (CCS) Posters/Demos*. ACM, 2021.

[HGX+17] Z. HUA, J. GU, Y. XIA, H. CHEN, B. ZANG, H. GUAN. **"vTZ: Virtualizing ARM TrustZone"**. In: *USENIX Security Symposium*. USENIX, 2017.

[HHC+23] A. HENZINGER, M. M. HONG, H. CORRIGAN-GIBBS, S. MEIKLEJOHN, V. VAIKUNTANATHAN. **"One server for the price of two: Simple and fast single-server private information retrieval"**. In: *USENIX Security Symposium*. USENIX, 2023.

[HKST22] K. HAMACHER, T. KUSSEL, T. SCHNEIDER, O. TKACHENKO. **"PEA: Practical private epistasis analysis using MPC"**. In: *European Symposium on Research in Computer Security (ESORICS)*. Springer, 2022.

[HLC+22] M. HAO, H. LI, H. CHEN, P. XING, G. XU, T. ZHANG. **"Iron: Private inference on transformers"**. In: *Advances in Neural Information Processing Systems (NeurIPS)*. Curran Associates, 2022.

[HLHD22] Z. HUANG, W.-j. LU, C. HONG, J. DING. **"Cheetah: Lean and fast secure two-party deep neural network inference"**. In: *USENIX Security Symposium*. USENIX, 2022.

[HMA+20] G. F. HATKE, M. MONTANARI, S. APPADWEDULA, M. WENTZ, J. MEKLENBURG, L. IVERS, J. WATSON, P. FIORE. **"Using Bluetooth Low Energy (BLE) signal strength estimation to facilitate contact tracing for COVID-19"**. https://arxiv.org/ftp/arxiv/papers/2006/2006.15711.pdf. 2020.

[HMM+21] K. HOGAN, B. MACEDO, V. MACHA, A. BARMAN, X. JIANG. **"Contact tracing apps: Lessons learned on privacy, autonomy, and the need for detailed and thoughtful implementation"**. In: *Medical Informatics*. JMIR Publications, 2021.

[HMSY21a] A. HEGDE, H. MÖLLERING, T. SCHNEIDER, H. YALAME. **"CONTRIBUTED TALK: SoK: Privacy-preserving clustering (Extended Abstract)"**. Privacy in Machine Learning Workshop (PriML@NeurIPS). 2021.

[HMSY21b] A. HEGDE, H. MÖLLERING, T. SCHNEIDER, H. YALAME. **"SoK: Efficient privacy-preserving clustering"**. In: *Proceedings on Privacy Enhancing Technologies (PoPETs)* 2021.4 (2021). Online: https://ia.cr/2021/809. Code: https://encrypto.de/code/SoK_ppClustering, S. 225–248. CORE Rank A. Appendix A.

[HR22] L. HELMINGER, C. RECHBERGER. **"Multi-party computation in the GDPR"**. In: *Privacy Symposium 2022: Data Protection Law International Convergence and Compliance with Innovative Technologies (DPLICIT)*. Springer, 2022.

[HRP20] V. HARALAMPIEVA, D. RUECKERT, J. PASSERAT-PALMBACH. **"A systematic comparison of encrypted machine learning solutions for image classification"**. In: *Workshop on Privacy-preserving Machine Learning in Practice (PPMPLP)*. ACM, 2020.

[HZNF15] J. HEURIX, P. ZIMMERMANN, T. NEUBAUER, S. FENZ. **"A taxonomy for privacy enhancing technologies"**. In: *Computers & Security*. Elsevier, 2015.

[ICÖ21] A. IBARRONDO, H. CHABANNE, M. ÖNEN. **"Practical privacy-preserving face identification based on function-hiding functional encryption"**. In: *Conference on Cryptology And Network Security (CANS)*. Springer, 2021.

[IF20] INRIA, FRAUNHOFER AISEC. *ROBust and privacy-presERving proximity Tracing protocol*. https://github.com/ROBERT-proximity-tracing/documents. 2020.

[JA18] A. JÄSCHKE, F. ARMKNECHT. **"Unsupervised machine learning on encrypted data"**. In: *International Conference on Selected Areas in Cryptography (SAC)*. Springer, 2018.

[JFK20] M. S. JERE, T. FARNAN, F. KOUSHANFAR. **"A taxonomy of attacks on federated learning"**. In: *Security & Privacy*. IEEE, 2020.

[JLL+19] K. JÄRVINEN, H. LEPPÄKOSKI, E.-S. LOHAN, P. RICHTER, T. SCHNEIDER, O. TKACHENKO, Z. YANG. **"PILOT: Practical privacy-preserving indoor localization using outsourcing"**. In: *European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2019.

[JVC18] C. JUVEKAR, V. VAIKUNTANATHAN, A. CHANDRAKASAN. **"GAZELLE: A low latency framework for secure neural network inference"**. In: *USENIX Security Symposium*. USENIX, 2018.

[KBSC18] T. KANSAL, S. BAHUGUNA, V. SINGH, T. CHOUDHURY. **"Customer segmentation using K-means clustering"**. In: *International Conference on Computational Techniques, Electronics and Mechanical Systems (CTEMS)*. IEEE. 2018.

[KC18] H.-J. KIM, J.-W. CHANG. **"A privacy-preserving K-means clustering algorithm using secure comparison protocol and density-based center point selection"**. In: *International Conference on Cloud Computing (CLOUD)*. IEEE, 2018.

[KDK11] K. KURSAWE, G. DANEZIS, M. KOHLWEISS. **"Privacy-friendly aggregation for the smart-grid"**. In: *Proceedings on Privacy Enhancing Technologies (PoPETs)*. Springer, 2011.

[Kel20] M. KELLER. **"MP-SPDZ: A versatile framework for multi-party computation"**. In: *Conference on Computer and Communications Security (CCS*. ACM, 2020.

[KMA+21] P. KAIROUZ, H. B. MCMAHAN, B. AVENT, A. BELLET, M. BENNIS, A. N. BHAGOJI, K. BONAWITZ, Z. CHARLES, G. CORMODE, R. CUMMINGS. **"Advances and open problems in federated learning"**. In: *Foundations and Trends® in Machine Learning*. Now Publishers, Inc., 2021.

[KMRB20] G. A. KAISSIS, M. R. MAKOWSKI, D. RÜCKERT, R. F. BRAREN. **"Secure, privacy-preserving and federated machine learning in medical imaging"**. In: *Nature Machine Intelligence*. Nature Publishing Group UK London, 2020.

[KMS22] K. KUŹNIEWSKI, K. MATUSIEWICZ, P. SAPIECHA. **"The high-level practical overview of open-source privacy-preserving machine learning solutions"**. In: *International Journal of Electronics and Telecommunications*. Polish Academy of Sciences Committee of Electronics und Telecommunications, 2022.

[KMSY21] H. KELLER, H. MÖLLERING, T. SCHNEIDER, H. YALAME. **"Balancing quality and efficiency in private clustering with affinity propagation"**. In: *International Conference on Security and Cryptography (SECRYPT)*. Full version: https://ia.cr/2021/825. Code: https://encrypto.de/code/ppAffinityPropagation. SciTePress, 2021, S. 173–184. CORE Rank B.

[KNL+19] Á. KISS, M. NADERPOUR, J. LIU, T. SCHNEIDER, N. ASOKAN. **"SoK: Modular and efficient private decision tree evaluation"**. In: *Proceedings on Privacy Enhancing Technologies (PoPETs)*. De Gruyter, 2019.

[KO97] E. KUSHILEVITZ, R. OSTROVSKY. **"Replication is not needed: Single database, computationally-private information retrieval"**. In: *Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 1997.

[KÖB21] F. KARAKOÇ, M. ÖNEN, Z. BILGIN. **"Secure aggregation against malicious users"**. In: *Symposium on Access Control Models and Technologies (SACMAT)*. ACM, 2021.

[Kol06] V. KOLESNIKOV. **"Secure two-party computation and communication"**. Diss. University of Toronto, 2006.

[KOR19] D. KALES, O. OMOLOLA, S. RAMACHER. **"Revisiting user privacy for certificate transparency"**. In: *European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2019.

[KR07] K. A. KUMAR, C. P. RANGAN. **"Privacy preserving DBSCAN algorithm for clustering"**. In: *International conference on advanced data mining and applications*. Springer, 2007.

[KR11] S. KAMARA, M. RAYKOVA. **"Secure outsourced computation in a multi-tenant cloud"**. In: *IBM Workshop on Cryptography and Security in Clouds*. 2011.

[KS08] V. KOLESNIKOV, T. SCHNEIDER. **"Improved garbled circuit: Free XOR gates and applications"**. In: *Automata, Languages and Programming: 35th International Colloquium (ICALP)*. Springer, 2008.

[KS14] M. KELLER, P. SCHOLL. **"Efficient, oblivious data structures for MPC"**. In: *International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT)*. Springer, 2014.

[KT21] K. KIM, H. C. TANUWIDJAJA. **"Privacy-preserving deep learning: A comprehensive survey"**. Springer, 2021.

[KTS+17] I. KAVAKIOTIS, O. TSAVE, A. SALIFOGLOU, N. MAGLAVERAS, I. VLAHAVAS, I. CHOUVARDA. **"Machine learning and data mining methods in diabetes research"**. In: *Computational and Structural Biotechnology Journal*. Elsevier, 2017.

[KV72] A. KERSHENBAUM, R. VAN SLYKE. **"Computing minimum spanning trees efficiently"**. In: *ACM Annual Conference*. ACM, 1972.

[KVH+21] B. KNOTT, S. VENKATARAMAN, A. HANNUN, S. SENGUPTA, M. IBRAHIM, L. v. d. MAATEN. **"CrypTen: Secure multi-party computation meets machine learning"**. In: *Advances in Neural Information Processing Systems (NeurIPS)*. Curran Associates, 2021.

[Lau22] K. LAUTER. "Private AI: Machine learning on encrypted data". In: *Recent Advances in Industrial and Applied Mathematics*. Springer, 2022.

[LCL+18] S.-I. LEE, S. CELIK, B. A. LOGSDON, S. M. LUNDBERG, T. J. MARTINS, V. G. OEHLER, E. H. ESTEY, C. P. MILLER, S. CHIEN, J. DAI. **"A machine learning approach to integrate big data for precision medicine in acute myeloid leukemia"**. In: *Nature Communications*. Nature Publishing Group UK London, 2018.

[LD10] A. H. LAND, A. G. DOIG. **"An automatic method for solving discrete programming problems"**. Springer, 2010.

[LDS+21] B. LIU, M. DING, S. SHAHAM, W. RAHAYU, F. FAROKHI, Z. LIN. **"When machine learning meets privacy: A survey and outlook"**. In: *Computing Surveys (CSUR)*. ACM, 2021.

[LGV+21] W. LUEKS, S. GÜRSES, M. VEALE, E. BUGNION, M. SALATHÉ, K. G. PATERSON, C. TRONCOSO. **"CrowdNotifier: Decentralized privacy-preserving presence tracing"**. In: *Proceedings on Privacy Enhancing Technologies (PoPETs)*. De Gruyter, 2021.

[LGY+22] Z. LIU, J. GUO, W. YANG, J. FAN, K.-Y. LAM, J. ZHAO. **"Privacy-preserving aggregation in federated learning: A survey"**. In: *Transactions on Big Data*. IEEE, 2022.

[LHLX12] J. LIU, J. Z. HUANG, J. LUO, L. XIONG. **"Privacy preserving distributed DBSCAN clustering"**. In: *Joint EDBT/ICDT Workshops*. 2012.

[Lin17] Y. LINDELL. **"How to simulate it– A tutorial on the simulation proof technique"**. In: *Tutorials on the Foundations of Cryptography*. Springer, 2017.

[Lin20] Y. LINDELL. **"Secure multiparty computation (MPC)"**. https://eprint.iacr.org/2020/300.pdf. 2020.

[LMSP21] R. LEHMKUHL, P. MISHRA, A. SRINIVASAN, R. A. POPA. **"Muse: Secure inference resilient to malicious clients"**. In: *USENIX Security Symposium*. USENIX, 2021.

[LYW+20] M. LI, Y. YAN, Q. WANG, M. DU, Z. QIN, C. WANG. **"Secure prediction of neural network in the cloud"**. In: *Network*. IEEE, 2020.

[LYZY20] L. LYU, H. YU, J. ZHAO, Q. YANG. **"Threats to federated learning"**. In: *Federated Learning: Privacy and Incentive*. Springer, 2020.

[MAA+16] F. MCKEEN, I. ALEXANDROVICH, I. ANATI, D. CASPI, S. JOHNSON, R. LESLIE-HURD, C. ROZAS. **"Intel® Software Guard Extensions (Intel® SGX) support for dynamic memory management inside an enclave"**. In: *Hardware and Architectural Support for Security and Privacy (HASP)*. 2016.

[MAB+13] F. MCKEEN, I. ALEXANDROVICH, A. BERENZON, C. V. ROZAS, H. SHAFI, V. SHANBHOGUE, U. R. SAVAGAONKAR. **"Innovative instructions and software model for isolated execution."** In: *Hardware and Architectural Support for Security and Privacy (HASP)*. 2013.

[MAZW22] L. MARTINICO, A. ABADI, T. ZACHARIAS, T. WIN. **"Glass-Vault: A generic transparent privacy-preserving exposure notification analytics platform"**. https://eprint.iacr.org/2022/1084.pdf. 2022.

[MBÖE20] M. MANSOURI, B. BOZDEMIR, M. ÖNEN, O. ERMIS. **"PAC: Privacy-preserving arrhythmia classification with neural networks"**. In: *Foundations and Practice of Security (FPS)*. Springer, 2020.

[MHJ+08] J. MOSSONG, N. HENS, M. JIT, P. BEUTELS, K. AURANEN, R. MIKOLAJCZYK, M. MASSARI, S. SALMASO, G. S. TOMBA, J. WALLINGA. **"Social contacts and mixing patterns relevant to the spread of infectious diseases"**. In: *PLoS Medicine*. Public Library of Science, 2008.

[MKA+17] A. J. MILLER, B. A. KIBERD, I. P. ALWAYN, A. ODUTAYO, K. K. TENNANKORE. **"Donor-recipient weight and sex mismatch and the risk of graft loss in renal transplantation"**. In: *Clinical Journal of the American Society of Nephrology (CJASN)*. American Society of Nephrology, 2017.

[MKA04] N. K. MALHOTRA, S. S. KIM, J. AGARWAL. **"Internet users' information privacy concerns (IUIPC): The construct, the scale, and a causal model"**. In: *Information systems research*. Informs, 2004.

[MMA+20] E. S. MCBRYDE, M. T. MEEHAN, O. A. ADEGBOYE, A. I. ADEKUNLE, J. M. CALDWELL, A. PAK, D. P. ROJAS, B. M. WILLIAMS, J. M. TRAUER. **"Role of modelling in COVID-19 policy development"**. In: *Paediatric Respiratory Reviews*. Elsevier, 2020.

[MMR+17] B. MCMAHAN, E. MOORE, D. RAMAGE, S. HAMPSON, B. A. y. ARCAS. **"Communication-efficient learning of deep networks from decentralized data"**. In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*. PMLR, 2017.

[MMW+10] W. C. MOORE, D. A. MEYERS, S. E. WENZEL, W. G. TEAGUE, H. LI, X. LI, R. D'AGOSTINO JR, M. CASTRO, D. CURRAN-EVERETT, A. M. FITZPATRICK. **"Identification of asthma phenotypes using cluster analysis in the severe asthma research program"**. In: *American Journal of Respiratory and Critical Care Medicine*. American Thoracic Society, 2010.

[MOJC23] M. MANSOURI, M. ONEN, W. B. JABALLAH, M. CONTI. **"SoK: Secure aggregation based on cryptographic schemes for federated learning"**. In: *Proceedings on Privacy Enhancing Technologies (PoPETs)*. De Gruyter, 2023.

[MP19] K.-K. MAK, M. R. PICHIKA. **"Artificial intelligence in drug development: present status and future prospects"**. In: *Drug Discovery Today*. Elsevier, 2019.

[MPOT21] X. MENG, D. PAPADOPOULOS, A. OPREA, N. TRIANDOPOULOS. **"Private hierarchical clustering and efficient approximation"**. In: *Cloud Computing Security Workshop (CCSW)*. ACM, 2021.

[MRRL23] A. MUÑOZ, R. RIOS, R. ROMÁN, J. LÓPEZ. **"A survey on the (in) security of trusted execution environments"**. In: *Computers & Security*. Elsevier, 2023.

[MRT20] P. MOHASSEL, M. ROSULEK, N. TRIEU. **"Practical privacy-preserving K-means clustering"**. In: *Proceedings on Privacy Enhancing Technologies (PoPETs)*. De Gruyter, 2020.

[MSDS19] L. MELIS, C. SONG, E. DE CRISTOFARO, V. SHMATIKOV. **"Exploiting unintended feature leakage in collaborative learning"**. In: *Symposium on Security and Privacy (SP)*. IEEE, 2019.

[MSST07] N. MISHRA, R. SCHREIBER, I. STANTON, R. E. TARJAN. **"Clustering social networks"**. In: *International Workshop on Algorithms and Models for the Web-Graph (WAW)*. Springer. 2007.

[MTV+20] F. MIRESHGHALLAH, M. TARAM, P. VEPAKOMMA, A. SINGH, R. RASKAR, H. ESMAEILZADEH. **"Privacy in deep learning: A survey"**. https://arxiv.org/pdf/2004.12254.pdf. 2020.

[MW22] S. J. MENON, D. J. WU. **"Spiral: Fast, high-rate single-server PIR via FHE composition"**. In: *Symposium on Security and Privacy (SP)*. IEEE, 2022.

[MWCB22] Z. Á. MANN, C. WEINERT, D. CHABAL, J. W. BOS. **"Towards practical secure neural network inference: The journey so far and the road ahead"**. https://eprint.iacr.org/2022/1483.pdf. 2022.

[MZRA22] Y. MA, K. ZHONG, T. RABIN, S. ANGEL. **"Incremental offline/online PIR"**. In: *USENIX Security Symposium*. USENIX, 2022.

[NA20] H. S. NOGAY, H. ADELI. **"Machine learning (ML) for the diagnosis of autism spectrum disorder (ASD) using brain imaging"**. In: *Reviews in the Neurosciences*. De Gruyter, 2020.

[NC23] L. K. NG, S. S. CHOW. **"SoK: Cryptographic neural-network computation"**. In: *Symposium on Security and Privacy (SP)*. IEEE, 2023.

[NMD+22] T. D. NGUYEN, M. MIETTINEN, A. DMITRIENKO, A.-R. SADEGHI, I. VISCONTI. **"Digital contact tracing solutions: Promises, pitfalls and challenges"**. In: *Transactions on Emerging Topics in Computing*. IEEE, 2022.

[NRC+22] T. D. NGUYEN, P. RIEGER, H. CHEN, H. YALAME, H. MÖLLERING, H. FEREIDOONI, S. MARCHAL, M. MIETTINEN, A. MIRHOSEINI, S. ZEITOUNI, F. KOUSHANFAR, A.-R. SADEGHI, T. SCHNEIDER. **"FLAME: Taming backdoors in federated learning"**. In: *USENIX Security Symposium (USENIX Security)*. Online: https://ia.cr/2021/025. USENIX, 2022, S. 1415–1432. CORE Rank A*. Appendix C.

[NSH19] M. NASR, R. SHOKRI, A. HOUMANSADR. **"Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning"**. In: *Symposium on Security and Privacy (SP)*. IEEE, 2019.

[OEC23] OECD. **"Emerging privacy-enhancing technologies"**. In: https://www.oecd-ilibrary.org/content/paper/bf121be4-en. Accessed: 06/27/23. 2023.

[Ope97] G. OPELZ. **"Impact of HLA compatibility on survival of kidney transplants from unrelated live donors"**. In: *Transplantation*. LWW, 1997.

[PAD+23] A. POLYCHRONIADOU, G. ASHAROV, B. DIAMOND, T. BALCH, H. BUEHLER, R. HUA, S. GU, G. GIMLER, M. VELOSO. **"Prime Match: A privacy-preserving inventory matching system"**. In: *USENIX Security Symposium*. USENIX, 2023.

[PCCS18] L. PANSART, H. CAMBAZARD, N. CATUSSE, G. STAUFFER. **"Kidney exchange problem: Models and algorithms"**. In: *ROADEF 2018-19ème congrès annuel de la société Française de Recherche Opérationnelle et d'Aide à la Décision*. Société française de Recherche Opérationnelle et d'Aide à la Décision, 2018.

[PJ89] T. N. PAPPAS, N. S. JAYANT. **"An adaptive clustering algorithm for image segmentation"**. In: *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* IEEE, 1989.

[PMSW18] N. PAPERNOT, P. MCDANIEL, A. SINHA, M. P. WELLMAN. **"Sok: Security and privacy in machine learning"**. In: *European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2018.

[PN23] O. PROCUREMENT, T. NETWORK. **"National data"**. In: https://optn.transplant.hrsa.gov/data/view-data-reports/national-data/. Accessed: 07/16/23. 2023.

[PR21] B. PINKAS, E. RONEN. **"Hashomer–Privacy-preserving Bluetooth based contact tracing scheme for Hamagen"**. In: *Real World Crypto (RWC), Workshop on Secure IT Technologies against COVID-19 (CoronaDef)*. 2021.

[PS19] S. PINTO, N. SANTOS. **"Demystifying ARM TrustZone: A comprehensive survey"**. In: *Computing Surveys (CSUR)*. ACM, 2019.

[PSSY21] A. PATRA, T. SCHNEIDER, A. SURESH, H. YALAME. **"ABY2.0: Improved mixed-protocol secure two-party computation"**. In: *USENIX Security Symposium*. USENIX, 2021.

[PT22] P. PANZADE, D. TAKABI. **"SoK: Privacy preserving machine learning using functional encryption: Opportunities and challenges"**. https://arxiv.org/pdf/2204.05136.pdf. 2022.

[PTC+21] B. PULIDO-GAYTAN, A. TCHERNYKH, J. M. CORTÉS-MENDOZA, M. BABENKO, G. RADCHENKO, A. AVETISYAN, A. Y. DROZDOV. **"Privacy-preserving neural networks with homomorphic encryption: Challenges and opportunities"**. In: *Peer-to-Peer Networking and Applications*. Springer, 2021.

[PTH21] R. PODSCHWADT, D. TAKABI, P. HU. **"SoK: Privacy-preserving deep learning with homomorphic encryption"**. https://arxiv.org/pdf/2112.12855.pdf. 2021.

[QWLQ19] P. QIU, D. WANG, Y. LYU, G. QU. **"VoltJockey: Breaching TrustZone by software-controlled voltage manipulation over multi-core frequencies"**. In: *Conference on Computer and Communications Security (CCS)*. ACM, 2019.

[RBS21a] L. REICHERT, S. BRACK, B. SCHEUERMANN. **"Lighthouses: A warning system for super-spreader events"**. In: *International Conference on Communications Workshops (ICC Workshops)*. IEEE, 2021.

[RBS21b] L. REICHERT, S. BRACK, B. SCHEUERMANN. **"Poster: Privacy-preserving contact tracing of COVID-19 patients"**. In: *Symposium on Security and Privacy (SP)*. IEEE, 2021.

[REE08] J. M. READ, K. T. EAMES, W. J. EDMUNDS. **"Dynamic social networks and the implications for the spread of infectious disease"**. In: *Journal of the Royal Society Interface*. The Royal Society London, 2008.

[Rey22] J. REYES. **"Building the next generation of digital advertising with MPC"**. In: *Real World Crypto Symposium (RWC)*. https://iacr.org/submit/files/slides/2022/rwc/rwc2022/104/slides.pdf. Accessed: 07/03/23. 2022.

[RF07] B. ROSENFELD, R. FELDMAN. **"Clustering for unsupervised relation identification"**. In: *Conference on Information and Knowledge Management (CIKM)*. ACM, 2007.

[RHW19] Y. ROH, G. HEO, S. E. WHANG. **"A survey on data collection for machine learning: a big data-AI integration perspective"**. In: *Transactions on Knowledge and Data Engineering*. IEEE, 2019.

[Ric21] N. RICHARDS. **"Why privacy matters"**. Oxford University Press, 2021.

[RRK19] M. S. RIAZI, B. D. ROUANI, F. KOUSHANFAR. **"Deep learning on private data"**. In: *Security & Privacy*. IEEE, 2019.

[RSB+15] F.-Y. RAO, B. K. SAMANTHULA, E. BERTINO, X. YI, D. LIU. **"Privacy-preserving and outsourced multi-user K-means clustering"**. In: *Conference on Collaboration and Internet Computing (CIC)*. IEEE, 2015.

[RSÜ04] A. E. ROTH, T. SÖNMEZ, M. U. ÜNVER. **"Kidney exchange"**. In: *The Quarterly Journal of Economics*. MIT Press, 2004.

[RWT+18] M. S. RIAZI, C. WEINERT, O. TKACHENKO, E. M. SONGHORI, T. SCHNEIDER, F. KOUSHANFAR. **"Chameleon: A hybrid secure computation framework for machine learning applications"**. In: *ASIA Conference on Computer and Communications Security (ASIACCS)*. ACM, 2018.

[SAA19] W. Z. SRINIVASAN, P. AKSHAYARAM, P. R. ADA. **"DELPHI: A cryptographic inference service for neural networks"**. In: *USENIX Security Symposium*. USENIX, 2019.

[Sab12] A. S. SABAU. **"Survey of clustering based financial fraud detection research"**. In: *Informatica Economica*. INFOREC Association, 2012.

[SAB15] M. SABT, M. ACHEMLAL, A. BOUABDALLAH. **"Trusted execution environment: what it is, and what it is not"**. In: *Trustcom/BigDataSE/Ispa*. IEEE, 2015.

[SAD+16] R. E. SHERMAN, S. A. ANDERSON, G. J. DAL PAN, G. W. GRAY, T. GROSS, N. L. HUNTER, L. LAVANGE, D. MARINAC-DABIC, P. W. MARKS, M. A. ROBB. **"Real-world evidence—What is it and what can it tell us"**. In: *N Engl J Med*. 2016.

[ŠBC+21] T. ŠUŠTERŠIČ, A. BLAGOJEVIĆ, D. CVETKOVIĆ, A. CVETKOVIĆ, I. LORENCIN, S. B. ŠEGOTA, D. MILOVANOVIĆ, D. BASKIĆ, Z. CAR, N. FILIPOVIĆ. **"Epidemiological predictive modeling of COVID-19 infection: Development, testing, and implementation on the population of the Benelux union"**. In: *Frontiers in Public Health*. 2021.

[SH20] H. STEVENS, M. B. HAINES. **"TraceTogether: Pandemic response, democracy, and technology"**. https://www.tracetogether.gov.sg. Accessed: 07/23/23. 2020.

[SHA+19] A. SANGERS, M. v. HEESCH, T. ATTEMA, T. VEUGEN, M. WIGGERMAN, J. VELDSINK, O. BLOEMEN, D. WORM. **"Secure multiparty PageRank algorithm for collaborative fraud detection"**. In: *Financial Cryptography and Data Security (FC)*. Springer, 2019.

[Sha79] A. SHAMIR. **"How to share a secret"**. In: *Communications of the ACM*. ACM, 1979.

[Shi20] E. SHI. **"Path oblivious heap: Optimal and practical oblivious priority queue"**. In: *Symposium on Security and Privacy (SP)*. IEEE, 2020.

[SK21] S. SAGAR, C. KEKE. **"Confidential machine learning on untrusted platforms: A survey"**. In: *Cybersecurity*. SpringerOpen, 2021.

[SKK+23] P. SHRIRAM, N. KOTI, V. B. KUKKALA, A. PATRA, B. R. GOPAL. **"Find thy neighbourhood: Privacy-preserving local clustering"**. In: *Proceedings on Privacy Enhancing Technologies (PoPETs)*. De Gruyter, 2023.

[SKKR02] B. M. SARWAR, G. KARYPIS, J. KONSTAN, J. RIEDL. **"Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering"**. In: *International Conference on Computer and Information Technology (ICCIT)*. 2002.

[SKKT22] E. U. SOYKAN, L. KARAÇAY, F. KARAKOÇ, E. TOMUR. **"A survey and guideline on privacy enhancing technologies for collaborative machine learning"**. In: *Access*. IEEE, 2022.

[SKL+10] M. SALATHÉ, M. KAZANDJIEVA, J. W. LEE, P. LEVIS, M. W. FELDMAN, J. H. JONES. **"A high-resolution human contact network for infectious disease transmission"**. In: *Proceedings of the National Academy of Sciences*. National Acad Sciences, 2010.

[SMS+22] M. SCHNEIDER, R. J. MASTI, S. SHINDE, S. CAPKUN, R. PEREZ. **"SoK: Hardware-supported trusted execution environments"**. https://arxiv.org/pdf/2205.12742.pdf. 2022.

[SS08] A.-R. SADEGHI, T. SCHNEIDER. **"Generalized universal circuits for secure evaluation of private functions with application to data classification"**. In: *International Conference on Information Security and Cryptology (ICISC)*. Springer, 2008.

[SSW09] A.-R. SADEGHI, T. SCHNEIDER, I. WEHRENBERG. **"Efficient privacy-preserving face recognition"**. In: *International Conference on Information Security and Cryptology (ICISC)*. Springer, 2009.

[ST19] T. SCHNEIDER, O. TKACHENKO. **"EPISODE: Efficient privacy-preserving similar sequence queries on outsourced genomic databases"**. In: *ASIA Conference on Computer and Communications Security (ASIACCS)*. ACM, 2019.

[Ste56] H. STEINHAUS. **"Sur la division des corp materiels en parties"**. In: *Bulletin L'Académie Polonaise des Science*. 1956.

[SV00] P. SAMARATI, S. C. d. VIMERCATI. **"Access control: Policies, models, and mechanisms"**. In: *Foundations of Security Analysis and Design (FOSAD)*. Springer, 2000.

[Swe02] L. SWEENEY. **"K-anonymity: A model for protecting privacy"**. In: *International Journal of Uncertainty, Fuzziness and Knowledge-based Systems*. World Scientific, 2002.

[SWRH23] L. SONG, H. WU, W. RUAN, W. HAN. **"SoK: Training machine learning models over multiple sources with privacy preservation"**. https://arxiv.org/pdf/2012.03386.pdf. 2023.

[SWX+21] R. SUN, W. WANG, M. XUE, G. TYSON, S. CAMTEPE, D. C. RANASINGHE. **"An empirical assessment of global COVID-19 contact tracing applications"**. In: *International Conference on Software Engineering (ICSE)*. IEEE, 2021.

[SZ13] T. SCHNEIDER, M. ZOHNER. **"GMW vs. Yao? Efficient secure two-party computation with low depth circuits"**. In: *Conference on Financial Cryptography and Data Security (FC)*. Springer, 2013.

[TCBK20] H. C. TANUWIDJAJA, R. CHOI, S. BAEK, K. KIM. **"Privacy-preserving deep learning on machine learning as a service—a comprehensive survey"**. In: *Access*. IEEE, 2020.

[TJS19] S. TOPLE, Y. JIA, P. SAXENA. **"PRO-ORAM: Practical read-only oblivious RAM"**. In: *Symposium on Research in Attacks, Intrusions and Defenses (RAID)*. ACM, 2019.

[TKRM22] Y. T. TAMER, A. KARAM, T. RODERICK, S. MIFF. **"Know thy patient: A novel approach and method for patient segmentation and clustering using machine learning to develop holistic, patient-centered programs and treatment plans"**. In: *NEJM Catalyst Innovations in Care Delivery*. Massachusetts Medical Society, 2022.

[TKTW21] S. TAN, B. KNOTT, Y. TIAN, D. J. WU. **"CryptGPU: Fast privacy-preserving machine learning on the GPU"**. In: *Symposium on Security and Privacy (SP)*. IEEE. 2021.

[TMSS22] A. TREIBER, D. MÜLLMANN, T. SCHNEIDER, I. SPIECKER GENANNT DÖHMANN. **"Data protection law and multi-party computation: Applications to information exchange between law enforcement agencies"**. In: *Workshop on Privacy in the Electronic Society (WPES)*. ACM, 2022.

[TMW+20] A. TREIBER, A. MOLINA, C. WEINERT, T. SCHNEIDER, K. KERSTING. **"CryptoSPN: Privacy-preserving sum-product network inference"**. In: *European Conference on Artificial Intelligence (ECAI)*. IOS Press, 2020.

[TNK+19] A. TREIBER, A. NAUTSCH, J. KOLBERG, T. SCHNEIDER, C. BUSCH. **"Privacy-preserving PLDA speaker verification using outsourced secure computation"**. In: *Speech Communication*. Elsevier, 2019.

[Tof09] T. TOFT. **"Solving linear programs using multiparty computation"**. In: *International Conference on Financial Cryptography and Data Security (FC)*. Springer, 2009.

[TÖHL22] R. TAIELLO, M. ÖNEN, O. HUMBERT, M. LORENZI. **"Privacy preserving image registration"**. In: *Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. Springer, 2022.

[TPH+20] C. Troncoso, M. Payer, J. Hubaux, M. Salathé, J. R. Larus, W. Lueks, T. Stadler, A. Pyrgelis, D. Antonioli, L. Barman, S. Chatel, K. G. Paterson, S. Capkun, D. A. Basin, J. Beutel, D. Jackson, M. Roeschlin, P. Leu, B. Preneel, N. P. Smart, A. Abidin, S. Gurses, M. Veale, C. Cremers, M. Backes, N. O. Tippenhauer, R. Binns, C. Cattuto, A. Barrat, D. Fiore, M. Barbosa, R. Oliveira, J. Pereira. **"Decentralized privacy-preserving proximity tracing"**. In: *Data Eng. Bull.* IEEE, 2020.

[TPH16] W. Tang, D. Pi, Y. He. **"A density-based clustering algorithm with sampling for travel behavior analysis"**. In: *Intelligent Data Engineering and Automated Learning (IDEAL)*. Springer. 2016.

[TPK15] L. Trevithick, J. Painter, P. Keown. **"Mental health clustering and diagnosis in psychiatric in-patients"**. In: *BJPsych Bulletin*. Cambridge University Press, 2015.

[Tre18] S. Trewin. **"AI fairness for people with disabilities: Point of view"**. https://arxiv.org/pdf/1811.10670.pdf. 2018.

[TS18] O. Tkachenko, T. Schneider. **"Towards efficient privacy-preserving similar sequence queries on outsourced genomic databases"**. In: *Workshop on Privacy in the Electronic Society (WPES)*. ACM, 2018.

[TSS+20] N. Trieu, K. Shehata, P. Saxena, R. Shokri, D. Song. **"Epione: Lightweight contact tracing with strong privacy"**. In: *Computer Society Technical Committee on Data Engineering*. IEEE, 2020.

[TTGL20] V. Tolpegin, S. Truex, M. E. Gursoy, L. Liu. **"Data poisoning attacks against federated learning systems"**. In: *European Symposium on Research in Computer Security (ESORICS)*. Springer, 2020.

[TWSH18] O. Tkachenko, C. Weinert, T. Schneider, K. Hamacher. **"Large-scale privacy-preserving statistical computations for distributed genome-wide association studies"**. In: *ASIA Conference on Computer and Communications Security (ASIACCS)*. ACM, 2018.

[Ult05] A. Ultsch. **"Clustering wih som: U* c"**. In: *Workshop on Self-Organizing Maps*. 2005.

[Vau20] S. Vaudenay. **"Centralized or decentralized? The contact tracing dilemma"**. https://eprint.iacr.org/2020/531.pdf. 2020.

[VBF+04] V. S. Verykios, E. Bertino, I. N. Fovino, L. P. Provenza, Y. Saygin, Y. Theodoridis. **"State-of-the-art in privacy preserving data mining"**. In: *Sigmod Record*. ACM, 2004.

[WCP+17] W. Wang, G. Chen, X. Pan, Y. Zhang, X. Wang, V. Bindschaedler, H. Tang, C. A. Gunter. **"Leaky cauldron on the dark land: Understanding memory side-channel hazards in SGX"**. In: *Conference on Computer and Communications Security (CCS)*. ACM, 2017.

[WSB+00] J. Waiser, M. Schreiber, K. Budde, L. Fritsche, T. Böhler, I. Hauser, H.-H. Neumayer. **"Age-matching in renal transplantation"**. In: *Nephrology Dialysis Transplantation*. Oxford University Press, 2000.

[WSR+20] H. Wang, K. Sreenivasan, S. Rajput, H. Vishwakarma, S. Agarwal, J.-y. Sohn, K. Lee, D. Papailiopoulos. **"Attack of the tails: Yes, you really can backdoor federated learning"**. In: *Advances in Neural Information Processing Systems (NeurIPS)*. Curran Associates, 2020.

[WTB+21] S. WAGH, S. TOPLE, F. BENHAMOUDA, E. KUSHILEVITZ, P. MITTAL, T. RABIN. **"Falcon: Honest-majority maliciously secure framework for private deep learning"**. In: *Proceedings on Privacy Enhancing Technologies (PoPETs)*. De Gruyter, 2021.

[WTC22] W. WEI, C. TANG, Y. CHEN. **"Efficient privacy-preserving K-means clustering from secret-sharing-based secure three-party computation"**. In: *Entropy*. MDPI, 2022.

[WZL+20] H. WEN, Q. ZHAO, Z. LIN, D. XUAN, N. SHROFF. **"A study of the privacy of COVID-19 contact tracing apps"**. In: *Security and Privacy in Communication Networks (SecureComm)*. Springer, 2020.

[XBJ21] R. XU, N. BARACALDO, J. JOSHI. **"Privacy-preserving machine learning: Methods, challenges and directions"**. https://arxiv.org/pdf/2108.04417.pdf. 2021.

[XHCL20] C. XIE, K. HUANG, P.-Y. CHEN, B. LI. **"DBA: Distributed backdoor attacks against federated learning"**. In: *International Conference on Learning Representations (ICLR)*. 2020.

[XHZ+23] G. XU, X. HAN, T. ZHANG, S. XU, J. NING, X. HUANG, H. LI, R. H. DENG. **"Simc 2.0: Improved secure ML inference against malicious clients"**. In: *IEEE Transactions on Dependable and Secure Computing*. IEEE, 2023.

[XJL19] R. XU, J. B. JOSHI, C. LI. **"CryptoNN: Training neural networks over encrypted data"**. In: *International Conference on Distributed Computing Systems (ICDCS)*. IEEE. 2019.

[XT15] D. XU, Y. TIAN. **"A comprehensive survey of clustering algorithms"**. In: *Annals of Data Science*. Springer, 2015.

[XW05] R. XU, D. WUNSCH. **"Survey of clustering algorithms"**. In: *Transactions on Neural Networks*. IEEE, 2005.

[XWRJ13] K. XIA, Y. WU, X. REN, Y. JIN. **"Research in clustering algorithm for diseases analysis"**. In: *Journal of Networks*. Academy Publisher, 2013.

[Yan20] R. YANG. **"Survey on privacy-preserving machine learning protocols"**. In: *Machine Learning for Cyber Security*. Springer, 2020.

[Yao82] A. C. YAO. **"Protocols for secure computations"**. In: *Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 1982.

[Yao86] A. C.-C. YAO. **"How to generate and exchange secrets"**. In: *Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 1986.

[YBDN11] L. YING-HUA, Y. BING-RU, C. DAN-YANG, M. NAN. **"State-of-the-art in distributed privacy preserving data mining"**. In: *International Conference on Communication Software and Networks (ICCSN)*. IEEE, 2011.

[YKX22] Y. YU, S. KHADIVI, J. XU. **"Can data diversity enhance learning generalization?"** In: *International Conference on Computational Linguistics (COLING)*. International Committee on Computational Linguistics, 2022.

[YZH21] X. YIN, Y. ZHU, J. HU. **"A comprehensive survey of privacy-preserving federated learning: A taxonomy, review, and future directions"**. In: *Computing Surveys (CSUR)*. ACM, 2021.

[ZAGK22] J. ZALONIS, F. ARMKNECHT, B. GROHMANN, M. KOCH. **"Report: State of the art solutions for privacy preserving machine learning in the medical context"**. https://arxiv.org/pdf/2201.11406.pdf. 2022.

[ZCWS18] D. ZHANG, X. CHEN, D. WANG, J. SHI. **"A survey on collaborative deep learning and privacy-preserving"**. In: *International Conference on Data Science in Cyberspace (DSC)*. IEEE, 2018.

[ZE13] S. ZAHUR, D. EVANS. **"Circuit structures for improving efficiency of security and privacy tools"**. In: *Symposium on Security and Privacy (SP)*. IEEE, 2013.

[Zha07] J. ZHAN. **"Privacy preserving K-medoids clustering"**. In: *International Conference on Systems, Man and Cybernetics (SMC)*. IEEE. 2007.

[ZHS+22] P. ZHANG, T. HUANG, X. SUN, W. ZHAO, H. LIU, S. LAI, J. K. LIU. **"Privacy-preserving and outsourced multi-party K-means clustering based on multi-key fully homomorphic encryption"**. In: *Transactions on Dependable and Secure Computing (TDSC)*. IEEE, 2022.

[ZJYC02] Z.-H. ZHOU, Y. JIANG, Y.-B. YANG, S.-F. CHEN. **"Lung cancer cell identification based on artificial neural network ensembles"**. In: *Artificial Intelligence in Medicine*. Elsevier, 2002.

[ZL09] P.-y. ZHANG, C.-h. LI. **"Automatic text summarization based on sentences clustering and extraction"**. In: *International Conference on Computer Science and Information Technology (ICCSIT)*. IEEE, 2009.

[ZPWV17] L. ZHOU, S. PAN, J. WANG, A. V. VASILAKOS. **"Machine learning on big data: Opportunities and challenges"**. In: *Neurocomputing*. Elsevier, 2017.

[ZRE15] S. ZAHUR, M. ROSULEK, D. EVANS. **"Two halves make a whole: Reducing data transfer in garbled circuits using half gates"**. In: *International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*. Springer, 2015.

[ZRL96] T. ZHANG, R. RAMAKRISHNAN, M. LIVNY. **"BIRCH: An efficient data clustering method for very large databases"**. In: *Sigmod Record*. ACM, 1996.

[ZRX+23] Y. ZHANG, D. RAMAGE, Z. XU, Y. ZHANG, S. ZHAI, P. KAIROUZ. **"Private federated learning in Gboard"**. https://arxiv.org/pdf/2306.14793.pdf. 2023.

[ZZCY20] J. ZHANG, J. ZHANG, J. CHEN, S. YU. **"Gan enhanced membership inference: A passive local attack in federated learning"**. In: *International Conference on Communications (ICC)*. IEEE, 2020.

List of Figures

1.1 Thesis Overview 4
1.2 Ideal Functionality of Secure Computation Protocols 5
1.3 Ideal Functionality for Private Information Retrieval (PIR) 8

2.1 Information Leakage Example in Privacy-Preserving Clustering 14
2.2 Density-based Spatial Clustering of Applications with Noise (DBSCAN) . . . 20
2.3 Effect of a Fixed Number of Iterations in ppDBSCAN 22

3.1 Kidney Exchange Cycle of Size 2 32
3.2 Overview of SPIKE's four phases 33
3.3 Overview of the RIPPLE Framework 37
3.4 Ideal Functionality of PIR−SUM 39

List of Tables

2.1 Comparison of Plaintext Clustering Algorithms 18
2.2 Related Work of ppDBSCAN 27

3.1 Related Work on the Kidney Exchange Problem (KEP) 44

List of Abbreviations

ML	Machine Learning
FL	Federated Learning
AI	Artificial Intelligence
KEP	Kidney Exchange Problem
SoK	Systematization of Knowledge
PETs	Privacy-Enhancing Technologies
HE	Homomorphic Encryption
SMPC	Secure Multi-Party Computation
2PC	Secure Two-Party Computation
3PC	Secure Three-Party Computation
ORAM	Oblivious RAM
FHE	Fully Homomorphic Encryption
GCs	Garbled Circuits
GMW	Goldreich-Micali-Wigderson
PPML	Privacy-Preserving Machine Learning
GDPR	General Data Protection Regulation
CCPA	California Consumer Privacy Act
HIPAA	Health Insurance Portability and Accountability Act
EHDS	European Health Data Space
TEE	Trusted Execution Environments
HLA	Human Leukocyte Antigens
SIMD	Single Instruction, Multiple Data
kNN	k-Nearest Neighbors
PIR	Private Information Retrieval
HDBSCAN	Hierarchical Density-Based Spatial Clustering of Applications with Noise
MST	Minimum Spanning Tree
RTT	Round-Trip Time
DBSCAN	Density-based Spatial Clustering of Applications with Noise
FSS	Function Secret Sharing
TNR	True Negative Rate
SPIKE	Secure and Private Investigation of the Kidney Exchange Problem
BGW	Ben-Or-Goldwasser-Wigderson
HC	Hierarchical Clustering

List of Own Publications

Peer-reviewed Publications

[AMMK21] S. Andreina, G. A. Marson, H. Möllering, G. Karame. **"BaFFLe: Backdoor detection via feedback-based federated learning"**. In: *International Conference on Distributed Computing Systems (ICDCS)*. Online: https://arxiv.org/pdf/2011.02167.pdf. IEEE. 2021, S. 852–863. CORE Rank A.

[BHK+22] T. Birka, K. Hamacher, T. Kussel, H. Möllering, T. Schneider. **"SPIKE: Secure and private investigation of the kidney exchange problem"**. In: *BMC Medical Informatics and Decision Making* 22.1 (2022). Online: https://arxiv.org/abs/2204.09937. Code: https://github.com/encryptogroup/ppke, S. 253. CORE Rank B. Appendix D.

[BCE+21] B. Bozdemir, S. Canard, O. Ermis, H. Möllering, M. Önen, T. Schneider. **"Privacy-preserving density-based clustering"**. In: *ASIA Conference on Computer and Communications Security (ASIACCS)*. Online: https://ia.cr/2021/612. Code: https://encrypto.de/code/ppDBSCAN. ACM, 2021, S. 658–671. CORE Rank A. Appendix B.

[FMM+21] H. Fereidooni, S. Marchal, M. Miettinen, A. Mirhoseini, H. Möllering, T. D. Nguyen, P. Rieger, A.-R. Sadeghi, T. Schneider, H. Yalame, S. Zeitouni. **"SAFELearn: Secure aggregation for private federated learning"**. In: *Deep Learning and Security Workshop (DLS@SP)*. Full version: https://ia.cr/2021/386. IEEE, 2021, S. 56–62.

[GHJ+22] D. Günther, M. Holz, B. Judkewitz, H. Möllering, B. Pinkas, T. Schneider, A. Suresh. **"POSTER: Privacy-preserving epidemiological modeling on mobile graphs"**. In: *Conference on Computer and Communications Security (CCS) Posters/Demos*. ACM, 2022, S. 3351–3353. CORE Rank A*.

[HMSY21a] A. Hegde, H. Möllering, T. Schneider, H. Yalame. **"CONTRIBUTED TALK: SoK: Privacy-preserving clustering (Extended Abstract)"**. Privacy in Machine Learning Workshop (PriML@NeurIPS). 2021.

[HMSY21b] A. Hegde, H. Möllering, T. Schneider, H. Yalame. **"SoK: Efficient privacy-preserving clustering"**. In: *Proceedings on Privacy Enhancing Technologies (PoPETs)* 2021.4 (2021). Online: https://ia.cr/2021/809. Code: https://encrypto.de/code/SoK_ppClustering, S. 225–248. CORE Rank A. Appendix A.

[KMSY21] H. KELLER, H. MÖLLERING, T. SCHNEIDER, H. YALAME. **"Balancing quality and efficiency in private clustering with affinity propagation"**. In: *International Conference on Security and Cryptography (SECRYPT)*. Full version: https://ia.cr/2021/825. Code: https://encrypto.de/code/ppAffinityPropagation. SciTePress, 2021, S. 173–184. CORE Rank B.

[NRC+22] T. D. NGUYEN, P. RIEGER, H. CHEN, H. YALAME, H. MÖLLERING, H. FEREIDOONI, S. MARCHAL, M. MIETTINEN, A. MIRHOSEINI, S. ZEITOUNI, F. KOUSHANFAR, A.-R. SADEGHI, T. SCHNEIDER. **"FLAME: Taming backdoors in federated learning"**. In: *USENIX Security Symposium (USENIX Security)*. Online: https://ia.cr/2021/025. USENIX, 2022, S. 1415–1432. CORE Rank A*. Appendix C.

In Submission

[BMP+23] Y. BEN-ITZHAK, H. MÖLLERING, B. PINKAS, T. SCHNEIDER, A. SURESH, O. TKACHENKO, S. VARGAFTIK, C. WEINERT, H. YALAME, A. YANAI. **"ScionFL: Efficient and robust secure quantized aggregation"**. https://eprint.iacr.org/2023/652.pdf. 2023.

[GHJ+23] D. GÜNTHER, M. HOLZ, B. JUDKEWITZ, H. MÖLLERING, B. PINKAS, T. SCHNEIDER, A. SURESH. **"Privacy-preserving epidemiological modeling on mobile graphs"**. https://ia.cr/2020/1546. Code: https://zenodo.org/record/6599225. 2023. Appendix E.

[KMT+23] H. KELLER, H. MÖLLERING, O. TKACHENKO, T. SCHNEIDER, L. ZHAO. **"Secure noise sampling for DP in MPC with finite precision"**. 2023.

[PZM+23] Q. PANG, J. ZHU, H. MÖLLERING, W. ZHENG, T. SCHNEIDER. **"BOLT: Privacy-preserving, accurate and efficient inference for transformers"**. 2023.

Helen Möllering

Personal Data

CITIZENSHIP: German
EMAIL: moellering@encrypto.cs.tu-darmstadt.de

Education

11/2019 - Today	PhD Studies Cryptography and Privacy Engineering Group (ENCRYPTO) **Technical University of Darmstadt**, Darmstadt, Germany Working Title: "Towards Practical Privacy-preserving Clustering and Health Care Data Analyses" Advisor: Prof. Dr.-Ing. Thomas Schneider
09/2017 - 09/2019	Master of Science in Security & Privacy (Double Degree), European Institute of Innovation and Technology's (EIT) Digital Master School **University of Twente**, Enschede, Netherlands **EURECOM**, Biot, France Thesis: "Thwarting Semantic Backdoor Attacks in Privacy-preserving Federated Learning" | Advisors: Prof. Dr. Melek Önen, Dr. Ghassan Karame, Dr. Giorgia Marson | Grade: 20.0 Final Grade: 8.75 (Twente, > 8: excellent) 19.5 (Eurecom, > 15: excellent)
04/2015 - Today	Bachelor of Science in Business Administration **University of Münster**, Münster, Germany Average Grade: 1.8 (< 2.1: very good)
07/2018	Summer School about Internet of Things for Industry 4.0 **Technische Universität München**, Munich, Germany
10/2013 - 11/2016	Bachelor of Science in Computer Science, **University of Münster**, Münster, Germany Thesis: "Comparing Methods of Application Development for Mobile Devices with an Example for iOS" | Advisor: Dr. Dietmar Lammers | Grade: 1.0 Final Grade: 1.5 (< 1.6: excellent)
08/2004 - 07/2013	Abitur, **Emsland-Gymnasium**, Rheine, Germany Focus: Computer Science & Mathematics Final Grade: 1.0 (< 1.6: excellent)

Work Experience

11/2019 - Today	Research Assistant, **Technical University of Darmstadt**, Darmstadt, Germany Cryptography and Privacy Engineering Group: • Design, prototyping, and benchmarking of efficient protocols for secure and privacy-preserving computation • Scientific service and acquisition of third-party funding • Teaching assistance for lectures, seminars, labs, and supervision of thesis students

01/2023 - 03/2023	Associate Intern, **McKinsey & Company**, Munich, Germany McKinsey Digital: • Interviews with senior female tech consultants and creation of a survey for the development of an internal strategy for enhancing diversity in tech talent • Development of a cost model and preparation of a board paper, as well as request for proposal documents for the transition of core infrastructure services for a major European airline
08/2022 - 11/2022	Visiting Scholar, **Carnegie Mellon University**, Pittsburgh, United States of America Cryptosystems Group (Prof. Dr. Wenting Zheng): • Design, prototyping, and benchmarking of efficient protocols for privacy-preserving natural language processing
03/2019 - 08/2019	Research Internship & Master Thesis, **NEC Laboratories Europe**, Heidelberg, Germany Security Group: • Design, development, and evaluation of defensive techniques against backdoor attacks on federated learning • Implementation of a federated learning process and semantic backdoor attacks on federated learning with several image data sets
07/2018 - 08/2018	Internship at **Volkswagen AG**, Hanover, Germany Team Information Security: • Conception of a security assessment of mobile applications • Security analysis of the used MDM solution and the penetration testing processes • Development of possibilities to improve and extend penetration testing and the security assessment of client software
09/2015 - 07/2017	Student Trainee at **Mintellity GmbH**, Münster, Germany iOS App Development: • Implementation of UI, API, and logic of a cash register system, a social network app, and a smart home application • Enhancement of knowledge about Objective-C, Cocoa Touch, and a range of iOS frameworks as well as third party frameworks • Solving general mobile development issues like limited resources, variable display sizes, and layout trends • Unit Testing for evaluating robustness, performance, and reliability • Customer meetings to define requirements and features of apps
11/2013 - 08/2015	Student Trainee at **LVM Versicherung**, Münster, Germany First Level Support: • Support for insurance agents and customers to resolve technical problems • Diagnosis of system malfunctions and defects • IBM Notes training for new colleagues

Project Affiliations

01/2021 - Today	**Engineering Private AI Systems (EPAI)** at the Private AI Collaborative Research Institute, funded by Intel, Avast, Borsetta, and VMware
04/2020 - 12/2022	**Practical Private Set Intersection for Data Protection** (as part of the mission "User-centered Security and Privacy (UCSP)") at the National Research Center for Applied Cybersecurity (ATHENE), funded by the German Federal Ministry of Education and Research (BMBF) and the Federal State of Hesse

11/2019 - Today	**Compiler for Privacy-Preserving Protocols** (E4) at the Collaborative Research Center (CRC) Cryptography-Based Security Solutions (CROSSING), funded by the German Research Foundation (DFG)

Teaching

Summer Term 23	**Teaching Assistant**, Cryptographic Protocols, Technical University of Darmstadt
	(Project) Lab, Development for Protecting Privacy (PrivDev), Technical University of Darmstadt
Summer Term 22	**Teaching Assistant**, Cryptographic Protocols, Technical University of Darmstadt
	(Project) Lab, Development for Protecting Privacy (PrivDev), Technical University of Darmstadt
Winter Term 21/22	**Seminar**, Privacy Preserving Technologies (PrivTech), Technical University of Darmstadt
Summer Term 21	**Teaching Assistant**, Cryptographic Protocols, Technical University of Darmstadt
	(Project) Lab, Development for Protecting Privacy (PrivDev), Technical University of Darmstadt
Winter Term 20/21	**Seminar**, Privacy Preserving Technologies (PrivTech), Technical University of Darmstadt
Summer Term 20	**Teaching Assistant**, Cryptographic Protocols, Technical University of Darmstadt
	Seminar, Privacy Preserving Technologies (PrivTech), Technical University of Darmstadt

Supervision

08/2023 - Today	**Klaus Hipp**, Master Thesis, *Attacks on Natural Language Processing*, co-supervised with Prof. Dr.-Ing. Thomas Schneider, Dr.-Ing. Christian Weinert, and M. Sc. Andreas Brüggemann
06/2023 - Today	**Thanh Long David Ngyuen**, Bachelor Thesis, *Information Extraction Attacks on Private Kidney Exchange (and Beyond)*, co-supervised with Prof. Dr.-Ing. Thomas Schneider and M. Sc. Andreas Brüggemann
06/2021 - 08/2022	**Liang Zhao**, Master Thesis, *Securely Realizing Output Privacy in MPC using Differential Privacy*, co-supervised with Prof. Dr.-Ing. Thomas Schneider and Dr.-Ing. Oleksandr Tkachenko
09/2021 - 07/2022	**Hannah Keller**, Master Thesis, *Understanding Privacy Leakage From Independently Computed Statistics*, co-supervised with Prof. Dr.-Ing. Thomas Schneider and Prof. Dr. Adam Smith
10/2021 - 03/2022	**Timm Birka**, Bachelor Thesis, *Efficient and Practical Privacy-Preserving Protocols for Organ Donations*, co-supervised with Prof. Dr.-Ing. Thomas Schneider and Dr. rer. nat. Tobias Kussel
05/2020 - 07/2020	**Aditya Hegde**, Research Internship, *Systematization of Knowledge: Efficient Privacy-preserving Clustering*, co-supervised with Prof. Dr.-Ing. Thomas Schneider and M. Sc. Hossein Yalame

Invited Talks/Presentations at Conferences

04/2023	Beyond contact tracing: How to effectively and privately model infectious diseases?, International Workshop on Cryptography, Robustness, and Provably Secure Schemes for Female* Young Researchers (CrossFyre@Eurocrypt '23), Lyon, France.
08/2022	Privacy-Preserving Epidemiological Modeling on Mobile Graphs, Massachusetts Institute of Technology (MIT) Media Lab, Boston, United States of America.

08/2021	A Systemization of Knowledge (SoK) of Efficient Privacy-preserving Clustering. Academic Techtalk, Private AI Collaborative Research Institute (Intel, Avast, Borsetta, and VMware), Virtual Event.
07/2021	SoK: Efficient privacy-preserving clustering. Privacy Enhancing Technologies Symposium (PETs), Virtual Event.
06/2021	Privacy-preserving density-based clustering. ACM ASIA Conference on Computer and Communications Security (ASIACCS '21), Virtual Event.

Patents

1. G. Karame, G. A. Marson, H. Möllering, Privacy-preserving machine learning, US Patent 11,470,053 B2, October 11, 2022.
2. G. Karame, G. A. Marson, H. Möllering, Thwarting model poisoning in federated learning. US Patent 11,616,804 B2, March 28, 2023.

Scientific Service

Reviewer	**ISIT '23** (International Symposium on Information Theory) **TIFS '22/'21'/20** (IEEE Transactions on Information Forensics & Security) **TDSC '21** (IEEE Transactions on Dependable and Secure Computing)
External Reviewer	**KDD '21** (ACM SIGKDD International Conference on Knowledge Discovery and Data Mining) **CRYPTO '23** (International Cryptology Conference) **INTERSPEECH Conference '23** **ACNS '22** (International Conference on Applied Cryptography and Network Security) **CCS '21** (ACM Conference on Computer and Communications Security) **NDSS '21** (Network and Distributed System Security Symposium) **EuroS&P '21** (IEEE European Symposium on Security and Privacy) **PPML@CCS '21** (Privacy Preserving Machine Learning Workshop) **ASIACRYPT '20** (International Conference on the Theory and Application of Cryptology and Information Security) **ICIP '20** (IEEE International Conference on Image Processing) **WPES '20** (Workshop on Privacy in the Electronic Society) **CCSW '20/'19** (ACM Cloud Computing Security Workshop) **IH&MMSec '20** (ACM Information Hiding and Multimedia Security Workshop) **CYSARM '19** (Workshop on Cyber-Security Arms Race)

Extracurricular activities, Scholarships, and Mentoring

04/2023 - Today	McKinsey & Company College
03/2023	Real World Crypto Symposium (RWC) Student Stipend
07/2021 - 06/2022	ProCareer.Doc Hessen Mentoring
2021 - Today	Deutschlandstipendium selection panel of the Computer Science Department at the Technical University of Darmstadt
2021	DISCOVER – Leadership Training through Arts (Excellence Program of the Association of Arts and Culture of the German Economy at the Federation of German Industries e. V.)
2020 - Today	Mentor in the Female Student Mentoring and Networking Program at the Collaborative Research Center (CRC) Cryptography-Based Security Solutions (CROSSING)
09/2017 - 09/2019	EIT Digital Master School Scholarship
06/2017 - 03/2023	Capstone McKinsey & Company
04/2015 - 11/2016	ProTalent – Deutschlandstipendium

Appendices

A SoK: Efficient Privacy-preserving Clustering (PoPETS'21)

[HMSY21] A. HEGDE, H. MÖLLERING, T. SCHNEIDER, H. YALAME. **"SoK: Efficient privacy-preserving clustering"**. In: *Proceedings on Privacy Enhancing Technologies (PoPETs)* 2021.4 (2021). Online: https://ia.cr/2021/809. Code: https://encrypto.de/code/SoK_ppClustering, S. 225–248. CORE Rank A. Appendix A.

https://doi.org/10.2478/popets-2021-0068

Aditya Hegde, Helen Möllering, Thomas Schneider, and Hossein Yalame

SoK: Efficient Privacy-preserving Clustering

Abstract: Clustering is a popular unsupervised machine learning technique that groups similar input elements into clusters. It is used in many areas ranging from business analysis to health care. In many of these applications, sensitive information is clustered that should not be leaked. Moreover, nowadays it is often required to combine data from multiple sources to increase the quality of the analysis as well as to outsource complex computation to powerful cloud servers. This calls for efficient privacy-preserving clustering. In this work, we systematically analyze the state-of-the-art in privacy-preserving clustering. We implement and benchmark today's four most efficient fully private clustering protocols by Cheon et al. (SAC'19), Meng et al. (ArXiv'19), Mohassel et al. (PETS'20), and Bozdemir et al. (ASIACCS'21) with respect to communication, computation, and clustering quality. We compare them, assess their limitations for a practical use in real-world applications, and conclude with open challenges.

Keywords: Privacy-preserving Protocols, Clustering, Secure Computation

DOI 10.2478/popets-2021-0068
Received 2021-02-28; revised 2021-06-15; accepted 2021-06-16.

1 Introduction

In today's world, machine learning (ML) algorithms are widely used to categorize and classify large amounts of data. Applications range from spam filtering over fraud detection, stock market analysis to health diagnostic [1–4]. Moreover, many large IT companies, including Microsoft, Facebook, Google, and Apple, collect massive amounts of data to perform analyses for their commercial benefit [5]. Clustering is a popular unsupervised learning technique and plays a crucial role in data processing and analysis. It divides a set of given input data into subgroups of elements with similar properties.

Cluster analysis is being utilized in various fields with extremely sensitive data such as medical imaging [4] and market research [6], to name a few. Moreover, data protection regulations such as the General Data Protection Regulation (GDPR) in the EU and the Health Insurance Portability and Accountability Act (HIPAA) in the US prohibit companies from sharing sensitive user information. Nevertheless, combining data from different sources, e.g., different hospitals, broadens the database and offers more meaningful, credible, and high-quality clustering results. Additionally, it is often needed to outsource the expensive clustering of large amounts of data to powerful cloud servers. These requirements emphasize the need for privacy-preserving clustering to preserve the privacy of data.

Consequently, a series of efforts have been made to protect the privacy of sensitive input data in clustering through two paradigms for secure computation that can also be combined. The first paradigm leverages homomorphic encryption (HE) [7–9]. HE allows to directly compute functions on encrypted data. The second paradigm uses secure multi-party computation (MPC) [10, 11]. MPC allows mutually distrusting parties to collaboratively compute a joint function over their respective private data. However, these works only cover a few clustering algorithms so far: K-means, K-medoid, Mean-shift, Gaussian Mixture Models Clustering (GMM), Density-Based Spatial Clustering of Applications with Noise (DBSCAN), hierarchical clustering (HC), Affinity Propagation, and Mean-shift. Moreover, we found that only ten works (cf. Tab. 1) provide full privacy protection according to the ideal functionality for privacy-preserving clustering, i.e., they leak nothing beyond the output (cf. §3.1).

Even revealing little and at the first glance minor information during the clustering can have severe consequences for the data privacy of individuals. For example, when using clustering for the segmentation of medical images [4] between two hospitals, revealing the cluster sizes and assignments in each clustering iteration leaks information about how many patients with similar characteristics are input by the other party even before the clustering stabilizes and a final result is reached. Unintended common characteristics between patients might

Aditya Hegde: IIIT-Bangalore, E-mail: aditya.shridhar@iiitb.org (This work was done when the author was intern at the Technical University of Darmstadt)
Helen Möllering, Thomas Schneider, Hossein Yalame: Technical University of Darmstadt, E-mail: lastname@encrypto.cs.tu-darmstadt.de

CC BY-NC-ND

Algorithm	Paper	PETs	Scenario	Data	Output	Efficiency
K-means	[12, CCS'07]	HE+ASS	2PC	a	final centroids	✗†
	[13, CIC'15]	HE	Outsourcing, 2 Servers	h	final centroids	✗†
	[14, SAC'18]	HE	Outsourcing, 1 Server	$-$	final centroids	✗⋆
	[15, CLOUD'18]	HE	Outsourcing, 2 Servers	$-$	cluster sizes	✗†
	MPC-KMeans [11, PETS'20]	GC	Outsourcing, 2 Servers *or* 2PC	h	final centroids	✓
Mean-shift	HE-Meanshift [9, SAC'19]	HE	Outsourcing, 1 Server	$-$	final centroids	✓
Affinity Propagation	[16, SECRYPT'21]	ASS	Outsourcing or MPC	a	final clusters	✗‡
DBSCAN	[17, S&P'13]	GC	2PC	h	cluster labels, centroids/size possible	✗¶
	ppDBSCAN [18, ASIACCS'21]	GC+ASS	Outsourcing, 2 Servers *or* 2PC	a	cluster labels, centroids/size possible	✓
Hierachical Clustering	PCA/OPT [19, ArXiv'19]	HE+GC	2PC	h	final dendogram	✓

† Computationally expensive due to use of Paillier's HE and no parallelization.
⋆ Costly computation due to use of bit-wise encryption. MPC-KMeans [11] outperforms this scheme by 5000× for 400 data records.
‡ [18] is 194× faster than this scheme for 400 data records.
¶ [18] is 5× faster than this scheme for a dataset size of 500 data records.

Table 1. Fully privacy-preserving clustering protocols (cf. §3.1). HE is homomorphic encryption [7], ASS is arithmetic secret sharing [20], and GC is garbled circuits [21]. v indicates vertically partitioned data, i.e., the data owners hold the values for a subset of parameters from all data records. h indicates horizontally partitioned data, where the data owners hold complete data records with all parameters, a is arbitrarily partitioned data, and " $-$ " indicates the scheme has only one data owner. Schemes that were implemented and benchmarked in §4 are highlighted in gray.

be leaked even though they are only temporarily assigned to one cluster (due to these characteristics which would not have been revealed in the final result). An even more severe privacy breach is demonstrated in [22] where leaking the results of comparison of distances between data records and a threshold can enable to accurately approximate the original data record held by another party. With this, complete patient records could be extracted when clustering medical data. To summarize, it is difficult to concretely determine the effects of leaking intermediate information in advance for all possible constellations. Hence, privacy research should focus on designing efficient private clustering protocols that do not leak anything beyond what can be inferred from the output, i.e., provide full privacy.

Related Work. Privacy-preserving machine learning (PPML) is a hot topic in recent privacy research [23–26]. To provide a better overview over the exploding research field, several surveys have been done. Haralampieva et al. [27] survey existing frameworks in the context of private image classification. An overview about frameworks for private neural network inference is given in [28]. Protocols used for private machine learning training are investigated in [29]. Similarily, Tanuwidjaja et al. [30] summarize existing works on privacy-preserving deep learning and issues when using these schemes as well as possible attacks on private deep learning. Kiss et al. [31] systematically review the state-of-the-art approaches to private decision tree evaluation.

All previous surveys focus on privacy-preserving supervised learning where a training dataset with labelled samples (i.e., known input-output pairs) is used to train a model that can later be used to classify new data records. In contrast, our survey focuses on clustering, a popular *unsupervised* machine learning (ML) technique, which detects unknown patterns in unlabelled data so no "training" of a model is needed. In our work, we systematically survey and evaluate the state-of-the-art in private clustering using secure computation techniques.

An orthogonal line of research uses differential privacy (DP) to protect privacy-preserving machine learning (PPML), including clustering [32–37], against information leakage. Abadi et al. [38] and Shokri et al. [39] provide comprehensive surveys on differentially private deep learning. Generally, the noise added to achieve DP reduces utility whereas secure computation has higher complexity. Hence, DP-based and secure computation-based protocols are not directly comparable and we leave a survey on DP-based clustering for future work.

Our Contributions and Outline. After presenting the preliminaries of privacy-preserving clustering in §2, our Systematization of Knowledge (SoK) paper provides the following core contributions:

– The first comprehensive review and analysis of existing techniques and protocols used for privacy-preserving clustering with respect to security models, privacy limitations, efficiency, and further aspects. We also provide guidelines on how to choose an appropriate privacy-preserving clustering scheme for a specific application (§3).

– An empirical evaluation of the four most efficient and fully private clustering schemes [9, 11, 18, 19], cf. Tab. 1, on a range of criteria, including clustering quality, security and privacy, and runtime/communication overhead (§4). Based on these insights, we provide an analysis of the practicality of the four protocols for real-world

applications based on our results from the benchmarking (§5).

– An implementation of the clustering protocol of [9] and [19] in C++17. Implementations of the remaining two protocols that we also evaluate [11, 18] are publicly available. Our code is available at https://encrypto.de/code/SoK_ppClustering.

2 Preliminaries

2.1 Clustering

Clustering is a well-known unsupervised machine learning (ML) technique, i.e., it deals with detecting unknown patterns in unlabeled data. Concretely, it groups similar input records (*internal homogeneity*) in *clusters* while records belonging to different clusters should be maximally different (*external separation*) [40–42].

Clustering consists of four components: feature selection/normalization, a proximity measure to determine similarity/dissimilarity, the clustering algorithm, and the output assessment [41, 42]. However, most prior works on privacy-preserving clustering mainly focus on a specific clustering algorithm. For example, the proximity measure is typically chosen to enable efficient computation using cryptographic techniques [14, 19]. Furthermore, mostly continuous values are considered while clustering can generally be applied to any kind of variable (i.e., also discrete or nominal values) [42].

Clustering algorithms can be split in two classes: hard and soft (fuzzy) clustering. In hard clustering, each input data record is assigned to exactly one cluster. In soft clustering, data records can be assigned to several clusters with a certain probability. All works on privacy-preserving clustering that we investigated in this work except from [43] have only tackled hard clustering.

Properties of Good Clustering. Records are assigned to the same cluster given they are *similar*. However, (dis)similarity heavily depends on the chosen proximity measure. Additionally, clustering algorithms were designed having specific problems in mind such that they exhibit biases that affect their performance when the assumed conditions are not fulfilled. Therefore, according to Xu and Wunsch [41], no clustering algorithm is universally superior and a good clustering algorithm should be able to cope with: 1) arbitrarily shaped clusters, 2) large datasets, 3) updates with new records without having to cluster old records again, 4) numerical (i.e., discrete and continuous) and nominal variables, and 5) outliers. Furthermore, it should: 6) be insensitive to the order of input records, 7) provide acceptable storage requirements, and 8) minimize the number of input parameters. Finally, it should also be able to handle 9) high-dimensional data records.

	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)
1) Cluster Shapes	−	−	○	+	+	−	−	−
2) Large Datasets	○	−	−	−	−	−	+	○
3) Update Input Data	+	−	−	○	+	+	+	+
4) Nominal Variables	−	+	+	−	+	+	−	−
5) Outliers	−	+	○	−	+	○	+	○
6) Input Order	+	+	+	+	○	+	−	+
7) Storage	+	−	−	+	+	−	+	+
8) # Parameters	−	○	−	○	○	−	○	−
Full privacy	✓	✓	✓	✓	✓	✗	✗	✗

Table 2. Comparison of clustering algorithms with respect to the aspects explained in §2.1: (a) K-means, (b) Affinity Propagation, (c) Single/Complete Linkage HC, (d) Mean-shift, (e) DBSCAN, (f) K-medoid, (g) BIRCH, and (h) GMM. + denotes that the clustering algorithm performs well with respect to the indicated aspect, ○ denotes an average performance, and − indicates that it has some weaknesses. ✓ indicates that a fully privacy-preserving clustering protocol is available and ✗ that it is not available yet.

Clustering Algorithms

In the context of privacy-preserving clustering, four different types of clustering have been studied so far: partitioning-based [8, 11, 14, 16], distribution-based [44, 45], density-based [18, 46, 47], and hierarchical clustering [19, 48–50]. In the following, we summarize these four clustering types and compare the respective algorithms w.r.t. the properties listed before in §2.1. Due to space limitations, we only provide the details of this evaluation for the three algorithms [51–53] for which fully private protocols were proposed (cf. §3.2) and that we benchmark in §4. Details of the other algorithms are given in Appx. A.

Partitioning-based Clustering. Partitioning-based clustering splits the input into K non-overlapping clusters. Typically, an initial random partition is iteratively improved given an objective function [54].

A well-known example is K-means [51]. It has a computational complexity of $\mathcal{O}(NKt)$ and a space complexity of $O(N)$ [40] for dataset size N, K clusters, and t clustering iterations. Furthermore, K-means can only cluster convexly-shaped clusters, cannot to appropriately handle outliers, and requires to pre-determine the number of clusters K [40]. If the initial partitioning,

i.e., the centroid initialization, is done at random, K-means is not deterministic. It may converge to a local optimum [55]. The input order does not affect the clustering result. As the centroids are determined by averaging, K-means is not suitable for nominal variables [56]. New data records typically require only a few additional clustering iterations because they normally do not significantly change the result. Other partitioning-based clustering algorithms that were investigated in the context of privacy-preserving clustering are the closely related K-medoids [57], Kernel K-means [58], Possibilistic C-means [43], as well as Affinity Propagation [16].

Hierarchical Clustering. Hierarchical Clustering (HC) algorithms can be classified into agglomerative and divisive approaches. In agglomerative algorithms, each data record forms an own cluster in the beginning and the clusters are then iteratively merged together based on their proximity. Divisive algorithms follow the opposite approach and start with all elements in one cluster which is then iteratively split up [52, p. 71-72]. HC algorithms output a binary tree/dendrogram[1] where each leaf represents a record and nodes indicate a merge of two similar clusters into one. The root combines all records into a single cluster [40].

As divisive HC exhibits an immense overhead for examining the optimal splits ($2^{N-1}-1$ possibilities [40], where N is the dataset size), mostly agglomerative algorithms have been observed in practice. Traditionally, three merging methods were used: (1) single, (2) complete, and (3) average linkage. Single linkage merges the two clusters with the closest two elements, complete linkage merges the two clusters whose maximally distant pair of elements are closest among all pairs of clusters, and average linkage merges the two clusters that have the smallest average of all pairwise distances of their elements [52, p. 76-77] [59].

Naive HC has computation complexity $\mathcal{O}(N^3)$ and space complexity $\mathcal{O}(N^2)$ [42]. Some HC-based algorithms (e.g., single linkage) cannot detect some cluster shapes. They do not incorporate a notion of noise, but are relatively insensentive to outliers. HC requires to pre-determine the number of clusters K that are obtained by cutting the tree at the respective level [40]. HC needs a restructuring of the tree if new data records are added after the first clustering. Nevertheless, HC can handle any type of variable and the input order does not affect the result.

1 A dendogram is a graph representing a tree structure.

Density-based Clustering. These algorithms use a density-based neighborhood notion such that input records that lay together in a dense area form a cluster. Examples are Mean-shift [53] and DBSCAN [60]. Mean-shift has time complexity $\mathcal{O}(N^2t)$, where N is the dataset size and t is the number of iterations, which makes it inefficient for large datasets. It can handle any cluster shape and flexibly determine the number of clusters K based on the input data. Additionally, the input order does not affect the results. However, the value of the bandwidth h in the Kernel Density Estimator (KDE) used in Mean-shift can significantly affect its performance. A too large h merges distinct clusters while a too small h splits one cluster into multiple smaller groups. The performance also deteriorates for high dimensional data due to the "curse of dimensionality" in the KDE. Similarly, noisy features can hamper the performance [61]. Mean-shift does not incorporate a notion of noise. An update with new records can change the KDE and the local maximas, thus requiring a rerun of the entire algorithm. However, in practice, the new points can be assigned to the cluster containing the nearest mode if the change in the KDE is not significant.

Density-Based Spatial Clustering of Applications with Noise (DBSCAN [62]) specifically recognizes noisy elements and marks them as outliers. It detects arbitrarily shaped clusters and flexibly determines the number of clusters in a dataset based on two input parameters, namely the minimal cluster size and the maximal distance between two clusters. Especially the second parameter can be difficult to determine and DBSCAN cannot correctly handle clusters with significantly different densities [63]. Generally, if appropriate distance measures are chosen, any type of parameter can be clustered. Moreover, the input order does only affect the clustering result in exceptional cases where border elements lay in the range of more than one cluster. If additional data records shall be clustered after a first clustering was finished, their neighbors have to be determined to assess if they can be added to previously created clusters. Otherwise, they may create a new cluster with other outliers, but no completely new clustering is needed. Naive DBSCAN needs $\mathcal{O}(N)$ memory and has computation complexity $\mathcal{O}(N^2)$ [62, 63].

Distribution-based Clustering. Distribution-based clustering algorithms assume that clusters are drawn from an unknown mixture of distributions and aim at approximating the original distributions (i.e., the type and parameters) as well as the number of different distributions (i.e., the number of clusters) [41, 42]. A well-

known example for distribution-based clustering algorithms are Gaussian Mixture Models (GMM) using the Expectation-Maximization (EM) algorithm [64].

Comparison of Clustering Algorithms. Modifications proposed for the clustering algorithms to fix some weaknesses of the original often introduce other problems. Therefore, it is difficult to evaluate them with respect to the general requirements for clustering algorithms (cf. §2.1).

In Tab. 2, we compare the eight baseline clustering algorithms for which privacy-preserving protocols have been proposed with respect to the properties of good clustering algorithms listed in §2.1. We did not include the effect of property 9), i.e., high dimensionality, because it is often not directly linked to the clustering algorithm. Instead, a large number of variables often requires using feature reduction techniques.

2.2 Cryptographic Building Blocks

In the following, we summarize secure computation techniques and respective security models.

Secure Computation. There are two main paradigms for secure computation: Homomorphic encryption (HE) and multi-party computation (MPC). HE [7, 65, 66] enables operations on a set of ciphertexts such that the resulting ciphertext contains the result of a function on the corresponding plaintexts. MPC allows two or more mutually distrusting parties to jointly compute a function on their private inputs. Two well-known generic approaches for MPC are based on garbled circuits (GC) [21] and secret-sharing (SS) [20, 67]. As an example for a SS-based technique, the GMW protocol [20] represents a function as Boolean/Arithmetic circuit and the values are secret-shared using XOR or Arithmetic secret sharing (ASS). Another type of SS is Shamir's secret sharing (SSS) [67].

Security Models. Two main security models have been considered in privacy-preserving clustering: In the *semi-honest/passive security model*, the adversary [68] is assumed to honestly follow the protocol, but tries to learn additional information about the private inputs of other parties. Though this model is weaker than the *malicious* model, that even protects against deviations from the protocol specification, it facilitates practically-efficient applications especially for privacy-preserving machine learning (PPML) [69]. *Full threshold* security means that up to $N-1$ parties can collude without jeopardising privacy while *honest majority* security requires the majority of the parties to not collude.

3 Privacy-preserving Clustering

In this section, we first define privacy-preserving clustering. Then, we categorize and analyse the existing privacy-preserving clustering protocols to conclude which protocols offer good efficiency with strong privacy guarantees. Afterwards, we discuss possible applications and provide indications on how to choose appropriate privacy-preserving clustering schemes for these.

3.1 Functionality and Requirements

In an ideal world with a trusted third party (TTP), all involved parties send their input data to the TTP. The TTP then performs the clustering and returns the output to the parties. The output can vary depending on the application requirements and clustering algorithm. For example, the output can be the cluster centroids or it can be the cluster label for each data record.

We identified the following requirements for privacy-preserving clustering:

Privacy. According to the *ideal functionality* a privacy-preserving clustering protocol must not leak information other than what can be derived from the output of the protocol to be considered as *fully privacy-preserving*. Importantly, this includes that all operations must be obliviously realized and all intermediate results must be kept private.

Efficiency. A privacy-preserving clustering scheme must be efficient in terms of communication and runtime. This means that it must scale well with respect to the dataset size N, the number of clusters K, and the dimensionality d of the input records.

Clustering Quality: A privacy-preserving clustering scheme must offer a good clustering quality of the results independent of a dataset's properties. Specifically, the requirements of good clustering listed in §2.1 should be fulfilled.

Flexibility. A privacy-preserving clustering scheme should ideally be flexibly usable for *outsourcing* [70] and *multi-party computation*. In an outsourcing scenario, one or multiple data owners outsource their data and the computation to untrusted non-colluding parties [70]. Here, the data owners can even be malicious (cf. §2.2). In multi-party computation, several parties interactively compute the clustering on their joint dataset.

3.2 Existing Private Clustering Protocols

In this subsection, we categorize the existing works on privacy-preserving clustering with respect to the underlying plaintext clustering algorithm, security model, scenarios for which protocols where designed, data distributions, used secure computation techniques as well as privacy and efficiency (cf. §3.1). We discuss the strengths and weaknesses of these schemes with respect to these criteria. Tab. 3 contains on overview of all 59 works on privacy-preserving clustering with secure computation techniques that we are currently aware of. It indicates the respective security model, used secure computation techniques, common types of leakages of intermediate values, the type of output, which and how many parties are involved in the protocol, the data partition, and other issues.

Plaintext Clustering Algorithms. Eight clustering algorithms have been investigated in the context of privacy-preserving clustering: K-means (including the two variants Kernel K-means [58] and Possibilistic C-means [43]), K-medoids [57, 71], GMM [44, 45], Mean-shift [9], DBSCAN [22, 46, 47, 72–76], baseline agglomerative HC (e.g., single linkage or complete linkage) [19, 48–50, 77, 78], BIRCH [79, 80], and Affinity Propagation [16, 81]. The vast majority of works focuses on the simple K-means algorithm [8, 11–15, 82–106], which enables an efficient parallelization of computation through packing with homomorphic encryption [8, 88, 95] or amortization through batched oblivious transfers [11]. However, as discussed in §2.1, K-means can be used only for very specific applications where the number of clusters is known in advance and the clusters are convexly shaped. We gave an overview of the strengths and weaknesses of these plaintext clustering algorithms in Tab. 2. Generally, the choice of the plaintext clustering algorithm heavily affects the quality of the clustering result. Some works on privacy-preserving clustering exactly reproduce the original algorithms and hence achieve the same accuracy, e.g., [19, 44, 45, 82]. Others deviate from the original algorithms such as when updating the centroids in K-means due to, e.g., normalization/quantization/specific encodings of the plaintext space [8, 9, 14, 22, 88, 95], adaptations of the original algorithm [14], or approximations [14, 43] which either enhance efficiency or are needed because of the underlying secure computation techniques.

Security Models. All works except for [16, 96, 105] consider only the semi-honest security model (cf. §2.2). A few even do not explicitly define their security model [43, 48, 57, 58, 71, 100, 108]. The semi-honest security model assumes that the adversary correctly follows the protocol while trying to gain additional information. However, this strong assumption is not always realistic. Concretely, the use of protocols that are secure against semi-honest adversaries is only acceptable in specific applications where the participants already generally trust each other but are legally not allowed to share data, e.g., hospitals conducting medical analysis or central banks for financial analytics on country-level. We discuss the requirements and implications of applications on the choice of a privacy-preserving clustering scheme in more detail in §3.3.

Scenarios. Generally, privacy-preserving clustering protocols have been designed for two scenarios: Firstly, multi-party computation (MPC, [16, 44–46, 50, 57, 71, 72, 74–76, 82, 85, 87, 89, 91, 94, 96, 99–101, 103]) with the special case of two-party computation (2PC, [11–13, 19, 22, 48, 49, 73, 74, 77–80, 83, 84, 86, 93]), where two or more data owners jointly perform a secure computation protocol ideally such that nothing beyond the output is leaked to each other (cf. §3.1). Some of these protocols [50, 75, 76, 88, 95, 100, 103] also involve one or more additional (semi-trusted) entities, e.g., represented by servers, that assist in the computation. In contrast, other protocols were designed for the outsourcing scenario where one or more data owners outsource computation (and storage) to external parties who ideally perform the clustering for them without learning anything about the input data [8, 9, 14, 15, 43, 47, 58, 90, 92, 97, 98, 102, 104–106, 108]. As outsourcing aims at using external resources, data owners should not be involved in the execution of the protocol and can go offline, but this is often not fulfilled, e.g., in [43, 47, 97, 98, 104, 105, 108]. Some MPC/2PC protocols can also be used for an outsourcing scenario where the data owners secret share their data among multiple non-colluding parties who then perform the clustering [8, 11, 105]. However, whether a 2PC/MPC clustering protocol is usable for outsourcing heavily depends on its design. This is hindered if data owners are actively involved by computing on plaintext input data, e.g., [22, 44–46, 48, 49, 72–75, 82, 88], or a data owner needs to perform intermediate decryptions, e.g., [86, 91, 95].

Data Partition. The data to be clustered in a private manner can be partitioned in three ways when provided by multiple parties. It is horizontally partitioned when each data owner holds complete (but different) data records [9, 13, 44, 45, 47, 49, 50, 73, 75, 77, 78, 84–86, 91, 92, 94–96, 98, 99, 101–103, 105, 106, 108]. The data is vertically partitioned when data owners hold mutually different parameters of the same data records [44,

Algorithm	Scheme	Privacy	Security	PETs	L1	L2	L3	L4	O1	O2	O3	Interactivity (Scenario)	Data	Other issues
K-means	[82, KDD'03]	✗	◐	HE+blinding	(✗)[1]	✗	✗	✗	✗	✓	✗	all data owners (≥ 3)	v	
	[83, KDD'05]	✗	◐	HE+ASS+GC	✓	✓	✗	✗	✓	✓	✗	2PC	a	wrong division
	[84, ESORICS'05]	✗	◐	HE *or* OPE	✗	✓	✓	✗	✗	✓	✗	2PC	h	
	[12, CCS'07]	✓	◐	HE+ASS	✓	✓	✓	✗	✗	✓	✗	2PC	a	
	[85, SECRYPT'07]	✗	◐	blinding	✗	✓	✗	✗	✓	✓	✗	all data owners	v/h	
	[86, AINAW'07]	✗	◐	HE+ASS+OPE	✓	✗	✗	✗	✓	✓	✗	2PC	h	
	[87, PAIS'08]	✗	◐	ASS	✓	✓	✗	✗	✓	✓	✗	all data owners (≥ 4)	v	
	[88, WIFS'09]	✗	◐	HE	✗	✓	✗	✓	✓	✗	✗	data owners + 1 server	h	
	[89, KAIS'10]	✗	◐	HE+ASS	✓	✓	✗	✗	✓	✗	✗	all data owners	h	
	[90, PAISI'10]	✗	◐	SS	✓	✗	✗	✗	✓	✓	✗	Outsourcing ≥ 3 servers	a	
	[91, ISPA'10]	✗	◐	HE	✓	✓	✗	✗	✗	✓	✗	all data owners	v/h	
	[92, WIFS'11]	✗	◐	HE+GC	✓	✗	✓	✓	✓	✗	✗	Outsourcing, 3 servers	h	
	[93, ISI'11]	✗	◐	HE+ASS	(✗)[1]	✗	✗	✗	✓	✗	✗	2PC	v	
	[94, TM'12]	✗	◐	SSS	✗	✗	✓	✗	✗	✓	✗	all data owners	h	distance calculation unclear
	[95, JIS'13]	✗	◐	HE	✗	✓	✓	✗	✓	✗	✗	data owners + 2 servers	h	
	[96, ICDCIT'13]	✗	●	SSS+ZKP	✗	✗	✓	✗	✗	✓	✗	all data owners	h	
	[97, ASIACCS'14]	✗	◐	HE	✗	✗	✗	✗	✓	✓	✗	outsourcing, 1 data owner + 1 server	–	insecure HE [107]
	[98, MSN'15]	✗	◐	HE	✗	✗	✗	✓	✗	✗	✗	outsourcing, data owners + 1 server	h	insecure HE [107]
	[99, IJNS'15]	✗	◐	HE	✗	✗	✗	✗	✗	✓	✗	all data owners	h	
	[13, CIC'15]	✓	◐	HE	✓	✓	✓	✗	✗	✓	✗	Outsourcing, 2 servers	h	
	[100, ICACCI'16]	✗	N/A	SS	✗	✗	✗	✗	✓	✗	✗	arbitrary number of servers	a	
	[101, ISPA'16]	✗	◐	blinding	✗	✗	✗	✓	✗	✓	✗	all data owners (≥ 3)	h	
	[102, SecComm'17]	✗	◐	HE	✓	✗	✗	✓	✗	✓	✗	outsourcing, ≥ 4 servers	h	
	[103, TII'17]	✗	◐	HE	✗	✗	✗	✗	✗	✗	✗	data owners + 1 server	h	
	[14, SAC'18]	✓	◐	HE	✓	✓	✓	✓	✗	✓	✗	Outsourcing, 1 server	–	
	[15, CLOUD'18]	✓	◐	HE	✓	✓	✓	✗	✗	✓	✗	Outsourcing, 2 servers	–	distance calculation unclear
	[108, CCPE'19]	✗	N/A	HE	✗	✗	✗	✗	✗	✓	✗	Outsourcing, 2 data owners + 1 server	h	insecure HE [107]
	[104, TCC'19]	✗	◐	HE	✓	✗	✗	✓	✓	✗	✗	Outsourcing, 1 data owner + $\geq$ 1server(s)	–	
	[105, Inf. Sci.'20]	✗	◐ (●)[2]	HE+GC	✗	✗	✗	✗	✗	✓	✗	Outsourcing, 2 data owners + 1 server	h	
	[106, SCN'20]	✗	◐	HE+SKC	✓	✗	✗	✓	✗	✓	✗	Outsourcing, 3 servers	h	
	[11, PETS'20]	✓	◐	GC	✓	✓	✓	✗	✗	✓	✗	2PC/Outsourcing	h	
	[8, TKDE'20]	✗	◐	HE	✓	✗[3]	✓	✗	✗	✓	✗	Outsourcing, 2 servers	a	
Kernel K-means	[58, KAIS'16]	✗	N/A	PKC	✓	✗	✗	✗	✓	✗	✗	Outsourcing, 1 server	–	security model
Possibilistic C-means	[43, TBD'17]	✗	N/A	HE	✗	✗	✗	✗	✓	✓	✗	Outsourcing, 1 data owner + 1 server	–	
K-medoids	[57, SMC'07]	✗	N/A	HE+blinding	✓	✗	✗	✓	✗	✗	✗	all data owners	v	exhaustive search
	[71, CCSEIT'12]	✗	N/A	HE+blinding	✓	✗	✗	✓	✗	✗	✗	all data owners	v	exhaustive search
GMM	[45, KAIS'05]	✗	◐	blinding	✓	✓	✗	✗	✓	✗	✗	all data owners	h	
	[44, DCAI'19]	✗	◐	ASS	✓	✓	✗	✗	✓	✗	✗	all data owners (> 2)	v/h	
Affinity Propagation	[81, INCoS'12]	✗	◐	HE + blinding	✓	✓	✗	✓	✓	✗	✗	all data owners	v	
	[16, SECRYPT'21]	✓	◐/●	ASS+GC	✓	✓	✓	✓	✓	✗	✗	all data owners/Outsourcing	a	
Mean-shift	**[9, SAC'19]**	✓	◐	HE	✓	✓	✓	✓	✓	✗	✗	Outsourcing, 1 server	–	
DBSCAN	[72, ISI'06]	✗	◐	blinding	✓	✓	✗	✓	✗	✗	✗	all data owners	v	lack of complete protocol
	[73, ADMA'07]	✗	◐	HE+blinding	✓	✗	✗	✓	✓	✗	✗	2PC	v/h	
	[74, IJSIA'07]	✗	◐	PKC+blinding	✓	✓	✗	✓	✓	✗	✗	all data owners	v	
	[75, ITME'08]	✗	◐	HE+blinding	✓	✗	✗	✓	✓	✗	✗	data owners + 1 server	h	
	[22, TDP'13]	✗	◐	HE+blinding	✓	✗	✗	✓	✓	✗	✗	2PC	a	
	[17, S&P'12]	✓	◐/●[5]	GC	✓	✓	✓	✓	✓	✓	✗	2PC	h	
	[46, SIBCON'17]	✗	◐	HE+PKC	✓	✓	✗	✓	✓	✗	✗	all data owners	v	cluster expansion missing
	[47, PRDC'17]	✗	◐	HE	✓	✗	✗	✓	✗	✗	✗	outsourcing, all data owners + 1 server	h	
	[76, AI'18]	✗	◐	HE	✓	✗	✗	✓	✓	✗	✗	data owners + 1 server	a	uses absolute distance
	[18, ASIACCS'21]	✓	◐	ASS+GC	✓	✓	✓	✓	✓	(✓)[4]	✗	2PC/Outsourcing	a	
HC	[77, SDM'06]	✗	◐	HE+ASS+GC	✓	✓	✗	✓	✗	✓	✗	2PC	h	
	[50, TKDE'07]	✗	◐	blinding *or* SKC	✓	✓	✗	✓	✓	✗	✗	data owners + 1 server	h	SKC not semantically secure
	[49, TDP'10]	✗	◐	HE+GC	✓	✓	✗	✓	✓	✓	✗	2PC	h	
	[48, ISI'14]	✗	N/A	HE	✓	✗	✗	✓	✓	✓	✗	2PC	v	
	[78, ISCC'17]	✗	◐	HE	✓	✓	✗	✓	✗	✗	✓	2PC	v/h	
	[19, ArXiv'19]	✓	◐	HE & GC	✓	✓	✓	✓	✗	✓	✓	2PC	h	
BIRCH	[79, SDM'06]	✗	◐	HE+ASS	✓	✓	✗	✓	✗	✗	✗	2PC	v	
	[80, ADMA'07]	✗	◐	HE+ASS	✓	✓	✗	✓	✗	✗	✗	2PC	a	

[1] Of the parameters hold by the respective data owner.
[2] Assuming max. 1 party deviates from the protocol.
[3] Leaks partial information about cluster sizes.
[4] Not implemented, but possible.
[5] Can be used with any security model of GCs.

Table 3. History overview of privacy-preserving clustering using secure computation techniques. Privacy indicates if fully privacy protection according to the ideal functionality for privacy-preserving clustering (§3.1) is provided (✗: leakage; ✓: no leakage). ◐ is the semi-honest security model, ● is the malicious security model, N/A indicates that no security model was defined. HE is homomorphic encryption, ASS additive secret sharing, SSS Shamir's secret sharing, GC garbled circuits, OPE oblivious polynomial evaluation, PKC public-key cryptography, SKC symmetric-key cryptography, ZKP zero-knowledge proof, blinding is the use of random values for blinding, and other types of secret sharing are summarized by SS. v indicates that the data that shall be clustered is vertically distributed, i.e., the data owners hold the values for a subset of parameters from all data records. h indicates horizontally partitioned data where the data owners hold complete data records with all parameters, and a is arbitrary data partitioning. L1 leaks intermediate centroids, L2 intermediate cluster sizes, L3 other intermediate values (e.g., intermediate cluster assignments or distance comparison results), and L4 the number of clustering iterations. O1 outputs the final cluster labels/assignments, O2 outputs the final centroids, and O3 outputs the final dendogram/tree structure. The schemes with the best privacy guarantees are marked in bold (we do not consider the number of clustering iterations as a severe leakage as it can be easily avoided, cf. §3.2). The efficient and fully private schemes that we implemented and benchmarked in §4 are highlighted in gray.

46, 48, 57, 71–74, 78, 79, 82, 85, 87–89, 91, 93]. An arbitrary partitioning is a mix of both vertical and horizontal data splitting [8, 12, 22, 76, 80, 83, 90, 100]. A realistic data partition depends on the specific application. We discuss this matter in more detail in §3.3.

Used Secure Computation Techniques. Existing privacy-preserving clustering protocols use two main cryptographic techniques. First, there is a range of works that use homomorphic encryption (HE), e.g., [8, 14, 48, 76, 78, 84, 97, 104], but most of them tend to be relatively slow due to the expensive cryptographic operations. Another research direction uses multi-party computation (MPC) techniques like Yao's Garbled Circuits [21], blinding with random numbers, and secret sharing to achieve better efficiency [11, 16, 18, 44, 45, 72, 83, 94]. These schemes tend to have better runtimes, but higher communication than using HE. However, some MPC techniques [10] also have to rely on non-collusion assumptions between (a subset of) the computing parties which can make them more difficult to deploy in real-world applications. Other protocols use a mix of these techniques aiming at combining the strengths of both approaches [12, 19, 46, 77, 79, 103].

Privacy. As discussed in §1, information leakage can cause severe privacy infringement. Ideally, no information beyond what can be extracted from the final output should be derivable (cf. §3.1). However, most of the proposed privacy-preserving clustering schemes leak intermediate values like the intermediate centroids [43, 84, 85, 88, 94, 97, 98, 100, 105, 108], cluster assignments [43, 57, 58, 71, 83, 86, 87, 89–91, 97, 98, 100, 102–106, 108], and/or cluster sizes [8, 57, 71] in each clustering iteration of K-means or K-medoids, thus, failing to provide full privacy protection. Similarly, both private GMM schemes [44, 45] leak the intermediate covariance matrices, means, and probability values for each Gaussian distribution. Many schemes originating from a round-based clustering algorithm such as K-means or GMM leak the number of clustering iterations until convergence, e.g., [12, 13, 15, 43–45, 82, 83, 94, 100]. However, this issue can be avoided by clustering for a fixed number of iterations independent of the input which must be large enough to reach a good clustering result. However, this results in a longer runtime as more iterations are done than normally with a convergence check. Also most DBSCAN-based and HC-based schemes leak information, e.g., distances between data records [46, 50], the comparison results of distances [48, 72, 75, 79, 80], cluster assignments [22, 46, 47, 49, 73, 75, 76, 79, 80], cluster sizes [22, 47, 73, 75, 76], or may even leak concrete input records for specific data constellations [73, 77] to at least one of the involved parties (independent of the party's data ownership). All in all, we only identified ten clustering protocols shown in Tab. 1 that provide fully privacy guarantees (maximally leaking the number of clustering iterations): [9, 11–19].

Efficiency. As stated before, homomorphic encryption-based protocols such as [14, 48, 76, 78, 84, 97, 104] tend to be computationally expensive and, thus, slower than MPC-based schemes, e.g., [11, 18, 44, 45, 72, 94], which require more communication. Due to space limitations, we will focus here on the ten protocols that provide full privacy (cf. §3.1) and compare them in terms of efficiency. Kim and Chang [15] observe an about 2.85× runtime improvement compared to [13] thanks to a more efficient secure comparison. They as well as Bunn and Ostrovsky [12] use the Paillier encryption scheme without any parallelization making it expensive and slow compared to the other more optimised protocols which use, for example, packing or batching of operations [11]. The K-means protocol by Mohassel et al. [11] was experimentally compared to [14] and [18]. It outperforms the K-means protocol by Jäschke et al. [14] by five orders of magnitude on a dataset with 400 elements thanks to an efficient batching and the usage of GC instead of HE. It is also 19× faster on the same dataset than the private DBSCAN protocol of Bozdemir et al. [18], but DBSCAN often achieves significantly better clustering quality [18]. [18] runs about 13 minutes for clustering 500 elements while Zahur and Evans [17] report more than 550 minutes for their private DBSCAN protocol with 480 records. [18] is also 194× faster than the fully-private affinity propagation protocol by Keller et al. [16] thanks to the use of optimized combinations of GC and ASS. No direct comparison between [9, 11, 19] was done so far.

Choice for Benchmarks. To summarize, the MPC-based protocol of Mohassel et al. [11] is the most advanced private K-means scheme w.r.t. privacy and efficiency. Meng et al. [19] and Cheon et al. [9] provide the only schemes that offer fully private single and complete linkage HC/Mean-shift. Bozdemir et al. [18] propose the most efficient fully privacy-preserving DBSCAN protocol. In §4, we focus on these four works by comparing their computation and communication efficiency, security and privacy, and clustering quality.

3.3 Private Clustering Applications

Privacy-preserving clustering can be generally used for two main purposes: to protect sensitive data when out-

sourcing the computation and storage and/or when multiple data owners provide input data to the clustering. For each of these scenarios, we discuss a few example applications in the following to give a guideline on how to choose appropriate private clustering protocols.

Example Applications. Multiple data owners who jointly cluster their combined data is an instance of *multi-party computation* (MPC). In the financial market, clustering is used to automatically detect correlations between securities' stock prices in pair trading, i.e., for investment strategies that leverage discrepancies between typically correlated securities [109, 110]. Additionally, it is used for outlier detection to identify credit card, insurance, or tax frauds and insider trading [3, 111]. In this context, it is typically necessary to cluster data from several sources like competing (investment) banks or insurance companies to detect suspicious behavior [112]. Thereby, the different entities might hold information about the same customer, i.e., they have vertically partitioned data. Furthermore, clustering can be used by companies for enhancing marketing measures, e.g., by market segmentation or personalization of recommendation systems [113, 114]. A larger database increases the quality and reliability of the result but business secrets and customer data must be protected. In such scenarios, a horizontal data partitioning where the companies provide data from different customers is more plausible. Additionally, clustering is also used in medical research and diagnosis [115, 116]. In this context, using data from several sources, e.g., several hospitals, reduces potential bias caused by demographics, ethnicities, or cultures.

MPC. For the aforementioned applications, the parties can always safely trust themselves. Thus, full threshold security (i.e., security against up to $N-1$ collusions, cf. §2.2) as provided by some MPC techniques [117, 118] would be an interesting option. Unfortunately, only Keller et al. [16] and schemes based on a threshold secret sharing (like SSS (cf. §2.2)) with the respective threshold can offer this. Additionally, again only Keller et al. [16] provide full privacy guarantees against malicious adversaries (cf. §2.2). However, if the other partners involved are generally trusted but strictly regulated by data protection laws like HIPAA or GDPR, hindering them from directly sharing data, a 2PC- or MPC-based clustering protocol with honest majority and secure against semi-honest adversaries, e.g., [9, 11, 19], might be sufficient and provides significantly better efficiency than MPC techniques that are secure against full threshold semi-honest or malicious adversaries [119].

Outsourcing. While running "generic" MPC protocols is the most straightforward approach to securely cluster on the joint database of data owners, it suffers from high computation and communication costs and might be practically infeasible for a large number of data owners as MPC protocols often scale quadratically in the number of parties. A more efficient alternative can be to outsource the evaluation.[2] Thereby, the data owners (e.g., competing companies conducting market analyses) might prefer to not rely on a non-collusion assumption needed for MPC-based protocols such as [11] where multiple providers must be found and trusted not to collude. Hence, in such a situation an HE-based outsourcing scheme like [9, 14] might be advantageous. Additionally, in an outsourcing scenario, no data owner should be required to be online or actively involved in the computation. This can only be achieved by some protocols: [11, 13–16, 18, 19, 37, 43, 58, 90, 92, 100, 102, 106].

Privacy vs. Efficiency. Furthermore, there exists a trade-off between data leakage and efficiency. Schemes that can leak complete data records (e.g., [73, 74]) should not be used. When generally trusted parties like hospitals are involved, leaking less critical information like the number of iterations (cf. §3.2) might be acceptable to reduce runtime. However, as pointed out in §1, it is not always possible to fully understand and anticipate the effects of leaking intermediate results. Generally, MPC-based protocols are considered to be faster but require more communication than HE-based protocols. We will give more insight on the efficiency of the four most efficient fully private clustering schemes in §4.

Algorithm Characteristics. Finally, another important aspect for choosing the right private clustering scheme are the input parameters. For instance, K-means, K-medoid, GMM, and HC require the number of clusters as input. This is not an issue when the number of clusters is fixed by the application, e.g., if the goal is to split bank customers' behavior into benign and suspicious. However, a more fine-grained analysis might be needed when different types of malicious behavior can occur that significantly differ from each other (e.g., credit card, tax, or insurance fraud), but it is unclear in advance how many untypical behaviors can occur. Clustering might even be used to detect and differentiate these outliers in the first place. In such a case, a protocol that originates from affinitey propogation, DBSCAN or Mean-shift should be chosen as they will flexibly detect the number of clusters. Furthermore, some algo-

2 A single data owner might of course also outsource clustering.

rithms like K-means and GMM can only detect clusters of convex shapes, while DBSCAN can detect arbitrarily shaped clusters [41]. If new data arrives regularly, it might be beneficial to avoid recomputing the complete clustering by using K-means-, K-medoid-, DBSCAN-, or GMM-based protocols (e.g., [8, 11, 43, 44, 46]). Then, different private clustering schemes give different outputs, e.g., centroids [8, 11] or dendograms [19]. For example, a medical analysis detecting typical characteristics for a specific disease should output centroids as they represent these characteristics. Additionally, centroids also allow to assign new data to the created clusters later on which is not possible with only the cluster labels.

To conclude, the following aspects need to be examined when deciding upon a private clustering protocol: scenario (MPC vs. outsourcing), security/privacy requirements, trust level among the data owners, data distribution and splitting, and the plaintext clustering algorithm's characteristics (e.g., required input parameters that can be anticipated in advance). Based on this information, our extensive summary in Tab. 3 can help to choose an appropriate protocol.

4 Evaluation

In this section, we compare the clustering quality (§4.1), security and privacy (§4.2), and efficiency (§4.3) of the four most efficient fully private clustering protocols [9, 11, 18, 19] identified in §3.2. Details about these protocols are provided in Appx. B.

Software Details. We implemented all four protocols in C++17 and instantiate all cryptographic building blocks with a security level of 128 bits. We instantiate all algorithms with optimal parameters to assess their performance assuming perfect conditions. All our implementations are single threaded for a fair comparison of the efficiency of protocols.

MPC-KMeans [11]. In the remainder of this work, we call the private K-means protocol by Mohassel et al. [11] MPC-KMeans. We use the publicly available implementation[3] from the authors of [11] with default parameter values. Specifically, the statistical security parameter is $\lambda = 40$, the computational security parameter $\kappa = 128$, and the bitlength $\ell = 32$.

HE-Meanshift [9]. We call the private Mean-shift protocol by Cheon et al. [9] HE-Meanshift. The implementation uses the HEAAN library [120] with the same parameters as [9] providing 128-bit security. Specifically, the degree of the polynomial modulus of the plaintext ring N_c is set to 2^{17} and the ciphertext modulus q_L is set to 2^{1480}. Thus, the number of plaintext slots in each ciphertext is 2^{16}. Unless explicitly stated, we set the degree parameter for the kernel $\Gamma = 6$, the MinIdx degree parameter $t = 5$, and the Inv iteration parameter $\zeta = 5$.

PCA/OPT [19]. We call the baseline private HC protocol by Meng et al. [19] PCA (complete linkage) and its extension OPT (single linkage). The implementation uses the ABY framework [10] for Yao's garbled circuits and the libpaillier library [121] for Paillier with identical parameters as [19]. Specifically, the symmetric-key security parameter is $\kappa = 128$ bits and the size of the RSA modulus in Paillier encryption is $\kappa_{\text{pub}} = 2048$ bits. The statistical security parameter is $\lambda = 40$.

ppDBSCAN [18]. We call the privacy-preserving DBSCAN protocol by Bozdemir et al. [18] ppDBSCAN. We use the publicly available C++ implementation[4] provided by the authors which is based on the ABY framework [10]. The computational security parameter is set to $\kappa = 128$ bits and the bitlength $\ell = 32$ bits. The parameter $maxIterations$ was set to 4 as also done in [18].

4.1 Clustering Quality

In this section, we evaluate the clustering quality of the four fully private clustering protocols.

Datasets. We use nine datasets from the well-known FCPS [122] and Graves [123] collections designed for benchmarking clustering algorithms. They also include the ground truth separation [124]. In Tab. 4, we summarize four of these datasets for which we present the results of the quality evaluation. The results of our evaluation on the remaining 5 datasets are given in Appx. D.

Metrics for Clustering Quality. We measure the quality of the clustering result using *clustering quality indices*. As no single index is superior [125], we use four well-known indices: Adjusted Rand Index (ARI) [126], Adjusted Mutual Information (AMI) [127], Silhouette Index (SI) [128], and Calinski-Harabasz Index (CHI) [129]. SI and CHI measure the output clusters' separation and compactness while ARI and AMI

3 https://github.com/osu-crypto/secure-kmean-clustering

4 https://encrypto.de/code/ppDBSCAN

Dataset	N	d	K	Property
Hepta	212	3	7	Well-defined clusters
Lsun	400	2	3	Different shapes
Chainlink	1000	3	2	Non-linearly separable clusters
Dense	200	2	2	Different cluster variances

Table 4. Datasets used for evaluating clustering quality, where N is the dataset size, d is the dimension of the data records, and K is the number of clusters.

compare the output clusters to the known ground truth to evaluate the clustering quality [125].

The results for the algorithms with random initialization, MPC-KMeans and HE-Meanshift are averaged over 10 runs. The iterative algorithms MPC-KMeans, HE-Meanshift, KMeans++, and Mean-shift are run for 20 iterations. The number of clusters K for the K-Means and HC protocols, i.e., MPC-KMeans and PCA/OPT, is set to the number of clusters in the ground truth. HE-Meanshift modifies the Mean-shift algorithm to run the mean-shift process on a small and random subset of datapoints, called dusts, to improve efficiency. The number of dusts is set to the largest power of 2 greater than or equal to the number of clusters, to ensure efficiency while maintaining the quality of the clustering.

Original vs. Private Algorithm. We also compare the differences in clustering quality between the private clustering protocols and the original plaintext algorithms to evaluate the *error arising by using privacy preserving techniques*. PCA, OPT and ppDBSCAN are identical to plaintext HC with complete and single linkage and DBSCAN respectively which is why we do not include the results for their plaintext implementations here. The underlying computations in MPC-KMeans are identical to the standard K-means protocol except for differences in the initialization of the centroids. We evaluate this effect using the plaintext KMeans++ [130] algorithm (a variant of K-Means with an improved cluster initialization where the centroids are initialized with data records far apart from each other). HE-Meanshift, in contrast, introduces several modifications to the standard Mean-shift algorithm [53] to make the computation HE friendly, e.g., using a polynomial kernel and adopting dust-sampling for efficient mode-seeking. We compare the clustering quality of HE-Meanshift to a plaintext implementation of Mean-shift to evaluate the combined effect of the changes.

Fig. 1 summarises the results of our evaluation of the clustering quality with the four quality indices.

Hepta Dataset. All algorithms achieve a relatively good clustering quality. PCA, OPT and ppDBSCAN output exactly the ground truth and achieve the best scores on the four indices. MPC-KMeans has a slightly worse clustering quality than the HC algorithms. HE-Meanshift achieves significantly lower scores and its *high standard deviation* in comparison to KMeans++ and Mean-shift indicates that the initialization of dusts has a significant impact on the clustering quality.

Lsun Dataset. OPT and ppDBSCAN output exactly the ground truth and, thus, achieve the maximal scores for the ARI and AMI. While the output of PCA significantly differs from the ground truth, it is noteworthy that it achieves similar scores as OPT and ppDBSCAN on the SI and CHI that measure internal cluster properties, i.e., separation and compactness. HE-Meanshift again shows large standard deviations due to the random initialization. The poor clustering quality achieved by MPC-KMeans and KMeans++ on the ARI and AMI is due to the non-convexly shaped clusters in the ground truth where K-means does not work well. The best scores achieved by HE-Meanshift are significantly higher than their plaintext counterparts, possibly due to favourable (random) initialization for the non-convexly shaped clusters in some of the runs.

Chainlink Dataset. OPT and ppDBSCAN have the same output as the groundtruth and achieve the highest ARI and AMI scores though the presence of non-linearly separable clusters in the dataset leads to lower SI and CHI scores. The remaining algorithms perform poorly. A poor clustering quality of MPC-KMeans and KMeans++ is expected since K-Means does not work well on non-linearly separable clusters.

Dense Dataset. MPC-KMeans, HE-Meanshift, ppDBSCAN, KMeans++, and Mean-shift achieve good scores on all indices while the HC protocols, i.e., PCA and OPT, have significantly lower scores in comparison. Intuitively, the poor clustering quality of HC-based protocols can be attributed to the large variance in one of the clusters of the Dense dataset. This can cause incorrect merging of clusters since clusters are merged based on their proximity, which can be large when the variance of the cluster is high.

Conclusion. ppDBSCAN consistently achieves the highest scores and is able handle different shapes, non-linear clusters, and high cluster variance well. PCA and OPT achieve a relatively good clustering quality on three out of four datasets, but they (completely) fail on the Dense dataset. The K-Means and Mean-shift protocols have comparable clustering quality that heavily varies between different datasets. The K-means-based protocols can only cluster very specific datasets that do not contain non-convexly shaped and non linearly-separable clusters.

HE-Meanshift tends to have large standard deviations which indicate a strong dependency on dust initialization. However, the highest score achieved by HE-Meanshift is comparable to that of plaintext Mean-shift which indicates that the modifications introduced for its HE-friendly computation do not decrease accuracy. In contrast, MPC-KMeans has a small standard deviation and achieves a similar clustering quality to KMeans++, which shows that the randomness used for centroid initialization has a smaller impact on final output.

4.2 Security & Privacy

In this section, we discuss the security and privacy of the four clustering protocols.

Security Model w.r.t Scenario. All four works are in the static semi-honest security model i.e., the adversary can corrupt some of the parties at the onset of the computation and correctly follows the protocol description, but attempts to learn information about the private inputs of the honest parties.

MPC-KMeans, PCA/OPT, and ppDBSCAN consider the outsourced two-party computation setting where multiple data owners secret share their input among two non-colluding servers to privately cluster the dataset. In contrast, in HE-Meanshift, a *single* data owner outsources its computation to a *single* server.

Informally, a protocol is said to be secure if anything that can be computed by a party participating in the protocol can also be derived from the input and output of this party. This is formalized by using a *simulator* which generates a view that is indistinguishable from a real protocol execution given the party's input and output [132]. MPC-KMeans [11] and PCA/OPT [19] provide such a formal proof of security.

The security of HE-Meanshift [9] follows directly from the security of the used CKKS encryption scheme since only the input and final output are sent. We note that the recent attack on the CKKS scheme by Li and Micciancio [133] does not affect the security of HE-Meanshift, as discussed by Cheon et al. [134]. Specifically, the attack requires access to a decryption oracle which is not available to the server in the outsourced single-server computation setting.

Similarly, the security of ppDBSCAN [18] follows directly from the security of the employed secure two-party computation techniques, specifically GC and ASS (cf. §2.2), as no intermediate values are opened and the conversions are provably secure [10].

Leakage from Outputs. Provable security of the protocols ensures that the computation does not leak anything more than what is revealed in an ideal world where a trusted third party obtains the inputs, computes the clustering functionality and returns the output. However, the information leaked from the clustering *output* is not captured in the security definition and we discuss this in the following.

HE-Meanshift outputs the cluster labels for every record in the dataset. However, this is not a privacy concern since the protocol is intended to be used in the outsourced single-server computation setting where the entire dataset is known to the client.

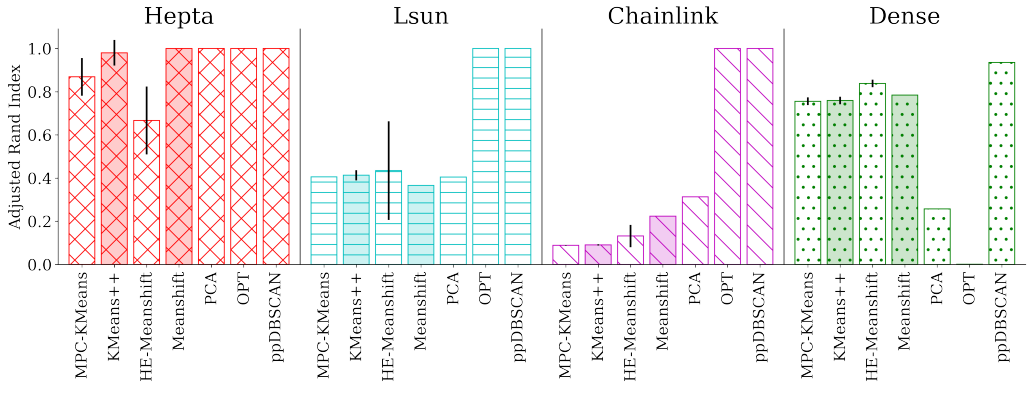
MPC-KMeans and ppDBSCAN can be adapted to output either the cluster centroids or cluster labels. MPC-KMeans also outputs the number of iterations for the clustering to converge which is related to the distribution of the underlying dataset. We have already discussed how to avoid this leakage in §3.2.

The PCA/OPT algorithms output a *point-agnostic dendrogram* in addition to the cluster centroids. The point-agnostic dendrogram is intended to be a privacy-preserving variant of the dendrogram output by a plaintext HC algorithm since the latter provides the complete merging history which leaks information in a setting with multiple data owners. The point-agnostic dendrogram is computed by first applying a random and private permutation on the input records to fuzz the merging history and by retaining the metadata of only sufficiently large clusters. Intuitively, this allows obtaining useful metadata akin to the plaintext computation while still preserving privacy. However, it is unclear how to formalize/measure the information leakage from the protocol output.

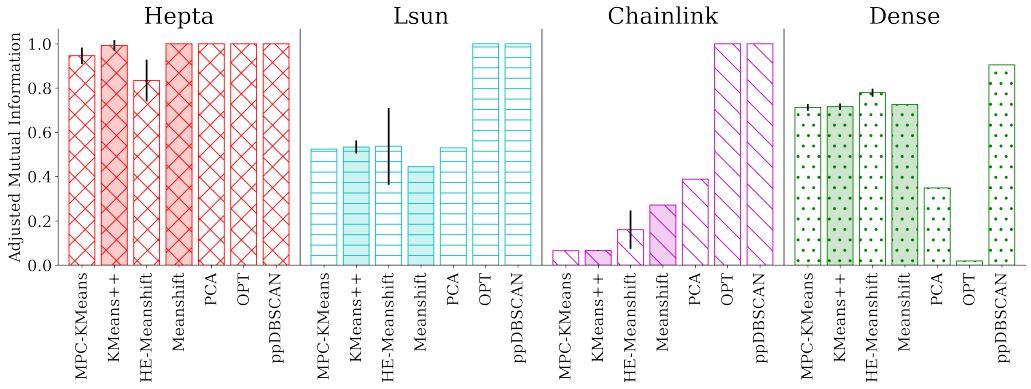
4.3 Efficiency

Asymptotic Analysis. First, we compare the asymptotic runtime, communication, and round complexity of the four investigated private clustering protocols and depict the results in Tab. 5. Asymptotically, MPC-KMeans is the most efficient with respect to communication and runtime in terms of dataset size N, input records' dimension d and number of clusters K.

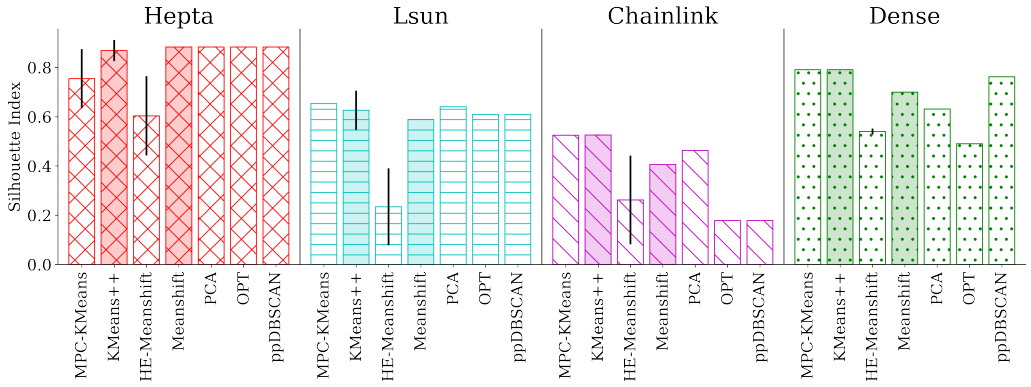
Hardware Details. All experiments are run on two machines (one for each party) each equipped with a 2.8 GHz Intel Core i9-7960X processor running Linux, 32 vCPUs and 128 GB RAM. We consider two network settings. The LAN setting has bandwidth 1 Gbps and



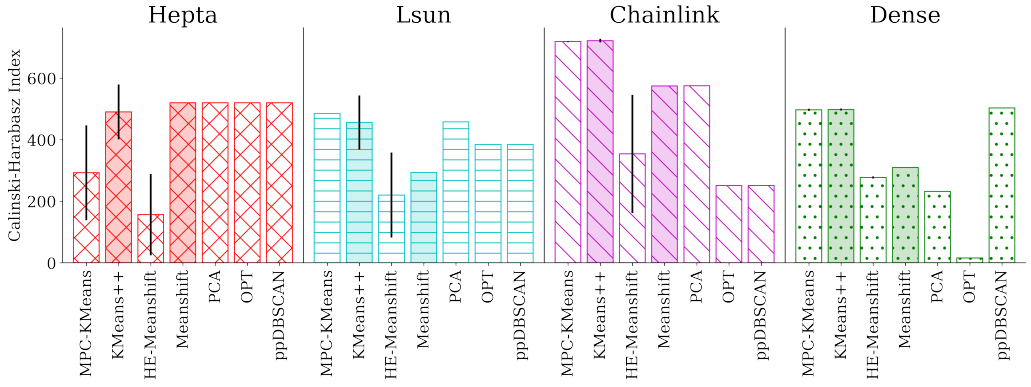
(a) Adjusted Rand Index (ARI)



(b) Adjusted Mutual Information (AMI)



(c) Silhouette Index (SI)



(d) Calinski-Harabasz Index (CHI)

Fig. 1. Clustering quality evaluation of the fully-private clustering protocols MPC-KMeans [11], HE-Meanshift [9], PCA/OPT [19], and ppDBSCAN [18] evaluated on datasets Hepta (red), Lsun (blue), Chainling (purple), and Dense (green). As comparison to the plaintext clustering algorithms (shaded bars), we also include KMeans++ and Mean-shift from Python Scikit-learn [131]. Each subfigure (a-d) corresponds to a different clustering quality index: ARI, AMI, SI, and CHI (cf. §4.1) and larger values indicate better clustering quality. The values are averaged over 10 runs and the error bar shows the standard deviation.

Protocol	Runtime	Communication	Rounds
MPC-KMeans [11]	$\Theta(NK(d+\ell)t)$	$\Theta\left(NK(d\ell^2+\ell\kappa)t\right)$	$\Theta(\lceil\log K\rceil t)$
HE-Meanshift [9]	$\Theta\left((NK_d d^2 t)/(N_c \log d)\right)$	$\Theta(NdK_d\kappa)$	2
PCA [19]	$\Theta(N^3\lambda)$	$\Theta(N^3\lambda\kappa)$	$\Theta(N^2)$
OPT [19]	$\Theta(N^2(\lambda+d))$	$\Theta(N^2(\lambda\kappa+\kappa_{\text{pub}}))$	$\Theta(N^2)$
ppDBSCAN [18]	$\Theta(N^2(N+d))$	$\Theta(N^2\ell\kappa)$	$\mathcal{O}(N^3)$

Table 5. Asymptotic runtime, communication, and round complexity of the private clustering protocols MPC-KMeans [11], HE-Meanshift [9], PCA/OPT [19], and ppDBSCAN [18]. N is the dataset size, d is the dimension, ℓ is the bitlength of the data records, K is the number of clusters, K_d is the number of dusts used in HE-Meanshift, $\kappa = 128$ is the computational security parameter, $\lambda = 40$ is the statistical security parameter, N_c is the number of plaintext slots in CKKS, $\kappa_{\text{pub}} = 2048$ is the size of the RSA modulus in Paillier encryption [65].

RTT 1 ms while the WAN setting has bandwidth 100 Mbps and RTT 100 ms.

Benchmarks. We evaluate the efficiency of the four privacy-preserving clustering protocols and their scalability to large datasets, datasets with many clusters, and multi-dimensional data. We generate synthetic datasets with $N \in \{50, 100, 150, 200, 250\}$ data points of dimension $d \in \{2, 8\}$ and bitlength $\ell = 32$, and number of clusters $K \in \{2, 5\}$. We run the protocols on all possible combinations of the above described parameters. The iterative protocols MPC-KMeans and HE-Meanshift are run for 5 iterations to reach a comparable clustering quality and enabling a fair comparison of their efficiency.

To analyze the scalability of the protocols to large multi-dimensional datasets, we generate a synthetic collection of *large datasets* with parameters $N \in \{2^{13}, 2^{14}, 2^{15}, 2^{16}\}$, $d \in \{1, 2, 4, 8, 16\}$ and $K \in \{2, 5, 10, 15, 20\}$. The memory consumption of MPC-KMeans was too large (greater than 128 GB) to benchmark on datasets where $N \cdot d > 2^{19}$. Similarly, the memory consumption of PCA/OPT [19] and ppDBSCAN [18] was too large even for the smallest dataset with $N = 2^{13}, d = 1$, and $K = 2$ due to the usage of the memory intensive ABY framework [10]. We thus exclude these protocols from our benchmarks on the large datasets. Fig. 4 in Appx. C presents the memory consumption of our implementations of the protocols for a small and large dataset.

Communication. We plot the communication costs in the bottom rows of Fig. 2 (small datasets) and Fig. 5 in Appx. C (large datasets).

The communication cost for HE-Meanshift is identical across different small datasets (Fig. 2) because the entire dataset can be encrypted in one ciphertext. This inefficient packing of the dataset leads to HE-Meanshift's communication being 2× higher than that of MPC-KMeans on average for small datasets. However, for large datasets (Fig. 5), HE-Meanshift's communication cost is up to 11.5× lower than that of MPC-KMeans on average due to optimal packing. The communication cost of PCA is 6× higher than that of ppDBSCAN on average while the communication of ppDBSCAN is 2× higher than that of OPT on average. While the communication cost increases linearly in the input records' dimension d for HE-Meanshift, d does not have a significant effect on the communication of MPC-KMeans since the communication during the assignment of input records to clusters is independent of d. This phase has the highest communication complexity and dominates the overall communication cost. In Fig. 5(f) HE-Meanshift's communication for $K = 10$ and $K = 15$ is identical as both use the same number of dusts $K_d = 16$ and hence send the same number of ciphertexts. The communication costs of PCA/OPT are independent of the input records' dimension d while the communication costs of PCA/OPT and ppDBSCAN are independent of the number of clusters K.

Runtimes. We plot the LAN runtimes in the top row of Fig. 2 (small datasets) and Fig. 5 in Appx. C (large datases), and the WAN runtimes for small datasets in Fig. 3 in Appx. C, all averaged over 10 runs.

MPC-KMeans has the lowest runtime and is up to 700× faster than HE-Meanshift on small datasets and 9.5× faster than HE-Meanshift on large datasets over LAN. Thus, HE-Meanshift scales well to large, multi-dimensional datasets due to lower communication costs, but is not suitable for small datasets. As expected, the runtime of MPC-KMeans in Fig. 5(b) is marginally affected by increasing the dimension of the input records d since its assignment of the input records (which is the bottleneck of the protocol) to the clusters is independent of d. On the other hand, the runtime of HE-Meanshift is linear in d since it directly affects the number of ciphertexts and the efficiency of the bootstrapping operation.

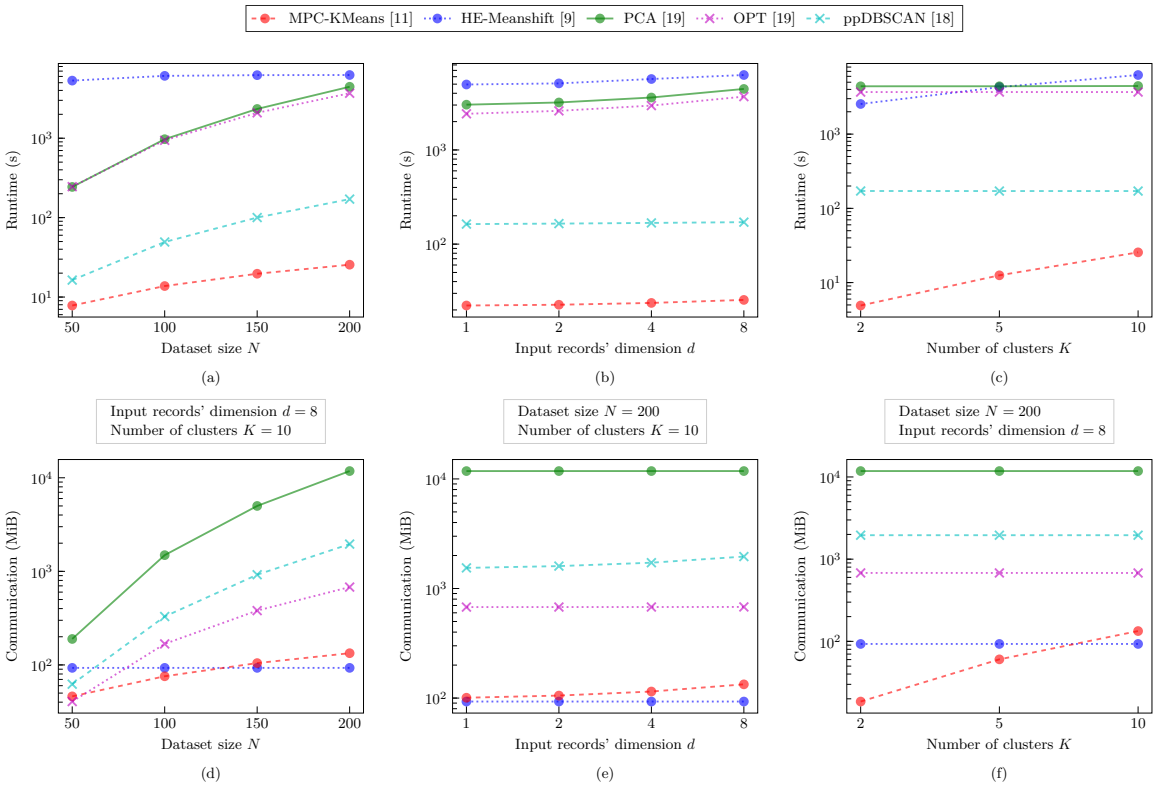


Fig. 2. LAN runtimes in seconds (top row) and **communication** in MiB (bottom row) of the fully-private clustering protocols MPC-KMeans [11], HE-Meanshift [9], PCA/OPT [19], and ppDBSCAN [18] for varying dataset size N, input records' dimension d, number of clusters K, and bitlength $\ell = 32$. In (a) and (d) $d = 8$ and $K = 10$, in (b) and (e) $N = 200$ and $K = 10$, and in (c) and (f) $N = 200$ and $d = 8$.

ppDBSCAN's average runtime is 14× higher over LAN and 2.5× higher over WAN than that of MPC-KMeans. However, since ppDBSCAN's runtime is independent of the number of clusters, this gap in runtime diminishes with the increase in number of clusters. OPT and PCA have similar runtimes wich are 15× higher than that of ppDBSCAN on average over LAN, even though OPT has less communication. Moreover, OPT has the highest runtime on average over WAN. One possible reason for this might be higher concrete computation costs for OPT which is not captured by the asymptotic complexity but highlighted due to benchmarking on small datasets.

5 Real-world Application and Open Challenges

In this section, we discuss the challenges that need to be solved to make privacy-preserving clustering practical for real-world applications.

Parameters. In a privacy-preserving setting, it is typically impossible to perform a preliminary analysis of the data since it is distributed among multiple data owners and not available to a single party. However, to set parameters like the number of clusters K for K-Means (i.e., in MPC-KMeans [11]) and HC (i.e., in PCA/OPT [19]), the distance parameter ϵ for ppDBSCAN [18], or the number of dusts for HE-Meanshift [9], insights about the dataset are often needed to achieve high clustering quality. We also observed in our experimental evaluation that the degree of the kernel and the value of the MinIdx parameter in HE-Meanshift has significant impact on the clustering quality by amplifying the distances between data records. In specific cases, some parameters like the number of clusters K can be given by the application (cf. §3.3). But this is not the case for less intuitive parameters like the initial distribution specifications of GMM or the neighborhood radius in DBSCAN. Only ppDBSCAN [18], out of the four protocols that we benchmarked, can determine these parameters when they are not fixed by the application and inputs are provided by more than one party.

Secure Clustering Quality Evaluation. The issue of dataset-dependent parameters is amplified by the lack of secure clustering quality evaluation techniques. Specifically, plaintext clustering algorithms are often simply run several times with a range of different parameter values and the best output is selected based on the score of a clustering quality index. This is not possible in privacy-preserving clustering. Firstly, as in plaintext clustering, no ground truth is known, so metrics like ARI or AMI that compare to the ground truth cannot be used. Secondly, as the private clustering result is typically split among the data owners or only consists of the centroids, the clusters' compactness and the separation between different clusters cannot be measured without additional secure computation. Thus, also internal indices like SI and CHI cannot be used easily. The inherent overhead of secure computation makes it expensive to perform multiple runs with different parameter values.

Clustering Quality and Efficiency. Clustering algorithms that make minimal assumptions about the shape of clusters and are robust to outliers are especially important in the case of privacy-preserving clustering, because it is impossible to analyze the dataset or remove noisy records before clustering. None of the privacy-preserving clustering protocols we consider investigate soft clustering (cf. §2.1) and only ppDBSCAN has the notion of noise. The K-means-based protocols, i.e., MPC-KMeans, are very sensitive to outliers. Additionally, as shown by our quality evaluation in §4.1 and as discussed in §2.1, K-means only succeeds on clustering convexly shaped clusters which is not the case for all datasets. HE-Meanshift's clustering quality also strongly fluctuates depending on the dataset's properties (cf. §4.1). This is especially problematic for privacy-preserving clustering where the dataset distribution is often not known in advance. In contrast, hierarchical clustering like PCA/OPT is less sensitive to noise and more flexible with respect to the data distribution, i.e., the cluster shapes. However, as shown in §4.3, PCA/OPT cannot be run on large datasets while MPC-KMeans and HE-Meanshift scale significantly better to large datasets. Our evaluation shows that ppDBSCAN performs well on different types of datasets while also having lower runtimes than other protocols (except MPC-KMeans).

Recommendations. Among the protocols we evaluate, MPC-KMeans seems to be the most efficient alternative when clustering large multi-dimensional datasets. HE-Meanshift might be a better choice when a *single* resource-constrained data-owner outsources clustering to a more powerful server over a high-latency and low-bandwidth network. For smaller datasets, ppDBSCAN seems to be the best option which performs well on a variety of dataset types and also achieves low runtimes. However, choosing the input parameter ϵ that determines the maximum distance between two data records to be considered as neighbors requires domain expertise and partial information about the dataset. This can be avoided by using MPC-KMeans which only requires setting the more intuitive number of clusters K which in some cases is also given by the application (cf. §3.3).

Open Challenges. To summarize, for practical application, privacy-preserving clustering protocols must be (1) efficient in terms of runtime and communication, (2) memory efficient, (3) only have parameters that are mostly independent of the input data, (4) insensitive to noise, and (5) flexible to cluster data of any distribution with high quality. Unfortunately, none of the state-of-the-art works can fulfill all these requirements simultaneously. Additionally, there is the need for a secure clustering quality evaluation to assess the quality of a clustering result run in a privacy-preserving manner. Finally, privacy-research has not tackled privacy-preserving soft clustering.

6 Conclusion

In this work, we systematically surveyed and analyzed the state-of-the-art in privacy-preserving clustering. We benchmarked and compared four efficient protocols [9, 11, 18, 19] that securely realize four different clustering algorithms, with respect to clustering quality, communication, and runtime to investigate their practicality for real-world applications. Finally, we discussed open challenges to make privacy-preserving clustering practical.

ACKNOWLEDGEMENTS

We thank Oliver Schick for his support with implementing PCA/OPT [19]. This project received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement No. 850990 PSOTI). It was co-funded by the Deutsche Forschungsgemeinschaft (DFG) – SFB 1119 CROSSING/236615297 and GRK 2050 Privacy & Trust/251805230, and by the BMBF and the HMWK within ATHENE.

References

[1] Z. Qian, Z. M. Mao, Y. Xie, and F. Yu, "On Network-level Clusters for Spam Detection." in *NDSS*, 2010.

[2] K. Kourou, T. P. Exarchos, K. P. Exarchos, M. V. Karamouzis, and D. I. Fotiadis, "Machine learning applications in cancer prognosis and prediction," *Computational and Structural Biotechnology Journal*, 2015.

[3] M. Ahmed, A. N. Mahmood, and M. R. Islam, "A Survey of Anomaly Detection Techniques in Financial Domain," in *Future Generation Computer Systems*, 2016.

[4] F. Masulli and A. Schenone, "A fuzzy clustering based segmentation system as support to diagnosis in medical imaging," *Artificial Intelligence in Medicine*, 1999.

[5] S. Gauch, M. Speretta, A. Chandramouli, and A. Micarelli, "User profiles for personalized information access," in *The adaptive web*, 2007.

[6] A. Chaturvedi, J. D. Carroll, P. E. Green, and J. A. Rotondo, "A feature-based approach to market segmentation via overlapping k-centroids clustering," *Journal of Marketing Research*, 1997.

[7] C. Gentry and D. Boneh, *A fully homomorphic encryption scheme*. Stanford university Stanford, 2009.

[8] W. Wu, J. Liu, H. Wang, J. Hao, and M. Xian, "Secure and efficient outsourced K-means clustering using fully homomorphic encryption with ciphertext packing technique," in *TDKE*, 2020.

[9] J. H. Cheon, D. Kim, and J. H. Park, "Towards a practical cluster analysis over encrypted data," in *SAC*, 2019.

[10] D. Demmler, T. Schneider, and M. Zohner, "ABY - A framework for efficient mixed-protocol secure two-party computation," in *NDSS*, 2015.

[11] P. Mohassel, M. Rosulek, and N. Trieu, "Practical privacy-preserving K-means clustering," in *PETS*, 2020.

[12] P. Bunn and R. Ostrovsky, "Secure two-party K-means clustering," in *CCS*, 2007.

[13] F.-Y. Rao, B. K. Samanthula, E. Bertino, X. Yi, and D. Liu, "Privacy-preserving and outsourced multi-user K-means clustering," in *CIC*, 2015.

[14] A. Jäschke and F. Armknecht, "Unsupervised Machine Learning on Encrypted Data," in *SAC*, 2018.

[15] H. Kim and J. Chang, "A privacy-preserving k-means clustering algorithm using secure comparison protocol and density-based center point selection," in *International Conference on Cloud Computing*, 2018.

[16] H. Keller, H. Möllering, T. Schneider, and H. Yalame, "Balancing quality and efficiency in private clustering with affinity propagation," in *SECRYPT*, 2021.

[17] S. Zahur and D. Evans, "Circuit structures for improving efficiency of security and privacy tools," in *IEEE S&P*, 2013.

[18] B. Bozdemir, S. Canard, O. Ermis, H. Möllering, M. Önen, and T. Schneider, "Privacy-preserving density-based clustering," in *ASIACCS*, 2021.

[19] X. Meng, D. Papadopoulos, A. Oprea, and N. Triandopoulos, "Private two-party cluster analysis made formal & scalable," *arXiv:1904.04475v2*, 2019.

[20] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game," in *STOC*, 1987.

[21] A. C.-C. Yao, "How to generate and exchange secrets," in *FOCS*, 1986.

[22] J. Liu, L. Xiong, J. Luo, and J. Z. Huang, "Privacy preserving distributed DBSCAN clustering," in *Transactions on Data Privacy*, 2013.

[23] N. Kumar, M. Rathee, N. Chandran, D. Gupta, A. Rastogi, and R. Sharma, "CrypTFlow: Secure TensorFlow inference," in *IEEE S&P*, 2020.

[24] D. Rathee, M. Rathee, N. Kumar, N. Chandran, D. Gupta, A. Rastogi, and R. Sharma, "CrypTFlow2: Practical 2-party secure inference," in *CCS*, 2020.

[25] P. Mishra, R. Lehmkuhl, A. Srinivasan, W. Zheng, and R. A. Popa, "Delphi: A cryptographic inference service for neural networks," in *USENIX Security*, 2020.

[26] A. Patra, T. Schneider, A. Suresh, and H. Yalame, "ABY2.0: Improved mixed-protocol secure two-party computation," in *USENIX Security*, 2021.

[27] V. Haralampieva, D. Rueckert, and J. Passerat-Palmbach, "A systematic comparison of encrypted machine learning solutions for image classification," in *PPMLP*, 2020.

[28] F. Boemer, R. Cammarota, D. Demmler, T. Schneider, and H. Yalame, "MP2ML: A mixed-protocol machine learning framework for private inference," in *ARES*, 2020.

[29] L. Song, H. Wu, W. Ruan, and W. Han, "SoK: Training machine learning models over multiple sources with privacy preservation," in *arXiv:2012.03386*, 2020.

[30] H. C. Tanuwidjaja, R. Choi, S. Baek, and K. Kim, "Privacy-preserving deep learning on machine learning as a service—a comprehensive survey," in *IEEE Access*, 2020.

[31] Á. Kiss, M. Naderpour, J. Liu, N. Asokan, and T. Schneider, "SoK: modular and efficient private decision tree evaluation," in *PETS*, 2019.

[32] U. Stemmer, "Locally private K-means clustering," in *ACM-SIAM Symposium on Discrete Algorithms*, 2020.

[33] L. Ni, C. Li, X. Wang, H. Jiang, and J. Yu, "DP-MCDBSCAN: Differential privacy preserving multi-core DBSCAN clustering for network user data," in *IEEE Access*, 2018.

[34] M.-F. Balcan, T. Dick, Y. Liang, W. Mou, and H. Zhang, "Differentially private clustering in high-dimensional Euclidean spaces," in *ICML*, 2017.

[35] D. Su, J. Cao, N. Li, E. Bertino, M. Lyu, and H. Jin, "Differentially private K-means clustering and a hybrid approach to private optimization," in *TOPS*, 2017.

[36] D. Su, J. Cao, N. Li, E. Bertino, and H. Jin, "Differentially private K-means clustering," in *Data and Application Security and Privacy*, 2016.

[37] W. Wu and H. Huang, "A DP-DBSCAN clustering algorithm based on differential privacy preserving," in *Computer Engineering and Science*, 2015.

[38] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *CCS*, 2016.

[39] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *CCS*, 2015.

[40] D. Xu and Y. Tian, "A comprehensive survey of clustering algorithms," in *Annals of Data Science*, 2015.

[41] R. Xu and D. Wunsch, "Survey of clustering algorithms," in *TNN*, 2005.

[42] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: A review," in *ACM Computing Surveys*, 1999.

[43] Q. Zhang, L. T. Yang, Z. Chen, and P. Li, "PPHOPCM: privacy-preserving high-order possibilistic c-means algorithm for big data clustering with cloud computing," *IEEE Transactions on Big Data*, 2017.

[44] M. Hamidi, M. Sheikhalishahi, and F. Martinelli, "Privacy preserving Expectation Maximization (EM) clustering construction," in *DCAI*, 2019.

[45] X. Lin, C. Clifton, and M. Zhu, "Privacy-preserving clustering with distributed EM mixture modeling," in *Knowledge and Information Systems*, 2005.

[46] I. V. Anikin and R. M. Gazimov, "Privacy preserving DBSCAN clustering algorithm for vertically partitioned data in distributed systems," in *International Siberian Conference on Control and Communications*, 2017.

[47] M. S. Rahman, A. Basu, and S. Kiyomoto, "Towards outsourced privacy-preserving multiparty DBSCAN," in *PRDC*, 2017.

[48] I. De and A. Tripathy, "A secure two party hierarchical clustering approach for vertically partitioned data set with accuracy measure," in *Recent Advances in Intelligent Informatics*, 2014.

[49] G. Jagannathan, K. Pillaipakkamnatt, R. Wright, and D. Umano, "Communication-efficient privacy-preserving clustering," in *Transactions on Data Privacy*, 2010.

[50] A. İnan, S. V. Kaya, Y. Saygın, E. Savaş, A. A. Hintoğlu, and A. Levi, "Privacy preserving clustering on horizontally partitioned data," in *TDKE*, 2007.

[51] H. Steinhaus, "Sur la division des corp materiels en parties," in *Bulletin L'Académie Polonaise des Science*, 1956.

[52] B. S. Everitt, S. Landau, M. Leese, and D. Stahl, "Cluster analysis," in *Wiley*, 2011.

[53] K. Fukunaga and L. Hostetler, "The estimation of the gradient of a density function, with applications in pattern recognition," in *TIT*, 1975.

[54] X. Xu, M. Ester, H.-P. Kriegel, and J. Sander, "A distribution-based clustering algorithm for mining in large spatial databases," in *ICDE*, 1998.

[55] J. M. Pena, J. A. Lozano, and P. Larranaga, "An empirical comparison of four initialization methods for the K-means algorithm," in *Pattern Recognition Letters*, 1999.

[56] Zhexue Huang and M. K. Ng, "A fuzzy k-modes algorithm for clustering categorical data," in *TFS*, 1999.

[57] J. Zhan, "Privacy preserving K-medoids clustering," in *SMC*, 2007.

[58] K.-P. Lin, "Privacy-preserving kernel K-means clustering outsourcing with random transformation," *Knowledge and Information Systems*, 2016.

[59] A. K. Jain and R. C. Dubes, *Algorithms for clustering data*. Prentice-Hall, 1988.

[60] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in Large spatial databases with noise." in *SIGKDD*, 1996.

[61] Y. Ren, C. Domeniconi, G. Zhang, and G. Yu, "A weighted adaptive mean shift clustering algorithm."

[62] M. Ester, "Density-based clustering," in *Encyclopedia of Database Systems*. Springer, 2009.

[63] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "OPTICS: ordering points to identify the clustering structure," in *ACM SIGMOD*, 1999.

[64] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," in *Journal of the Royal Statistical Society*, 1977.

[65] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *EUROCRYPT*, 1999.

[66] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers," in *ASIACRYPT*, 2017.

[67] A. Shamir, "How to share a secret," in *Communication of the ACM*, 1979.

[68] O. Goldreich, *Foundations of cryptography: volume 2, basic applications*. Cambridge university press, 2009.

[69] P. Mohassel and Y. Zhang, "SecureML: A system for scalable privacy-preserving machine learning," in *IEEE S&P*, 2017.

[70] S. Kamara and M. Raykova, "Secure outsourced computation in a multi-tenant cloud," in *IBM Workshop on Cryptography and Security in Clouds*, 2011.

[71] S. K. Dash, D. P. Mishra, R. Mishra, and S. Dash, "Privacy preserving K-medoids clustering: An approach towards securing data in mobile cloud architecture," in *Conference on Computational Science, Engineering and Information Technology*, 2012.

[72] A. Amirbekyan and V. Estivill-Castro, "Privacy preserving DBSCAN for vertically partitioned data," in *Intelligence and Security Informatics*, 2006.

[73] K. A. Kumar and C. P. Rangan, "Privacy preserving DBSCAN algorithm for clustering," in *Advanced Data Mining and Applications*, 2007.

[74] W.-j. Xu, L.-s. Huang, Y.-l. Luo, Y.-f. Yao, and W. Jing, "Protocols for privacy-preserving DBSCAN clustering," in *International Journal of Security and Its Applications*, 2007.

[75] D. Jiang, A. Xue, S. Ju, W. Chen, and H. Ma, "Privacy-preserving DBSCAN on horizontally partitioned data," in *International Symposium on IT in Medicine and Education*, 2008.

[76] N. Almutairi, F. Coenen, and K. Dures, "Secure third party data clustering using ϕ data: Multi-user order preserving encryption and super secure chain distance matrices," in *International Conference on Innovative Techniques and Applications of Artificial Intelligence*, 2018.

[77] G. Jagannathan, K. Pillaipakkamnatt, and R. N. Wright, "A new privacy-preserving distributed K-clustering algorithm," in *SDM*, 2006.

[78] M. Sheikhalishahi and F. Martinelli, "Privacy preserving clustering over horizontal and vertical partitioned data," in *Symposium on Computers and Communications*, 2017.

[79] P. K. Prasad and C. P. Rangan, "Privacy preserving birch algorithm for clustering over vertically partitioned databases," in *Workshop on Secure Data Management*, 2006.

[80] K. Prasad and P. Rangan, "Privacy preserving birch algorithm for clustering over arbitrarily partitioned databases," *ADMA*, 2007.

[81] X. Zhu, M. Liu, and M. Xie, "Privacy-preserving affinity propagation clustering over vertically partitioned data," in *International Conference on Intelligent Networking and Collaborative Systems*, 2012.

[82] J. Vaidya and C. Clifton, "Privacy-preserving K-means clustering over vertically partitioned data," in *SIGKDD*, 2003.

[83] G. Jagannathan and R. N. Wright, "Privacy-preserving distributed K-means clustering over arbitrarily partitioned data," in *SIGKDD*, 2005.

[84] S. Jha, L. Kruger, and P. McDaniel, "Privacy preserving clustering," in *ESORICS*, 2005.

[85] S. Samet, A. Miri, and L. Orozco-Barbosa, "Privacy preserving K-means clustering in multi-party environment," in *SECRYPT*, 2007.

[86] C. Su, F. Bao, J. Zhou, T. Takagi, and K. Sakurai, "Privacy-preserving two-party K-means clustering via secure approximation," in *AINA*, 2007.

[87] M. C. Doganay, T. B. Pedersen, Y. Saygin, E. Savaş, and A. Levi, "Distributed privacy preserving K-means clustering with additive secret sharing," in *International Workshop on Privacy and Anonymity in Information Society*, 2008.

[88] Z. Erkin, T. Veugen, T. Toft, and R. L. Lagendijk, "Privacy-preserving user clustering in a social network," in *Information Forensics and Security*, 2009.

[89] J. Sakuma and S. Kobayashi, "Large-scale k-means clustering with user-centric privacy-preservation," in *Knowledge and Information Systems*, 2010.

[90] M. Upmanyu, A. M. Namboodiri, K. Srinathan, and C. V. Jawahar, "Efficient privacy preserving K-means clustering," in *Pacific-Asia Workshop on Intelligence and Security Informatics*, 2010.

[91] T.-K. Yu, D. Lee, S.-M. Chang, and J. Zhan, "Multi-party K-means clustering with privacy consideration," in *ISPA*, 2010.

[92] M. Beye, Z. Erkin, and R. L. Lagendijk, "Efficient privacy preserving K-means clustering in a three-party setting," in *Information Forensics and Security*, 2011.

[93] Z. Lin and J. W. Jaromczyk, "Privacy preserving two-party K-means clustering over vertically partitioned dataset," in *ISI*, 2011.

[94] S. Patel, S. Garasia, and D. Jinwala, "An efficient approach for privacy preserving distributed K-means clustering based on shamir's secret sharing scheme," in *Trust Management VI*, 2012.

[95] Z. Erkin, T. Veugen, T. Toft, and R. L. Lagendijk, "Privacy-preserving distributed clustering," in *EURASIP Journal on Information Security*, 2013.

[96] S. Patel, V. Patel, and D. Jinwala, "Privacy preserving distributed K-means clustering in malicious model using zero knowledge proof," in *Distributed Computing and Internet Technology*, 2013.

[97] D. Liu, E. Bertino, and X. Yi, "Privacy of outsourced K-means clustering," in *ASIACCS*, 2014.

[98] X. Liu, Z. L. Jiang, S. M. Yiu, X. Wang, C. Tan, Y. Li, Z. Liu, Y. Jin, and J. Fang, "Outsourcing two-party privacy preserving K-means clustering protocol in wireless sensor networks," in *MSN*, 2015.

[99] S. J. Patel, D. Punjani, and D. C. Jinwala, "An efficient approach for privacy preserving distributed clustering in semi-honest model using elliptic curve cryptography," *International Journal of Network Security*, 2015.

[100] V. Baby and N. S. Chandra, "Distributed threshold K-means clustering for privacy preserving data mining," in *ICACCI*, 2016.

[101] Z. Gheid and Y. Challal, "Efficient and privacy-preserving K-means clustering for big data mining," in *IEEE TrustCom/BigDataSE/ISPA*, 2016.

[102] H. Rong, H. Wang, J. Liu, J. Hao, and M. Xian, "Outsourced k-means clustering over encrypted data under multiple keys in spark framework," in *Security and Privacy in Communication Networks*, 2017.

[103] K. Xing, C. Hu, J. Yu, X. Cheng, and F. Zhang, "Mutual privacy preserving K-means clustering in social participatory sensing," in *TII*, 2017.

[104] J. Yuan and Y. Tian, "Practical privacy-preserving MapReduce based K-means clustering over large-ccale dataset," in *TCM*, 2019.

[105] Z. L. Jiang, N. Guo, Y. Jin, J. Lv, Y. Wu, Z. Liu, J. Fang, S. Yiu, and X. Wang, "Efficient two-party privacy-preserving collaborative k-means clustering protocol supporting both storage and computation outsourcing," *Information Sciences*, 2020.

[106] Y. Zou, Z. Zhao, S. Shi, L. Wang, Y. Peng, Y. Ping, and B. Wang, "Highly secure privacy-preserving outsourced k-means clustering under multiple keys in cloud computing," in *Security and Communication Networks*, 2020.

[107] Y. Wang, "Notes on two fully homomorphic encryption schemes without bootstrapping." Cryptology ePrint Archive, Report 2015/519.

[108] Y. Cai and C. Tang, "Privacy of outsourced two-party k-means clustering," *Concurrency and Computation: Practice and Experience*, 2019.

[109] S. M. Sarmento and N. Horta, "Enhancing a pairs trading strategy with the application of machine learning," in *Expert Systems with Applications*, 2020.

[110] R. Adusumilli, "DBSCAN Clustering for Trading," 2020, https://towardsdatascience.com/dbscan-clustering-for-trading-4c48e5ebffc8.

[111] S. Panigrahi, A. Kundu, S. Sural, and A. K. Majumdar, "Credit card fraud detection: A fusion approach using dempster–shafer theory and bayesian learning," in *Information Fusion*, 2009.

[112] A. Sangers, M. van Heesch, T. Attema, T. Veugen, M. Wiggerman, J. Veldsink, O. Bloemen, and D. Worm, "Secure Multiparty PageRank Algorithm for Collaborative Fraud Detection," in *FC*, 2019.

[113] A. Chaturvedi, J. Carroll, P. Green, and J. A. Rotondo, "A feature-based approach to market segmentation via overlapping k-centroids clustering," *Journal of Marketing Research*, 1997.

[114] Y. S. Cho, S. C. Moon, S. C. Noh, and K. H. Ryu, "Implementation of personalized recommendation system using k-means clustering of item category based on rfm," in *ICMIT*, 2012.

[115] Q. Guo, X. Lu, Y. Gao, J. Zhang, B. Yan, D. Su, A. Song, X. Zhao, and G. Wang, "Cluster Analysis: A New Approach for Identification of Underlying Risk Factors for Coronary Artery Disease in Essential Hypertensive Patients," in *Scientific Reports*, 2017.

[116] F. Masulli and A. Schenone, "A fuzzy clustering based segmentation system as support to diagnosis in medical imaging," *Artificial Intelligence in Medicine*, 1999.

[117] L. Braun, D. Demmler, T. Schneider, and O. Tkachenko, "MOTION - A framework for mixed-protocol multi-party computation," Cryptology ePrint Archive, Report 2020/1137.
[118] M. Keller, "MP-SPDZ: a versatile framework for multi-party computation," in *CCS*, 2020.
[119] A. Dalskov, D. Escudero, and M. Keller, "Secure evaluation of quantized neural networks," *PETS*, 2020.
[120] "HEAAN," https://github.com/snucrypto/HEAAN, 2020.
[121] "Paillier library," http://acsc.cs.utexas.edu/libpaillier, 2010.
[122] A. Ultsch, "Clustering with SOM," in *Workshop on Self-Organizing Maps*, 2005.
[123] D. Graves and W. Pedrycz, "Kernel-based fuzzy clustering and fuzzy clustering: A comparative experimental study," in *Fuzzy Sets and Systems*, 2010.
[124] M. Gagolewski, "Benchmark suite for clustering algorithms version 1," 2020, https://github.com/gagolews/clustering_benchmarks_v1.
[125] O. Arbelaitz, I. Gurrutxaga, J. Muguerza, J. M. Pérez, and I. Perona, "An extensive comparative study of cluster validity indices," *Pattern Recognition*, 2013.
[126] L. Hubert and P. Arabie, "Comparing partitions," *Journal of Classification*, 1985.
[127] N. X. Vinh, J. Epps, and J. Bailey, "Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance," *Journal of Machine Learning Research*, 2010.
[128] P. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics*, 1987.
[129] T. Caliński and J. Harabasz, "A dendrite method for cluster analysis," in *Communications in Statistics-theory and Methods*, 1974.
[130] D. Arthur and S. Vassilvitskii, "K-means++: The advantages of careful seeding," *ACM-SIAM Symposium on Discrete Algorithms*, 2007.
[131] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in Python," *JMLR*, 2011.
[132] Y. Lindell, "How to simulate it–a tutorial on the simulation proof technique," *Tutorials on the Foundations of Cryptography*, 2017.
[133] B. Li and D. Micciancio, "On the security of homomorphic encryption on approximate numbers," Cryptology ePrint Archive, Report 2020/1533.
[134] J. H. Cheon, S. Hong, and D. Kim, "Remark on the security of CKKS scheme in practice," Cryptology ePrint Archive, Report 2020/1581.
[135] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," *Science*, 2007.
[136] X. Liu, M. Yin, J. Luo, and W. Chen, "An improved affinity propagation clustering algorithm for large-scale data sets," in *International Conference on Natural Computation*, 2013.
[137] F. Shang, L. Jiao, J. Shi, F. Wang, and M. Gong, "Fast affinity propagation clustering: A multilevel approach," *Pattern Recognition*, 2012.
[138] D. Dueck, *Affinity propagation: clustering data by passing messages*, 2009.
[139] A. Rodriguez and A. Laio, "Clustering by fast search and find of density peaks," *Science*, 2014.
[140] T. Zhang, R. Ramakrishnan, and M. Livny, "Birch: An efficient data clustering method for very large databases," *ACM SIGMOD*, 1996.
[141] C. E. Rasmussen *et al.*, "The infinite Gaussian mixture model," in *NIPS*, 1999.
[142] X. He, D. Cai, Y. Shao, H. Bao, and J. Han, "Laplacian regularized gaussian mixture model for data clustering," *IEEE Transactions on Knowledge and Data Engineering*, 2010.
[143] J. P. Patist, W. Kowalczyk, and E. Marchiori, "Maintaining gaussian mixture models of data streams under block evolution," in *International Conference on Computational Science*, 2006.
[144] R. C. Pinto and P. M. Engel, "A fast incremental gaussian mixture model," *PloS one*, 2015.
[145] J. H. Cheon, K. Han, A. Kim, M. Kim, and Y. Song, "Bootstrapping for approximate homomorphic encryption," in *EUROCRYPT*, 2018.
[146] J. H. Cheon, D. Kim, D. Kim, H. H. Lee, and K. Lee, "Numerical method for comparison on homomorphically encrypted numbers," in *ASIACRYPT*, 2019.

A Discussion of Strengths and Weaknesses of Additional Clustering Algorithms

Affinity Propagation. Affinity Propagation [135] is a deterministic partitioning-based clustering algorithm that has a computational complexity of $\mathcal{O}(N^2t)$ and space complexity of $\mathcal{O}(N^2)$, where N is the dataset size and t the number of clustering iterations [136]. It flexibly determines the required number of clusters based on the input data such that outliers that do not fit into any cluster form a a cluster on their own [135]. The clustering result is independent of the input order of the data records, as Affinity Propagation always iterates through the complete dataset in each iteration. Affinity Propagation requires the input of a *preference* value for each input record that indicates its likelihood to be chosen as exemplar (similar to a centroid in K-means) of a cluster. If the preference values are not well chosen, it can lead to suboptimal clustering results [137]. If all records are equally likely the preference values are set to the same value for all records, e.g., the median or minimum of the distances [135]. The respective distance measure can be freely chosen, thus, also any variable type could be clustered [138]. On the downside, Affinity Propagation can only detect spherical clusters [139] and a re-clustering is needed if new data records are added to the input dataset after it has already been clustered, as this changes the responsibility and availability matrices.

BIRCH. Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) is a divisive HC algorithm [140]. Its computation and space complexity is linear in the dataset size [40]. Generally, BIRCH is relatively insensitive to noisy elements as it allows to remove elements in sparse regions [40, 140]. Because

each input record is processed incrementally and inserted into the subtree representing the assigned closest cluster, BIRCH can handle new data records well, but is affected by the input order [40, 140]. Moreover, it can only detect convexly shaped clusters with records with metric attributes [40, 140]. Additionally, BIRCH requires to input a threshold for the maximal cluster size and the branching factor of the tree. If the number of cluster K is not given, all sub-clusters in the tree are returned.

GMM. Gaussian Mixture Models (GMM) Clustering is a distribution-based clustering algorithm that uses the Expectation-Maximization (EM) algorithm [64, 141]. GMM has a computational complexity of $\mathcal{O}(NKd^3t)$, where N is the dataset size, K is the number of cluster, d is the data dimension, and t the number of clustering iterations and its space complexity is also linear in N [142–144]. The assumption of a Gaussian distribution of cluster elements restricts the type of the variables to real numbers. GMM fails when clusters have specific constellations, e.g., when one cluster is surrounded by another one, or if the clusters are not convexly shaped. It takes the number of clusters K as input and is relatively sensitive to the selection of the initial parameters of the cluster distributions [41]. Although GMM does not explicitly acknowledge the notion of noise, its result is relatively insensitive to outliers [40, 54], but Keller et al. [16] demonstrate that outliers can still cause significant misclassifcations and incorrect merges between different clusters. As it process the whole data set in each iteration, GMM is not affected by the input order. A few new data records can be clustered by GMM with only a few additional iterations.

B Summaries of Fully-Private Clustering Protocols

MPC-KMeans [11]. Mohassel et al. [11] propose a secure two-party K-means (cf. §2.1) protocol in the semi-honest security model using the ABY framework [10] for secure two-party computation. We call this protocol MPC-KMeans in the following. MPC-KMeans can also be used in an outsourcing scenario [70] where multiple data owners outsource the clustering to two non-colluding servers. The authors also propose a multi-party variant where parties first locally run the plaintext K-means on their local datasets and then proceed to securely compute the joint clustering result based on the previously determined local centroids of all parties.

In the two-party protocol, each data owner runs the plaintext K-means algorithm on its local datasets to compute $\frac{K}{2}$ local clusters. The centroids of these clusters are then secret-shared and used to initialize the centroids for the clustering over the combined dataset.

MPC-KMeans' building blocks are optimized for two computational settings: the amortized setting where the same function is evaluated multiple times on different inputs and the adaptive setting where the inputs to multiple evaluations of the function depend on the output of previous evaluations. Intuitively, the updates of the centroids are non-adaptive in one clustering iteration but adaptive across several iterations. Therefore, the authors introduce efficient protocols for secure multiplication and the calculation of the squared Euclidean distance in the adaptive amortized setting. They also propose an efficient protocol for computing the index of the minimum element in a list of t values using a recursive tree evaluation of a customized Garbled Circuit. This increases the number of rounds by $\lceil \log t \rceil$, but it reduces the communication costs by a factor of 2. MPC-KMeans terminates when the difference between new and old centroids is less than a predefined threshold.

The authors use the squared Euclidean distance. They also benchmark the Manhattan distance ($\max_{i \in [1,d]} |x_i - y_i|$, where d is the dimension) and Chessboard distance ($\sum_{i=1}^{d} |x_i - y_i|$), but show that computing squared Euclidean distance in the adaptive amortized setting is faster than the other two distances. The computation is done on fixed point numbers using the truncation method of [69] where each party locally truncates its share with an error of at most one bit in the least significant bit of the fractional part. The authors show that truncation has a negligible impact on the accuracy of clustering.

HE-Meanshift [9]. Cheon et al. [9] propose a HE-friendly variant of the Mean-shift clustering algorithm (cf. §2.1) in the semi-honest security model using the fully homomorphic encryption (FHE) scheme CKKS [66, 145]. We call this protocol HE-Meanshift in the following. The protocol is designed for the outsourced computation setting where a single, possibly resource-constrained, data owner securely outsources the computation to a server.

CKKS computes on real numbers, but it supports only addition and multiplication. Thus, HE-Meanshift replaces the non-polynomial operations in Mean-shift by polynomial operations. The gradient ascent algorithm used in Mean-shift for mode-seeking requires computing the derivative of the kernel. The authors of [9] propose the HE-friendly polynomial kernel G_{he} shown in Eq. 1. Given the degree parameter $\Gamma \in \mathbb{N}$, the derivative of G_{he} can be computed with a constant multiplicative factor using $\Gamma + 1$ multiplications and 2 subtraction operations.

$$G_{\text{he}}(\boldsymbol{x}, \boldsymbol{y}) = (1 - \|\boldsymbol{x} - \boldsymbol{y}\|^2)^{2^{\Gamma}+1}. \tag{1}$$

HE-Meanshift uses a fixed bandwidth parameter $h = 1$ in the kernel density estimator (KDE) used to compute the density function in Mean-shift. h is a smoothing parameter for the density function and it is the only parameter for the classical Mean-shift algorithm. Although h is fixed in HE-Meanshift, the Γ parameter still offers some flexibility by amplifying the distance between the data points.

Due to the computation overhead of FHE, the authors adopt a random sampling strategy called *dust sampling* which involves sampling K_d points called dusts from the dataset to reduce the $\mathcal{O}(N^2)$ complexity of the original Mean-shift algorithm. HE-Meanshift then performs the mode-seeking on the dusts instead on all points in the dataset. Recall that modes are points corresponding to local maxima in the KDE and represent areas of high density. In Mean-shift, each point is mapped to the cluster containing the closest mode and the number of clusters is equal to the number of distinct modes. Thus, HE-Meanshift can use a relatively low value for K_d that is at least equal to the number of clusters K to compute all clusters in the dataset. This not only improves the efficiency of the mode-seeking but also reduces the costs for bootstrapping, which is now proportional to K_d. However, setting K_d requires prior information about the

number of clusters in the dataset in contrast to the plaintext Mean-shift where only a value for h is needed.

After a predefined number of iterations, each point in the input dataset is assigned a cluster label based on the final value of the dusts. Since two or more sampled dusts might converge to the same mode and hence the same cluster, HE-Meanshift uses a secure PointLabeling algorithm to robustly assign records to clusters. The Inv and MaxIdx protocols of [146] are used for division and comparison.

HE-Meanshift does not require communication except from sending and receiving the data to/from the untrusted processing party. It is usable for a single data owner outsourcing the clustering. However, the protocol is not usable for most outsourcing scenarios where multiple data owners cluster their joint data since the encrypted output can be decrypted only by a single data owner.

PCA/OPT [19]. Meng et al. [19] introduce two-party privacy-preserving hierarchical clustering (HC) protocols with single and complete linkage (cf. §2.1) in the semi-honest security model using additively homomorphic encryption [65] and Yao's Garbled Circuits [21]. In contrast to the protocols discussed before in §3, they do not return the resulting clusters/its indices, but a dendrogram (cf. §2.1) indicating the clustering's merging history and metadata containing statistical information about each merge like the new cluster's size and a representative element/centroid. To limit information leakage through this returned metadata, the protocols output only metadata of sufficiently large merges or of the final clusters.

In the baseline protocol, called *PCA*, the two parties calculate the pairwise squared Euclidean distances between the clusters using the additively homomorphic property of Paillier encryption [65]. Then, both parties get access to the plaintext values of the distance matrix blinded with random values such that they can collaboratively cluster the input elements and update the merging dendrogram leveraging two GC-protocols that determine the minimum/maximum distance.

The authors introduce an extension of *PCA* called *OPT* that reduces HC's computation complexity of $O(N^3)$, where N is the dataset size, with single linkage by leveraging the symmetry of the minimum distance. This accelerates the search for the next pair of clusters that have to be merged by a factor of N.

PCA can also be extended to the outsourcing scenario [70] where an arbitrary number of data owners secret share their input data among two non-colluding servers that run the privacy-preserving clustering. However, an extension to more than two servers is not straightforward due to the usage of GCs.

ppDBSCAN [18]. Bozdemir et al. [18] propose a privacy-preserving DBSCAN [60] protocol in the semi-honest security setting using the ABY framework [10]. We call this protocol ppDBSCAN in the following. ppDBSCAN can either be used as secure two-party computation protocol or in an outsourcing scenario with two computing parties, e.g., servers, and an arbitrary number of data owners. The authors point out that the post-processing can be adapted to provide an arbitrary output, e.g., cluster labels, cluster sizes, etc. By assessing the needed recursive depth of the neighborhood exploration ppDBSCAN's complexity can be reduced to a low cubic complexity (from normal cubic complexity). All computations are done on integers.

Initially, the data owners arithmetically share their input records among the two non-colluding parties (which are potentially represented by themselves). Then, the pair-wise squared Euclidean distances are computed between all data records in Arithmetic Sharing [20] to assess which elements have sufficiently many neighbors (i.e., lie in a dense area) to form a cluster. The results are stored as binary values to enhance the the efficiency of the clustering process mostly done with GC [21]. Additionally, the distance computation and the cluster expansion is also parallelized with SIMD operations.

C Additional Benchmarking Results

Fig. 3 summarizes the WAN runtimes of the fully private clustering protocols on small datasets. Fig. 4 depicts the memory consumption of the fully private clustering protocols for a small and large dataset. Fig. 5 summarizes the runtime of HE-Meanshift and MPC-KMeans on large datasets over LAN network.

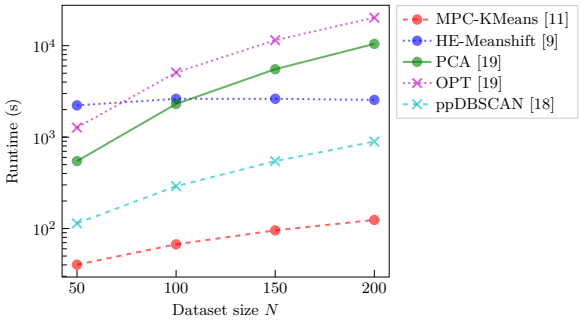


Fig. 3. WAN runtime in seconds of the private clustering protocols MPC-KMeans [11], HE-Meanshift [9], PCA/OPT [19], and ppDBSCAN [18] for varying dataset size N, K=2 clusters, dimension $d = 8$, and bitlength $\ell = 32$.

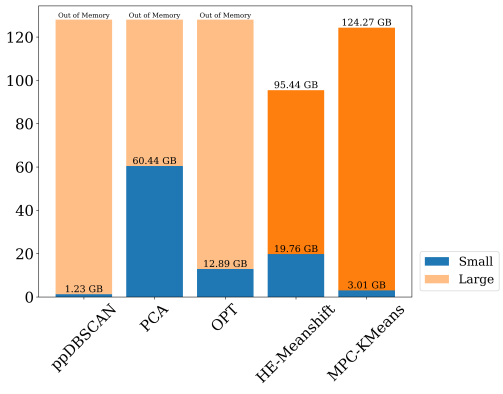


Fig. 4. Memory consumption in GB of the privacy-preserving clustering protocols ppDBSCAN [18], PCA/OPT [19], HE-Meanshift [9], and MPC-KMeans [11] for a small ($N = 200, d = 8, K = 10$) and large ($N = 65536, d = 4, K = 20$) dataset.

D Additional Clustering Quality Evaluation

We compare the clustering quality on nine widely used datasets: Hepta (Tab. 6), Lsun (Tab. 7), Target (Tab. 8), Wingnut (Tab. 9), Tetra (Tab. 10), Chainlink (Tab. 11), and EngyTime (Tab. 12) from [122]; and Dense (Tab. 13) and ZigZag Noisy (Tab. 14) from [123]. Each dataset has different characteristics such as different cluster shapes or different densities.

Algorithm	ARI	AMI	SI	CHI
Ground Truth	-	-	0.883	519.937
MPC-KMeans	0.869	0.946	0.754	292.539
KMeans++	1.0	1.0	0.883	519.937
HE-Meanshift	0.667	0.834	0.603	156.518
Mean-shift	1.0	1.0	0.883	519.937
ppDBSCAN	1.0	1.0	0.609	384.439
OPT	1.0	1.0	0.883	519.937
PCA	1.0	1.0	0.883	519.937

Table 6. Hepta

Algorithm	ARI	AMI	SI	CHI
Ground Truth	-	-	0.609	384.439
MPC-KMeans	0.405	0.524	0.653	485.003
KMeans++	0.405	0.524	0.653	485.003
HE-Meanshift	0.434	0.537	0.234	220.118
Mean-shift	0.366	0.445	0.589	293.479
ppDBSCAN	1.0	1.0	0.609	384.439
OPT	1.0	1.0	0.609	384.439
PCA	0.405	0.529	0.641	458.157

Table 7. Lsun

Algorithm	ARI	AMI	SI	CHI
Ground Truth 1	-	-	0.260	27.869
Ground Truth 2	-	-	0.249	0.494
MPC-KMeans	0.534	0.615	0.678	709.651
KMeans++	0.611	0.639	0.742	738.291
HE-Meanshift	0.215	0.311	0.383	101.586
Mean-shift	0.626	0.645	0.766	590.057
ppDBSCAN	1.0	1.0	0.249	0.494
OPT	1.0	1.0	0.249	0.494
PCA	0.207	0.377	0.506	90.502

Table 8. Dataset Target

Algorithm	ARI	AMI	SI	CHI
Ground Truth	-	-	0.630	1061.016
MPC-KMeans	0.417	0.326	0.567	805.066
KMeans++	0.425	0.334	0.570	815.492
HE-Meanshift	0.475	0.451	0.373	611.832
Mean-shift	0.638	0.538	0.621	1027.873
ppDBSCAN	1.0	1.0	0.630	1061.016
OPT	1.0	1.0	0.630	1061.016
PCA	1.0	1.0	0.630	1061.016

Table 9. Dataset Wingnut

Algorithm	ARI	AMI	SI	CHI
Ground Truth	-	-	0.726	418.391
MPC-KMeans	0.961	0.975	0.689	390.920
KMeans++	1.0	1.0	0.726	418.391
HE-Meanshift	0.518	0.587	0.124	109.916
Mean-shift	1.0	1.0	0.726	418.391
ppDBSCAN	0.94	0.94	0.694	391.819
OPT	0.000	0.000	-0.436	1.472
PCA	0.987	0.982	0.718	409.221

Table 10. Dataset Tetra

Algorithm	ARI	AMI	SI	CHI
Ground Truth	-	-	0.179	250.865
MPC-KMeans	0.088	0.065	0.525	718.934
KMeans++	0.087	0.064	0.525	718.788
HE-Meanshift	0.132	0.160	0.262	353.929
Mean-shift	0.223	0.272	0.405	574.488
ppDBSCAN	1.0	1.0	0.179	250.865
OPT	1.0	1.0	0.179	250.865
PCA	0.313	0.388	0.463	575.488

Table 11. Chainlink

Algorithm	ARI	AMI	SI	CHI
Ground Truth 1	-	-	0.557	2921.700
Ground Truth 2	-	-	0.577	3075.082
MPC-KMeans	0.844	0.783	0.578	3158.931
KMeans++	0.843	0.783	0.578	3158.931
HE-Meanshift	0.801	0.720	0.198	1681.223
Mean-shift	0.833	0.769	0.578	3176.809
ppDBSCAN	0.612	0.493	-0.416	115.717
OPT	0.000	0.000	0.479	8.044
PCA	0.042	0.150	0.472	1318.72

Table 12. Dataset EngyTime

Algorithm	ARI	AMI	SI	CHI
Ground Truth	-	-	0.740	433.656
MPC-KMeans	0.756	0.713	0.790	497.182
KMeans++	0.768	0.723	0.790	499.284
HE-Meanshift	0.838	0.779	0.540	277.155
Mean-shift	0.784	0.725	0.699	309.728
ppDBSCAN	0.935	0.904	0.762	503.083
OPT	0.000	0.019	0.490	15.626
PCA	0.257	0.348	0.631	231.817

Table 13. Dense

Algorithm	ARI	AMI	SI	CHI
Ground Truth 1	-	-	-0.050	30.858
Ground Truth 2	-	-	0.500	317.869
MPC-KMeans	0.497	0.636	0.489	321.627
KMeans++	0.519	0.655	0.540	362.227
HE-Meanshift	0.498	0.648	0.538	368.285
Mean-shift	0.542	0.699	0.510	351.971
ppDBSCAN	1.0	1.0	-0.050	30.858
OPT	1.0	1.0	-0.040	30.858
PCA	0.521	0.672	0.504	371.251

Table 14. Dataset ZigZag Noisy

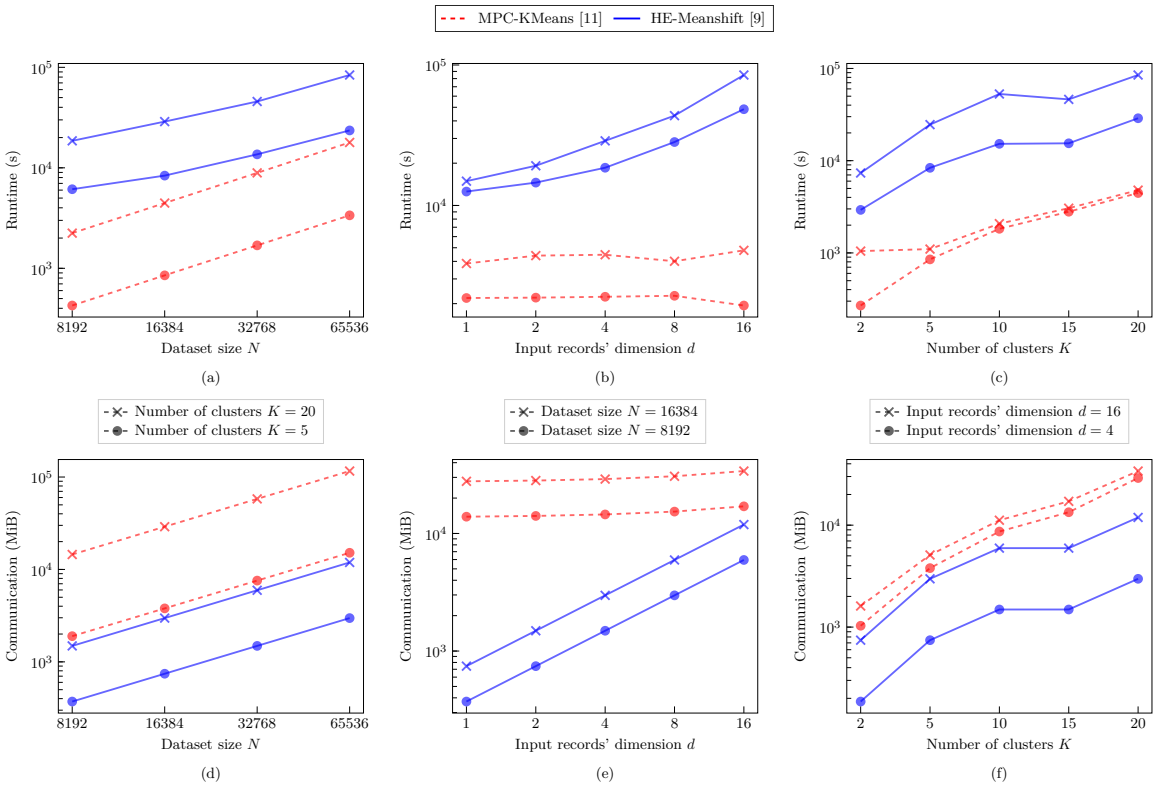


Fig. 5. LAN runtimes in seconds (top row) and **communication** in MiB (bottom row) of the fully-private clustering protocols MPC-KMeans [11] and HE-Meanshift [9] for varying dataset size N, input records' dimension d, number of clusters K, and bitlength $\ell = 32$. In (a) and (d) is $d = 4$, in (b) and (e) $K = 20$, and in (c) and (f) $N = 16384$.

B Privacy-preserving Density-based Clustering (ASIACCS'21)

[BCE+21] B. Bozdemir, S. Canard, O. Ermis, H. Möllering, M. Önen, T. Schneider. **"Privacy-preserving density-based clustering"**. In: *ASIA Conference on Computer and Communications Security (ASIACCS)*. Online: https://ia.cr/2021/612. Code: https://encrypto.de/code/ppDBSCAN. ACM, 2021, S. 658–671. CORE Rank A. Appendix B.

https://doi.org/10.1145/3433210.3453104

Privacy-preserving Density-based Clustering

Beyza Bozdemir
EURECOM
Sophia Antipolis, France
beyza.bozdemir@eurecom.fr

Sébastien Canard
Applied Crypto Group, Orange Labs
Caen, France
sebastien.canard@orange.com

Orhan Ermis
EURECOM
Sophia Antipolis, France
orhan.ermis@eurecom.fr

Helen Möllering*
Technical University of Darmstadt
Darmstadt, Germany
moellering@encrypto.cs.tu-darmstadt.de

Melek Önen
EURECOM
Sophia Antipolis, France
melek.onen@eurecom.fr

Thomas Schneider
Technical University of Darmstadt
Darmstadt, Germany
schneider@encrypto.cs.tu-darmstadt.de

ABSTRACT

Clustering is an unsupervised machine learning technique that outputs clusters containing similar data items. In this work, we investigate privacy-preserving density-based clustering which is, for example, used in financial analytics and medical diagnosis. When (multiple) data owners collaborate or outsource the computation, privacy concerns arise. To address this problem, we design, implement, and evaluate the first practical and fully private density-based clustering scheme based on secure two-party computation. Our protocol privately executes the DBSCAN algorithm without disclosing any information (including the number and size of clusters). It can be used for private clustering between two parties as well as for private outsourcing of an arbitrary number of data owners to two non-colluding servers. Our implementation of the DBSCAN algorithm privately clusters data sets with 400 elements in 7 minutes on commodity hardware. Thereby, it flexibly determines the number of required clusters and is insensitive to outliers, while being only factor 19x slower than today's fastest private K-means protocol (Mohassel et al., PETS'20) which can only be used for specific data sets. We then show how to transfer our newly designed protocol to related clustering algorithms by introducing a private approximation of the TRACLUS algorithm for trajectory clustering which has interesting real-world applications like financial time series forecasts and the investigation of the spread of a disease like COVID-19.

CCS CONCEPTS

• **Security and privacy** → **Privacy-preserving protocols**; • **Computing methodologies** → *Cluster analysis.*

KEYWORDS

Private Machine Learning, Clustering, Secure Computation

*Contact Author.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
ASIA CCS '21, June 7–11, 2021, Virtual Event, Hong Kong
© 2021 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-8287-8/21/06…$15.00
https://doi.org/10.1145/3433210.3453104

ACM Reference Format:
Beyza Bozdemir, Sébastien Canard, Orhan Ermis, Helen Möllering, Melek Önen, and Thomas Schneider. 2021. Privacy-preserving Density-based Clustering. In *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security (ASIA CCS '21), June 7–11, 2021, Virtual Event, Hong Kong*. ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3433210.3453104

1 INTRODUCTION

The availability of vast amounts of data and cloud computing power nowadays has lead to hype around machine learning (ML). Supervised ML techniques like neural networks use labeled data records (i.e., known input-output pairs) to train a model later utilized, e.g., for the classification of new records. In contrast, unsupervised ML techniques have no "training" phase and aim at detecting unknown patterns and structures in the unlabeled input data. Clustering is a widespread unsupervised ML technique that partitions data into groups of elements with similar properties. It has many privacy-critical applications where sensitive business or personal data must be protected spanning from financial analytics over market research to medical diagnoses [1, 25, 55].

For such analyses, data from several sources is often needed, for example, from competing (investment) banks or insurance companies to detect suspicious behavior [62] or from several hospitals to get a diverse data set that is not biased due to diverse backgrounds and demographics. Generally, clustering more data from several sources commonly enhances the quality of analyses. Moreover, it can be attractive to outsource computation and data due to high requirements for storage and costly computation. However, in both collaborative analyses and when outsourcing computation, the sensitive input data requires protection against untrusted servers and other data owners. Secure computation techniques can protect against these parties to preserve data privacy.

Several optimized private clustering protocols using secure computation have already been proposed for the well-known K-means algorithm [48, 70]. However, K-means is relatively simple and can only cluster specific data. It only detects convexly shaped clusters such that it is only suitable for specific data collections. Additionally, the number of clusters K needs to be determined in advance which requires domain knowledge. Access to just a subset of the input data makes it difficult to determine K. Also, K-means does not include the notion of noise and its result is highly sensitive to outliers as it has to assign every input to a cluster. Hence, even if a

record does not fit into any cluster, it will be assigned to the least distant one and heavily affect this cluster's centroid (i.e., the mean of all assigned elements).

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is a more flexible clustering algorithm introduced by Ester et al. in [19] to address the weaknesses of previously known clustering algorithms like K-means. DBSCAN is able to detect arbitrarily shaped clusters. Additionally, the number of clusters is flexibly determined such that it fits to the data. The algorithm is insensitive to outliers and specifically marks them as noise. Fig. 1 demonstrates the advantages of DBSCAN over K-means on four different data sets.

TRACLUS [44], an extension of DBSCAN, is specifically designed for clustering trajectories (i.e., sequences of multidimensional points). Trajectories intuitively do not form convex clusters. Furthermore, as trajectories often have different lengths, K-means cannot straightforwardly cluster them, because it requires the same fixed number of parameters for all inputs.

Private trajectory clustering can be a valuable mean for financial time series forecasts [28, 53] which are used by investors for decision making and rely on the input of sensitive business data. Additionally, it can be used for the privacy-preserving analysis of location data collected by telecommunication providers to optimize the services of the travel industry by determining typical traveling routes. Another interesting application is the analysis of the movement of (infected) people to control the spread of COVID-19 while maintaining privacy in accordance to regulations like the GDPR. In this context, it enables to detect people who used similar routes while the service provider cannot learn individual movements. Moreover, policy makers could use privacy-preserving trajectory clustering to privately determine if people comply to social distancing in the current pandemic before deciding about further regulations.

Need for Full Privacy-preservation. As we will discuss in §2, many previous "private" clustering protocols leak information beyond the output. We now demonstrate why leaking such intermediate information can cause a severe privacy breach using an example from [46].

Liu et al. [46] demonstrate how leaking which elements are neighbors, as done in [42], can be used to approximate data records of other parties. We depict the attack in Fig. 2. Let us consider two data owners, Alice and Bob, who hold a horizontally partitioned

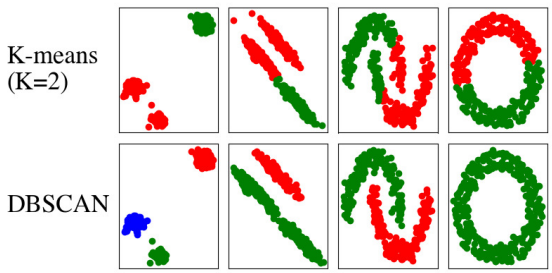


Figure 1: Comparison of clustering results with K-means and DBSCAN.

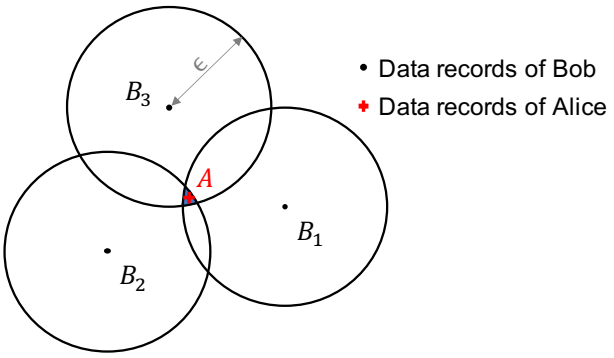


Figure 2: Approximating data records of other parties from leaked information (cf. [46]). ϵ is a DBSCAN parameter indicating the maximal distance between two data records to be considered as neighbors (cf. §3.1).

data set, i.e., different data records that must not be leaked to each other. Bob holds three data records B_1, B_2, and B_3 that are too far from each other (i.e., their mutual distances are greater than ϵ, cf. §3.1) to create a cluster themselves. However, when Alice holds a data record A that lies exactly in the intersection of the neighborhoods of B_1, B_2, and B_3 indicated by the circles with radius ϵ, Bob learns that all three of his data records are neighbors of the same element of Alice. From this information and if this intersection is small, Bob can accurately approximate A.

To summarize, such information leakage beyond the clustering output can cause serious privacy infringements, and hence should be avoided. In this work, we provide the first fully privacy-preserving DBSCAN algorithm that does not leak any information beyond the output of the clustering.

Our Contributions and Outline

After summarizing related works in §2, we provide the following contributions:

- **Fully private DBSCAN:** We design and implement the first fully private DBSCAN scheme based on secure two-party computation (S2PC) that achieves the same clustering quality as the plaintext algorithm (cf. §5.2). For this, we introduce dedicated protocols for all components of DBSCAN including a *partial parallelization* (cf. §4.3). Our solution can be used for clustering between two parties that want to keep their inputs private as well as for outsourcing scenarios where many data owners secret-share their private input records among two non-colluding servers [37] (cf. §4.1). Due to the usage of generic S2PC, our protocols leak no information beyond the output. They can flexibly be used for arbitrarily partitioned inputs (cf. §4.3.1) and complemented with arbitrary private post-processing of the output (cf. §4.3.5).
- **Privacy-preserving trajectory clustering:** We show that our protocols can be used for the privacy-preserving design of particular instantiations of density-based clustering schemes by introducing the first privacy-preserving trajectory clustering protocol based on the TRACLUS algorithm.

TRACLUS is a DBSCAN-based scheme optimized for clustering trajectories, for which we design and implement an approximated distance to improve its secure computation efficiency (cf. §4.4).

- **Open-source implementation and comprehensive empirical evaluation:** We benchmark our implementation[1] and show that our protocols have practical runtimes on public data sets (cf. §5.4). We compare the efficiency of privacy-preserving DBSCAN to state-of-the-art privacy-preserving solutions of the simpler and limited K-means algorithm [34, 48] (cf. §5.3). Furthermore, we show that our approximated TRACLUS distance has similar clustering quality as the original distance on public and real-world data sets (cf. §5.2.2).

2 RELATED WORK

In this section, we discuss related work on privacy-preserving K-means and DBSCAN clustering. Moreover, we give a brief overview on privacy-preserving trajectory analysis.

K-means. Privacy-preserving K-means clustering protocols were intensively discussed in the literature. They use different techniques like homomorphic encryption (HE), random blinding, various secure two-party computation (S2PC) and multi-party computation (MPC) techniques, and combinations of the aforementioned.

Most early works assume that all parties have access to a subset of the plaintext input and require that data owners actively participate in the clustering. Private K-means clustering on horizontally partitioned data, i.e., each party holds a subset of the data records, was addressed in [23, 35, 60]. [60, 68] investigate private K-means clustering on vertically partitioned data, i.e., the data owners hold the values of disjoint subsets of attributes of all data records. A flexible mix of both partitioning types is called arbitrary partitioning covered in [13, 32]. Unfortunately, many proposed schemes leak intermediate values such as centroids [23, 35, 45, 60, 68], clusters' sizes [23, 60], merging patterns [31], use incorrect calculations [32], or require active participation of the data owner [45] (which makes them not suitable for the outsourcing scenario). Another research direction leverages differential privacy [8, 63–65] to protect individuals' data, but these works trade accuracy for better performance. Only the following works provide full privacy guarantees while achieving a high accuracy: [13, 34, 48].

Bunn and Ostrovsky [13] create a two-party private K-means protocol using additive homomorphic encryption (AHE) in combination with S2PC for arbitrarily partitioned data. However, the usage of expensive AHE results in impractical runtimes.

Jäschke et al. [34] use fully homomorphic encryption (FHE) for a protocol that enables a data owner to outsource the K-means computation in a privacy-preserving way. To improve runtime, they simplify the required calculation of the original K-means clustering by approximating the centroid determination and distance comparisons, but still the overhead is far from practical for larger data sets.

Mohassel et al. [48] provide an efficient privacy-preserving K-means protocol in the semi-honest security model using Yao's Garbled Circuits. In their work, they improve efficiency by utilizing that K-means requires to calculate the same distance function with one fixed input, namely a point of the input data, to all (repeatedly updated) centroids.

DBSCAN. Although density-based clustering has several advantages as detailed in the beginning of §1, its privacy-preserving realization is barely studied.

Anikin and Gazimov [5] design a privacy-preserving DBSCAN protocol between an arbitrary number of parties with vertically partitioned data in the semi-honest security model. Their scheme uses additive blinding and HE. Besides expensive encryption operations, their protocol requires all parties to have plaintext access to the data records, to be online, and to communicate extensively which makes it not applicable to a privacy-preserving outsourcing scenario. Additionally, one party executes the main part of the computation and obtains the plaintext distances between the input records, cluster assignments, and hence, cluster sizes. The authors do neither provide a concrete instantiation of their protocol nor a performance evaluation.

Similar reasoning applies for [4, 36, 42, 46, 72] who assume vertically/horizontally partitioned data and for [3, 46] with arbitrary data partitioning. They use HE, blinding with random values, and/or partially trusted third parties to protect data privacy. However, they leak information like cluster sizes and neighborhood patterns [3, 36, 42, 46, 57] or distances between input records [4]. Under specific circumstances, the leaked information can be used to concretely identify other parties' input records [42, 46]. In [72], complete data records are revealed when they belong to a cluster. Again, no implementation and performance evaluation were performed in these works.

Rahman et al. [57] introduce a privacy-preserving DBSCAN protocol using key-HE[2] for an outsourcing scenario where an untrusted server conducts the clustering with the help of the data owners. Unfortunately, it leaks information such as the clusters' sizes and neighborhood patterns to the server. Again, no implementation and performance evaluation were performed in that work.

In [51, 71], the authors leverage differential privacy [22] to protect the privacy of individuals' data records. In these works, the data owner performs the clustering, and an untrusted party later requests access to parts of the results. This scenario is not transferable to executing the clustering with inputs from two or more parties. Moreover, the level of noise added to the distance in each parameter dimension depends on the privacy requirements and will inevitably affect the meaningfulness of the data and, hence, the quality of the clustering result.

To summarize, all existing protocols on private density-based clustering assume that the involved parties have access to a fraction of the plaintext input data (meaning that the data owners actively participate in the protocol), use differential privacy (sacrificing accuracy), are not fully private (leaking information), and/or make heavy use of public-key cryptography (making the protocol inefficient and not scalable). Our protocol is the first to be fully precise (i.e., the same clustering quality as with plaintext DBSCAN is achieved), fully private, and efficient for both the data owners and server(s). It is usable in the outsourcing scenario and for two-party computation (cf. §4.1).

[1] https://encrypto.de/code/ppDBSCAN

[2] Such a HE scheme allows to combine ciphertexts created with different keys to produce an encryption of the sum of the messages decodable with sum of the keys.

Trajectory Analysis. Very few previous studies investigate the problem of private trajectory analysis. Private-Hermes [54] and Hermes++ [40] use anonymisation techniques to generate crafted, realistic fake trajectories to allow users to securely query mobility (trajectory) data sets. Other works such as [2, 26, 52] explore the problem of private ride sharing which consists of finding a match between parts of trajectories. To the best of our knowledge, no previous work has investigated private trajectory clustering. Our privacy-preserving TRACLUS protocol is the first that enables fully privacy-preserving and efficient trajectory clustering between two data owners and in the outsourcing scenario.

3 PRELIMINARIES

In this section, we introduce the original DBSCAN algorithm and its adaptation TRACLUS. We also give the cryptographic building blocks used in our protocols.

3.1 Clustering

3.1.1 DBSCAN. Density-based Spatial Clustering of Applications with Noise (DBSCAN) [19] relies on a density-based neighborhood notion. Elements that are located with many other elements in a dense area form a cluster while data records in a sparse area are marked as noise. This intuition is realized by determining *core points* that have at least minPts other elements in a range of ϵ around them. The range can be quantified with any distance measure. All elements that are located in the neighborhood of a core point p are *directly density-reachable* from p. If an element q is connected via a chain of core points $p_1p_2...p_z$, with $p_z = q$, where each p_{i+1} is directly density-reachable from p_i, then q is called *density-reachable*. All points in the chain have to be core points except from q. A cluster consists of a core point p and all density-reachable elements from p. Elements of a cluster that are not core elements but belong to the neighborhood of a core element are *border elements*. If an element is neither a core element nor a border element it is marked as *noise*.

As mentioned above, DBSCAN has two parameters minPts and ϵ. minPts determines how many data records need to lie in a neighborhood to form a cluster. The larger the value of minPts, the more elements are marked as noise while the clusters get more dominant. Ester et al. [19] suggest minPts = 4 for 2-dimensional data, Sander et al. [61] recommend minPts = $2 \cdot$ dim where dim denotes the dimensionality of the data, but depending on the application, data set size, and the noisiness of data, the optimal value can vary. ϵ determines the maximum distance between two data records to be considered as neighbors. A large value of ϵ results in large clusters whereas a small value of ϵ marks many elements as *noise*.

DBSCAN can be applied on data of an arbitrary dimension. Its worst case complexity in the cleartext is $O(n^2)$, where n is the number of input elements.

3.1.2 TRACLUS. TRAjectory CLUStering (TRACLUS) [44] optimizes DBSCAN for sequences of multidimensional points (trajectories) and consists of two phases: *partitioning* and *grouping*. In the partitioning phase, the algorithm divides trajectories into sub-trajectories that are called *line segments*. Each line segment is represented by two points. The partitioning phase ensures that the output is close to the original trajectory (*preciseness*) and that the number of line segments created is minimal (*conciseness*). Afterwards in the *grouping* (*clustering*) phase, TRACLUS uses DBSCAN with the same parameters (minPts is named minLns for the minimal number of line segments). An optimal ϵ value can be determined with simulated annealing [38]: It is set to the value that minimizes the entropy of the clustering [44].

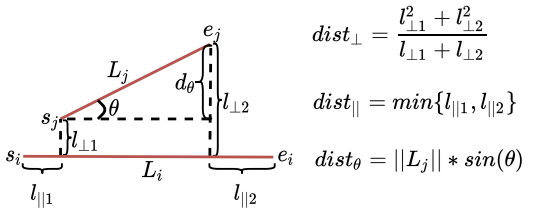


Figure 3: TRACLUS' tripartite distance. L_i and L_j denote two line segments with start points s_i/s_j and end points e_i/e_j (cf. [44]).

For the neighborhood determination, TRACLUS uses a special tripartite distance that is illustrated in Fig. 3:

$$dist(L_i, L_j) = w_\perp \cdot dist_\perp + w_{||} \cdot dist_{||} + w_\theta \cdot dist_\theta, \quad (1)$$

where L_i and L_j are two line segments. The perpendicular distance $dist_\perp$ measures the vertical distance between L_i and L_j. The parallel distance $dist_{||}$ is the horizontal distance between L_i and L_j. The angular distance $dist_\theta$ measures the directional difference between L_i and L_j. The components are weighted with $w_\perp$, $w_{||}$, and w_θ (set to 1 by default) and summed up. See [44] for more details. In §4.4, we give a secure computation-friendly approximation of the tripartite TRACLUS distance.

3.2 Clustering Quality Indices

To evaluate the quality of a clustering method, several quality assessment measures have been proposed. We employ the commonly used Adjusted Rand Index (ARI), Silhouette Coefficient (SC), and Density-Based Clustering Validation (DBCV).

3.2.1 Adjusted Rand Index (ARI). The ARI [29] is an external clustering validation index. External means in this context that the real partitioning, called the ground truth, must be known. It takes two partitionings as an input and evaluates how many input pairs are assigned to the same cluster by both partitionings, how many pairs are assigned to different clusters by both, and how many pairs belong to the same cluster in one partitioning but to different clusters in the other. Additionally, it corrects the count for chance. The ARI is 1 for a perfect clustering. Worse clustering results yield smaller ARIs.

3.2.2 Silhouette Coefficient (SC). As clustering is an unsupervised machine learning technique, a ground truth is normally not available. The most frequently used methodology to evaluate the quality of the clustering result is the silhouette analysis [59]. This methodology analyzes the resulting clusters by evaluating the similarity between the elements within the cluster and the elements of other

clusters. The output is called the *silhouette coefficient* (SC) which is close to 1 for a good clustering.

Although SC allows insights about how tightly elements are clustered, it does not take outliers that remain unclustered, called noise, into account. Thus, a penalty is needed to measure the SC with noise as described in [49]. Let n be the data set size and u be the number of not clustered elements. Then, multiplying the SC with $(n-u)/n$ gives the SC with noise penalty, SC_{noise}.

3.2.3 Density-Based Clustering Validation (DBCV). Another method specially designed to assess the quality of density-based clustering algorithms is the DBCV [49]. In contrast to the standard SC, DBCV also takes noise into account. Similar to the other evaluation metrics, a higher DBCV indicates a good clustering quality.

3.3 Secure Two-Party Computation

Secure computation allows two or multiple parties to securely compute on private inputs. In this work, we use secure two-party computation (S2PC) techniques. Typically, S2PC is used for two-party applications, where two parties provide private inputs. Alternatively, it can be extended to an outsourcing scenario [37], where an arbitrary number of input parties secret-share their private input data among two non-colluding servers. Secret-sharing allows to split data into two random-looking values called secret-shares and each of the two servers obtains one of these shares. To recover the data, both servers would have to put their shares together (which by the non-collusion assumption they do not do). The two servers then jointly evaluate a function using S2PC on the shares without being able to learn anything about the inputs or intermediate values.

The non-collusion assumption could be avoided by using homomorphic encryption (HE) [22] but currently available HE schemes have a significant computational overhead as they use expensive cryptographic primitives. Alternatively, using more than two servers with tolerating one corruption yields better efficiency, e.g., [14, 47], but have a larger attack surface as the corruption of any pair of the multiple servers is sufficient to break privacy. Therefore, S2PC is a reasonable trade-off between security and efficiency. Still, our protocols in §4 are general and can easily be extended to more than two parties/servers using hybrid secure multi-party computation frameworks [14, 47].

Security Model. We consider the semi-honest security model where the two non-colluding parties honestly follow the protocol while attempting to collect information about the other party's private inputs. This security model also protects against passive attacks by curious administrators or accidental data leakage. The non-collusion assumption can, e.g., be guaranteed between two competing companies as it is in their interest to not give their customer's data to the competitor and to protect their business secrets. Additionally, in an outsourcing scenario where one or several data owners outsource the expensive computation to (cloud) servers, it is reasonable to assume that the two servers are semi-honest, because cloud providers are strictly regulated and threatened with severe financial damage. Moreover, their reputation is damaged when malicious behavior is detected. Hence, it is in the providers' economic interest to behave semi-honestly. In the outsourcing scenario, our protocols in §4 are even secure against any number of *malicious* data owners who can *arbitrarily* deviate from the protocol [37]. We provide concrete examples how several applications can be realized with S2PC in §4.

ABY. We use the S2PC framework ABY [16] that implements three types of S2PC protocols, namely Yao's Garbled Circuits [73], Boolean GMW [24], and Arithmetic sharing, which is a generalization of GMW, including state-of-the-art optimizations. Moreover, it enables flexible conversions between the three sharing types. ABY supports Single Instruction Multiple Data (SIMD) operations that efficiently apply the same functionality on multiple inputs in parallel. SIMD operations reduce memory usage and evaluation time. We use Yao's Garbled Circuits (i.e., Yao sharing) and Arithmetic sharing in this work (cf. §B for details).

4 PRIVACY-PRESERVING DENSITY-BASED CLUSTERING

In this section, we present our protocols for privacy-preserving DBSCAN (ppDBSCAN) clustering, which for the first time provides scalability with respect to data set size, the number of clusters, and input records' dimension as well as full privacy while maintaining the clustering quality of the plaintext algorithm. We extend our protocol to the private clustering of trajectories using the TRACLUS algorithm (ppTRACLUS).

4.1 Application Scenarios

As already indicated in §3.3, our ppDBSCAN and ppTRACLUS can be used in two application scenarios:

- **Two-party Computation (2PC)**: Here, two data owners privately perform the clustering on their vertically or horizontally partitioned data. In such a classical secure two-party computation, each of them runs one semi-honest party itself. The data owners could, for example, be two banks who jointly investigate credit card frauds.
- **Outsourcing**: Here, one or multiple data owners outsource their data as random-looking secret-shares (cf. §3.3) to two non-colluding semi-honest parties (e.g., cloud servers) that perform the private clustering via S2PC. The data owners want to hide their sensitive data from any other party. As explained in §3.3, it is in the cloud providers' interest to behave semi-honestly and to not collude. The data owners can even be malicious and arbitrarily deviate from the protocol [37]. Considering the application examples in §1, the data owners can be, for example, multiple hospitals that join forces to improve medical diagnosis or a telecommunication provider that lawfully wants to monetize location information of customers by selling privacy-preserving analytics to the travel industry. To analyze people's movements in a pandemic, a governmental institution and a non-profit organization like the EFF[3] could operate the two non-colluding servers.

In the following, a *party* denotes one of the two non-colluding and semi-honest parties that perform the privacy-preserving clustering.

4.2 Notation

We use the following notation in the remainder of this work: A multiplexer gate (MUX) is denoted by $a?b:c$ which corresponds to

[3] https://www.eff.org

if a then b; else c. An AND gate is denoted by $\wedge$ and an OR gate by $\vee$.

4.3 Privacy-preserving DBSCAN

In this section, we present our secure two-party computation (S2PC) protocol for fully privacy-preserving DBSCAN including a partial parallelization of the clustering. A discussion on its security properties is given in §C.

4.3.1 Flexible Inputs. Initially, the data owners use Arithmetic sharing (cf. §3.3) to secret-share their data among each of the two parties. Each of the two parties collects the secret-shares of the inputs it receives from all data owners in a vector X that is input to the clustering. Additionally, the two processing parties can also be provided with secret-shares of DBSCAN's parameters minPts and ϵ (cf. §3.1) such that they do not know them in the clear.

Note that the usage of Arithmetic sharing for the input data that shall be clustered enables an *arbitrary* number of data owners to easily join the clustering as no encryption key is needed. Moreover, our ppDBSCAN can support *any* data splitting, i.e., horizontally or vertically distributed data or an arbitrary mix of both, because thanks to the Arithmetic sharing the data owners can simply secret-share the attributes of the data records they are holding and send these shares to the two parties who can sort them according to the respective data records for the clustering.[4]

After receiving all shares from the data owners, the clustering starts. DBSCAN can be split into two main building blocks, the *distance calculation* and the *clustering* process, for which we design novel S2PC protocols enabling a partial parallelization.

Listing 1: Privacy-preserving Squared Euclidean Distance (SED) calculation.

```
1  Input: vector X // n-dim. vector with input data
2         ε²       // sq. threshold
3  Output: dist    // n-dim. vector with SIMD -values:
4                  // dist[l]=(SED(X[k],X[l])< ε²), 0 ≤ k < n*/
5
6  sharedElements = combineToSIMD(X)
7  Vector dist;
8  for(i=0; i < n; i++):
9    simdElement = [X[i]]*n // n copies of X[i]
10   sed = 0
11   for(j=0; j < dim; j++):
12     temp = sharedElements[j] - simdElement[j] // Arithm
13     sed += temp * temp
14   dist[i] = sed < ε² // Yao
15 return dist
```

4.3.2 Parallelized Distance Calculation. In DBSCAN, pairwise distances between all input records are calculated to determine neighborhoods. In general, any distance measure can be used. Ester et al. [19] originally use the Euclidean distance (ED), but for more efficient evaluation we use the squared Euclidean distance (SED) which is common for S2PC, e.g., privacy-preserving face recognition [17]. To determine if two elements are neighbors, the ED of two input values x and y has to be smaller than ϵ which is equivalent to $SED(x, y) < \epsilon^2$.

List. 1 presents our private SED protocol. To massively parallelize the computation, each party combines its secret-share of the jth coordinates of each data record in X with *combineToSIMD(..)* in one SIMD arithmetic share (cf. §3.3) and collects these SIMD-shares in a j-dimensional vector *sharedElements*. For every execution of the *for* loop in Line 8, a j-dimensional vector *simdElement* is created by each party where the jth entry contains n times its secret-share of the jth coordinate of the ith input, where n denotes the data set size. In Lines 11-13, the SED is jointly calculated between the two parties. Then, they jointly compare the result to the threshold ϵ^2 and only the comparison result (secret-shared between the two parties) is saved in the vector *dist* to reduce the storage requirement to 1 bit per input record pair for each party. Given the symmetry of the distance measure, this approach causes an overhead as it calculates n^2 instead of $\approx 0.5n^2$ distances. But this is compensated by enabling parallel distance computation and a partial parallelization of the clustering which substantially reduces memory consumption and improves runtime. For multiplications and additions, Arithmetic sharing is most efficient (addition can even be done for free), and we convert the resulting shared SED to Yao sharing for efficient comparison [16].

We point out that further optimizations for distance calculations from [33, 48] can improve the runtime and/or communication costs of our protocol. However, as our experiments in §5.4 show, computing SEDs is only a small fraction of the total complexity, so we did not implement further optimizations.

4.3.3 Parameter Estimation. As described in §3.1, DBSCAN has two parameters: ϵ (now ϵ^2 because of the SED) determines the maximal distance between two input records to consider them as neighbors, and minPts is the minimal cluster size.

The minimal cluster size minPts is strongly dependent on the respective use case. For example, if for the containment of COVID-19 contact restrictions have been put into place and meetings with only less than 10 people are allowed, minPts could be set to 10 to check if and how many "illegal" meetings happened. We recommend to choose minPts based on the use case and its privacy requirements as well as the data set size. Generally, it is advantageous to aim for larger clusters (i.e., to choose a sufficiently large minimal cluster size minPts) as they generalize better and leak less information. For our DBSCAN experiments in §5, we follow the recommendation by [19] and set minPts = $\frac{n}{100}$ (resp. minPts = 4 for the smaller data set), where n is the data set size.

Similarly, for some use cases the value for ϵ^2 might already be given by the application itself: When investigating the movements of people for the containment of COVID-19, ϵ^2 can be set to 2 meters similarly as it is done in contact tracing apps [66]. Alternatively, ϵ^2 can be estimated by plotting a sorted k-distance graph as suggested in [19]. In an outsourcing scenario with a single data owner, the data owner can determine the value of ϵ^2 locally. When several data owners provide input data and the data is expected to be approximately independent and identically distributed, it can be sufficient that one data owner DO_i plots the graph with his data setting $k = \text{minPts}/\frac{n_i}{n}$, where n_i denotes the size of DO_i's input set. The proportion $\frac{n_i}{n}$ and n itself may also be approximated.

[4] Even the data set size can be hidden from the two processing parties by using dummy data records, but one has to ensure that the added data records do not change the clustering result. One option can be to duplicate data records in combination with increasing the value of minPts. Also the number of parameters per data record can be hidden by adding dummy parameters that are set to the same value for all input records such that they do not affect the clustering result.

Otherwise, the data owners can also agree on a value for ϵ^2 after each data owner analyzed its data locally by a secure aggregation of their local results.[5] Furthermore, the calculation of a k-distance graph and determination of ϵ^2 can also be realized with a secure computation ahead of the clustering. However, such a secure computation protocol for calculating ϵ^2 is not the focus of our work and we leave it open for future work. Some works have already investigated private sorting [27, 33] and computing of the k-th nearest neighbor [15, 33, 56].

The minimal possible k is 2 and the minimal meaningful value for minPts is 3. As noted in the beginning of §4.3, both parameters can be secret-shared and no information is leaked.

Listing 2: Shared input element.

```
1  class SharedInputRecord(minPts, // (minimal neighborhood
       size
2      splittedD): /* (n-dim. vector with subset of splitted
           dist output by Listing 1)*/
3    clusterId = 0
4    isNoise = 0
5    notProcessed = 1
6    Vector neighbors = splittedD
7    validNeighborhoodSize = minPts < hamming_weight(
         splittedD)
```

4.3.4 Partially Parallelized & Private Clustering. To make DBSCAN fully private, it is not sufficient to have a private distance calculation, but also all intermediate information such as the clusters' sizes, assignment patterns, and number of clusters must not be leaked as these can contain sensitive information.

The plaintext DBSCAN clustering is given in §A. We significantly reduce the overhead of the neighborhood queries for every record by calculating all distances in a parallelized fashion before starting the clustering instead of doing it several times for every single record like in the plaintext. For every input record, each party creates an object of *SharedInputRecord* (cf. List. 2) which contains its secret-shares of the object's attributes' values. The secret-shares of the results of the comparison of the record's distances to all other records with ϵ^2 are stored in *neighbors*. The combined secret-shares of both parties of *validNeighborhoodSize* denote if an input has more than minPts neighbors, i.e., whether it is a core element (cf. §3.1). It follows that minPts has to be one less than in the plaintext DBSCAN to obtain the same clustering result.

Each party collects all objects of *SharedInputRecord* in a vector *sharedElements* and inputs it into our private clustering protocol in List. 3. In Lines 7 to 14 in List. 3, the parties jointly check if the data record *elem* was assigned to a cluster before and, if this is not the case, they check if *elem* is a core element such that it creates a new cluster or it is marked as noise. If a new cluster is created, the unclustered neighbors of *elem* are also added to this cluster (Lines 23-26 in List. 3). If the neighbors themselves are core elements, their neighbors also have to be analyzed and added to the cluster (Lines 28-36 in List. 3) as they are density-reachable (cf. §3.1). In plaintext DBSCAN, this is realized with a queue which leaks neighborhood patterns (cf. Lines 19-24 in List. 4). Therefore, for a privacy-preserving realization, all elements always have to be checked instead to obliviously expand a cluster. This allows us to parallelize these steps

[5] Secure aggregation protocols have been thoroughly investigated in the context of smart metering, e.g., [18, 21, 43].

Listing 3: Privacy-preserving DBSCAN (ppDBSCAN) with partial parallelization.

```
1  Input: vector sharedElements /* n-dim. vector with
       SharedInputRecords (cf. Listing 2)*/
2  Output: sharedElements
3
4  // clustering in Yao
5  currentClusterId = 0
6  for each elem in sharedElements:
7    elem.clusterId = elem.notProcessed?
8          (elem.validNeighborhoodSize?currentClusterId:0):
9          elem.clusterId
10   isCoreElement = elem.validNeighborhoodSize∧
11         elem.notProcessed
12   elem.isNoise = elem.notProcessed?
13         (!elem.validNeigborhoodSize):elem.isNoise
14   elem.notProcessed = 0
15
16   sIsCoreElement = [isCoreElement]*n        // n copies
17   sCurrentClusterId = [currentClusterId]*n // n copies
18
19   for(i=0; i < maxIterations; i++):
20     /* create SIMD values with the value of the
           respective attribute of all sharedElements */
21     sClusterIds, sIsNoise, sNotProcessed,
           sValidNeighborhoodSize = combineSharesToSIMD(
           sharedElements)
22
23     sValidNeighbor = ((sIsNoise∨sNotProcessed)∧
           sIsCoreElement)∧elem.neighbors
24     sClusterIds = sValidNeighbor?sCurrentClusterId:
           sClusterIds
25     sNotProcessed = sValidNeighbor?0:sNotProcessed
26     sIsNoise = sValidNeighbor?0:sIsNoise
27     // check if neighborhood size and if valid neighbor
28     sNeighborsAreCoreElement = sValidNeighbor?
           sValidNeighborhoodSize:0
29
30     for (k=0; k < n; k++):
31       sUpdateResultDistanceComp =
32       sNeighborsAreCoreElement∧sharedElements[k].
             neighbors
33       elem.neighbors[k] = ORTREE(
             sUpdateResultDistanceComp)?1:elem.neighbors[k]
34
35     // split simd values and update sharedElements
36     updateSharedElements({sIsNoise, sNotProcessed},
           sharedElements)
37   updateSharedElements({sClusterIds}, sharedElements)
38   currentClusterId = currentClusterId+isCoreElement //
       Arithm
39 return sharedElements
```

shown in Lines 19 to 36 in List. 3 with SIMD operations. However, as *elem*'s neighbors are changing, one has to repeat the loop starting in Line 19 in List. 3 maximally n times to accommodate the neighbors of neighbors that are core elements through the update in Line 33 in List. 3. This would increase the complexity of the protocol to $O(n^3)$ to make it fully oblivious. But depending on the data distribution few *maxIterations* are already sufficient. This yields a complexity of $O(maxIterations \cdot n^2)$. In our tests on the data sets used in §5, setting $maxIterations = 4$ detects all clusters correctly. Generally, many iterations are only needed if clusters consist of core elements that are chained in a row; thus, simply speaking, if clusters are particularly elongated. If the data owners cannot estimate the needed *maxIterations* in advance, a privacy-preserving check can be added that tests if the vector *elem.neighbors* in Line 33 in List. 3

changed in the last iteration (or after a specific number of iterations). This leaks one bit of information to the two parties to inform them if they have to continue further iterations. This additional information leaks only vaguely how the input data is distributed which can be acceptable for applications as a trade-off for better efficiency. The function $combineSharesToSIMD(..)$ takes a vector with n $SharedInputRecord$s as input and combines their $clusterId$-, $isNoise$-, $notProcessed$-, and $validNeigborhoodSize$-values to SIMD shares each containing n-values. $updateSharedElements(..)$ does the opposite, it takes SIMD shares as input, splits them, and updates the corresponding $SharedInputRecord$'s objects.

All bit-operations of our ppDBSCAN protocol in List. 3 are done in Yao sharing and the addition in Line 38 is done with Arithmetic sharing to optimize efficiency [16]. Furthermore, note that List. 3 is independent of the number of clusters and the dimensionality of the input. Only the distance calculation in §4.3.2 is influenced by the dimensionality while the number of clusters has no effect on the performance of ppDBSCAN. Its clustering's complexity solely depends on the input data set's size n.

4.3.5 Flexible Output and Post-Processing. Note that our protocol is applicable for both the outsourcing scenario where one or more (possibly malicious) data owners outsource computation to two non-colluding semi-honest (cloud) servers as well as for two-party applications (cf. §3.3 and §4.1). In both scenarios, the output can be provided to a third party, one or multiple data owners, or one of the servers depending on the use case.

Furthermore, the output of a clustering can flexibly be adapted to the application without leaking intermediate results. For example, the clustering can return a cluster label for each data record. Alternatively or additionally, it can return the average (i.e., centroid) or medoid of each cluster, or the number of clusters and respective cluster sizes.

Our secure two-party computation (S2PC) protocols for ppDBSCAN allow to realize each of these options as well as any further post-processing in a secure and efficient way by simply defining what is sent back to the respective parties that shall receive the output and optionally securely evaluating a post-processing circuit via S2PC.

4.4 Privacy-preserving TRACLUS

In this section, we provide a S2PC protocol for privacy-preserving trajectory clustering. For trajectory clustering with our privacy-preserving clustering protocol based on TRACLUS (cf. §3.1.2), we assume that the data owners hold a horizontally split data set, i.e., each data owner holds full trajectories.[6] They locally run the TRACLUS partitioning phase and outsource the clustering phase by providing arithmetic secret-shares (cf. §B) of the line segments to the two processing parties.

Simplified and Approximated Distance Measure. As described in §3.1.2, the computation of the TRACLUS distance involves complex operations like sine computations which are relatively expensive in S2PC (cf. §3.1.2). In order to ensure data privacy and execute the clustering phase in an efficient manner, we simplify this distance calculation using an approximation. Afterwards, ppDBSCAN — as presented in List. 3 — is executed.

As the Euclidean Distance (ED) is the most common measure used for trajectory clustering [12], we propose to replace TRACLUS' original distance with a combination of EDs. The perpendicular, parallel, and angular distances are still partially taken into account when using the EDs between each pair of points of the two line segments. As already explained in §4.3.2, ED can be replaced by the Squared Euclidean Distance (SED). Therefore, given line segment L_i with start point $s_i = (x_{si}, y_{si})$ and end point $e_i = (x_{ei}, y_{ei})$ and similar for line segment L_j, our approximated distance measure realized with arithmetic sharing for privacy-preserving TRACLUS (ppTRACLUS) is defined as:

$$d_{pptrac}(L_i, L_j) = SED(s_i, s_j) + SED(s_i, e_j) + SED(e_i, s_j) + SED(e_i, e_j).$$

To summarize, the two processing parties first jointly compute d_{pptrac} and compared it then to ϵ' in order to check whether two line segments are neighbors.[7]

5 EXPERIMENTAL EVALUATION

In this section, we present the experimental results of our ppDBSCAN and ppTRACLUS. We run the experiments with four public data sets and one private real-world data set to show clustering quality and to benchmark runtime and communication costs. Additionally, we compare to previous work on private K-means to show the practicability of our ppDBSCAN.

5.1 Experimental Setup

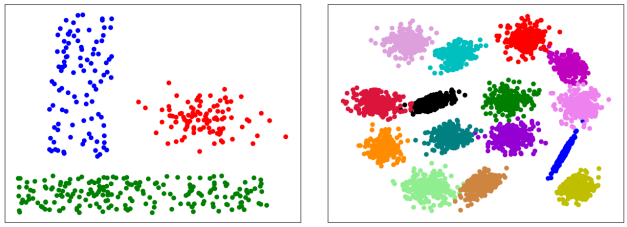
Server Configuration. We implement our protocols using the ABY framework [16] (cf. §3.3) which is written in C++ and uses 64-bit precision. The experiments are performed on two separate servers each equipped with Intel Core i9-7960X CPUs with 2.8 GHz and 128 GB RAM. As in an outsourcing scenario, the two servers are well-connected over a 10 Gbit/s LAN with 0.2 ms RTT.

Scenario. As detailed in §4.1, ppDBSCAN can be run in an outsourcing or a 2PC scenario. Moreover, the data to be clustered can be arbitrarily split among the data owners for ppDBSCAN. For ppTRACLUS, we require a horizontal data splitting to enable each data owner to locally execute the partitioning phase. In our benchmarks, we secret-share and outsource all data records to the two non-colluding servers to assess the efficiency and quality of our ppDBSCAN and ppTRACLUS. This models an outsourcing scenario with a single data owner, but we stress that the clustering (and hence its efficiency and quality) is *independent* of the number of data owner(s) and the data splitting between the data owner(s) in the outsourcing scenario as well as of the data splitting between the two parties in the 2PC scenario. It follows that our experimental results are directly transferable to other scenarios and more data owners as long as the data set size, the number of parameters, and the number of clusters is equal to our reported setting.

Data Sets. For the evaluation of our private clustering protocols, we use five data sets from different sources. Two are chosen based on

[6] Considering, e.g., the use case of collecting human movements trajectories for analysis purposes related to the containment of COVID-19, this trajectory data is typically collected by telecommunication providers such that assuming horizontally split data is plausible [9].

[7] We indicate the ϵ of ppTRACLUS with ϵ' to unambiguously differentiate it from the ϵ^2 used in ppDBSCAN.



(a) Lsun (b) S1

Figure 4: Ground truth of data sets Lsun and S1.

Table 1: Clustering quality evaluation.

Data Set	K-means	DBSCAN
Lsun	0.4386	1.0
S1	0.9254	0.9757

previous work on private K-means [34, 48] to evaluate and compare the efficiency and practicality of our ppDBSCAN. Additionally, we evaluate the clustering quality of our simplification of the original TRACLUS distance with a real world data set that contains location data extracted from mobile phones and two public data sets, namely the Hurricane and the Deer data sets used in the original TRACLUS paper [44].

- **Lsun:** The Lsun data set [67] in Fig. 4a was used in [34, 48] for evaluating private K-means protocols. It contains 400 2-dimensional data points. Its ground truth contains 3 clusters with different variances and cluster distances. The clusters have 200, 100, and 100 elements.
- **S1:** S1 [20] in Fig. 4b is a 2-dimensional synthetic data set with 5 000 samples that can be split into 15 spherical Gaussian clusters with between 300 and 350 elements. The clusters have an overlap of 9%. The set was used in [48, 65] to evaluate private K-mean protocols.
- **Travel:** This "synthetic" data set (not related to real individuals) is a 2-dimensional trajectory data corresponding to location of people's mobile phones. It has been created and provided by Orange S.A.[8], a major telecom provider, based on anonymized indicators of real trajectories. It consists of 40 000 line segments.
- **Hurricane:** This data set [44] has 2-dimensional track data of Atlantic hurricanes from 1950 to 2006. It has 608 trajectories with 18 343 line segments.
- **Deer:** This data set [44] corresponds to 2-dimensional movements of deers in 1995. There exist 32 trajectories which correspond to 20 033 line segments.

Encoding. The Lsun data set consists of rational numbers. Similarly as in previous work [34], we scaled the data to an integer representation with a factor of 10^6 to use efficient S2PC protocols on integers [16]. The S1, Deer, Hurricane, and Travel data sets were already represented by integers.

[8]https://www.orange.fr

5.2 Clustering Quality

In the following, we compare the clustering quality of the output of DBSCAN (which is exactly realized by ppDSCAN) to the results of the well-known K-means algorithm on the data sets Lsun and S1. Afterwards, we demonstrate that our simplified TRACLUS distance (cf. §4.4) provides a better clustering quality than the original tripartite distance of TRACLUS (cf. §3.1.2) on our tested data sets.

5.2.1 Comparison between DBSCAN and K-means. As clustering is an unsupervised ML technique, it is normally not possible to calculate an *accuracy*, meaning the proportion of correct predictions/classifications in supervised machine learning. However, Lsun and S1 are artificial data sets, created to benchmark machine learning algorithms, so their ground truths are known. For this reason, we are able to evaluate the clustering quality of the outputs of K-means and DBSCAN with the Adjusted Rand Index (ARI, cf. §3.2) which is a widely used *external* clustering quality measure [6, 69].

For a fair comparison, we provide both clustering algorithms with optimal values for their parameters. For K-means, centroids are initialized at random, and we provide the right amount of clusters ($K = 3/15$). For DBSCAN, we choose the parameters as described in §4.3.3: $\epsilon = 2 \times 10^{11}$ and $\mathsf{minPts} = 4$ with Lsun, $\epsilon = 2.25 \times 10^9$ and $\mathsf{minPts} = 50$ with S1.

Note that we use the Squared Euclidean Distance for DBSCAN such that the output of DBSCAN is equal to the output of ppDBSCAN. Thus, the insights derived from the clustering quality evaluation in this section are directly transferable to the clustering quality of ppDSBCAN.

The results shown in Tab. 1 are averaged across 10 experiments. DBSCAN finds a better partitioning than K-means for both data sets. Especially for Lsun, DBSCAN performs significantly better than K-means. This is easily explainable by the data distribution of both data sets (cf. Fig. 4a and Fig. 4b). K-means cannot correctly split Lsun, because it only detects convex clusters which works relatively well for the mostly round shaped clusters in S1, but not for the two rectangular shaped ones in Lsun.

5.2.2 Validation of Simplified and Approximated TRACLUS Distance. We compare the clustering quality of ppTRACLUS with the original TRACLUS. As no ground truth is known for the trajectory data sets Hurricane, Deer, and Travel, we rely for this on well established *internal* clustering quality indices [6, 41]: the Silhouette Coefficient (SC) [59], $\mathrm{SC}_{\mathrm{noise}}$, and Density-Based Clustering Validation (DBCV) [41] (cf. §3.2). We conduct experiments with various values for ϵ' and minLns for the Hurricane data set [44], the Deer data set [44], and the Travel data set. As in [44], the optimal values for ϵ' and minLns are computed with simulated annealing. The ϵ' values differ for TRACLUS and ppTRACLUS because of the different distance measures. Note that all ϵ' for an Experiment $i \in \{1, 2\}$ result in the same entropy level in simulated annealing.

Tab. 2 contains the results for the Hurricane data set. In Experiment 1, TRACLUS outputs one cluster less than ppTRACLUS. Moreover, the number of elements marked as noise and the number of clusters are larger with ppTRACLUS than with the original plaintext TRACLUS. Nevertheless, we observe that ppTRACLUS outputs better results with respect to SC, $\mathrm{SC}_{\mathrm{noise}}$, and DBCV than the original TRACLUS. Results for the Deer data are given in §D.1.

Table 2: Clustering quality assessment for TRACLUS and ppTRACLUS on the Hurricane data set. For all scores larger values are better (best marked in bold).

Data Set	Score	Experiment 1		Experiment 2	
		TRACLUS	ppTRACLUS	TRACLUS	ppTRACLUS
Hurricane	(ϵ', minLns)	(24, 5)	(5000, 5)	(4, 5)	(2250, 5)
	# of Clusters	2	3	11	13
	Noise	129	150	597	645
	SC	0.27	**0.87**	0.44	0.79
	SC_{noise}	0.27	**0.86**	0.42	0.76
	DBCV	0.72	**0.97**	0.64	0.76

Table 3: Results of Experiment 1 of the clustering quality assessment for TRACLUS, ppTRACLUS, and ppTRACLUS' on the Travel data set. The best results are marked in bold.

	Experiment 1		
	TRACLUS	ppTRACLUS	ppTRACLUS'
(ϵ', minLns)	(4200, 3)	$(450 \times 10^6, 3)$	$(450 \times 10^6, 3)$
# of Clusters	1	2	48
Noise	796	13092	13506
SC	N/A	0.83	**0.98**
SC_{noise}	N/A	0.56	**0.65**
DBCV	N/A	**0.67**	0.37

Tab. 3 contains the results of Experiment 1 of ppTRACLUS and TRACLUS for the Travel data set. Tab. 6 in §D.1 contains the results of Experiment 2 of ppTRACLUS and TRACLUS for the Travel data set. We notice the same behavior with respect to the quality evaluation metrics as for the Hurricane and Deer data sets shown in Tab. 2 and Tab. 5 in §D.1. In Experiment 1, the number of input records marked as noise by ppTRACLUS is larger than the number of elements marked as noise by the original plaintext TRACLUS. However, when fine-tuning the values of ϵ' and minLns, it is possible to decrease the number of elements marked as noise (cf. Tab. 6 in §D.1 and §5.4 in [44]). Both algorithms output relatively few clusters while this particular data set (illustrating peoples' travel patterns) may need more precision (i.e., more clusters). One illustrative example is the identification of typical routes between two locations A and B. In this example and for this particular data set, ppTRACLUS groups all line segments in one cluster because of its expansion method (cf. Line 19 of List. 3). We propose to reduce *maxIterations* to 1 in order to take only the first-level neighbors into account and to obtain a larger variety of clusters (containing trajectories that reach B by passing through different locations $L_i, 0 \leq i \leq maxLoc$) and call this adaption ppTRACLUS'. We run the same two experiments for ppTRACLUS'. Our results show that the number of clusters increases with good results for SC, SC_{noise}, and DBCV.

To summarize, our simplified and approximated distance measure d_{pptrac} tends to create a greater number of clusters (resulting in smaller clusters in average) and marks more elements as outliers. Nevertheless, our quality evaluation with well-established clustering quality indices surprisingly gives even better results for the three analyzed data sets showing that its performance is comparable to the original tripartite distance measure of TRACLUS.

Intuition for better Clustering Quality of ppTRACLUS. Intuitively, our approximated distance d_{pptrac} summarizes all possible location-based differences between two line segments while the original tripartite distance explicitly focuses on horizontal, vertical, and angular distance. Hence, the original distance might unnecessarily amplify differences of the three different distance types although those might be partially equalized when considering the combined effect as done by d_{pptrac}. It follows that our approximation might generalize better than the original which could be the reason for the better clustering results. To conclude, ppTRACLUS with our d_{pptrac} offers efficient privacy protection with high clustering quality.

5.3 Runtime Comparison and Evaluation

In this section, we compare the runtime of our ppDBSCAN to two state-of-the-art fully privacy-preserving clustering schemes presented in [34, 48]. These two works focus on the K-means algorithm which is significantly less powerful than DBSCAN and, thus, often yields worse clustering results than DBSCAN (cf. §1). We do on purpose not experimentally compare to the previous works on private DBSCAN as all of these works leak intermediate information whereas our protocol does not (cf. §2) . Moreover, none of these works was implemented and experimentally evaluated.

Tab. 4 contains the runtimes of two state-of-the-art private K-means protocols [34, 48] with the Lsun data set and of [48] with the S1 data set. Both works use a better network than we do: [34] uses a single Intel i7-3770 with 3.4 GHz and 20 GB RAM. [48] ran their experiments locally on an Intel Core i7, 2.6 GHz with 12 GB RAM over a simulated LAN with 10 Gbit/s and 0.02 ms RTT. For both works, we present the best achieved runtimes, which is in [48] an exact calculation with 15/30 iterations. [34] simplifies the original K-means algorithm to get rid of divisions with encrypted denominators that are originally needed to update the centroids as they are expensive to realize with HE. Their fastest approximation needs 15.47 h for one iteration but it takes 40 iterations until convergence and sacrifices some accuracy for this speed-up. Our runtime for Lsun is averaged across 10 experiments. Because of time constraints, we approximate the overall runtime of our protocol for the S1 data set by multiplying the average runtime of one iteration by the data set size and adding the average distance calculation time. We use the same DBSCAN parameters as in §5.2.

Table 4: Runtime comparison of private clustering schemes.

Algorithm	Privacy-preserving K-means		ppDBSCAN
Data Set	Jäschke et al. [34]	Mohassel et al. [48]	This work
Lsun	25.79 days	22.21*s*	420.72*s*
S1	-	1,472.60*s*	620,912.70*s*

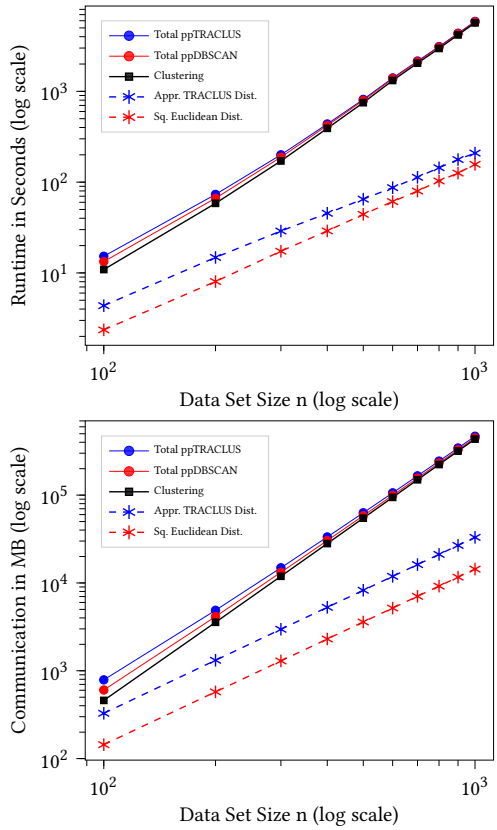


Figure 5: Runtimes and communication of our ppDBSCAN and ppTRACLUS.

Tab. 4 shows that our ppDBSCAN is about 19x (for Lsun) and 422x (for S1) slower than the private K-means protocol of [48], but more than 5 000x faster than the HE-based private K-means protocol of [34] on the small data set Lsun. While in K-means optimizations like batching [48] reduce runtime, making DBSCAN fully private adds additional computation compared to the plaintext algorithm. Again, we want to emphasize that DBSCAN is more complex than K-means but also more powerful since it can detect arbitrarily shaped clusters, automatically determine the required number of clusters, and handle noise, which results in a higher clustering quality.

5.4 Scalability

Although DBSCAN has an inherent worst case complexity of $O(n^2)$, it is one of the most used clustering algorithms because of its favorable properties. Making it private inherently implies additional overhead. Still, our protocol is practical as shown in Fig. 5 (exact numbers are provided in §D). The complexities for computing the two distances scale quadratically in the data set size, while the clustering has a low cubic complexity.

The squared Euclidean distance (SED) is applied on 2-dimensional data, d_{pptrac} on 4-dimensional line segments. *maxIterations* is set to 4. As discussed in §4.3.4, an increase of the inputs' dimension will only affect the runtime and communication of the distance calculation, but not of the clustering process. The distance calculation scales linearly in the dimensionality of the input records. Moreover, a larger number of clusters does not change our clustering and therefore also not the clustering's costs. To summarize, the private clustering component of ppDBSCAN is *independent* of the number of clusters and the data dimensionality. In contrast, private K-means [34, 48] requires to newly calculate the distances to the centroids in every iteration, such that its efficiency is heavily affected by an (1) increased input dimension, (2) a higher number of clusters (i.e., more centroids), and (3) it leaks the number of clusters K by design.

6 CONCLUSION

In this work, we presented the first fully private DBSCAN based on secure two-party computation. We designed efficient protocols for ppDBSCAN and introduced a partial parallelization for the clustering. Furthermore, we showed that our protocols can be extended to other density-based clustering algorithms by introducing the first private trajectory clustering which has interesting real-world applications for financial time series forecasts or analyzing people's movements in a pandemic. We designed a S2PC-friendly approximated distance measure for trajectories and evaluated its quality showing that it can even offer a better clustering quality than the original TRACLUS distance. Finally, we demonstrated ppDBSCAN's efficiency in terms of runtime and communication with benchmarks on real-world and public data sets and compared its overhead to state-of-the-art private K-means [34, 48] whose limitations we overcome.

ACKNOWLEDGMENTS

This project received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement no. 850990 PSOTI) and from the PAPAYA project funded by the European Union's Horizon 2020 research and innovation program (grant agreement no. 786767). It was co-funded by the Deutsche Forschungsgemeinschaft (DFG) – SFB 1119 CROSSING/236615297 and GRK 2050 Privacy & Trust/251805230, and by the German Federal Ministry of Education and Research and the Hessen State Ministry for Higher Education, Research and the Arts within ATHENE.

REFERENCES

[1] M. Ahmed, A. N. Mahmood, and Md. R. Islam. 2016. A Survey of Anomaly Detection Techniques in Financial Domain. In *Future Generation Computer Systems*.
[2] U. M. Aïvodji, K. Huguenin, M. Huguer, and M. Killijian. 2018. Sride: A Privacy-Preserving Ridesharing System. In *WISEC*. ACM.
[3] N. Almutairi, F. Coenen, and K. Dures. 2018. Secure Third Party Data Clustering Using Φ Data: Multi-User Order Preserving Encryption and Super Secure Chain Distance Matrices. In *International Conference on Innovative Techniques and Applications of Artificial Intelligence*.
[4] A. Amirbekyan and V. Estivill-Castro. 2006. Privacy Preserving DBSCAN for Vertically Partitioned Data. In *Intelligence and Security Informatics*. Springer.

[5] I. V. Anikin and R. M. Gazimov. 2017. Privacy Preserving DBSCAN Clustering Algorithm for Vertically Partitioned Data in Distributed Systems. In *International Siberian Conference on Control and Communications*. IEEE.

[6] O. Arbelaitz, I. Gurrutxaga, J. Muguerza, J. M. PéRez, and I. Perona. 2013. An Extensive Comparative Study of Cluster Validity Indices. *Pattern Recognition* (2013).

[7] G. Asharov, Y. Lindell, T. Schneider, and M. Zohner. 2013. More Efficient Oblivious Transfer and Extensions for Faster Secure Computation. In *CCS*. ACM.

[8] M.-F. Balcan, T. Dick, Y. Liang, W. Mou, and H. Zhang. 2017. Differentially Private Clustering in High-Dimensional Euclidean Spaces. In *International Conference on Machine Learning (ICML)*. PMLR.

[9] A. Bampoulidis, A. Bruni, L. Helminger, D. Kales, C. Rechberger, and R. Walch. 2020. Privately Connecting Mobility to Infectious Diseases via Applied Cryptography. https://eprint.iacr.org/2020/522.

[10] D. Beaver. 1991. Efficient Multiparty Protocols Using Circuit Randomization. In *CRYPTO*. Springer.

[11] M. Bellare, V. T. Hoang, S. Keelveedhi, and P. Rogaway. 2013. Efficient Garbling from a Fixed-Key Blockcipher. In *S&P*. IEEE.

[12] P. Besse, B. Guillouet, J.-M. Loubes, and F. Royer. 2016. Review & Perspective for Distance Based Clustering of Vehicle Trajectories. In *Transactions on Intelligent Transportation Systems*. IEEE.

[13] P. Bunn and R. Ostrovsky. 2007. Secure Two-Party K-means Clustering. In *CCS*. ACM.

[14] H. Chaudhari, R. Rachuri, and A. Suresh. 2020. Trident: Efficient 4PC Framework for Privacy Preserving Machine Learning. In *NDSS*. The Internet Society.

[15] H. Chen, I. Chillotti, Y. Dong, O. Poburinnaya, I. Razenshteyn, and M. S. Riazi. 2020. SANNS: Scaling Up Secure Approximate k-Nearest Neighbors Search. In *USENIX Security*. USENIX.

[16] D. Demmler, T. Schneider, and M. Zohner. 2015. ABY - A Framework for Efficient Mixed-Protocol Secure Two-Party Computation. In *NDSS*. The Internet Society.

[17] Z. Erkin, M. Franz, J. Guajardo, S. Katzenbeisser, I. Lagendijk, and T. Toft. 2009. Privacy-Preserving Face Recognition. In *PoPETS*. Springer.

[18] Z. Erkin, J. R. Troncoso-pastoriza, R. L. Lagendijk, and F. Perez-Gonzalez. 2013. Privacy-preserving Data Aggregation in Smart Metering Systems: An Overview. In *IEEE Signal Processing Magazine*.

[19] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al. 1996. A Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *SIGKDD Conference on Knowledge Discovery and Data Mining*. ACM.

[20] P. Fränti and S. Sieranoja. 2018. K-means Properties on Six Clustering Benchmark Datasets. In *Applied Intelligence*. Springer.

[21] F. D. Garcia and B. Jacobs. 2010. Privacy-friendly Energy-metering via Homomorphic Encryption. In *International Workshop on Security and Trust Management*. Springer.

[22] C. Gentry. 2009. *A fully Homomorphic Encryption Scheme*. Stanford University.

[23] Z. Gheid and Y. Challal. 2016. Efficient and Privacy-Preserving K-means Clustering for Big Data Mining. In *TrustCom/BigDataSE/ISPA*. IEEE.

[24] O. Goldreich, S. Micali, and A. Wigderson. 1987. How to Play ANY Mental Game. In *STOC*. ACM.

[25] Q. Guo, X. Lu, Y. Gao, J. Zhang, B. Yan, D. Su, A. Song, X. Zhao, and G. Wang. 2017. Cluster Analysis: A New Approach for Identification of Underlying Risk Factors for Coronary Artery Disease in Essential Hypertensive Patients. In *Scientific Reports*.

[26] P. Hallgren, C. Orlandi, and A. Sabelfeld. 2017. PrivatePool: Privacy-Preserving Ridesharing. In *Computer Security Foundations (CSF)*. IEEE.

[27] K. Hamada, R. Kikuchi, D. Ikarashi, K. Chida, and K. Takahashi. 2012. Practically Efficient Multi-party Sorting Protocols from Comparison Sort Algorithms. In *International Conference on Information Security and Cryptology (ICISC)*. Springer.

[28] M. Huang, Q. Bao, Y. Zhang, and W. Feng. 2019. A Hybrid Algorithm for Forecasting Financial Time Series Data Based on DBSCAN and SVR. In *Information*.

[29] L. Hubert and P. Arabie. 1985. Comparing Partitions. In *Journal of Classification*. Springer.

[30] Y. Ishai, J. Kilian, K. Nissim, and E. Petrank. 2003. Extending Oblivious Transfers Efficiently. In *CRYPTO*. Springer.

[31] G. Jagannathan, K. Pillaipakkamnatt, R. Wright, and D. Umano. 2010. Communication-efficient Privacy-Preserving Clustering. In *Transactions on Data Privacy*. Springer.

[32] G. Jagannathan and R. N. Wright. 2005. Privacy-Preserving Distributed k-Means Clustering over Arbitrarily Partitioned Data. In *SIGKDD International Conference on Knowledge Discovery in Data Mining*. ACM.

[33] K. Järvinen, H. Leppäkoski, E. S. Lohan, P. Richter, T. Schneider, O. Tkachenko, and Z. Yang. 2019. PILOT: Practical Privacy-Preserving Indoor Localization using OuTsourcing. In *EuroS&P*. IEEE.

[34] A. Jäschke and F. Armknecht. 2018. Unsupervised Machine Learning on Encrypted Data. In *SAC*. Springer.

[35] S. Jha, L. Kruger, and P. McDaniel. 2005. Privacy Preserving Clustering. In *ESORICS*. Springer.

[36] D. Jiang, A. Xue, S. Ju, W. Chen, and H. Ma. 2008. Privacy-preserving DBSCAN on Horizontally Partitioned Data. In *International Symposium on IT in Medicine and Education*. IEEE.

[37] S. Kamara and M. Raykova. 2011. Secure Outsourced Computation in a Multi-Tenant Cloud. In *IBM Workshop on Cryptography and Security in Clouds*.

[38] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. 1983. Optimization by Simulated Annealing. In *SCIENCE*.

[39] V. Kolesnikov and T. Schneider. 2008. Improved Garbled Circuit: Free XOR Gates and Applications. In *ICALP*. Springer.

[40] D. Kopanaki, N. Pelekis, A. Gkoulalas-Divanis, M. Vodas, and Y. Theodoridis. 2012. A Framework for Mobility Pattern Mining and Privacy- Aware Querying of Trajectory Data. In *Hellenic Data Management Symposium*.

[41] H.-P. Kriegel and M. Pfeifle. 2005. Density-Based Clustering of Uncertain Data. In *SIGKDD Conference on Knowledge Discovery and Data Mining*. ACM.

[42] K. A. Kumar and C. P. Rangan. 2007. Privacy Preserving DBSCAN Algorithm for Clustering. In *Advanced Data Mining and Applications*. Springer.

[43] K. Kursawe, G. Danezis, and M. Kohlweiss. 2011. Privacy-friendly Aggregation for the Smart-Grid. In *PETS*. Springer.

[44] J.-G. Lee, J. Han, and K.-Y. Whang. 2007. Trajectory Clustering: a Partition-and-Group Framework. In *SIGMOD International Conference on Management of Data*. ACM.

[45] D. Liu, E. Bertino, and X. Yi. 2014. Privacy of Outsourced K-Means Clustering. In *ASIACCS*. ACM.

[46] J. Liu, L. Xiong, J. Luo, and J. Z. Huang. 2013. Privacy Preserving Distributed DBSCAN Clustering. In *Transactions on Data Privacy*.

[47] P. Mohassel and P. Rindal. 2018. ABY^3: A Mixed Protocol Framework for Machine Learning. In *CCS*. ACM.

[48] P. Mohassel, M. Rosulek, and N. Trieu. 2020. Practical Privacy-Preserving K-means Clustering. In *PoPETS*. Sciendo.

[49] D. Moulavi, P. A. Jaskowiak, R. J. G. B. Campello, A. Zimek, and J. Sander. 2014. Density-based clustering validation. In *International Conference on Data Mining*. SIAM.

[50] M. Naor and B. Pinkas. 1999. Oblivious Transfer and Polynomial Evaluation. In *STOC*. ACM.

[51] L. Ni, C. Li, X. Wang, H. Jiang, and J. Yu. 2018. DP-MCDBSCAN: Differential Privacy Preserving Multi-Core DBSCAN Clustering for Network User Data. In *IEEE Access*. IEEE.

[52] E. Pagnin, G. Gunnarsson, P. Talebi, C. Orlandi, and A. Sabelfeld. 2019. TOPPool: Time-aware Optimized Privacy-Preserving Ridesharing. In *PoPETS*. Sciendo.

[53] N. G. Pavlidis, V. P. Plagianakos, D. K. Tasoulis, and M. N. Vrahatis. 2006. Financial Forecasting through Unsupervised Clustering and Neural Networks. *Operational Research* (2006).

[54] N. Pelekis, A. Gkoulalas-Divanis, M. Vodas, A. Plemenos, D. Kopanaki, and Y. Theodoridis. 2012. Private-HERMES: A Benchmark Framework for Privacy-Preserving Mobility Data Querying and Mining Methods. In *Extending Database Technology*. ACM.

[55] G. Punj and D. W. Stewart. 1983. Cluster Analysis in Marketing Research: Review and Suggestions for Application. In *Journal of Marketing Research*.

[56] Y. Qi and M. J. Atallah. 2008. Efficient Privacy-preserving K-nearest Neighbor Search. In *International Conference on Distributed Computing Systems*. IEEE.

[57] M. S. Rahman, A. Basu, and S. Kiyomoto. 2017. Towards Outsourced Privacy-Preserving Multiparty DBSCAN. In *Pacific Rim International Symposium on Dependable Computing*. IEEE.

[58] D. Rathee, T. Schneider, and K. K. Shukla. 2019. Improved Multiplication Triple Generation over Rings via RLWE-Based AHE. In *CANS*. Springer.

[59] P. Rousseeuw. 1987. Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis. In *Journal of Computational and Applied Mathematics*.

[60] S. Samet, A. Miri, and L. Orozco-Barbosa. 2007. Privacy Preserving K-means Clustering in Multi-Party Environment. In *SECRYPT*.

[61] J. Sander, M. Ester, H.-P. Kriegel, and X. Xu. 1998. Density-Based Clustering in Spatial Databases: The Algorithm GDBSCAN and Its Applications. In *SIGKDD Conference on Knowledge Discovery and Data Mining*. ACM.

[62] A. Sangers, M. van Heesch, T. Attema, T. Veugen, M. Wiggerman, J. Veldsink, O. Bloemen, and D Worm. 2019. Secure Multiparty PageRank Algorithm for Collaborative Fraud Detection. In *FC*. Springer.

[63] U. Stemmer. 2020. Locally Private K-means Clustering. In *SIAM Symposium on Discrete Algorithms*. ACM.

[64] D. Su, J. Cao, N. Li, E. Bertino, and H. Jin. 2016. Differentially Private K-Means Clustering. In *Conference on Data and Application Security and Privacy*. ACM.

[65] D. Su, J. Cao, N. Li, E. Bertino, M. Lyu, and H. Jin. 2017. Differentially Private K-Means Clustering and a Hybrid Approach to Private Optimization. In *Transactions on Privacy and Security*. ACM.

[66] C. Troncoso, M. Payer, J.-P. Hubaux, M. Salathé, J. Larus, E. Bugnion, W. Lueks, T. Stadler, A. Pyrgelis, D. Antonioli, et al. 2020. Decentralized Privacy-Preserving Proximity Tracing. *IEEE Data Engineering Bulletin* (2020).

[67] A. Ultsch. 2005. Clustering wih som: U*c. In *Workshop on Self-Organizing Maps*.

[68] J. Vaidya and C. Clifton. 2003. Privacy-Preserving k-Means Clustering over Vertically Partitioned Data. In *SIGKDD Conference on Knowledge Discovery and Data Mining*. ACM.

[69] N. X. Vinh, J. Epps, and J. Bailey. 2010. Information Theoretic Measures for Clusterings Comparison: Variants, Properties, Normalization and Correction for Chance. *The Journal of Machine Learning Research* (2010).
[70] W. Wu, J. Liu, H. Wang, J. Hao, and M. Xian. 2020. Secure and Efficient Outsourced K-means Clustering using Fully Homomorphic Encryption with Ciphertext Packing Technique. In *Transactions on Knowledge and Data Engineering*. IEEE.
[71] W. M. Wu and H. K. Huang. 2015. A DP-DBScan Clustering Algorithm based on Differential Privacy Preserving. In *Computer Engineering and Science*.
[72] W. Xu, L.. Huang, Y. Luo, Y.. Yao, and W. W. Jing. 2007. Protocols for Privacy-Preserving DBSCAN Clustering. In *Int. Journal of Security and Its Applications*.
[73] A. C. Yao. 1986. How to Generate and Exchange Secrets. In *FOCS*. IEEE.
[74] S. Zahur, M. Rosulek, and D. Evans. 2015. Two Halves Make a Whole - Reducing Data Transfer in Garbled Circuits Using Half Gate. In *EUROCRYPT*. Springer.

A PLAINTEXT DBSCAN CLUSTERING

List. 4 shows the plaintext DBSCAN clustering as described in §3.1. For each unclassified data record *elem* (i.e., it was not assigned to a cluster yet) of the input data set, it is checked if *elem* has at least minPts neighbors (i.e., at least minPts elements in a radius of ϵ, cf. Line 9). If this is the case, a new cluster c containing *elem* and its neighbors is created in Lines 10 to 12. Otherwise, *elem* is considered as outlier and marked as noise (cf. Line 14). The function *expandCluster*(..) recursively iterates through all neighbors of *elem* and the neighbors of the neighbors to check if they are unclassified and core elements (cf. Line 21). If this is the case, they will also be added to cluster c.

Listing 4: Plaintext DBSCAN Clustering

```
1  Input: input, eps, minPts
2  Output: clusters
3
4  dbscan(vector data,eps,minPts):
5    vector clusters
6    for each elem in input:
7      if(elem.isUnclassified()): // no cluster assigned
8        neighbors = elem.getNeighbors()
9        if (|neighbors| >= minPts): // big enough?
10         c = new Cluster()
11         clusters.push(c)
12         expandCluster(elem,neighbors,eps,minPts,c)
13       else://noise
14         elem[i].markAsNoise()
15   return clusters
16
17 expandCluster(p,queue,eps,minPts,c):
18   c.add(p)
19   while (queue.notEmpty()):
20     q = queue.pop()
21     if (q.isUnclassified() && |q.getNeighbors()|>minPts):
22       queue.append(q.getNeighbors())
23     if (q.isUnclassified() || q.isNoise()):
24       c.add(q)
```

B S2PC TECHNIQUES

Garbled Circuits. Yao's Garbled Circuits (GC) [73] allow to securely evaluate Boolean circuits between two parties. One party, called *garbler*, encrypts the gates of the circuit using random keys for each wire and sends it (now called *garbled circuit*) together with the keys associated to his inputs to the other party, called *evaluator*. The evaluator receives the keys associated to his inputs via Oblivious Transfer (OT) [7, 30, 50] from the garbler. After decrypting the garbled circuit, the parties jointly decode the output keys. ABY includes state-of-the-art optimizations like Free-XOR [39], fixed-key AES [11], and Half-Gates [74]. XOR gates are for free and AND gates cost 2κ bits of communication in an input-independent setup phase, where $\kappa = 128$ is the symmetric security parameter [16].

Arithmetic Sharing. Arithmetic sharing uses additive shares of l-bit integers in the ring $\mathbb{Z}_{2^l}$. To share the secret value x, the data owner (who can be one of the parties) chooses a random value $r \in_R \mathbb{Z}_{2^l}$ and sets $\langle x \rangle_i^A = x - r$ and $\langle x \rangle_{i-1}^A = r$. Party P_i receives share $\langle x \rangle_i^A$ and party P_{i-1} gets $\langle x \rangle_{i-1}^A$. When provided with both shares, any party can reconstruct the secret by addition of the shares. While secure addition can be executed locally, secure multiplication with arithmetic shares requires interaction and multiplication triples [10] that can be efficiently precomputed with OTs [16] or homomorphic encryption [58].

Conversions. Converting an Arithmetic share to Yao sharing (A2Y) costs $6l\kappa$ communication bits and $12l$ AES operations [16]. Yao to Arithmetic (Y2A) sharing conversion costs $l\kappa + (l2 + l)/2$ and $6l$ AES operations. Both conversions require two messages. l is the bit-length of the Arithmetic share.

C SECURITY DISCUSSION

In this section, we sketch why ppDBSCAN (§4.3) and ppTRACLUS (§4.4) are privacy-preserving, i.e., a semi-honest adversary learns nothing beyond what can be inferred from the output of the protocols. Generally, the security of ppDBSCAN and ppTRACLUS follows directly from the provable security of the used S2PC techniques (cf. §B): Yao's Garbled circuits (GC) [73] and Arithmetic sharing [24].

At the beginning of the protocol, the data owners secret-share their input data records among them (2PC) or among two non-colluding parties (Outsourcing [37]). Hence, those have only access to indistinguishable random secret-shares that do not leak anything about the input data. Parameters can similarly be input as random secret-shares that do not leak any information. In the first step, the distance calculation (cf. §4.3.2 for ppDBSCAN and §4.4 for ppTRACLUS), no secret-shares are opened and only conversions between Arithmetic sharing and Yao's GC are run which are provably secure [16]. Similar reasoning applies for the second phase, the clustering (cf. §4.3), if *maxIterations* is set to the data set size n. To enhance efficiency, *maxIterations* can be fixed or it can be checked in a privacy-preserving manner using secure computation if the vector *elem.neighbors* has changed. This, however, leaks one bit of information implying some information about the data distribution (e.g., how elongated clusters can be).

To summarize, ppDBSCAN and ppTRACLUS can be instantiated fully privacy-preserving such that all information remain secret-shared during the execution of the entire protocol.

D BENCHMARK RESULTS

In this section, we provide more details and additional results of our benchmarks of ppDBSCAN and ppTRACLUS.

D.1 Additional Experiments for Clustering Quality of ppTRACLUS

Tab. 5 presents the results of TRACLUS and ppTRACLUS on the Deer data set. It contains the number of clusters created from the Deer data set which are equal with TRACLUS and ppTRACLUS. When only one cluster is found, SC and DBCV cannot be computed.

Table 5: Clustering quality assessment for TRACLUS and ppTRACLUS on the Deer data set. For all scores larger values are better (best marked in bold).

Data Set	Score	Experiment 1		Experiment 2	
		TRACLUS	ppTRACLUS	TRACLUS	ppTRACLUS
Deer	(ϵ', minLns)	$(400, 3)$	$(1 \times 10^6, 3)$	$(282, 3)$	$(550 \times 10^3, 3)$
	# of Clusters	1	1	2	2
	Noise	1	480	20	1333
	SC	N/A	N/A	0.089	**0.36**
	SC_{noise}	N/A	N/A	0.089	**0.34**
	DBCV	N/A	N/A	0.47	**0.79**

Nevertheless, having only one cluster does not necessarily mean that the quality of the clustering algorithm is low. It simply says that the algorithm found only one group of similar elements.

Tab. 6 contains the results of Experiment 2 of TRACLUS, ppTRACLUS, and ppTRACLUS' on the Travel data set.

Table 6: Results of Experiment 2 of the clustering quality assessment for TRACLUS, ppTRACLUS, and ppTRACLUS' on the Travel data set. The best results are marked in bold.

	Experiment 2		
	TRACLUS	ppTRACLUS	ppTRACLUS'
(ϵ', minLns)	$(47 \times 10^3, 3)$	$(13.5 \times 10^9, 3)$	$(13.5 \times 10^9, 3)$
# of Clusters	1	1	2
Noise	0	359	361
SC	N/A	N/A	**0.82**
SC_{noise}	N/A	N/A	**0.81**
DBCV	N/A	N/A	**0.98**

D.2 Runtime and Communication Costs

Tab. 7 contains the runtimes (in seconds) and Tab. 8 contains the communication costs (in MB) of the SED, the approx. distance of TRACLUS, and clustering averaged over 10 experiments. The SED distance is applied on 2-dimensional data, the approx. TRACLUS distance on 4-dimensional line segments. *maxIterations* is set to 4.

Table 7: Runtimes for ppDBSCAN/ppTRACLUS.

Data Set Size n	Runtimes (s)		
	Squared Euclidean Distance	Appr. TRACLUS-Distance	Density-based Clustering
100	2.36	4.36	10.9
200	8.04	14.82	58.43
300	17.35	28.94	171.12
400	29.03	45.39	391.69
500	44.17	64.98	750.284
600	61.04	86.91	1317.70
700	79.76	113.07	2043.01
800	102.99	144.09	2970.52
900	125.82	177.48	4187.54
1000	157.07	208.92	5682.425

Table 8: Communication Costs for ppDBSCAN/ppTRACLUS.

Data Set Size n	Communication (MB)		
	Squared Euclidean Distance	Appr. TRACLUS-Distance	Density-based Clustering
100	144	328	460
200	575	1317	3566
300	1294	2968	11912
400	2300	5280	28090
500	3594	8253	54694
600	5176	11889	94315
700	7044	16185	149548
800	9201	21143	222983
900	11645	26762	317214
1000	14376	33043	434835

C FLAME: Taming Backdoors in Federated Learning (USENIX Security'22)

[NRC+22] T. D. Nguyen, P. Rieger, H. Chen, H. Yalame, H. Möllering, H. Fereidooni, S. Marchal, M. Miettinen, A. Mirhoseini, S. Zeitouni, F. Koushanfar, A.-R. Sadeghi, T. Schneider. **"FLAME: Taming backdoors in federated learning"**. In: *USENIX Security Symposium (USENIX Security)*. Online: https://ia.cr/2021/025. USENIX, 2022, S. 1415–1432. CORE Rank A*. Appendix C.

https://encrypto.de/papers/NRCYMFMMMZKSS22.pdf

FLAME: Taming Backdoors in Federated Learning

Thien Duc Nguyen and Phillip Rieger, *Technical University of Darmstadt;* Huili Chen, *University of California San Diego;* Hossein Yalame, Helen Möllering, and Hossein Fereidooni, *Technical University of Darmstadt;* Samuel Marchal, *Aalto University and F-Secure;* Markus Miettinen, *Technical University of Darmstadt;* Azalia Mirhoseini, *Google;* Shaza Zeitouni, *Technical University of Darmstadt;* Farinaz Koushanfar, *University of California San Diego;* Ahmad-Reza Sadeghi and Thomas Schneider, *Technical University of Darmstadt*

https://www.usenix.org/conference/usenixsecurity22/presentation/nguyen

This paper is included in the Proceedings of the 31st USENIX Security Symposium.

August 10–12, 2022 • Boston, MA, USA

978-1-939133-31-1

Open access to the Proceedings of the 31st USENIX Security Symposium is sponsored by USENIX.

FLAME: Taming Backdoors in Federated Learning

Thien Duc Nguyen[1], Phillip Rieger[1], Huili Chen[2], Hossein Yalame[1], Helen Möllering[1], Hossein Fereidooni[1], Samuel Marchal[3], Markus Miettinen[1], Azalia Mirhoseini[4], Shaza Zeitouni[1], Farinaz Koushanfar[2], Ahmad-Reza Sadeghi[1], and Thomas Schneider [1]
[1]*Technical University of Darmstadt, Germany*[*]; [2]*University of California San Diego, USA*; [3]*Aalto University and F-Secure, Finland*; [4]*Google, USA*

Abstract

Federated Learning (FL) is a collaborative machine learning approach allowing participants to jointly train a model without having to share their private, potentially sensitive local datasets with others. Despite its benefits, FL is vulnerable to so-called *backdoor attacks*, in which an adversary injects manipulated model updates into the federated model aggregation process so that the resulting model will provide targeted false predictions for specific adversary-chosen inputs. Proposed defenses against backdoor attacks based on detecting and filtering out malicious model updates consider only very specific and limited attacker models, whereas defenses based on differential privacy-inspired noise injection significantly deteriorate the benign performance of the aggregated model. To address these deficiencies, we introduce FLAME, a defense framework that estimates the sufficient amount of noise to be injected to ensure the elimination of backdoors. To minimize the required amount of noise, FLAME uses a model clustering and weight clipping approach. This ensures that FLAME can maintain the benign performance of the aggregated model while effectively eliminating adversarial backdoors. Our evaluation of FLAME on several datasets stemming from application areas including image classification, word prediction, and IoT intrusion detection demonstrates that FLAME removes backdoors effectively with a negligible impact on the benign performance of the models.

1 Introduction

Federated learning (FL) is an emerging collaborative machine learning trend with many applications, such as next word prediction for mobile keyboards [39], medical imaging [49], and intrusion detection for IoT [44] to name a few. In FL, clients locally train models based on local training data and then provide these model updates to a central aggregator who combines them into a global model. The global model is then propagated back to the clients for the next training iteration.

FL promises efficiency and scalability as the training is distributed among many clients and executed in parallel. In particular, FL improves privacy by enabling clients to keep their training data locally [38]. Despite its benefits, FL has been shown to be vulnerable to so-called *poisoning attacks* where the adversary manipulates the local models of a subset of clients participating in the federation so that the malicious updates get aggregated into the global model. Untargeted poisoning attacks merely aim at deteriorating the performance of the global model and can be defeated by validating the performance of uploaded models [12]. In this paper, we therefore focus on the more challenging problem of *backdoor attacks* [7, 45, 57, 59], i.e., targeted poisoning attacks in which the adversary seeks to stealthily manipulate the resulting global model in a way that attacker-controlled inputs result in incorrect predictions chosen by the adversary.

Deficiencies of existing defenses. Existing defenses against backdoor attacks can be roughly divided into two categories: The first one comprises anomaly detection-based approaches [4,9,22,51] for identifying and removing potentially poisoned model updates. However, these solutions are effective only under very specific adversary models, as they make detailed assumptions about the attack strategy of the adversary and/or the underlying distribution of the benign or adversarial datasets. If these very specific assumptions do not hold, the defenses may fail. The second category is inspired by differential privacy (DP) techniques [7,56], where individual weights[1] of model updates are clipped to a maximum threshold and random noise is added to the weights for diluting/reducing the impact of potentially poisoned model updates on the aggregated global model. In contrast to the first category, DP techniques [7,56] are applicable in a generic adversary model without specific assumptions about adversarial behavior and data distributions and are effective in eliminating the impact of malicious model updates. However, straightforward application of DP approaches severely deteriorates the benign

[*]Emails: {ducthien.nguyen, ahmad.sadeghi}@trust.tu-darmstadt.de

[1]Parameters of neural network models typically consist of 'weights' and 'biases'. For the purposes of this paper, however, these parameters can be treated identically and we will refer to them as 'weights' for brevity.

performance of the aggregated model because the amount of noise required to ensure effective elimination of backdoors also results in significant modifications of individual weights of benign model updates [7, 57].

In this paper, we develop a resilient defense against backdoors by combining the benefits of both defense types without suffering from the limitations (narrow attacker model, assumptions about data distributions) and drawbacks (loss of benign performance) of existing approaches. To this end, we introduce an approach in which detection of anomalous model updates and tuned clipping of weights are combined to minimize the amount of noise needed for backdoor removal of the aggregated model while preserving its benign performance.

Our Goals and Contributions. We present FLAME, a resilient aggregation framework for FL that eliminates the impact of backdoor attacks while maintaining the benign performance of the aggregated model. This is achieved by three modules: DP-based noising of model updates to remove backdoor contributions, automated model clustering approach to identify and eliminate potentially poisoned model updates, and model weight clipping before aggregation to limit the impact of malicious model updates on the aggregation result. The last two modules can significantly reduce the amount of random noise required by DP noising for backdoor elimination. In particular, our contributions are as follows:

- We present FLAME, a defense framework against backdoor attacks in FL that is capable of eliminating backdoors without impacting the benign performance of the aggregated model. Contrary to earlier backdoor defenses, FLAME is applicable in a *generic* adversary model, i.e., it does not rely on strong assumptions about the attack strategy of the adversary, nor about the underlying data distributions of benign and adversarial datasets (§4.1).
- We show that the amount of required Gaussian noise can be radically reduced by: a) applying our clustering approach to remove potentially malicious model updates and b) clipping the weights of local models at a proper level to constrain the impact of individual (especially malicious) models on the aggregated model. (§4.3)
- We provide a noise boundary proof for the amount of Gaussian noise required by noise injection (inspired by DP) to eliminate backdoor contributions (§5).
- We extensively evaluate our defense framework on real-world datasets from three very different application areas. We show that FLAME reduces the amount of required noise so that the benign performance of the aggregated model does not degrade significantly, providing a crucial advantage over state-of-the-art defenses using straightforward injection of DP-based noise (§7).

As an orthogonal aspect, we also consider how the privacy of model updates against an honest-but-curious aggregator can be preserved and develop a secure multi-party computation approach that can preserve the privacy of individual model updates while realizing our backdoor defense approach (§8).

2 Background and Problem Setting

2.1 Federated Learning

Federated Learning [38, 50] is a concept for distributed machine learning that links n clients and an aggregator to collaboratively build a global model G. In a training iteration $t \in \{1,\ldots,T\}$, each client $i \in \{1,\ldots,n\}$ locally trains a local model W_i with p parameters (indicating both weights and biases) $w_i^1,\ldots,w_i^p$ based on the previous global model G_{t-1} using its local data D_i and sends it to the aggregator which aggregates the received models W_i into the global model G_t.

Several aggregation mechanisms have been proposed recently: 1) *Federated Averaging* (FedAvg) [38], 2) *Krum* [9], 3) *Adaptive Federated Averaging* [42], and 4) *Trimmed mean or median* [60]. Although we evaluate FLAME's effectiveness on several aggregation mechanisms in §7.1, we generally focus on FedAvg in this work as it is commonly applied in FL [21, 28, 39, 44, 47, 50, 54] and related work on backdoor attacks [7, 22, 51, 57, 59]. In FedAvg, the global model is updated by averaging the weighted models as follows: $G_t = \Sigma_{i=1}^{n} s_i \times W_i / s$, where $s_i = \|D_i\|, s = \Sigma_{i=1}^{n} s_i$. However, in practice, a malicious client might provide falsified information about its dataset size (i.e., a large number) to amplify the relative weight of its updates [57]. Previous works often employed equal weights ($s_i = 1/n$) for the contributions of all clients [7, 51, 59]. We adopt this approach in this paper, i.e., we set $G_t = \Sigma_{i=1}^{n} W_i / n$. Further, other state-of-the-art aggregation rules, e.g., *Krum* [9], *Adaptive Federated Averaging* [42], and *Trimmed mean or median* [60] also do not consider the sizes of local training datasets by design.

2.2 Backdoor Attacks on Federated Learning

In backdoor attacks, the adversary $\mathcal{A}$ manipulates the local models W_i of k compromised clients to obtain poisoned models W_i' that are then aggregated into the global model G_t and thus affect its properties. In particular, $\mathcal{A}$ wants the poisoned model G_t' to behave normally on all inputs except for specific attacker-chosen inputs $x \in I_{\mathcal{A}}$ (where $I_{\mathcal{A}}$ denotes the so-called *trigger set*) for which attacker-chosen (incorrect) predictions should be output. Figure 1 shows common techniques used in FL backdoor attacks, including 1) data poisoning, e.g., [45, 51, 59], where $\mathcal{A}$ manipulates training datasets of models, and 2) model poisoning, e.g., [7, 57] where $\mathcal{A}$ manipulates the training process or the trained models themselves. Next, we will briefly discuss these attack techniques.

Data Poisoning. In this attack, $\mathcal{A}$ adds manipulated data $D^{\mathcal{A}}$ to the training datasets of compromised clients i by flipping data labels, e.g., by changing the labels of a street sign database so that pictures showing a 30 km/h speed limit are labeled as 80 km/h [51], or, by adding triggers into data samples (e.g., a specific pixel pattern added to images [59]) in combination with label flipping. We denote the fraction

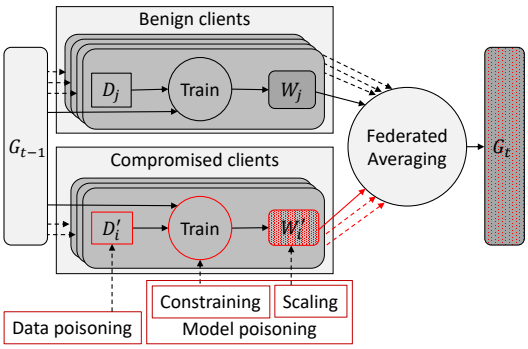


Figure 1: An overview of backdoor attacks.

of injected poisoned data $D_i^{\mathcal{A}}$ in the overall poisoned training dataset D'_i of client i as *Poisoned Data Rate (PDR)*, i.e., $PDR_i = |D_i^{\mathcal{A}}|/|D'_i|$.

Model Poisoning. This attack technique requires that $\mathcal{A}$ can fully control a number of clients. $\mathcal{A}$ poisons the training datasets of these clients and manipulates how they execute the training process by modifying parameters and scaling the resulting model update to maximize the attack impact while evading the aggregator's anomaly detector [7, 57]. This is done by (1) scaling up the weights of malicious model updates to maximize attack impact (e.g., *model-replacement* attack [7], or, *projected gradient descent (PGD) attack with model replacement* [57]), or, scaling down model updates to make them harder to detect (e.g., *train-and-scale* [7]) and (2) constraining the training process itself to minimize the deviation of malicious models from benign models to evade anomaly detection (e.g., *constrain-and-scale* attack [7]).

2.3 Adversary Goals and Capabilities

The goals of the adversary are two-fold:

Impact: The adversary $\mathcal{A}$ aims to manipulate the global model G so that the modified model G' provides incorrect predictions $f(G',x) = c' \neq f(G,x)$ for any inputs $x \in I_{\mathcal{A}}$, where $I_{\mathcal{A}}$ is the so-called *trigger set* consisting of specific attacker-chosen inputs and c' denotes the incorrect prediction chosen by the adversary.

Stealthiness: To make the poisoned model G' hard to detect by aggregator A, it should closely mimic the behavior of G on all other inputs not in $I_{\mathcal{A}}$, i.e.:

$$f(G',x) = \begin{cases} c' \neq f(G,x) & \forall x \in I_{\mathcal{A}} \\ f(G,x) & \forall x \notin I_{\mathcal{A}} \end{cases} \quad (1)$$

Additionally, to make poisoned models as indistinguishable as possible from benign models, the distance (e.g., euclidean) between a poisoned model W' and a benign model W must be smaller than a threshold η denoting the distinction capability of the anomaly detector of aggregator A, i.e., $dist(W,W') < \eta$. The adversary can estimate this distance by comparing the local malicious model to the global model or to a local model trained on benign data.

Adversarial Capabilities. In this paper, we make no specific assumptions about the adversary's behavior. We assume that the adversary $\mathcal{A}$ has full control over $k < \frac{n}{2}$ clients and their training data, processes, and parameters [7, 59]. We denote the fraction of compromised clients as Poisoned Model Rate $PMR = \frac{k}{n}$. Furthermore, $\mathcal{A}$ has full knowledge of the aggregator's operations, including potentially applied backdoor defenses. However, $\mathcal{A}$ has no control over any processes executed at the aggregator nor over the honest clients.

2.4 Preliminaries

HDBSCAN [11] is a density-based clustering algorithm that uses the distance of data points in n-dimensional space to group data points that are located near each other together into a cluster. Hereby the number of clusters is determined dynamically. Data points that do not fit to any cluster are considered outliers. However, while HDBSCAN's predecessor DBSCAN [19] uses a predefined maximal distance to determine whether two points belong to the same cluster, HDBSCAN determines this maximal distance for each cluster independently, based on the density of points. Thus, in HDBSCAN, neither the maximal distance nor the total number of clusters need to be predefined.

Differential Privacy (DP). DP is a privacy technique that aims to ensure that the outputs do not reveal individual data records of participants. DP is formally defined as follows:

Definition 1 ((ε,δ)-differential privacy)**.** *A randomized algorithm $\mathcal{M}$ is (ε,δ)-differentially private if for any datasets D_1 and D_2 that differ on a single element, and any subset of outputs $S \in Range(\mathcal{M})$, the following inequality holds:*

$$Pr[\mathcal{M}(D_1) \in S] \leq e^{\varepsilon} \cdot Pr[\mathcal{M}(D_2) \in S] + \delta.$$

Here, ε denotes the privacy bound and δ denotes the probability of breaking this bound [18]. Smaller values of ε and δ indicate stronger privacy. A commonly used approach to enforce differential privacy is adding random Gaussian noise $N(0,\sigma^2)$ to the output of the algorithm [3, 18].

3 Problem Setting and Objectives

Backdoor Characterization. Following common practice in FL-related papers (e.g., [7, 12, 22]), we represent Neural Networks (NNs) using their weight vectors, in which the extraction of weights is done identically for all models by flattening/serializing the weight/bias matrices in a predetermined order. Figure 2 shows an abstract two-dimensional representation of the weight vectors of local models compared to the global model G_{t-1} of the preceding aggregation round. Each model W_i can be characterized with two factors: *direction (angle)* and *magnitude (length)* of its weight vector $(w^1, w^2, \ldots, w^p)$. The angle between two updates W_i and W_j can be measured, e.g., by using the cosine distance metric c_{ij} as defined in (2) while their magnitude difference is measured by the L_2-norm e_{ij} as defined in (3).

$$c_{ij} = 1 - \frac{W_i W_j}{\|W_i\| \|W_j\|} = 1 - \frac{\sum_{k=1}^{p} w_i^k w_j^k}{\sqrt{\sum_{k=1}^{p} (w_i^k)^2} \sqrt{\sum_{k=1}^{p} (w_j^k)^2}} \quad (2)$$

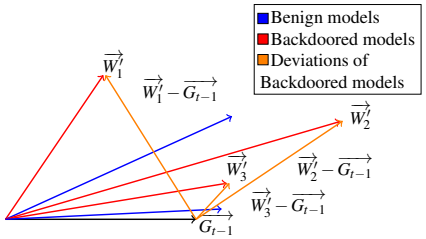


Figure 2: Weight vectors of benign and backdoored models.

$$e_{ij} = \|W_i - W_j\| = \sqrt{\sum_{k=1}^{p} (w_i^k - w_j^k)^2} \quad (3)$$

Benign and backdoored local models are shown in blue and red colors and are labeled with W_i or W'_i, respectively. Note that the benign models are typically not identical due to the potentially partially non-iid nature of their training data.

The impact of the adversarial goal (injection of a backdoor) causes a deviation in the model parameters that manifests itself as a difference in the direction and/or magnitude of the backdoored model's weight vector in comparison to benign models, e.g., the deviations among local models and to the global model G_{t-1} of the previous aggregation round. Since the adversary has full control over the training process of compromised clients, he can fully control these distances, e.g., by changing the direction (in the case of W'_1) or magnitude (in the case of W'_2) of the backdoored models' weight vectors.

Figure 2 also shows three kinds of backdoored models resulting from different types of backdoor attacks. The first type W'_1 has a similar weight vector, but a *large angular deviation* from the majority of local models and the global model. This is because such models are trained to obtain high accuracy on the backdoor task, which can be achieved by using a large poisoned data rate (*PDR*) or a large number of local training epochs (cf. Distributed Backdoor Attack (DBA) [59]). The second backdoor type W'_2 has a small angular deviation but a *large magnitude* to amplify the impact of the attack. Such models can be crafted by the adversary by *scaling up* the model weights to boost its effect on the global model (cf. *Model-replacement* attack in [7]). The third backdoor type W'_3 has a similar weight vector as benign models, the angular difference and the magnitude are not substantially different compared to benign models and, thus less distinguishable from benign models. Such *stealthy* backdoored models can be crafted by the adversary by carefully constraining the training process or scaling down the poisoned model's weights (cf. *Constrain-and-scale* attack [7] or FLIoT attack [45]).

Defense Objectives. A generic defense that can effectively mitigate backdoor attacks in the FL setting needs to fulfill the following objectives: (i) *Effectiveness*: To prevent the adversary from achieving its attack goals, the impact of backdoored model updates must be eliminated so that the aggregated global model does not demonstrate backdoor behavior. (ii) *Performance*: Benign performance of the global model must be preserved to maintain its utility. (iii) *Independence from data distributions and attack strategies*: The defense method must be applicable to generic adversary models, i.e., it must not require prior knowledge about the backdoor attack method, or make assumptions about specific data distributions of local clients, e.g., whether the data are iid or non-iid.

4 FLAME Overview and Design

We present the high-level idea of FLAME and the associated design challenges to fulfill the objectives identified in §3.

4.1 High-level Idea

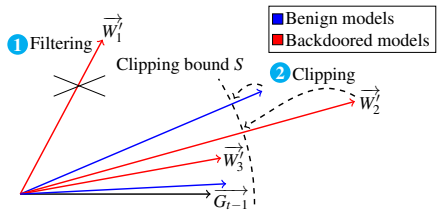
Motivation. Earlier works (e.g., Sun *et al.* [56]) use differential privacy-inspired noising of the aggregated model for eliminating backdoors. They determine the sufficient amount of noise to be used empirically. In the FL setting this is, however, challenging, as one cannot in general assume the aggregator to have access to training data, in particular to poisoned datasets. What is therefore needed is a generic method for determining how much noise is sufficient to remove backdoors effectively. On the other hand, the more noise is injected into the model, the more its benign performance will be impacted.

FLAME Overview. FLAME estimates the noise level required for backdoor removal in the FL setting without extensive empirical evaluation and having access to training data (this noise bound is formally proven in §5). In addition, to effectively limit the amount of required noise, FLAME uses a novel clustering-based approach to identify and remove adversarial model updates with high impact and applies a dynamic weight-clipping approach to limit the impact of models that the adversary has scaled up to boost their performance. As discussed in §3, one cannot guarantee that all backdoored models can be detected since the adversary can fully control both the angular and magnitude deviation to make the models arbitrarily hard to detect. Our clustering approach therefore aims to remove models with high attack impact (having larger angular deviation) rather than all malicious models. Fig. 3 illustrates the high-level idea of FLAME consisting of the above three components: filtering, clipping, and noising. We emphasize, however, that each of these components needs to be applied with great care, since, a naïve combination of noising with clustering and clipping leads to poor results as it easily fails to mitigate the backdoor and/or deteriorates the benign performance of the model, as we show in §C. We detail the design of each component and its use in the FLAME defense approach in §4.3.

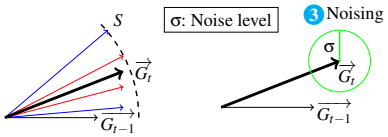
4.2 Design Challenges

To realize the high-level idea presented above, we need to solve the following technical challenges.

C_1- Filtering out backdoored models with large angular deviations in dynamic scenarios. As discussed in §3, the weight vector of a well-trained backdoored model, W', has a *higher angular difference* in comparison to weight vectors of benign models W. FLAME deploys a clustering approach to



(a) Applying clustering and clipping.



(b) Applying noising

Figure 3: High-level idea of FLAME defense.

identify such poisoned models and *remove* them from FL aggregation (detailed in §4.3.1). The effect of clustering-based filtering is shown in Fig. 3a where model W_1' is removed from the aggregated model as it does not align with the directions of benign models. In contrast to existing clustering-based defenses, we need an approach that can also work in a *dynamic attack setting*, i.e., the number of injected backdoors is unknown and may vary between training rounds. To this end, we make a key observation: clustering approaches using a fixed number of clusters $n_{cluster}$ for identifying malicious models are inherently vulnerable to attacks with varying numbers of backdoors[2] $n_{backdoor}$. This is because the adversary can likely cause at least one backdoor model to be clustered together with benign models due to the pigeonhole principle by simultaneously injecting $n_{backdoor} \geq n_{cluster}$ backdoors. We seek to solve this challenge by employing a clustering solution that dynamically determines the clusters for model updates, thereby allowing it to adapt to dynamic attacks.

C_2-Limiting the impact of scaled-up backdoors. To limit the impact of backdoored models that the adversary artificially scales up to boost the attack (e.g., W_2' in Fig. 2), the weight vectors of models with high magnitudes can be clipped [56]. The effect of clipping is shown in Fig. 3a where the weight vectors of all models with a magnitude beyond the clipping bound S (in particular, backdoored model W_2') are clipped to S by scaling down the weight vectors. The resulting clipped weight vectors are shown on the left side of Fig. 3b. The challenge here is how to select a proper clipping bound without empirically evaluating its impact on the training datasets (which are not available in the FL setting). If the applied clipping bound is too large, an adversary can boost its model W' by scaling its weights up to the clipping bound, thereby maximizing the backdoor impact on the aggregated global model G. However, if the applied clipping bound is too small, a large fraction of benign model updates W will be clipped, thereby leading to performance deterioration of the aggregated global model G on the main task. We tackle this challenge in §4.3.2, where we show how to select a clipping bound that can not be influenced by the adversary and that effectively limits the impact of scaled-up backdoored models.

C_3-Selecting suitable noise level for backdoor elimination. As mentioned in §4.1, FLAME uses model noising that applies Gaussian noise with noise level σ to mitigate the adversarial impact of backdoored models (e.g., W_3' in Fig. 2). Similar to the clipping bound, however, also here the noise level σ must be carefully selected, as it has a direct impact on the effectiveness of the defense and the model's benign performance. If it is too low, the aggregated model might retain backdoor behavior after model noising, rendering the defense ineffective, while excessive noise will degrade the utility of the aggregated model. To address this challenge, we develop an approach for reliably estimating a sufficient but minimal bound for the applied noise in §5.

4.3 FLAME Design

As discussed in §4.1, our defense consists of three main components: filtering, clipping, and noising. Figure 4 shows these components and the workflow of FLAME during training round t. Algorithm 1 outlines the procedure of FLAME. In the rest of this section, we detail the design of these components to resolve the challenges in §4.2.

Algorithm 1 FLAME

Input: n, G_0, T ▷ n is the number of clients, G_0 is the initial global model, T is the number of training iterations
Output: G_T^* ▷ G_T^* is the updated global model after T iterations
for each training iteration t in $[1,T]$ **do**
  for each client i in $[1,n]$ **do**
    $W_i \leftarrow \text{CLIENTUPDATE}(G_{t-1}^*)$ ▷ The aggregator sends G_{t-1}^* to Client i who trains G_{t-1}^* using its data D_i locally to achieve local modal W_i and sends W_i back to the aggregator.
  $(c_{11},\ldots,c_{nn}) \leftarrow \text{COSINEDISTANCE}(W_1,\ldots,W_n)$ ▷ $\forall i,j \in (1,\ldots,n)$, c_{ij} is the cosine distance between W_i and W_j
  $(b_1,\ldots,b_L) \leftarrow \text{CLUSTERING}(c_{11},\ldots,c_{nn})$ ▷ L is the number of admitted models, b_l is the index of the l^{th} model
  $(e_1,\ldots,e_n) \leftarrow \text{EUCLIDEANDISTANCES}(G_{t-1}^*,(W_1,\ldots,W_n))$ ▷ e_i is the Euclidean distance between G_{t-1}^* and W_i
  $S_t \leftarrow \text{MEDIAN}(e_1,\ldots,e_n)$ ▷ S_t is the adaptive clipping bound at round t
  for each client l in $[1,L]$ **do**
    $W_{b_l}^c \leftarrow G_{t-1} + (W_{b_l} - G_{t-1}) \cdot \text{MIN}(1,\gamma)$ ▷ Where γ $(= S_t/e_{b_l})$ is the clipping parameter, $W_{b_l}^c$ is the admitted model after clipped by the adaptive clipping bound S_t
  $G_t \leftarrow \sum_{l=1}^{L} W_{b_l}^c / L$ ▷ Aggregating, G_t is the plain global model before adding noise
  $\sigma \leftarrow \lambda \cdot S_t$ where $\lambda = \frac{1}{\varepsilon} \cdot \sqrt{2ln\frac{1.25}{\delta}}$ ▷ Adaptive noising level
  $G_t^* \leftarrow G_t + N(0,\sigma^2)$ ▷ Adaptive noising

4.3.1 Dynamic *Model Filtering*

The *Model Filtering* component of FLAME utilizes a *dynamic clustering* technique based on HDBSCAN [11] that

[2] We consider two backdoors to be independent if they use different triggers.

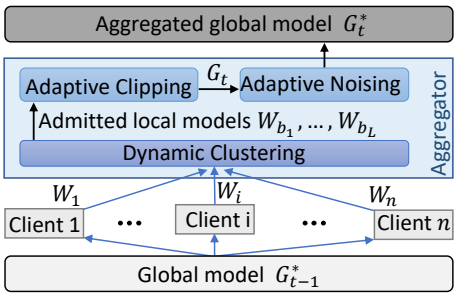


Figure 4: Illustration of FLAME's workflow in round t.

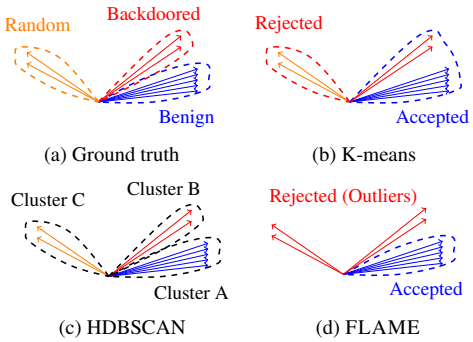


Figure 5: Comparison of clustering quality for (a) ground truth, (b) using K-means with 2 clusters as in Auror [51], (c) straightforward applied HDBSCAN and (d) our approach as in FLAME.

identifies poisoned models with high angular deviations from the majority of updates (e.g., W_1' in Fig. 3a). Existing clustering-based defenses [9, 51] identify potentially malicious model updates by clustering them into two groups where the smaller group is always considered malicious and thus removed. However, if no malicious models are present in the aggregation, this approach may lead to many models being incorrectly removed and thus a reduced accuracy of the aggregated model. These approaches also do not protect against attacks in which adversary $\mathcal{A}$ simultaneously injects multiple backdoors by using different groups of clients to inject different backdoors. If the number of clusters is fixed, there is the risk that poisoned and benign models end up in the same cluster, in particular, if models with different backdoors differ significantly. Consequently, existing model clustering methods do not adequately address challenge C_1 (§4.2). Fig. 5 shows the behavior of different clustering methods on a set of model updates' weight vectors. Fig. 5a shows the ground truth of an attack scenario where $\mathcal{A}$ uses two groups of clients: one group is used to inject a backdoor, whereas the other group provides random models with the goal of fooling clustering-based defenses. Fig. 5b shows how in this setting, K-means (as used in Auror [51]) fails to successfully separate benign and poisoned models as all poisoned models end up in the same cluster with the benign models.

To overcome the limitations of existing defenses, we design our clustering solution and ensure that: (i) it is able to handle dynamic attack scenarios where multiple backdoors are injected simultaneously, and (ii) it minimizes false positives of poisoned model identification. In contrast to existing approaches that try to place poisoned models into one cluster, our approach considers each poisoned model individually as an outlier, so that it can gracefully handle multiple simultaneous backdoors and thus address challenge C_1.

FLAME uses pairwise cosine distances to measure the angular differences between all model updates and applies the HDBSCAN clustering algorithm [11]. The advantage here is that cosine distances are not affected even if the adversary scales up model updates to boost their impact as this does not change the angle between the updates' weight vectors. Since the HDBSCAN algorithm clusters the models based on their density of the cosine distance distribution and *dynamically determines the required number of clusters*, we leverage it for our dynamic clustering approach. We describe HDBSCAN and how we apply it in detail in §E. In particular, HDBSCAN labels models as outliers if they do not fit into any cluster. This allows FLAME to effectively handle multiple poisoned models with different backdoors by labeling them as outliers. To realize this, we set the minimum cluster size to be at least 50% of the clients, i.e., $\frac{n}{2}+1$, so that the resulting cluster will contain the majority of updates (which we assume to be benign, cf. §2.3). All remaining (potentially poisoned) models are marked as *outliers*. This behavior is depicted in Fig. 5d where all the models from Clusters B and C from Fig. 5c are considered as outliers. Hence, to the best of our knowledge, our approach is the first FL backdoor defense that is able to gracefully handle also dynamic attacks in which the number of injected backdoors may vary. The clustering step is shown in lines 6-7 of Alg. 1 where L models are retained after clustering.

4.3.2 Adaptive Clipping and Noising

As discussed in §4.2 (challenges C_2 and C_3), determining a proper clipping bound and noise level for model weight clipping and noising is not straightforward. We present our new approach for selecting an effective clipping bound and reliably estimating a sufficient noise level that can effectively eliminate backdoors while preserving the performance of the main task. Furthermore, our defense approach is resilient to adversaries that dynamically adapt their attacks.

Adaptive Clipping. Fig. 6 shows the variation of the average L_2-norms of model updates of benign clients in three different datasets (cf. §6) over subsequent training rounds. We can observe that the *L_2-norms of benign model updates become smaller in later training rounds*. To effectively remove backdoors while minimizing the impact on benign updates, the clipping bound S needs to be dynamically adapted to this decreasing trend of the L_2-norm. Recall that clipping is performed after clustering by scaling down model weights so that the L_2-norm of the scaled model becomes smaller or equal to the clipping threshold. We describe how FLAME determines a proper scaling factor for each model update W_i in t^{th} training round as follows: Given the index set $(b_1, \ldots b_L)$ of the models admitted by the clustering method (line 7 of

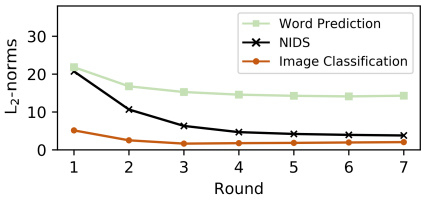


Figure 6: L_2-norms of model updates depending on the number of training rounds for different datasets.

Alg. 1), the aggregator first computes the clipping bound S_t as the median of the L_2-norms of all n model updates: $S_t = \text{MEDIAN}(e_1, \ldots, e_n)$. It should be noted that for determining the clipping bound, the rejected models are also considered to ensure that even if benign models were filtered, the computed median S_t is still determined based on benign values. However, after determining the clipping bound, only the admitted models $W_1, \ldots, W_L$ are considered for later processing. The scaling factor for the l^{th} admitted model is computed as $\gamma = \frac{S_t}{e_{b_l}}$ where e_{b_l} is the L_2-norm of the model update W_{b_l}. Clipping scales down model updates as follows: $W^c_{b_l} = G_{t-1} + (W_{b_l} - G_{t-1}) \cdot \text{MIN}(1, \gamma)$ (detailed in line 8-11 of Alg. 1) where the multiplication is computed coordinate-wise. It is worth noting that weighting contributions (i.e., adjusting scaling factor) based on client data sizes is insecure. As we point out in §2.1, the reported dataset sizes by clients cannot be trusted, i.e., the adversary can lie about their dataset sizes to maximize attack impact [57]. Hence, we follow common practice in literature and weight the contributions of all clients equally regardless of their dataset size [7, 9, 12, 59]. By using the median as the clipping bound S_t, we ensure that S_t is always in the range of the L_2-norms between benign models and the global model since we assume that more than 50% of clients are benign (cf. §2.3). We evaluate the effectiveness of the clipping approach in §B.2.

Adaptive Noising. It has been shown that by adding noise to a model's weights, the impact of outlier samples can be effectively mitigated [17]. Noise can also be added to poisoned samples (special cases of outliers) used in backdoor injection. The more noise is added to the model during the training process, the less responsive the model will be to the poisoned samples. Thus, increasing model robustness against backdoors. Eliminating backdoors utilizing noise addition is conceptually the same in a centralized or federated setting (e.g., [7, 17]): In both cases, noise is added to the model weights to smooth out the effect of poisoned data (cf. Eq. 5). The challenge is to determine as small a noise level as possible to eliminate backdoors and at the same time not deteriorate the benign performance of the model. As we discuss in detail in §5.1, the amount of noise is determined by estimating the sensitivity based on the differences (distances) among local models, which can be done without access to training data. We then add Gaussian noise to the global model G_t to yield a noised global model G^*_t as follows: $G^*_t = G_t + N(0, \sigma^2)$, see Lines 13-14 of Alg. 1 for more details. This ensures that backdoor contributions are effectively eliminated from the aggregated model. In particular, we show in §5.1 how the noise-based backdoor elimination technique can be transferred from a centralized to a federated setting by analysing the relationship between aggregated Gaussian noise applied to the global model and individual noising of each local model.

5 Security Analysis

5.1 Noise Boundary Proof of FLAME

In this section, we provide a proof to corroborate that FLAME can neutralize backdoors in the FL setting by applying strategical noising with bound analysis on the noise level. We first formulate the noise boundary guarantee of FLAME in Theorem 1. Subsequently, we explain related parameters and prove how the noise level bound for σ can be estimated. This is done by generalizing theoretical results from previous works [17, 18] to the FL setting. Then, we show how the filtering and clipping component of FLAME helps to effectively reduce the noise level bound in Theorem 2. We provide a formal proof for linear models and extend the proof to DNNs using empirical evaluation. This is because providing formal proof for DP-based backdoor security for DNN models is still an open research problem even for centralized settings.

Theorem 1. *A (ε, δ)-differentially private model with parameters G and clipping bound S_t is backdoor-free if random Gaussian noise is added to the model parameters yielding a noised version G^* of the model: $G^* \leftarrow G + N(0, \sigma^2_G)$ where the noise scale σ_G is determined by the clipping bound S_t and a noise level factor λ: $\sigma_G \leftarrow \lambda \cdot S_t$ and $\lambda = \frac{1}{\varepsilon} \cdot \sqrt{2ln\frac{1.25}{\delta}}$.*

We explore the key observation that an ML model with a sufficient level of differential privacy is *backdoor-free*. With this new definition of backdoor-free models in the DP domain, the main challenge to defeat backdoors in the FL setting is to decide a proper noise scale for the global model without knowledge of the training datasets. Furthermore, we need to minimize the amount of noise added to the global model to preserve its performance on the main task. None of the prior DP-based FL backdoor defense techniques provide a solution to the noise determination problem [56]. For the first time, FLAME presents an approach to *estimate* the proper noise scale that ensures the global model is backdoor-free. The noise boundary proof in Theorem 1 consists of two steps:
Step 1 (S1). By introducing the data hiding property of DP (Def. 1) and its *implication as the theoretical guarantee for backdoor-free* models. We also discuss function sensitivity (Def. 2) which is an important factor for selection of the DP parameters (ε, δ).
Step 2 (S2). We show how FLAME *generalizes backdoor elimination from centralized setting to federated setting* with theoretical analysis of the noise boundary (Eq. 5 and 6). FLAME *is the first FL defense against backdoors that provides noise level proof with bounded backdoor effectiveness.*

(S1) DP foundations and re-interpretation as Backdoor-free. As discussed in §2.4, by definition, DP makes the difference between data points indistinguishable. FLAME leverages this property of DP for backdoor elimination. In particular, we can consider D_1 and D_2 in Def. 1 as the benign and backdoored dataset. The inequality of DP suggests that algorithm $\mathcal{M}$ has a high probability of producing the same outputs on the benign and the poisoned dataset, meaning that the backdoor is eliminated. The noise level σ is determined based on the DP parameters (ε, δ) and the *sensitivity* of the function f defined below:

Definition 2 (Sensitivity)**.** *Given the function $f : \mathcal{D} \to \mathbb{R}^d$ where $\mathcal{D}$ is the data domain and d is the dimension of the function output, the sensitivity of the function f is defined as:*

$$\Delta = \max_{D_1,\, D_2 \in \mathcal{D}} ||f(D_1) - f(D_2)||_2, \tag{4}$$

where D_1 and D_2 differs on a single element $||D_1 - D_2||_1 = 1$.

As shown in Lemma 1 [18], this definition can be extended to datasets differing by more than one element, i.e., can be generalized to the DP in the multiple-point-difference setting.

(S2) Generalizing backdoor resilience from centralized to federated setting (FLAME). In the centralized setting, the defender has access to the model to be protected, the benign dataset, and the outlier (backdoored) samples. As such, he can estimate the sensitivity Δ for (ε, δ)-DP. When applying Gaussian noise with the noise scale $\sigma = \frac{\Delta}{\varepsilon}\sqrt{2ln\frac{1.25}{\delta}}$, the defender can enforce a lower bound on the prediction loss of the model on the backdoored samples for backdoor elimination [28]. However, this robustness rationale *cannot* be directly transferred from the centralized setting to the FL setting since the defender in the federated scenario (i.e., aggregator) only has access to received model updates, but not the datasets to estimate the sensitivity Δ for the global model.

FLAME extends DP-based noising for backdoor elimination to the federated setting based on the following observation: if one can ensure that all aggregated models are benign (i.e., backdoor-free), then it is obvious that the aggregated global model will also be backdoor-free. This intuition can be formally proven if the FL aggregation rule is Byzantine-tolerant. To ensure that any backdoor potentially present in the model is eliminated and the aggregated model is benign, a sufficient DP noise level is added to individual local models. However, since the local models are independent, adding noise to each local model is mathematically equivalent to the case where aggregated noise is added to the global model. This is conceptually equivalent to the conventional centralized setting, for which it has been formally shown that DP noise can eliminate backdoors [17]. In the following, we therefore show that adding DP noise to local models is equivalent to adding 'aggregated' DP noise to the global model.

We write the standard deviation of noise for the local models in the form $\sigma_i \leftarrow \frac{\alpha_i \cdot e_i}{\varepsilon} \cdot \sqrt{2ln\frac{1.25}{\delta}}$ where $\alpha_i = \frac{\Delta_i}{e_i}$, Δ_i and e_i is the sensitivity and the L_2 norm of the model W_i, respectively. Mathematically, the FL system with FLAME has:

$$\begin{aligned} G^* &= \frac{1}{n}\Sigma_{i=1}^n W_i^* = \frac{1}{n}[\,\Sigma_{i=1}^n\, W_i + N(0,\sigma_i^2)] \\ &= \frac{1}{n}\,\Sigma_{i=1}^2 W_i + \frac{1}{n}\Sigma_{i=1}^n N(0,\sigma_i^2) \\ &= \frac{1}{n}\,\Sigma_{i=1}^2 W_i + N(0, \frac{1}{n}\Sigma_{i=1}^n \sigma_i^2) \\ &= G + N(0,\sigma_G^2) \end{aligned} \tag{5}$$

in which W_i^* are local models and G^* the global model after adding noise $N(0,\sigma_i^2)$. Equation 5 represents the fact that adding DP noise to each local model (i.e., $W_i + N(0,\sigma_i^2)$) is equivalent to adding an 'aggregated' DP noise on the global model (i.e., $G + N(0,\sigma_G^2)$). More specifically, this equivalent Gaussian noise on the global model is the sum of Gaussian noise applied on each local model with a scaling factor $N_G = \frac{1}{n}\Sigma_{i=1}^n N_i$. Here, N_G and N_i are random variables with distribution $N(0,\sigma_G^2)$ and $N(0,\sigma_i^2)$, respectively. As such, we can compute the equivalent noise scale for the global model:

$$\begin{aligned} \sigma_G^2 &= \frac{1}{n^2}\Sigma_{i=1}^n \sigma_i^2 = (\frac{1}{\varepsilon}\sqrt{2ln\frac{1.25}{\delta}})^2 \cdot \frac{1}{n^2}\Sigma_{i=1}^n \Delta_i^2 \\ &= (\frac{1}{\varepsilon}\sqrt{2ln\frac{1.25}{\delta}})^2 \cdot \frac{1}{n^2}\Sigma_{i=1}^n \alpha_i^2 e_i^2. \end{aligned} \tag{6}$$

Equation 6 describes the relation between the DP noise added on FLAME's global model and the DP noise added on each local model. This noise scale relation in Eq. 6 together with the transformation in Eq. 5 enable FLAME to provide guaranteed security for the global model against backdoors, thereby addressing Challenge C_3 .

In Alg. 1, we use the median of Euclidean distances e_i as the upper bound S_t to clip the admitted local models (line 9-11). We hypothesize that the sensitivity of a model W_i is positively correlated with its weight magnitude $|W_i|$ (see Theorem 2 for details). In the case of linear models, the sensitivity Δ has a *linear* relation with the model weight $|\overrightarrow{w}|$ (see Eq. 8). Therefore, we use the following approximation:

$$\frac{1}{n^2}\Sigma_{i=1}^n \alpha_i^2 e_i^2 = \frac{1}{n^2}\Sigma_{i=1}^n \Delta_i^2 \approx S_t^2,$$

where S_t is the weight clipping bound. Having substituted the above approximation into Eq. 6, we can compute the noise scale of DP that FLAME deploys on the global model N_G:

$$\sigma_G \approx \frac{S_t}{\varepsilon}\sqrt{2ln\frac{1.25}{\delta}} \tag{7}$$

This concludes the proof of Theorem 1. □

FLAME's adaptive noising step applies the Gaussian noise with the noise scale computed in Eq. 7 on the global model for backdoor elimination as shown in Alg. 1, line 13-14. Note that FLAME's noising scheme is *adaptive* since the clipping bound S_t is obtained dynamically in each t^{th} epoch.

Next, we present Theorem 2 and justify how FLAME design reduces the derived noise level with *step 3 (S3)* below.

(S3) Clustering and clipping components in FLAME help to reduce the DP noise boundary. Recall that FLAME protects the FL system against backdoor attacks using three steps: clustering, clipping, and adding DP noise. The overall workflow of FLAME is shown in Fig. 4. If multiple backdoors exist in the FL system, the first two steps (clustering and clipping) can remove a subset of backdoors as shown in Fig. 3a. Note that the remaining backdoors are *'closer'* to the benign model updates in terms of both magnitude and direction. This gives us the *intuition* that removing the remaining backdoors by adding DP noise becomes easier (i.e., the noise scale σ_G is smaller) after the first two steps of FLAME.

We can see from Theorem 1 that the Gaussian noise scale σ required for backdoor resilience increases with the sensitivity of each local model Δ_i. We describe two characteristics of the model parameter W, i.e., direction and magnitude in §4. We discuss how these two factors impact the sensitivity of the model defined in Eq. 4 below.

Theorem 2. *Backdoor models with large angular deviation from benign ones, or with large parameter magnitudes have high sensitivity values* Δ.

Proving DP-based backdoor security for DNN models is still an open problem, even in the centralized setting. We, therefore, adopt a common approach in literature (e.g., [17]) by providing theoretical proof for linear models and validating it for DNNs empirically.

Proof: for a linear model f where the function output is determined by the inner product of model weight vector $\overrightarrow{w}$ and the data vector $\overrightarrow{x}$, we have

$$f(w;\, x) = \overrightarrow{w} \cdot \overrightarrow{x} = |w| \cdot |x| \cdot cos\theta, \tag{8}$$

where $\theta = < \overrightarrow{w}, \overrightarrow{x} >$ is the angle between two vectors. In this case, it is straightforward to see that if the backdoor attack changes the parameter magnitude $|w|$ or the direction θ of the model f, the resulting poisoned model f' has a large sensitivity value based on the definition in Eq. 4. □

This analysis suggests that backdoor models with large angular deviations or with large weight magnitudes have a high sensitivity value Δ. Recall that FLAME deploys dynamic clustering (§4.3.1) to remove poisoned models with large cosine distances, and employs adaptive clipping (§4.3.2) to remove poisoned models with large magnitudes. Therefore, the sensitivity of the remaining backdoor models is lower compared to the one before applying these two steps. As a result, FLAME *can use a small Gaussian noise to eliminate the remaining backdoors after applying clustering and clipping, which is beneficial for preserving the main task accuracy.*

We empirically show how the noise scale for backdoor elimination changes after applying each step of FLAME. Particularly, we measure the *smallest* Gaussian noise scale σ required to defeat *all backdoors* (i.e., $BA = 0\%$) in three settings: i) No defense components applied (which is equivalent to the previous DP-based defense [7, 18]); ii) After applying dynamic clustering; iii) After applying both dynamic clustering and adaptive clipping (which is the setting of FLAME).

Table 1: Effect of clustering and clipping in FLAME on minimal Gaussian noise level σ for backdoor elimination in the NIDS scenario, in terms of Backdoor Accuracy (*BA*) and Main Task Accuracy (*MA*).

σ	Only Noising		After Clustering		After Clustering & Clipping	
	BA	MA	BA	MA	BA	MA
0.01	100.0%	100.0%	0.0%	80.5%	0.0%	100.0%
0.08	3.5%	66.7%	0.0%	66.7%	0.0%	100.0%
0.10	0.0%	54.2%	0.0%	66.1%	0.0%	87.6%

We conduct this comparison experiment on the IoT-Traffic dataset (cf. §6). For each communication round, 100 clients are selected where $k = 40$ are adversaries. We remove the backdoor by adding Gaussian noise $N(0, \sigma^2)$ to the aggregated model. Table 1 summarizes the evaluation results in the above three settings. We can observe from the comparison results that the noise scale required to eliminate backdoors decreases after individual deployment of clustering and clipping. This corroborates the correctness of Theorem 2.

5.2 Attack and Data Distribution Assumption

In FLAME, we do not make specific assumptions about the attack and data distribution compared to the existing clustering-based defenses. Let $X = (X_1, \ldots, X_b)$ be a set of distributions of benign models $(W_1, \ldots, W_{n-k})$ where $b \leq n-k$. The deviation in X is caused by the diversity of the data. Let $X' = (X'_1, \ldots, X'_a)$ be a set of distributions of poisoned models $(W'_1, \ldots, W'_k)$ where $a \leq k$. The deviation in X' is caused by the diversity of the benign data and backdoors (e.g., poisoned data or model crafting). Existing works assume that $X'_i \approx X'_j \quad (\forall i, j : 1 \leq i, j \leq a)$ (see e.g., [22] or $X' \neq X$ [9,51]). However, this assumption does not hold in many situations because (i) there can be one or multiple attackers injecting multiple backdoors [7], or (ii) the adversary can inject one or several random (honeypot) models having a distribution X'_r that is significantly different from $X \cup (X' \setminus X'_r)$, and (iii) the adversary can control how much the backdoored models deviate from benign ones as discussed in §3. Therefore, approaches that purely divide models into two groups, e.g., K-means [51] will incorrectly classify models having distribution X'_r into the malicious group and all remaining models (having distributions drawn from $(X \cup (X' \setminus X'_r))$ into the benign group. As a result, all backdoored models having distributions drawn from $(X' \setminus X'_r)$ are classified as benign, as demonstrated in Fig. 5b. In contrast, FLAME does not rely on such specific assumptions (the adversary can arbitrarily choose X'). If the distribution X'_i of a poisoned model is similar to benign distributions in X, FLAME will falsely classify X'_i as being. But if the distribution X'_j of a poisoned model is different from the distributions in X, FLAME will identify X'_j as an outlier and classify the associated model as malicious. To identify deviating and thus potentially malicious models, FLAME leverages the HDBSCAN algorithm to identify regions of high density in the model space. Any models that are not located in the dense regions will be categorized as out-

liers, as shown in Fig. 5d. As discussed in §3, FLAME aims to remove models with distributions X'_j that have a higher attack impact compared to models with distribution X'_i. It is worth noting, however, that the impact of such remaining backdoored models will be eliminated by the noising component as shown in §5.1

Striking a balance between accuracy and security: Clustering and DP-based approaches affect model accuracy as discussed in §4.2 (Challenges C_2 and C_3). In particular, an approach that aims to maximize the number of filtered malicious models may lead to many false positives, i.e., many benign models being filtered out. Moreover, applying a very low clipping bound or a very high level of injected noise will degrade model accuracy. To address these problems, FLAME is configured so that the clustering component removes only models with high attack impact rather than all malicious models, i.e., it aims to remove the first backdoor type W'_1 as shown in Fig. 3. In addition, FLAME carefully estimates the clipping bound and noise level to ensure backdoor elimination while preserving model performance. As discussed in §4.3.2, the L_2-norms of model updates depend on the number of training rounds, dataset types, and type of backdoors. Consequently, the clipping threshold and noise level should be adapted to L_2-norms. We therefore apply the median of the L_2-norms of model updates as the clipping bound S_t (cf. Lines 9-11 of Alg. 1). This ensures that S_t is always computed between a benign local model and the global model since we assume that more than 50% of clients are benign (cf. §2.3). Further, estimating noise level based on S_t (cf. Lines 13-14 of Alg. 1) also provides a noise boundary that ensures that the global model is resilient against backdoors as discussed in §5.1. Moreover, our comparison of potential values for S_t presented in §B.2 and §B.3 shows that the chosen clipping bound and noise level provide the best balance between accuracy and security, i.e., FLAME eliminates backdoor while retaining the global model's performance on the main task.

6 Experimental Setup

We conduct all the experiments using the PyTorch deep learning framework [2] and use the source code provided by Bagdasaryan *et al.* [7], Xie *et al.* [59] and Wang *et al.* [57] to implement the attacks. We reimplemented existing defenses to compare them with FLAME.

Datasets and Learning Configurations. Following recent research on poisoning attacks on FL, we evaluate FLAME in three typical application scenarios: word prediction [35, 38–40], image classification [13, 49, 50], and an IoT intrusion detection [44, 47, 48, 54] as summarized in Tab. 2. Verification of the effectiveness of FLAME against state-of-the-art attacks in comparison to existing defenses (cf. Tab. 3 and Tab. 4) are conducted on these three datasets in the mentioned application scenarios. Experiments for evaluating specific performance aspects of FLAME are performed on the IoT dataset as it represents a very diverse and real-world setting with clear security implications.

Table 2: Datasets used in our evaluations.

Application	Datasets	#Records	Model	#params
WP	Reddit	20.6M	LSTM	∼20M
NIDS	IoT-Traffic	65.6M	GRU	∼507k
IC	CIFAR-10	60k	ResNet-18 Light	∼2.7M
	MNIST	70k	CNN	∼431k
	Tiny-ImageNet	120k	ResNet-18	∼11M

Evaluation Metrics. We consider a set of metrics for evaluating the effectiveness of backdoor attack and defense techniques as follows: *BA - Backdoor Accuracy* indicates the accuracy of the model in the backdoor task, i.e., it is the fraction of the trigger set for which the model provides the wrong outputs as chosen by the adversary. The adversary aims to maximize *BA*, while an effective defense prevents the adversary from increasing it. *MA - Main Task Accuracy* indicates the accuracy of a model in its main (benign) task. It denotes the fraction of benign inputs for which the system provides correct predictions. The adversary aims at minimizing the effect on *MA* to reduce the chance of being detected. The defense system should not negatively impact *MA*. *TPR - True Positive Rate* indicates how well the defense identifies poisoned models, i.e., the ratio of the number of models correctly classified as poisoned (True Positives - TP) to the total number of models being classified as poisoned: $TPR = \frac{TP}{TP+FP}$, where FP is False Positives indicating the number of benign clients that are wrongly classified as malicious. *TNR - True Negative Rate* indicates the ratio of the number of models correctly classified as benign (True Negatives - TN) to the total number of benign models: $TNR = \frac{TN}{TN+FN}$, where FN is False Negatives indicating the number of malicious clients that are wrongly classified as benign.

7 Experimental Results

In this section, we evaluate FLAME against backdoor attacks in the literature (§7.1) and demonstrate that our defense mechanism is resilient to adaptive attacks (§7.2). In addition, we show the effectiveness of each of FLAME's components in §B and FLAME overhead in §D. Finally, we evaluate the impact of the number of clients (§7.3) as well as the degree of non-IID data (§7.4).

7.1 Preventing Backdoor Attacks

Effectiveness of FLAME. We evaluate FLAME against the state-of-the-art backdoor attacks called *constrain-and-scale* [7], DBA [59], PGD and Edge-Case [57] and an untargeted poisoning attack [20] (cf. §F) using the same attack settings as in the original works with multiple datasets. The results are shown in Tab. 3. FLAME completely mitigates the *constrain-and-scale* attack ($BA = 0\%$) for all datasets. Moreover, our defense does not affect the Main Task Accuracy (*MA*) of the system as MA reduces by less than 0.4% in all experiments. The DBA attack as well as the Edge-Case attack [57] are also successfully mitigated ($BA = 3.2\%/4.0\%$). Further, FLAME is also effective against PGD attacks ($BA =$

Table 3: Effectiveness of FLAME against state-of-the-art attacks for the respective dataset, in terms of Backdoor Accuracy (*BA*) and Main Task Accuracy (*MA*). All metric values are reported as percentages.

Attack	Dataset	No Defense		FLAME	
		BA	*MA*	*BA*	*MA*
Constrain-and-scale [7]	Reddit	100	22.6	0	22.3
	CIFAR-10	81.9	89.8	0	91.9
	IoT-Traffic	100.0	100.0	0	99.8
DBA [59]	CIFAR-10	93.8	57.4	3.2	76.2
Edge-Case [57]	CIFAR-10	42.8	84.3	4.0	79.3
PGD [57]	CIFAR-10	56.1	68.8	0.5	65.1
Untargeted Poisoning [20]	CIFAR-10	-	46.72	-	91.31

Table 4: Effectiveness of FLAME in comparison to state-of-the-art defenses for the *constrain-and-scale* attack on three datasets, in terms of Backdoor Accuracy (*BA*) and Main Task Accuracy (*MA*). All values are percentages.

Defenses	Reddit		CIFAR-10		IoT-Traffic	
	BA	*MA*	*BA*	*MA*	*BA*	*MA*
Benign Setting	-	22.7	-	92.2	-	100.0
No defense	100.0	22.6	81.9	89.8	100.0	100.0
Krum [9]	100.0	9.6	100.0	56.7	100.0	84.0
FoolsGold [22]	**0.0**	**22.5**	100.0	52.3	100.0	99.2
Auror [51]	100.0	**22.5**	100.0	26.1	100.0	96.6
AFA [42]	100.0	22.4	**0.0**	91.7	100.0	87.4
DP [18]	14.0	18.9	**0.0**	78.9	14.8	82.3
Median [60]	**0.0**	22.0	**0.0**	50.1	0.0	87.7
FLAME	**0.0**	22.3	**0.0**	**91.9**	**0.0**	**99.8**

0.5 %). It should be noted that suggesting words is a quite challenging task, causing the MA even without attack to be only 22.7%, aligned with previous work [7].

We extend our evaluation to various backdoors on three datasets. For NIDS, we evaluate 13 different backdoors (Mirai malware attacks) and 24 device types (78 IoT devices). The results show that FLAME is able to mitigate all backdoor attacks completely while achieving a high MA=99.8%. We evaluate 5 different word backdoors for WP, and 90 different image backdoors for IC, which change the output of a whole class to another class. In all cases, FLAME successfully mitigates the attack while still preserving the *MA*.

Comparison to existing defenses. We compare FLAME to existing defenses: Krum [9], FoolsGold [22], Auror [51], Adaptive Federated Averaging (AFA) [42], Median [60] and a generalized differential privacy (DP) approach [7, 40]. Tab. 4 shows that FLAME is effective for all 3 datasets, while previous works either fail to mitigate backdoors or reduce the main task accuracy. Krum, FoolsGold, Auror, and AFA do not effectively remove poisoned models and *BA* often remains at 100%. Also, some defenses make the attack even more successful than without defense. Since they remove many benign updates (cf. §B) but fail to remove a sufficient number of poisoned updates, these defenses increase the *PMR* and, therefore, also the impact of the attack. Some defenses, e.g., Krum [9], Auror [51] or AFA [42] are not able to handle non-iid data scenarios like Reddit. In contrast, FoolsGold is only effective on the Reddit dataset ($TPR = 100\%$) because it works well on highly non-independent and identically distributed (non-IID) data (cf. §9). Similarly, AFA only mitigates backdoors on the CIFAR-10 dataset since the data are highly IID (each client is assigned a random set of images) such that the benign models share similar distances to the global model (cf. §9). Additionally, the model's *MA* is negatively impacted. The DP-based defense is effective, but it significantly reduces *MA*. For example, it performs best on the CIFAR-10 dataset with $BA = 0$, but *MA* decreases to 78.9% while FLAME increases *MA* to 91.9% which is close to the benign setting (no attacks), where $MA = 92.2\%$.

Effectiveness of FLAME's Components. Further, we have also conducted an extensive evaluation of the effectiveness of each of FLAME's components. Due to space limitations, we would like to refer to §B for the details.

7.2 Resilience to Adaptive Attacks

Given sufficient knowledge about FLAME, an adversary may seek to use adaptive attacks to bypass the defense components. In this section, we analyze such attack scenarios and strategies including *changing the injection strategy*, *model alignment*, and *model obfuscation*.

Changing the Injection Strategy. The adversary $\mathcal{A}$ may attempt to inject several backdoors simultaneously to execute different attacks on the system in parallel or to circumvent the clustering defense (cf. §2.2). FLAME is also effective against such attacks (cf. Fig. 5). To further investigate the resilience of FLAME against such attacks, we conduct two experiments: 1) assigning different backdoors to malicious clients and 2) letting each malicious client inject several backdoors. To ensure that each backdoor is injected by a sufficient number of clients, we increased the PMR for this experiment. We conducted these experiments with $n = 100$ clients of which $k = 40$ are malicious on the IoT-Traffic dataset with each type of Mirai attack representing a backdoor. First, we evaluate FLAME for $0, 1, 2, 4$, and 8 backdoors, meaning that the number of malicious clients for each backdoor is $0, 40, 20, 10$, and 5. Our experimental results show that our approach is effective in mitigating the attacks as $BA = 0\% \pm 0.0\%$ in all cases, with $TPR = 95.2\% \pm 0.0\%$, and $TNR = 100.0\% \pm 0.0\%$. For the second experiment, 4 backdoors are injected by each of the 40 malicious clients. Also, in this case, the results show that FLAME can completely mitigate the backdoors.

Model Alignment. Using the same attack parameter values, i.e., *PDR* (cf. §2.2), for all malicious clients can result in high distances between benign and poisoned models. Those high distances can be illustrated as a gap between poisoned and benign models, s.t. the clustering can separate them. Therefore, a sophisticated adversary can generate models that bridge the gap between them such that they are merged to the same cluster in our clustering. We evaluate this attack on the IoT-Traffic dataset for $k = 80$ malicious clients and $n = 200$ clients in total. To remove the gap, each malicious client is assigned a random amount of malicious data, i.e., a random *PDR* ranging from 5% to 20%. As Tab. 5 shows, when we apply model

Table 5: Resilience to model alignment attacks in terms of Backdoor Accuracy (*BA*), Main Task Accuracy (*MA*), True Positive Rate (*TPR*), True Negative Rate (*TNR*) in percent.

	BA	*MA*	*TPR*	*TNR*
Model Filtering	100.0	91.98	0.0	33.04
FLAME	**0.0**	**100.0**	**5.68**	**33.33**

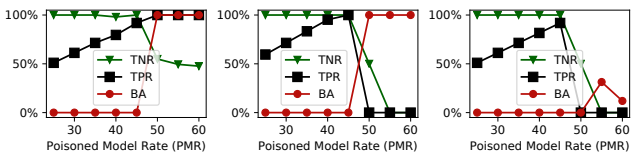
filtering only, our clustering component cannot identify the malicious clients well ($TPR = 0\%$), resulting in $BA = 100\%$. However, when we apply FLAME, although *TPR* remains low (5.68%) FLAME still mitigates the attack successfully (*BA* reduces from 100% to 0%). This can be explained by the fact that when the adversary $\mathcal{A}$ tunes malicious updates to be close to the benign ones, the attack's impact is reduced and consequently averaged out by our noising component.

Model Obfuscation. $\mathcal{A}$ can add noise to the poisoned models to make them difficult to detect. However, our evaluation of such an attack on the IoT-Traffic dataset shows that this strategy is not effective. We evaluate different noise levels to determine a suitable standard deviation for the noise. Thereby, we observe that a noise level of 0.034 causes the models' cosine distances in clustering to change without significantly impacting *BA*. However, FLAME can still efficiently defend this attack: *BA* remains at 0% and *MA* at 100%.

7.3 Effect of Number of Clients

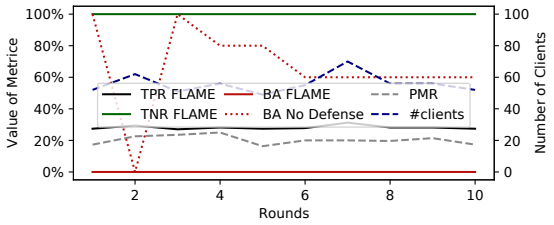
Impact of Number of Malicious Clients. We assume that the number of benign clients is more than half of all clients (cf. §2.2) and our clustering is only expected to be successful when $PMR = \frac{k}{n} < 50\%$ (cf. §4.3.1). We evaluate FLAME for different *PMR* values. Figure 7 shows how *BA*, *TPR*, and *TNR* change in the IC, NIDS, and WP applications for *PMR* values from 25% to 60%. It shows that FLAME is only effective if $PMR < 50\%$ so that only benign clients are admitted to the model aggregation ($TNR = 100\%$) and thus $BA = 0\%$. However, if $PMR > 50\%$, FLAME fails to mitigate the attack because the majority of poisoned models will be included resulting in low *TNR*. Interestingly, FLAME accepted all models for $PMR = 50\%$ ($TPR = 0\%$ and $TNR = 100\%$). For the IC application, since the IC data are non-IID, poisoned models are not similar. Therefore, some poisoned models were excluded from the cluster resulting in a high *TPR* even for *PMRs* higher than 50%. However, the majority of poisoned models were selected resulting in the drop in the *TNR*.

Varying number of clients in different training rounds.

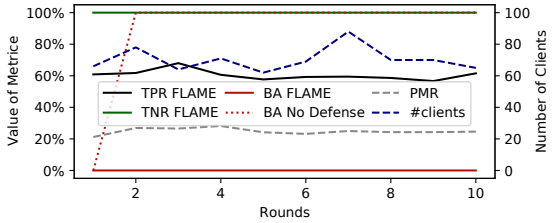


(a) IC (b) NIDS (c) WP

Figure 7: Impact of the poisoned model rate $PMR = \frac{k}{n}$ on the evaluation metrics. *PMR* is the fraction of malicious clients k per total clients n.



(a) Image Classification



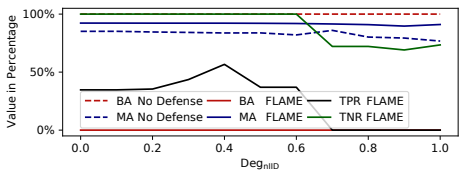
(b) Network Intrusion Detection

Figure 8: Impact of the number of clients on FLAME

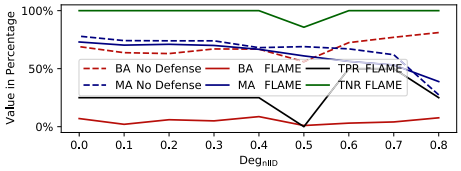
In general, FLAME is a round-independent defense, i.e., it does not use information from previous rounds such as which clients were excluded in which rounds. Therefore, FLAME will not be affected if the number of clients or number of malicious clients varies as long as the majority of clients remain benign. To demonstrate this, we simulate realistic scenarios in which clients can join and drop out dynamically. We conducted an experiment where during each round, the total number of available clients is randomly selected. As the result, the number of malicious clients will also be random. In this experiment, we used a population of 100 clients in total, out of which 25 are malicious. In each round, a random number (from 60 to 90) of clients are selected, so that the fraction of malicious clients (*PMR*) varies in each round. Figure 8 shows the experimental results. One can see that the proportion of malicious clients (*PMR*) does not affect the effectiveness of FLAME, i.e., the backdoor is completely removed ($BA = 0\%$) in every round. Since all poisoned models are detected, their negative effect on the aggregated model is removed. Therefore, the MA with FLAME is better than the one without defense, and is almost always 100 % aligned with the results in Tab. 4.

7.4 Impact of the Degree of non-IID Data

Since clustering is based on measuring differences between benign and malicious updates, the distribution of data among clients might affect our defense. We conduct two experiments for both *Constrain-and-scale* and Edge-Case PGD on the CIFAR-10 dataset. For Reddit and IoT datasets, changing the degree of non-IID data is not meaningful since the data have a natural distribution as every client obtains data from different Reddit users or traffic chunks from different IoT devices. Following previous works [20,57], we vary the degree of non-IID data Deg_{nIID} by changing the fraction of images belonging to a specific class assigned to clients. In particular, we divide the clients into 10 groups corresponding to the 10 classes of



(a) *Constrain-and-scale*



(b) Edge-Case PGD

Figure 9: Impact of degree of non-IID data on FLAME for *constrain-and-scale* using the Deg_{nIID} and for the Edge-Case PGD attack using the α parameter of the Dirichlet distribution.

CIFAR-10. The clients of each group are assigned a fixed fraction of $\text{Deg}_{\text{nIID}}$of the images from its designated image class, while the rest of the images will be assigned to it at random. Consequently, the data distribution is random, i.e., completely IID if $\text{Deg}_{\text{nIID}} = 0\%$ (all images are randomly assigned) and completely non-IID if $\text{Deg}_{\text{nIID}} = 100\%$ (a client only gets images from its designated class).

Figure 9a shows the evaluation results for the *constrain-and-scale* attacks. Although FLAME does not detect the poisoned models for very non-IID scenarios, it still mitigates the attack as the BA remains 0% for all values of Deg_{nIID}. For low Deg_{nIID}, FLAME effectively identifies the poisoned models ($TNR = 100\%$) and the MA remains on almost the same level as without defense. As shown in Fig. 9b, FLAME also mitigates the Edge-Case PGD attack effectively for all α values of the Dirichlet distribution and the MA also stays on the same level as without defense. However, since not all poisoned models are detected, a higher σ is determined dynamically to mitigate the *constrain-and-scale* backdoor, resulting in a slightly reduced MA for $\text{Deg}_{\text{nIID}} \geq 0.7$ (MA is 91.9% for $\text{Deg}_{\text{nIID}} = 0.6$, and is reduced to 91.0% for $\text{Deg}_{\text{nIID}} = 1.0$). Note that Fig. 9 shows the evaluation results in a training round t where the global model G_t is close to convergence [7], thus even though the TNR decreases with a large value of Deg_{nIID}, the drop of MA with FLAME is not substantial.

8 Privacy-preserving Federated Learning

A number of attacks on FL have been proposed that aim to infer from parameters of a model the presence of a specific training sample in the training dataset (*membership inference attacks*) [41,46,52], properties of training samples (*property inference attacks*) [23,41], try to assess the proportion of samples of a specific class in the data (*distribution estimation attacks*) [58]. Inference attacks by the aggregator $\mathcal{A}^s$ are significantly stronger, as $\mathcal{A}^s$ has access to the local models [43] and can also link gained information to a specific user, while the global model anonymizes the individual contributions.

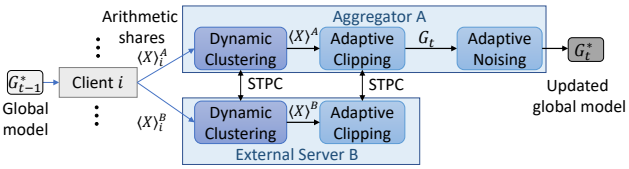


Figure 10: Overview of private FLAME in round t using Secure-Two-Party Computation (STPC).

Therefore, enhanced privacy protection for FL is needed that prohibits access to the local model updates.

Adversary Model (privacy). In this adversary type, $\mathcal{A}^s$ attempts to infer sensitive information about clients' data D_i from their model updates W_i [23,41,46,52] by maximizing the information $\phi_i = \text{INFER}(W_i)$ that $\mathcal{A}^s$ gains about the data D_i of client i by inferring from its corresponding model W_i.

Deficiencies of existing defenses. Generally, there are two approaches to protect the privacy of clients' data: differential privacy (DP; [18]) and cryptographic techniques such as homomorphic encryption [24] or multi-party computation [14]. DP is a statistical approach that can be efficiently implemented, but it can only offer high privacy protection at the cost of a significant loss in accuracy due to the noise added to the models [6,61]. In contrast, cryptographic techniques provide strong privacy guarantees as well as high accuracy at the cost of reduced efficiency.

Private FLAME. To securely implement FLAME using STPC, we use an optimized combination of three prominent STPC techniques as implemented with state-of-the-art optimizations in the ABY framework [14]. Fig. 10 shows an overview of private FLAME. It involves n clients and two non-colluding servers, called aggregator A and external server B. Each client $i \in \{1,...,n\}$ splits the parameters of its local update W_i into two Arithmetic shares $\langle X \rangle_i^A$ and $\langle X \rangle_i^B$, such that $W_i = \langle X \rangle_i^A + \langle X \rangle_i^B$, and sends $\langle X \rangle_i^A$ to A and $\langle X \rangle_i^B$ to B. A and B then privately compute the new global model via STPC. We co-design the distance calculation, clustering, adaptive clipping, and aggregation of FLAME (cf. Alg. 1) of FLAME as efficient STPC protocols.To further improve performance, we approximate HDBSCAN with the simpler DBSCAN [10] to avoid the construction of the minimal spanning tree in HDBSCAN as it is very expensive to realize with STPC. See §G for more details on private FLAME evaluation of its accuracy and performance.

9 Related Work

In general, existing backdoor defenses can roughly be divided into two main categories. The first one aims to distinguish malicious updates and benign updates by 1) clustering model updates [9,15,22,29,33,34,51], 2) changing aggregation rules [25,60], and 3) using root dataset [4]. The second category is based on differential privacy techniques [7,56]. Next, we will discuss these points in detail.

Clustering model updates. Several backdoor defenses, such as Krum [9], AFA [42], and Auror [51], aim at separat-

ing benign and malicious model updates. However, they only work under specific assumptions about the underlying data distributions, e.g., Auror and Krum assume that data of benign clients are iid. In contrast, FoolsGold and AFA [42] assume that benign data are non-iid. In addition, FoolsGold assumes that manipulated data are iid. As a result, these defenses are only effective under specific circumstances (cf. §7.1) and cannot handle the simultaneous injection of multiple backdoors (cf. §4.3.1). Moreover, such defenses cannot detect stealthy attacks, e.g., where the adversary constrains their poisoned updates within benign update distribution such as *Constrain-and-scale* attacks [7]. In contrast, FLAME does not make any assumption about the data distribution, clipping, and noising components can also mitigate stealthy attacks, and FLAME can defend against injection of multiple backdoors (cf. §4.3.1).

Changing aggregation rules. Instead of using FedAvg [38], Yin *et al.* [60] and Guerraoui *et al.* [25] propose using the median parameters from all local models as the global model parameters, i.e., $G_t = \text{MEDIAN}(W_1^t, \ldots, W_n^t)$. However, the adversary can bypass it by injecting stealthy models like W_3' (cf. Fig. 2), in which the parameters of poisoned model will be selected to be incorporated into the global model. Further, our evaluation in §7.1 shows that Median also reduces the performance of the model significantly.

Using root data. Although FLTrust [12] can defend against byzantine clients (with arbitrary behavior) and detect poisoning attacks including backdoors, it requires the aggregator to have access to a benign root dataset. Baffle [4] utilizes clients using their own data to evaluate the performance of the aggregated model to detect backdoors. However, this approach has two limitations, e.g., (i) the backdoor triggers are only known to the attacker, i.e., one cannot ensure that the benign clients would have such trigger data to activate the backdoor, and (ii) Baffle does not work in a non-IID data scenario with a small number of clients as clients cannot distinguish deficits in model performance due to the backdoor from lack of data.

Differential Privacy-based approaches. Clipping and noising are known techniques to achieve differential privacy (DP) [18]. However, directly applying these techniques to defend against backdoor attacks is not effective because they significantly decrease the Main Task Accuracy (§7.1) [7]. FLAME tackles this by i) identifying and filtering out potential poisoned models that have a high attack impact (cf. §4.3.1), and ii) eliminating the residual poison with an appropriate adaptive clipping bound and noise level, such that the Main Task Accuracy is retained (cf. §4.3.2).

10 Conclusion

In this paper, we introduce FLAME, a resilient aggregation framework for FL that eliminates the impact of backdoor attacks while maintaining the performance of the aggregated model on the main task. We propose a method to approximate the amount of noise that needs to be injected into the global model to neutralize backdoors. Furthermore, in combination with our dynamic clustering and adaptive clipping, FLAME can significantly reduce the noise scale for backdoor removal and thus preserve the benign performance of the global model. In addition, we design, implement, and benchmark efficient secure two-party computation protocols for FLAME to ensure the privacy of clients' training data and to impede inference attacks on client updates.

Acknowledgments

This research was funded by the Deutsche Forschungsgemeinschaft (DFG) SFB-1119 CROSSING/236615297, the European Research Council (ERC, grant No. 850990 PSOTI), the EU H2020 project SPATIAL (grant No. 101021808), GRK 2050 Privacy & Trust/251805230, HMWK within ATHENE project, NSF-TrustHub (grant No. 1649423), SRC-Auto (2019-AU-2899), Huawei OpenS3 Lab, and Intel Private AI Collaborative Research Center. We thank the anonymous reviewers and the shepherd, Neil Gong, for constructive reviews and comments.

References

[1] Reddit dataset, 2017. https://bigquery.cloud.google.com/dataset/fh-bigquery:reddit_comments.

[2] Pytorch, 2019. https://pytorch.org.

[3] Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *CCS*. ACM, 2016.

[4] Sebastien Andreina, Giorgia Azzurra Marson, Helen Möllering, and Ghassan Karame. BaFFLe: Backdoor Detection via Feedback-based Federated Learning. In *ICDCS*, 2021.

[5] Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J. Alex Halderman, Luca Invernizzi, Michalis Kallitsis, Deepak Kumar, Chaz Lever, Zane Ma, Joshua Mason, Damian Menscher, Chad Seaman, Nick Sullivan, Kurt Thomas, and Yi Zhou. Understanding the Mirai Botnet. In *USENIX Security*, 2017.

[6] Yoshinori Aono, Takuya Hayashi, Lihua Wang, and Shiho Moriai. Privacy-preserving Deep Learning via Additively Homomorphic Encryption. In *TIFS*, 2017.

[7] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How To Backdoor Federated Learning. In *AISTATS*, 2020.

[8] Moran Baruch, Gilad Baruch, and Yoav Goldberg. A Little Is Enough: Circumventing Defenses For Distributed Learning. In *NIPS*, 2019.

[9] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. Machine Learning with Adversaries: Byzantine Tolerant Gradient Descent. In *NIPS*, 2017.

[10] Beyza Bozdemir, Sébastien Canard, Orhan Ermis, Helen Möllering, Melek Önen, and Thomas Schneider. Privacy-preserving density-based clustering. In *ASIACCS*, 2021.

[11] Ricardo J. G. B. Campello, Davoud Moulavi, and Joerg Sander. Density-Based Clustering Based on Hierarchical Density Estimates. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 2013.

[12] Xiaoyu Cao, Minghong Fang, Jia Liu, and Neil Zhenqiang Gong. Fltrust: Byzantine-robust federated learning via trust bootstrapping. In *NDSS*, 2021.

[13] Trishul Chilimbi, Yutaka Suzue, Johnson Apacible, and Karthik Kalyanaraman. Project Adam: Building an Efficient and Scalable Deep Learning Training System. In *USENIX Operating Systems Design and Implementation*, 2014.

[14] Daniel Demmler, Thomas Schneider, and Michael Zohner. ABY - A Framework for Efficient Mixed-Protocol Secure Two-Party Computation. In *NDSS*, 2015.

[15] Ilias Diakonikolas, Gautam Kamath, Daniel Kane, Jerry Li, Jacob Steinhardt, and Alistair Stewart. Sever: A robust meta-algorithm for stochastic optimization. In *ICML*, 2019.

[16] Rohan Doshi, Noah Apthorpe, and Nick Feamster. Machine Learning DDoS Detection for Consumer Internet of Things Devices. In *arXiv preprint:1804.04159*, 2018.

[17] Min Du, Ruoxi Jia, and Dawn Song. Robust anomaly detection and backdoor attack detection via differential privacy. In *ICLR*, 2020.

[18] Cynthia Dwork and Aaron Roth. The Algorithmic Foundations of Differential Privacy. In *Foundations and Trends in Theoretical Computer Science*, 2014.

[19] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *KDD*, 1996.

[20] Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. Local Model Poisoning Attacks to Byzantine-Robust Federated Learning. In *USENIX Security*, 2020.

[21] Hossein Fereidooni, Samuel Marchal, Markus Miettinen, Azalia Mirhoseini, Helen Möllering, Thien Duc Nguyen, Phillip Rieger, Ahmad-Reza Sadeghi, Thomas Schneider, Hossein Yalame, et al. Safelearn: secure aggregation for private federated learning. In *2021 IEEE Security and Privacy Workshops (SPW)*, pages 56–62. IEEE, 2021.

[22] Clement Fung, Chris JM Yoon, and Ivan Beschastnikh. The limitations of federated learning in sybil settings. In *RAID*, 2020. originally published as arxiv:1808.04866.

[23] Karan Ganju, Qi Wang, Wei Yang, Carl A Gunter, and Nikita Borisov. Property Inference Attacks on Fully Connected Neural Networks Using Permutation Invariant Representations. In *CCS*, 2018.

[24] Craig Gentry. *A Fully Homomorphic Encryption Scheme*. PhD thesis, Stanford University, Stanford, CA, USA, 2009.

[25] Rachid Guerraoui, Sébastien Rouault, et al. The hidden vulnerability of distributed learning in byzantium. In *ICML*. PMLR, 2018.

[26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *CVPR*, 2016.

[27] Stephen Herwig, Katura Harvey, George Hughey, Richard Roberts, and Dave Levin. Measurement and Analysis of Hajime, a Peer-to-Peer IoT Botnet. In *NDSS*, 2019.

[28] Li Huang, Yifeng Yin, Zeng Fu, Shifa Zhang, Hao Deng, and Dianbo Liu. LoAdaBoost: Loss-Based AdaBoost Federated Machine Learning on medical Data. In *arXiv preprint:1811.12629*, 2018.

[29] Youssef Khazbak, Tianxiang Tan, and Guohong Cao. Mlguard: Mitigating poisoning attacks in privacy preserving distributed collaborative learning. In *International Conference on Computer Communications and Networks (ICCCN)*. IEEE, 2020.

[30] Constantinos Kolias, Georgios Kambourakis, Angelos Stavrou, and Jeffrey Voas. DDoS in the IoT: Mirai and Other Botnets. In *IEEE Computer*, 2017.

[31] Alex Krizhevsky and Geoffrey Hinton. Learning Multiple Layers of Features from Tiny Images. Technical report, 2009.

[32] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 1998.

[33] Suyi Li, Yong Cheng, Yang Liu, Wei Wang, and Tianjian Chen. Abnormal client behavior detection in federated learning. *arXiv preprint arXiv:1910.09933*, 2019.

[34] Suyi Li, Yong Cheng, Wei Wang, Yang Liu, and Tianjian Chen. Learning to detect malicious clients for robust federated learning. *arXiv preprint arXiv:2002.00211*, 2020.

[35] Yujun Lin, Song Han, Huizi Mao, Yu Wang, and William J. Dally. Deep Gradient Compression: Reducing the Communication Bandwidth for Distributed Training. In *ICLR*, 2018.

[36] Leland McInnes and John Healy. Accelerated hierarchical density based clustering. In *Data Mining Workshops (ICDMW), 2017 IEEE International Conference on*. IEEE, 2017.

[37] Leland McInnes, John Healy, and Steve Astels. hdbscan: Hierarchical density based clustering. *The Journal of Open Source Software*, 2(11):205, 2017.

[38] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *AISTATS*, 2017.

[39] Brendan McMahan and Daniel Ramage. Federated learning: Collaborative Machine Learning without Centralized Training Data. Google AI, 2017.

[40] H. Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning Differentially Private Language Models Without Losing Accuracy. In *ICLR*, 2018.

[41] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting Unintended Feature Leakage in Collaborative Learning. In *IEEE S&P*, 2019.

[42] Luis Muñoz-González, Kenneth T. Co, and Emil C. Lupu. Byzantine-Robust Federated Machine Learning through Adaptive Model Averaging. In *arXiv preprint:1909.05125*, 2019.

[43] M. Nasr, R. Shokri, and A. Houmansadr. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *IEEE S&P*, 2019.

[44] Thien Duc Nguyen, Samuel Marchal, Markus Miettinen, Hossein Fereidooni, N. Asokan, and Ahmad-Reza Sadeghi. DÏoT: A Federated Self-learning Anomaly Detection System for IoT. In *ICDCS*, 2019.

[45] Thien Duc Nguyen, Phillip Rieger, Markus Miettinen, and Ahmad-Reza Sadeghi. Poisoning Attacks on Federated Learning-Based IoT Intrusion Detection System. In *Workshop on Decentralized IoT Systems and Security*, 2020.

[46] Apostolos Pyrgelis, Carmela Troncoso, and Emiliano De Cristofaro. Knock Knock, Who's There? Membership Inference on Aggregate Location Data. In *NDSS*, 2018.

[47] Jianji Ren, Haichao Wang, Tingting Hou, Shuai Zheng, and Chaosheng Tang. Federated Learning-Based Computation Offloading Optimization in Edge Computing-Supported Internet of Things. In *IEEE Access*, 2019.

[48] Sumudu Samarakoon, Mehdi Bennis, Walid Saad, and Merouane Debbah. Federated Learning for Ultra-Reliable Low-Latency V2V Communications. In *GLOBCOM*, 2018.

[49] Micah Sheller, Anthony Reina, Brandon Edwards, Jason Martin, and Spyridon Bakas. Federated Learning for Medical Imaging. In *Intel AI*, 2018.

[50] Micah Sheller, Anthony Reina, Brandon Edwards, Jason Martin, and Spyridon Bakas. Multi-Institutional Deep Learning Modeling Without Sharing Patient Data: A Feasibility Study on Brain Tumor Segmentation. In *Brain Lesion Workshop*, 2018.

[51] Shiqi Shen, Shruti Tople, and Prateek Saxena. Auror: Defending Against Poisoning Attacks in Collaborative Deep Learning Systems. In *ACSAC*, 2016.

[52] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership Inference Attacks Against Machine Learning Models. In *IEEE S&P*, 2017.

[53] Arunan Sivanathan, Hassan Habibi Gharakheili, Franco Loi, Adam Radford, Chamith Wijenayake, Arun Vishwanath, and Vijay Sivaraman. Classifying IoT Devices in Smart Environments Using Network Traffic Characteristics. In *TMC*, 2018.

[54] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S Talwalkar. Federated Multi-Task Learning. In *NIPS*, 2017.

[55] Saleh Soltan, Prateek Mittal, and Vincent Poor. BlackIoT: IoT Botnet of High Wattage Devices Can Disrupt the Power Grid. In *USENIX Security*, 2018.

[56] Ziteng Sun, Peter Kairouz, Ananda Theertha Suresh, and H Brendan McMahan. Can you really backdoor federated learning? *arXiv preprint arXiv:1911.07963*, 2019.

[57] Hongyi Wang, Kartik Sreenivasan, Shashank Rajput, Harit Vishwakarma, Saurabh Agarwal, Jy-yong Sohn, Kangwook Lee, and Dimitris Papailiopoulos. Attack of the tails: Yes, you really can backdoor federated learning. In *NeurIPS*, 2020.

[58] Lixu Wang, Shichao Xu, Xiao Wang, and Qi Zhu. Eavesdrop the Composition Proportion of Training Labels in Federated Learning. In *arXiv preprint:1910.06044*, 2019.

[59] Chulin Xie, Keli Huang, Pin-Yu Chen, and Bo Li. DBA: Distributed Backdoor Attacks against Federated Learning. In *ICLR*, 2020.

[60] Dong Yin, Yudong Chen, Ramchandran Kannan, and Peter Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. In *ICML*. PMLR, 2018.

[61] Chengliang Zhang, Suyi Li, Junzhe Xia, Wei Wang, Feng Yan, and Yang Liu. BatchCrypt: Efficient Homomorphic Encryption for Cross-Silo Federated Learning. In *USENIX ATC*, 2020.

A Datasets and Learning Configurations

Word Prediction (WP). We use the Reddit dataset of November 2017 [1] with the same settings as state-of-the-art works [7, 38, 40] for comparability. In particular, each user in the dataset with at least 150 posts and not more than 500 posts is considered as a client. This results in 80 000 clients' datasets with sizes between 298 and 32 660 words.

The model consists of two LSTM layers and a linear output layer [7, 38]. To be comparable to the attack setting in [7], we evaluate FLAME on five different backdoors, each with a different trigger sentence corresponding to a chosen output.

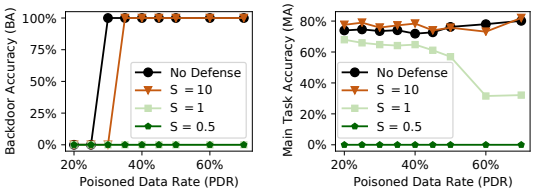
Image Classification (IC). For image classification, we use mainly the CIFAR-10 dataset [31], a standard benchmark dataset for image classification, in particular for FL [38] and backdoor attacks [7, 8, 42]. It consists of 60 000 images of 10 different classes. The adversary aims at changing the predicted label of one class of images to another class of images. We use a lightweight version of the ResNet18 model [26] with 4 convolutional layers with max-pooling and batch normalization [7]. The experimental setup consists of 100 clients and uses a PMR of 20%. In addition to the CIFAR-10 dataset, we also evaluate FLAME's effectiveness on two further datasets for image classification. The *MNIST* dataset consists of 70 000 handwritten digits [32]. The learning task is to classify images to identify digits. The adversary poisons the model by mislabeling labels of digit images before using it for training [51]. We use a convolutional neural network (CNN) with 431000 parameters. The *Tiny-ImageNet* [3] consists of 200 classes and each class has 500 training images, 50 validation images, and 50 test images. We used ResNet18 [26] model.

Network Intrusion Detection System (NIDS). We test backdoor attacks on IoT anomaly-based intrusion detection systems that often represent critical security applications [5, 16, 27, 30, 44, 45, 55]. Here, the adversary aims at causing incorrect classification of anomalous traffic patterns, e.g., generated by IoT malware, as benign patterns. Based on the FL anomaly detection system DÏoT [44], we use three datasets called DIoT-Benign, DIoT-Attack, and UNSW-Benign [44, 53] from real-world home and office deployments (four homes and two offices located in Germany and Australia). DIoT-Attack contains the traffic of 5 anomalously behaving IoT devices, infected by the Mirai malware [44]. Moreover, we collected a fourth IoT dataset containing communication data from 24 typical IoT devices (including IP cameras and power plugs) in three different smart home settings and an office setting. Following [44], we extracted

[3] https://tiny-imagenet.herokuapp.com

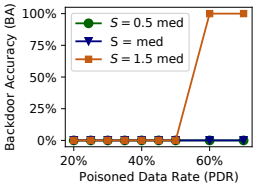
Table 6: Effectiveness of the clustering component, in terms of True Positive Rate (*TPR*) and True Negative Rate (*TNR*), of FLAME in comparison to existing defenses for the *constrain-and-scale* attack on three datasets. All values are in percentage and the best results of the defenses are marked in bold.

Defenses	Reddit		CIFAR-10		IoT-Traffic	
	TPR	TNR	TPR	TNR	TPR	TNR
Krum	9.1	0.0	8.2	0.0	24.2	0.0
FoolsGold	**100.0**	**100.0**	0.0	90.0	32.7	84.4
Auror	0.0	90.0	0.0	90.0	0.0	70.2
AFA	0.0	88.9	**100.0**	**100.0**	4.5	69.2
FLAME	22.2	**100.0**	23.8	86.2	**59.5**	**100.0**

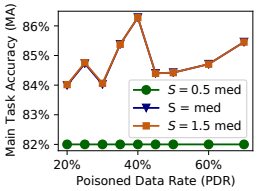


(a) BA for static bound

(b) MA for static bound



(c) BA for dynamic bound



(d) MA for dynamic bound

Figure 11: Effectiveness of FLAME's clipping bound in terms of Backdoor Accuracy (*BA*) and Main Task Accuracy (*MA*). *S* is the clipping bound and *med* the L_2-norm median.

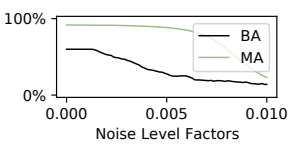
device-type-specific datasets capturing the devices' communication behavior. We simulate the FL setup by splitting each device type's dataset among several clients (from 20 to 200). Each client has a training dataset corresponding to three hours of traffic measurements containing samples of roughly 2 000-3 000 communication packets. The learning model consists of 2 GRU layers and a fully connected layer.

B Effectiveness of FLAME's Components

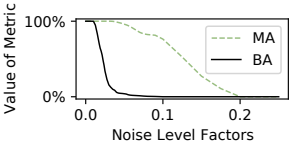
B.1 Effectiveness of the Clustering Component

We show the results for the clustering component in Tab. 6. As shown there, our filtering achieves $TNR = 100\%$ for the Reddit and IoT-Traffic datasets, i.e., FLAME only selects benign models in this attack setting. Recall that the goal of clustering is to filter out the poisoned models with high attack impact, i.e., not necessarily all poisoned models (cf. §4.1). This allows FLAME to defend backdoor attacks effectively, even if not all poisoned models are filtered. For example, although for the CIFAR-10 dataset in Tab. 6 the TNR is not 100 % (86.2%), the attack is still mitigated by the noising component, such that the BA is 0 % (cf. Tab. 4).

B.2 Effectiveness of Clipping



(a) Image Classification



(b) Network Intrusion Detection

Figure 12: Impact of different noise level factors on the Backdoor Accuracy (*BA*) and Main Task Accuracy (*MA*).

Fig. 11 demonstrates the effectiveness of FLAME's dynamic clipping where S is the median of L_2-norms compared to a static clipping bound [7] and different choices for a dynamic clipping boundary (i.e., median, half of median, median multiplied by 1.5). The experiments are conducted for the IoT-Traffic dataset, which is non-iid. Fig. 11a and Fig. 11b show that a small static bound $S = 0.5$ is effective to mitigate the attack ($BA = 0\%$), but *MA* drops to 0% rendering the model useless. Moreover, a higher static bound like $S = 10$ is ineffective as $BA = 100\%$ if the Poisoned Data Rate (PDR) $\geq 35\%$. In contrast, FLAME's dynamic clipping threshold performs significantly better as *BA* consistently remains at 0% while *MA* remains high (cf. Fig. 11c and Fig. 11d).

B.3 Effectiveness of Adding Noise

Fig. 12 shows the impact of adding noise to the intermediate global models with respect to different noise level factors λ to determine the standard deviation of the noise σ dynamically based on the median L_2-norm of the updates S_t as $\sigma = \lambda S_t$. As it can be seen, increasing λ reduces the *BA*, but it also negatively impacts the performance of the model in the main task (*MA*). Therefore, the noise level must be dynamically tuned and combined with the other defense components to optimize the overall success of the defense. The noise level factor is determined by $\lambda = \frac{1}{\varepsilon}\sqrt{2ln\frac{1.25}{\delta}}$ for (ε, δ)-DP. We use standard DP parameters and set $\varepsilon = 3705$ for IC, $\varepsilon = 395$ for the NIDS and $\varepsilon = 4191$ for the NLP scenario. Accordingly, $\lambda = 0.001$ for IC and NLP, and $\lambda = 0.01$ for the NIDS scenario. The DP budget is dependent on the considered dataset scenario. It is determined based on the median of the dataset sizes of the clients and the size of the model used. It can thus be empirically determined by the aggregator. Analogous to determining the clipping boundary S (cf. 4.3.2), using the median ensures that the used dataset size is within the range of benign values.

C Naïve Combination

Furthermore, we test a naïve combination of the defense components by stacking clipping and adding noise (using a fixed clipping bound of 1.0 and a standard deviation of 0.01 as in [7]) on top of a clustering component using K-means. However, this naïve approach still allows a *BA* of 51.9% and a *MA* of 60.24%, compared to a *BA* of 0.0% and a *MA* of 89.87% of FLAME in the same scenario for the CIFAR-10 dataset. Based on our evaluations in §7.1, it becomes apparent that

FLAME's dynamic nature goes beyond previously proposed defenses that consist of static baseline ideas, which FLAME significantly optimizes, extends, and automates to offer a comprehensive dynamic and private defense against sophisticated backdoor attacks.

D Overhead of FLAME

We evaluated FLAME for 6 different device types from the IoT dataset. In this experiment, only benign clients participated and the model was randomly initialized. The highest observed overhead was 4 additional rounds. In average, 1.67 additional training rounds were needed to achieve at least 99% of the *MA* that was achieved without applying the defense, i.e., FLAME does not prevent the model from converging.

E HDBSCAN

HDBSCAN [11] is a density-based clustering technique that classifies data samples in different clusters without predefined the maximum distance and the number of clusters. In the following, we describe HDBSCAN in detail, following the implementation of McInnes *et al.* [36, 37]. However, we focus on the behavior of HDBSCAN for the parameters that FLAME uses, i.e., when `min_cluster_size`=$N/2+1$ and `min_samples=1`, e.g., because of the choice for `min_cluster_size` we skip parts that deal with multiple clusters. HDBSCAN first uses the given distances to build a minimal spanning tree (MST), where the vertices represent the individual data points and the edges are weighted by the distances between the respective points. Then it uses the MST to build a binary tree where the leaf nodes represent the vertices of the MST and the non-leaf nodes represent the edges of the MST. For this, first, all vertices are considered as separate trees (of size 1). For this, first, all vertices are considered as separate trees (of size 1) and then, starting from the edge with the lowest weight, iteratively the trees are merged by creating a non-leaf-node for each edge of the MST and set the (previously not connected) subtrees containing the endpoints of the edge as children for the new node (represented by calling the function `make_binary_tree`. In the next step, HDBSCAN collects all nodes of the binary tree as candidates, that cover at least $N/2+1$ data points. Since only non-leaf nodes fulfill the requirement of covering at least $N/2+1$ data points, each cluster candidate is based on a node, representing an edge in the MST. It uses the weight of the edge and the number of covered points to calculate a so-called stability value. Then HDBSCAN uses the stability value to determine the cluster candidate with the most homogeneous density and returns this candidate as majority cluster. Finally, it assigns the cluster label to all data points inside this cluster and labels all points outside of this cluster as noise.

F Effectiveness of FLAME against untargeted poisoning attacks

Another attack type related to backdooring is *untargeted poisoning* [8, 9, 20]. Unlike backdoor attacks that aim to incorporate specific backdoor functionalities, untargeted poisoning aims at rendering the model unusable. The adversary uses crafted local models with low Main Task Accuracy to damage the global model G. Fang at el. [20] propose such an attack bypassing state-of-the-art defenses. Although we do not focus on untargeted poisoning, our approach intuitively defends it since, in principle, this attack also trade-offs attack impact against stealthiness. To evaluate the effectiveness of FLAME against this attack, we test the Krum-based attack proposed by [20] on FLAME. Since [20]'s evaluation uses image datasets, we evaluate FLAME's resilience against it with CIFAR-10. The evaluation results show that although the attack significantly damages the model by reducing *MA* from 92.16% to 46.72%, FLAME can successfully defend against it and *MA* remains at 91.31%.

G Performance of Private FLAME

For our implementation, we use the STPC framework ABY [14] which implements the three sharing types, including state-of-the-art optimizations and flexible conversions and the open-source privacy-preserving DBSCAN by Bozdemir et al. [10]. All STPC results are averaged over 10 experiments and run on two separate servers with Intel Core i9-7960X CPUs with 2.8 GHz and 128 GB RAM connected over a 10 Gbit/s LAN with 0.2 ms RTT.

Approximating HDBSCAN by DBSCAN. We measure the effect of approximating HDBSCAN by DBSCAN including the binary search for the neighborhood parameter ε. The results show that our approximation has a negligible loss of accuracy. For some applications, the approximation even performs slightly better than the standard FLAME, e.g., for CIFAR-10, private FLAME correctly filters all poisoned models, while standard FLAME accepts a small number ($TNR = 86.2\%$), which is still sufficient to achieve $BA = 0.0\%$.

Runtime of Private FLAME. We evaluate the runtime in seconds per training iteration of the cosine distance, Euclidean distance + clipping + model aggregation, and clustering steps of Alg. 1 in standard (without STPC) and in private FLAME (with STPC). The results show that private FLAME causes a significant overhead on the runtime by a factor of up to three orders of magnitude compared to the standard (non-private) FLAME. However, even if we consider the largest model (Reddit) with $K = 100$ clients, we have a total server-side runtime of 22 081.65 seconds ($\approx$ 6 hours) for a training iteration with STPC. Such runtime overhead would be acceptable to maintain privacy, especially since mobile phones, which would be a typical type of clients in FL [38], are not always available and connected so that there will be delays in synchronizing clients' model updates in FL. These delays can then also be used to run STPC. Furthermore, achieving provable privacy by using STPC may even motivate more clients to contribute to FL in the first place and provide more data.

D SPIKE: Secure and Private Investigation of the Kidney Exchange problem (BMC'22)

[BHK+22] T. Birka, K. Hamacher, T. Kussel, H. Möllering, T. Schneider. **"SPIKE: Secure and private investigation of the kidney exchange problem"**. In: *BMC Medical Informatics and Decision Making* 22.1 (2022). Online: https://arxiv.org/abs/2204.09937. Code: https://github.com/encryptogroup/ppke, S. 253. CORE Rank B. Appendix D.

https://www.doi.org/10.1186/s12911-022-01994-4

Birka *et al.*
BMC Medical Informatics and Decision Making (2022) 22:253
https://doi.org/10.1186/s12911-022-01994-4

BMC Medical Informatics and Decision Making

RESEARCH Open Access

SPIKE: secure and private investigation of the kidney exchange problem

Timm Birka[1†], Kay Hamacher[2], Tobias Kussel[2], Helen Möllering[1*] and Thomas Schneider[1]

Abstract

Background: The kidney exchange problem (KEP) addresses the matching of patients in need for a replacement organ with compatible living donors. Ideally many medical institutions should participate in a matching program to increase the chance for successful matches. However, to fulfill legal requirements current systems use complicated policy-based data protection mechanisms that effectively exclude smaller medical facilities to participate. Employing secure multi-party computation (MPC) techniques provides a technical way to satisfy data protection requirements for highly sensitive personal health information while simultaneously reducing the regulatory burdens.

Results: We have designed, implemented, and benchmarked SPIKE, a secure MPC-based privacy-preserving KEP protocol which computes a locally optimal solution by finding matching donor–recipient pairs in a graph structure. SPIKE matches 40 pairs in cycles of length 2 in less than 4 min and outperforms the previous state-of-the-art protocol by a factor of 400× in runtime while providing medically more robust solutions.

Conclusions: We show how to solve the KEP in a robust and privacy-preserving manner achieving significantly more practical performance than the current state-of-the-art (Breuer et al., WPES'20 and CODASPY'22). The usage of MPC techniques fulfills many data protection requirements on a technical level, allowing smaller health care providers to directly participate in a kidney exchange with reduced legal processes. As sensitive data are not leaving the institutions' network boundaries, the patient data underlie a higher level of protection than in the currently employed (centralized) systems. Furthermore, due to reduced legal barriers, the proposed decentralized system might be simpler to implement in a transnational, intereuropean setting with mixed (national) data protecion laws.

Keywords: Kidney-exchange, Privacy, Secure multi-party computation (MPC)

Introduction

Around 7% of U.S. adults are affected by chronic kidney disease [1]. With the increasing age of the population in most countries, end-stage renal disease constitutes a rapidly increasing challenge for health care systems [2]. Humans are able to live a normal life with at least one functioning kidney [3]. However, when both kidneys of a person are malfunctioning, this person requires kidney replacement therapy to survive, i.e., either dialysis or the donation of a functioning kidney.

Transplantations of deceased donor organs unfortunately imply long waiting times, as transplant waiting lists grow, given that the number of donations significantly exceed supply [4]. The other option is to find a living person that is willing to donate one of their kidneys. In general, living donor donations result in shorter waiting times and tend to have better long term outcomes compared to deceased donor donations [5]. Unfortunately, finding a willing, living donor does not guarantee (medical) compatibility with the recipient. Hence, the living donor exchange system was introduced in 1991 [6], which allows recipients with incompatible living donors, in the

†Timm Birka lead author

*Correspondence: moellering@encrypto.cs.tu-darmstadt.de

[1] ENCRYPTO, Technical University of Darmstadt, Darmstadt, Germany
Full list of author information is available at the end of the article

© The Author(s) 2022. **Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/. The Creative Commons Public Domain Dedication waiver (http://creativecommons.org/publicdomain/zero/1.0/) applies to the data made available in this article, unless otherwise stated in a credit line to the data.

following referenced as *pairs*, to exchange their donors, such that ideally each recipient can receive a compatible kidney donation. In most European kidney exchange programs the kidney transplantations of an exchange are executed simultaneously. Simultaneous operations limit the length of exchange cycles due to scarcity of medical staff. Additionally, exchanges of an exchange cycle that were initially deemed compatible in SPIKE can still be deemed incompatible after the required further assessment done by medical professionals. In case of longer exchange cycles, this leads to more pairs not receiving a kidney due to the failing of the whole exchange cycle. For these reasons, many European countries with kidney exchange programs limit the length of cycles to $L = 3$ or even $L = 2$, ensuring a practical feasibility [7]. In order for SPIKE to be applicable in European kidney exchange programs[1], we decided to limit the maximum length of cycles to $L = 3$.

In this work, we consider a scenario, in which several pairs exchange their donors in a cyclic fashion, so that each donating pair receives a compatible kidney. These cycles are called *exchange cycles* [7].

As a first step for finding possible exchange cycles, we have to evaluate the donors' and recipients' medical data to determine compatibility between pairs. Afterwards, we have to identify possible exchange cycles. This problem is known as the kidney exchange problem (KEP) [7] and can be described as finding cycles in a directed graph, where each vertex represents a pair and a directed edge describes the compatibility between two pairs. A schematic view of the protocol can be seen in Fig. 1.

The process requires the analysis of highly sensitive medical health data, which makes it crucial that no information is leaked accidentally or to unauthorized personnel. Thus, the KEP requires the implementation of strong privacy-preserving solutions, where the plaintext health information remains locally at each medical institution and the analysis is only run on "encrypted" data, which is leaking no sensitive data beyond the output: an exchange cycle with high transplantation success likelihood.[2] Note that such a distributed solution also enhances security against data breaches, as having to attack multiple parties is significantly harder than a single target. Similarly, it also simplifies the compliance with regulatory requirements potentially complicating or even prohibiting data sharing among facilities.

[1] For example, France and Sweden only accept a cycle Length of $L = 2$, while Spain, the Netherlands, and the United Kingdom accept longer cycles but prefer $L = 2$. For details see [8] and [7].

[2] This cycle still requires a final check by medical experts.

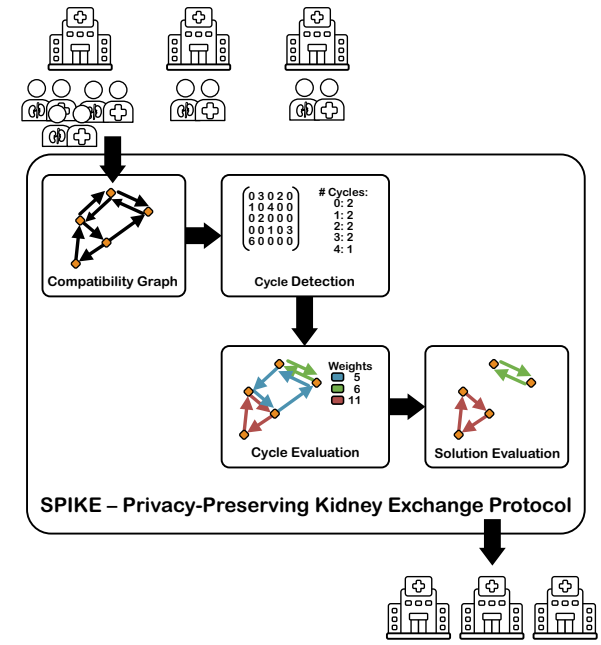


Fig. 1 Overview of our privacy-preserving kidney exchange protocol SPIKE. The best set of exchange cycles are calculated, while the patients' data remain strictly private

Contributions and outline

In this work, we provide the following contributions:

- *Efficient Privacy-Preserving Kidney Exchange protocol:* We design and implement SPIKE, a distributed, privacy-preserving protocol for solving the kidney exchange problem in the semi-honest security model. In contrast to the current state-of-the-art [9, 10], SPIKE improves efficiency as well as the medical compatibility matching by considering additional factors, namely, age, sex, human leukocyte antigens, and weight, that significantly affect compatibility between potential donors and recipients and is, thus, more robust than previous solutions by reducing the risk of failing procedures.
- *Comprehensive Empirical Evaluation:* We implement and extensively benchmark SPIKE and show that it significantly improves runtimes and communication costs compared to the state-of-the-art. We achieve about 30000× speedup over [9] and 400× over [10] thanks to our carefully optimized hybrid secure multi-party computation (MPC) protocols. Further, we provide additional (micro-) benchmarks and net-

work settings to further demonstrate scalability and practicality of SPIKE.
- *Open-source Implementation:* SPIKE is available under the GNU LGPL v3 license[3] here: https://encrypto.de/code/PPKE.

Related work

In this section, we summarize the related work on the Kidney Exchange Problem (KEP) with and without considering data privacy.

Robust KEP solutions

One major issue in kidney exchange programs is the potential cancellation of transplantations *after* having already determined exchange cycles of compatible donors and recipients. Reasons for such cancellations are manifold, e.g., a donor withdraws his consent, as his non-compatible relative has already received a kidney via the waiting list from a deceased donor in the meantime [11]. These issues call for *robust* solutions to the KEP, i.e., flexibility for recipient/donor dropouts and including as much as possible medical factors that can be algorithmically evaluated.

Carvalho et al. [12] propose three policies that are able to cope with dropouts within an kidney exchange cycle. One takes the costs (or missed gains) of planned transplants that do not proceed into account to find a solution with high probability of being successfully executed. The other two policies investigate strategies for recovering exchange cycles after dropouts. The plaintext algorithms in [12] are computationally expensive and, thus, cannot be trivially realized as secure computation protocols.

Ashby et al. [13] introduce a calculator for determining compatibility in kidney exchange, which they use to evaluate the importance of various medical factors, such as age, sex, obesity, weight, height, human leukocyte antigen (HLA) mismatches and ABO blood groups (see "Medical Background" Section). In our work, we increase the robustness of our privacy-preserving kidney exchange protocol by including the additional important biomedical factors from [13]. Furthermore, we recommend to use cycle sizes of two or three to reduce the impact of withdrawals [11]. The size is also beneficial for practical considerations with respect to medical staff and other resources needed for transplantations, as all operations of one exchange cycle should ideally be executed simultaneously. This recommendation reflects current best practices [14].

Privacy-preserving linear programming

Most currently used KEP solutions are based on Integer Linear Programming (ILP) formulations of the optimization problem. However, due to its superpolynomial complexity this is a largely unsolved space in the domain of privacy-preserving protocols. While multiple works considered secure linear programming using MPC (e.g., [15–17]), to our knowledge no results considering *integer* linear programming where some or all variables are not elements of a continuous field but must be integers. This research gap exsists for a good reason: Most exact ILP solving algorithms are based on "Branch and Bound" methods [18–20]. These methods find hyperplanes in the parameter space enclosing possible solutions, thus, pruning large sections of the parameter space. Unfortunately, a direct translation into the privacy-preserving realm would be vulnearble to timing attacks, hence insecure. Circuit-based MPC methods must exhibit deterministic runtimes, regardles of the specific inputs. Unfortunately, this disqualifies the privacy-preserving ILP approach for this work, as the presented algorithms inherently contain integer values in boundary conditions (e.g., encoding the graph structure).

Privacy-preserving KEP protocols

Just two works, both by Breuer et al. [9, 10], investigate how to solve the kidney exchange problem in a decentralized privacy-preserving manner. Both consider the semi-honest security model.

Privacy-preserving KEP protocol with HE

The first protocol [9] uses homomorphic encryption (concretely, a threshold variant of the Paillier cryptosystem [21]). It instantiates a computing party for each pair of a non-compatible donor and recipient at the providing hospital, thus, effectively creating a multi-party computation (MPC) protocol.

The protocol first pre-computes a set of all possible exchange constellations independent of any input data. Cycles of all lengths up to 3 are computed (but an arbitrary value could be chosen). Next, the pairs jointly compute an adjacency matrix with the pair-wise compatibility based on HLA crossmatching and ABO blood groups. Combining the results with the exchange constellations, the graph with the maximal size is delivered as the output. The protocol's runtime scales exponentially with the number of pairs: starting with a runtime of 14

[3] https://www.gnu.org/licenses/lgpl-3.0

seconds for two pairs it increases to 13 h for nine pairs. Unfortunately, such runtimes are prohibitive for practical deployment.

Privacy-preserving KEP protocol with Shamir's secret sharing

In a concurrent work to ours, Breuer et al. [10] introduced a privacy-preserving KEP protocol for crossover kidney exchanges with polynomial computation complexity. "Crossover" hereby means that the kidney exchange is done among two pairs, i.e., the exchange cycle size is limited to two in contrast to [9]. This limitation, however, enables a significant efficiency improvement for matchings with more than 13 pairs. For example, with 15 pairs it reduces the runtime of the old protocol [9] from 8.5 h to 30 min. Additionally, the new protocol enables a dynamic setting, where donor-recipient pairs can be added/removed from the exchange graph at any point in time. On the technical side, the authors replace HE and fully rely on a MPC-technique called Shamir's Secret Sharing (SSS) [22] implemented with the MP-SPDZ framework [23]. Beyond the dynamic setting and the change to MPC, the new protocol employs the graph matching algorithm by Pape and Conradt [24] for better efficiency in the matching between compatible donors and recipients.

Our privacy-preserving KEP protocol SPIKE offers significantly improved runtimes for real-world deployment. Our runtimes outperform the measured runtimes of previous works [9, 10], e.g., by a factor of hundreds/thousands for 9 recipient-donor pairs with a cycle length of 2. This is due to an efficient symbiosis of three MPC techniques and protocol optimizations that we will detail in the next section. Furthermore, we improve the robustness of SPIKE by including four additional biological factors notably impacting the transplantation success rate [13]. Thus, our protocol focuses on high medical quality rather than pure size, while also significantly improving efficiency.

Background

In this work, we present a privacy-preserving solution to the *kidney exchange problem* (KEP). We interpret the KEP as an optimization problem, specifically finding cycles with a maximal coverage of nodes on a compatibility graph and a maximal aggregated edge weight. The graph is constructed according to medical compatibility factors. This section gives the required background information to understand the underlying aspects of biomedicine, graph theory, as well as the used privacy-preservation techniques of secure multi-party computation (MPC).

Table 1 HLA split antigens assessed for biomedical donor – recipient compatibility testing in SPIKE

Class I			Class II	
HLA-A	HLA-B		HLA-DR	HLA-DQ
A23	B38	B60	DR11	DQ5
A24	B39	B61	DR12	DQ6
A25	B44	B62	DR13	DQ7
A26	B45	B63	DR14	DQ8
A29	B49	B64	DR15	DQ9
A31	B50	B65	DR16	
A32	B51	B71	DR17	
A33	B52	B72	DR18	
A34	B54	B75		
A66	B55	B76		
A68	B56	B77		
A69	B57			
A74	B58			

Medical background

In the following, we introduce the medical background, i.e., biological factors used in our protocol that cause general immunological incompatibility or influence success likelihood for a kidney transplantation.

General immunological compatibility

While many medical factors are involved in the definite assessment of donor-recipient compatibility, some can be algorithmically determined. For example, one key factor in avoiding allograft rejection—immunological compatibility—can be evaluated following evidence-based guidelines. Our kidney exchange protocol uses a specific form of immunological compatibility, the HLA crossmatch, as a transplant prohibiting factor.

Human leukocyte antigens crossmatch

The human immune system is responsible for the protection of the organism against potentially harmful invaders (called *pathogens*). Antigens are molecular structures often found on the surface of pathogens, but also naturally occurring in the body. *Antibodies* can attach to those structures, preventing the pathogens from docking, thus inhibiting their harmful effect. One important group of endogenous antigens, which occur in varying numbers in every human, forming the immunological "fingerprint" the immune system recognizes as normal, are the *human leukocyte antigens*. Out of the three classes of HLA [25], only classes I and II are of interest in this work.

With a *HLA crossmatch* general compatibility between recipient and donor can be determined: The human

leukocyte *antigens* of a donor are matched against existing human leukocyte *antibodies* of a possible recipient [26]. HLA crossmatch positive kidney transplants carry a significantly higher risk of antibody-mediated rejection or allograft rejection due to already existing antibodies [27, 28]. Modern immunosupressants might make such a procedure possible [29], but those cases require specialized, in-depth medical assessment and are out of scope of a general, algorithmic evaluation.

Following Eurotransplant's guidelines [26], we consider HLA groups, which are also most frequently screened in preparation for kidney replacement therapy [30]: the HLA encoded at HLA-A, -B, and -DR loci. Additionally, we consider the HLA-DQ coded antigens, which are related to some cases of antibody-mediated rejection [31]. The full list of HLA considered in SPIKE can be seen in Table 1.

Match quality estimation

Additionally to the previously introduced procedure that prevents immunological incompatibility, we strive to find the medically best/most robust solution to the kidney exchange problem – that includes maximal survival probability. For that, we calculate a match quality index, based on the following additional medical factors:

(i) *HLA match*

Additionally to the HLA crossmatch, HLAs influence the probability of a successful transplantation. Concretely, it increases if the donor has a subset or the same HLA as the recipient. The number of "mismatches" is associated with increased allograft rejection rates, as the probability that a recipient develops antibodies to those mismatched antigens increases [32]. HLA mismatches do not constitute exclusion criteria, as immunosupressants can reduce the rejection probability. The use of immunosupressants, however, is itself linked to higher rejection rates [32–34]. Special importance comes to the HLA-DQ group, as mismatches of it are strongly linked to antibody-mediated rejections [31].

Each person can inherit up to two types of HLA per group. Hence, at most two mismatches can occur per group [35]. The impact of HLA mismatches can be categorized in four bins: having no mismatch, a very rare case and mostly occurring in twin donor-recipient pairs, having 1–2 mismatches, having 3–4 mismatches, and, worst of all, having more than 5 mismatches [32]. The last group shows a more than 6% cumulative risk for death with a functioning graft during the first year. We weight HLA mismatches according to those four categories.

Table 2 ABO compatibility [36]

Blood group	Can receive from	Can donate to
O	O	O, A, B, AB
A	O, A	A, AB
B	O, B	B, AB
AB	O, A, B, AB	AB

(ii) *ABO blood type*

The ABO blood type system is based on the presence or absence of two types of antigens on the surface of the red blood cells [36]. The absence of both type A and type B antigens mark blood type O, the presence of both mark blood type AB, and the presence of only one mark blood type A and B, respectively. Receiving blood with an incompatible blood type leads to blood clumping due to an immune reaction and a possibly failed procedure. Compatible pairings are given in Table 2.

By pre-processing the donor organ, grafts from ABO incompatible donors are possible [37], although linked to severe adversary reactions during the first year post transplantation. These reactions include a higher risk of allograft loss, severe viral infections, antibody-mediated rejections, and postoperative bleeding. After this first year, however, the long-term survival rate is comparable to ABO compatible transplants [37].

(iii) *Age*

According to Waiser et al. [38], also age disparity influences allograft survival post transplant. The authors examined the role of age of the donor and recipient using two categories: *junior* participants aged below 55 years and *seniors* participants older than 55 years. The results show that intra-categorical transplants show the best outcomes, followed by pairings of junior donors and senior recipients. The worst outcomes were observed for pairings with senior donors and junior recipients.

(iv) *Sex*

As shown by Zhou et al. [39], the combination of donor-recipient sexes impact the transplant success probability. The worst allograft survival rates were observed in male recipients for female donor organs, while same-sex pairs performed slightly better than female recipients for male donor organs.

(v) *Weight*

Recipients, who received a kidney from a donor, who weighs less, have higher chances of allograft loss than other recipients [40]. El-Agroudy et al. [41]

reason that the allograft loss for recipients with kidneys from lighter donors might be caused by the kidney being unable to support the body functions of a heavier recipient.

Graph theory

We represent the structure of the kidney exchange problem (KEP), as a (bipartite) graph problem. A graph $\mathcal{G}$ consists of a set of vertices $\mathcal{V}$ and an edge set $\mathcal{E}$ connecting the vertices. Technically, we deal with a *bipartite* graph, i.e., consisting of two different sets of vertices (donors and recipients), but as those register pairwise for the kidney exchange, we can "collapse" each donor-recipient pair into one vertex in $\mathcal{V}$. If two vertices $v, u \in \mathcal{V}$ are connected by an edge, then $(v, u) \in \mathcal{E}$. We consider a directed graph with directed edges from v to u. Furthermore, we use *weighted* edges by associating a scalar weight to each edge, according to its "importance" in the network. The weights represent the degree of medical compatibility. We only allow positive edge weights.

Our goal is to find all cycles within the graph. A cycle c is a list of vertices $\{v_1, v_2, ..., v_m\}$, where an edge exists from vertex v_i to v_{i+1} for $i \in \{1, ..., m-1\}$ and, to close the "loop", from vertex v_m back to vertex v_1. In a vertex disjoint cycle, each vertex appears at most once within the cycle. We define the length of a cycle as the number of edges that are used to form that cycle.

One representation of a (weighted) graph structure is the *adjacency matrix*, a square matrix A with one row/column for each vertex. If an edge exists between vertices i and j, then, the entry $a_{ij} = w$, with $w > 0$ being the edge weight and $a_{ij} = 0$ otherwise. This work uses the fact that by raising the adjacency matrix to the ℓth power, one can quickly compute the number of paths between two vertices with a given length ℓ. That means, that vertices i and j are connected by $(A^\ell)_{ij}$ paths of length ℓ. The diagonal elements give the number of cycles of length ℓ by finding paths starting and ending on the same vertex.

Secure computation

Secure computation techniques enable multiple parties to securely evaluate an arbitrary function on their private inputs. Ideally nothing is leaked beyond what can be inferred from the output. A secure computation protocol must be able to realize this functionality without relying on a trusted party. To verify its security, it is typically compared to the so-called *ideal functionality*, which is a trusted third party that runs the computation on behalf of the data owners.

Privacy research has mainly worked on two paradigms for secure computation: Homomorphic Encryption (HE) and Secure Multi-Party Computation (MPC). HE schemes are special public-key encryption schemes that allow to realize (some limited) mathematical operations under encryption. However, they tend to be computing intensive making them (yet) often unsuitable for real-world applications. In contrast, MPC techniques are typically more efficient with respect to computation, as they are mainly based on efficient symmetric encryption and secret sharing. Additionally, MPC protocols can compute arbitrary functions. They are typically split into a setup and an online phase, where the setup phase is independent of the input data and, thus, can be precomputed. This separation enables to significantly speed up the time-critical online phase as pre-computation can be done in idle times when input data is not yet available. However, MPC involves two or more parties, who jointly evaluate the desired function in a secure manner, hence, it requires communication among the parties. Both paradigms have already been used in the context of privacy-preserving genome-wide association studies [42–44], as well as other applications in the health care area [45–48].

To have provably secure privacy guarantees while achieving practical efficiency, SPIKE efficiently combines multiple MPC techniques, which we introduce in the following.

Secure multi-party computation (MPC)

Introduced by Andrew Yao's seminarial work "How to Generate and Exchange Secrets" [49] in 1986, secure Multi-Party Computation (MPC) was considered a theoretical field first. MPC are cryptographic protocols that can securely compute an arbitrary function among two or more parties on their private inputs. Enabled by the rapid development of computer hardware and the development of the first MPC compiler "Fairplay" [50], first practical uses were demonstrated around the year 2004. Since then, MPC is a flourishing research field and due to novel protocols and optimizations, such as "Free XOR" [51] or "Halfgates" [52], practical applications in many fields were shown [45, 53, 54].

In this work, we rely on three well established secure *two-party* computation techniques, i.e., the secure computation protocols are run among exactly two parties: Arithmetic Secret Sharing ($\mathcal{A}$), Boolean Sharing ($\mathcal{B}$), both based on [55], and Yao's Garbled Circuits (Y), originally introduced in [49]. Each technique differs in how it creates (*shares*) and reconstructs secrets, but also how (efficiently) certain types of operations can be realized.

Secure outsourcing

Although we use two-party MPC to perform the computation, *any* number of parties can provide input data. This method of *secure outsourcing* [56] works by all *data owners* secret sharing their data and sending one share to each of the two non-colluding *computation servers*.

Secret sharing, thereby, means that the sensitive data is split into two random looking shares and each of the two computation servers receives only one of those. Specifically, a single computing server cannot infer any information about the secret input data from its share. Instead, the sensitive information can only be obtained when the secret shares of both servers are combined. The two computing servers, then, perform the actual secure computation on behalf of the data owners on the random looking secret shares, while not being able to learn anything about the private input data. To summarize, in the outsourcing scenario an arbitrary number of data owners can participate without leaking/uploading *any* sensitive information to an external party.

This scenario has three main benefits:

- The communication of N-party MPC scales at least linearly, often quadratic in the number of participating parties [57]. By outsourcing the N-party computation to $M \ll N$ parties, here $M = 2$, the complexity is improved substantially.
- As the input owners do not participate in the computation itself, the outsourced protocol provides security against malicious data owners [56]. At most they can corrupt the correctness of the calculation, but not the privacy.
- The location of the computation servers can be chosen pragmatically, e.g., two locations with high bandwidth and low latency network connection. Of course, the computation servers are assumed to not collude.
- Compared to N-party MPC setups, two-party MPC requires to trust exactly one computation server. A data owner can also run one himself. Using $N > 2$ non-colluding parties can be more efficient [58, 59], but ensuring the non-collusion among all N parties is more challenging/might not be realistic. Full threshold N-party MPC schemes [60], i.e., where all but one party can be compromised, significantly reduces efficiency/increase communication.[4] To summarize, outsourcing to two non-colluding servers offers a good trade-off between efficiency and security.

[4] As mentioned above, the communication of MPC protocols generally scales quadratically in the number of parties, thus, more parties significantly increase communication cost. When considering full threshold security, runtimes significantly increase as well. The state-of-the-art MPC framework MOTION [60] which offers full threshold security provides several benchmarks comparing to the two-party MPC framework ABY [61]. In a LAN setting, it takes ABY less than 0.1 seconds (online runtime) to securely compute an AES-128 circuit with two parties while MOTION requires at least twice (resp. four times) the time with three (resp. five) parties.

Table 3 Garbled AND gate

Input w_0	Input w_1	Output w_2	Garbled value
$k_0^{w_0}$	$k_0^{w_1}$	$k_0^{w_2}$	$\mathrm{Enc}_{k_0^{w_0},k_0^{w_1}}(k_0^{w_2})$
$k_0^{w_0}$	$k_1^{w_1}$	$k_0^{w_2}$	$\mathrm{Enc}_{k_0^{w_0},k_1^{w_1}}(k_0^{w_2})$
$k_1^{w_0}$	$k_0^{w_1}$	$k_0^{w_2}$	$\mathrm{Enc}_{k_1^{w_0},k_0^{w_1}}(k_0^{w_2})$
$k_1^{w_0}$	$k_1^{w_1}$	$k_1^{w_2}$	$\mathrm{Enc}_{k_1^{w_0},k_1^{w_1}}(k_1^{w_2})$

Security model

In our work, we consider the *semi-honest* security model, where the two computation servers are assumed to be honestly following the protocol, while trying to learn as much information as possible. By "honestly following the protocol" we, thereby, mean that they adhere to the specifications of the protocol, e.g., they do not manipulate local calculations or provide inconsistent data. Additionally, the two computation servers are assumed to not collude. This security model provides protection against curious personnel or accidental data leakage and the omission of a trusted third party further reduces the impact of a potential data breach. Although weaker than the malicious security model, where the parties might arbitrarily deviate from the protocol, the semi-honest security model is sufficient for our use case, as hospitals are generally trusted, but legally not allowed to simply share the data among each other. Furthermore, the semi-honest security model enables significantly more efficient computation than the malicious model and, hence, provides a good efficiency-privacy trade-off. While the European Data Protection Board recommends security against malicious adversaries when performing joint calculations with parties under jurisdiction of insecure countries [62], the semi-honest security model is the predominant model in data protection concepts for federated medical research[5]. Hence, it is a valuable security model in our application scenario. Previous works on privacy-preserving KEP protocols [9, 10] are also in the semi-honest security model.

Notation

In the following, $\langle x \rangle_i^s$ denotes a secret share of x shared using MPC technique $s \in \{A, B, Y\}$ and held by party P_i, where $i \in \{0, 1\}$.

Yao's garbled circuits ($\mathcal{Y}$)

Yao's Garbled Circuits enable two parties, called the *garbler* and the *evaluator*, to securely evaluate a function f represented as *Boolean circuit*, i.e., a directed acyclic

[5] For examples see [63] and [64] (in German language).

graph, where the nodes are logic gates and the edges (called *wires*) are the Boolean in- and outputs. For functional completeness AND and XOR gates are sufficient. The garbler generates random keys for each possible state of each wire $k_0^w, k_1^w \in \{0,1\}^\kappa$, where κ is the symmetric security parameter (set to $\kappa = 128$ in our implementation) and w is the respective wire. For all input combinations of each gate in the circuit, it uses the input keys to encrypt the corresponding output key (cf. Table 3). The order of the four ciphertexts is then permuted randomly and the *garbled circuit* is sent to the evaluator together with the keys associated to the garbler's input. As those keys look random, the evaluator cannot extract any information about the input of the garbler. Next, the evaluator engages in an oblivious transfer [65, 66] to receive the keys for his input without revealing it to the garbler. Equipped with all keys, it evaluates the garbled circuit to receive the circuit's output keys, which the parties jointly decode. Thanks to several optimizations, e.g., [51, 52, 67], $\mathcal{Y}$ requires no communication for the evaluation of an XOR gate and only 1.5κ bits of communication for AND gates. $\mathcal{Y}$ needs a constant number of communication rounds independent of the circuit depth.

Boolean and arithmetic secret sharing ($\mathcal{B}/\mathcal{A}$)

In Additive Arithmetic Secret Sharing ($\mathcal{A}$) operations on ℓ-bit inputs are done in an algebraic ring $\mathbb{Z}_{2^\ell}$, where ℓ is the bit length. Although the technique can also be used among an arbitrary number of parties [68], we focus here on the two party setting as introduced by Goldreich et al. [55].

To share a secret value x, party P_i, $i \in \{0,1\}$, generates a random value $r \in_R \mathbb{Z}_p$ and sets its arithmetic share to $\langle x \rangle_i^A = r$. Then, P_i also determines party P_{1-i}'s share $\langle x \rangle_{1-i}^A = x - r \mod 2^\ell$ and sends it to P_{1-i}. To reconstruct the secret, one needs to know *both* shares and compute $x = \langle x \rangle_0^A + \langle x \rangle_1^A \mod 2^\ell$. Boolean Secret Sharing ($\mathcal{B}$) describes the special case, where $\ell = 1$, viz. $\mathbb{Z}_2 = \{0,1\}$.

Note that a share $\langle x \rangle_i^A$ (resp. $\langle x \rangle_i^B$) is random and does not leak anything about the secret x. Secure addition (respectively, XORing in $\mathcal{B}$) can be executed locally, that is without communication between the parties. Secure multiplication (respectively, AND in $\mathcal{B}$) is done in an interactive protocol among the two parties using so-called multiplication triples [61, 69, 70]. Using only addition and multiplication (similarly, AND and XOR) arbitrary functions can be calculated.

ABY framework

All three MPC techniques are implemented in the state-of-the-art secure two-party computation framework ABY [61], which we use in our experiments[6]. Additionally, ABY also contains efficient conversions between them and supports Single Instruction Multiple Data (SIMD) operations to parallelize identical operations on different data, while reducing memory usage and runtime. Arithmetic Secret Sharing in ABY is performed on the ring $\mathbb{Z}_{2^\ell}$, that is with 2^ℓ elements, where ℓ is the bitlength of the data type (most often $\ell = 32$bit). A recent work by Patra et al. [53] improves [61] by making the online communication of scalar multiplication independent of the vector dimensions and reducing online communication for AND gates with two inputs in $\mathcal{B}$ by a factor of 2. Unfortunately, these protocols have been implemented only very recently in MOTION2NX [72], which is why we use [61] in our implementation.

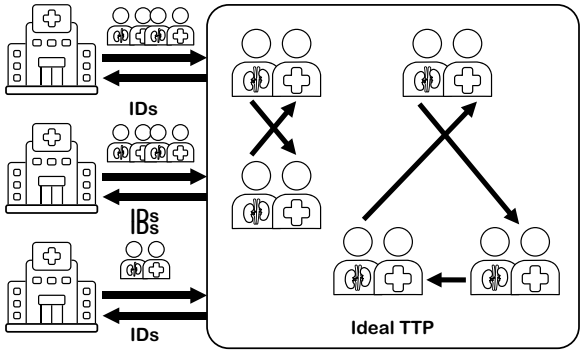


Fig. 2 Ideal functionality for a secure privacy-preserving protocol solving the kidney exchange problem (KEP)

Methods

In this section, we first define the privacy-preserving Kidney Exchange problem (KEP) and its requirements. Then, we present our solution, which we name SPIKE, consisting of tailored modular secure MPC protocols and include a complexity analysis.

Problem statement

Figure 2 shows the ideal functionality for solving the privacy-preserving KEP in a provably secure way. Assuming the (not realistic) availability of a trusted third party (TTP), hospitals send the data of recipients and donors to the TTP, which calculates cycles of pairs of recipients and donors with the highest probability to be compatible.

[6] Note that our protocols in Section "Methods" can also be instantiated with other MPC frameworks. For example, an instantiation with MP-SPDZ would also enable the switch to a malicious security model but at the costs of significantly reduced efficiency [71]. However, considering the security and efficiency requirements discussed in the Subsections "Outsourcing Data-Model" and "Security Model", 2-party MPC with the ABY framework offers the best trade-off.

Then, the TTP returns for each recipient the information about his/her donor to the respective hospital. Note that a final evaluation must still be done by medical experts. A privacy-preserving KEP protocol is meant for automatizing and, thus, accelerating, the process of the creation of the kidney exchange cycles.

Requirements We define the following requirements for a secure privacy-preserving KEP protocol:

- *Privacy* The privacy-preserving KEP protocol must realize the same functionality as described in the ideal functionality, while removing the problematic assumption of a TTP, i.e., it must leak nothing beyond what can be inferred from the output.
- *Efficiency* The privacy-preserving KEP protocol must be efficient in terms of communication and computation, such that it can be run in reasonable time on standard server hardware.
- *Decentralization* The privacy-preserving KEP protocol must be decentralized, i.e., the highly sensitive medical information of donors and patients must remain locally at the respective medial institution (inherently being compliant with the data minimisation principle).
- *Adaptability for Medical Experts* The priva-cy-preserving KEP protocol must be flexibly adaptable for medical experts with respect to the selection of biological factors for the algorithmic evaluation of compatibility. They must be able to adjust the weighting between the included factors and cycle lengths according to state-of-the-art medical advancements. The protocol must be easily extendable to new factors and additional HLA groups.

SPIKE: a MPC-based privacy-preserving KEP protocol

In this section, we provide the building blocks for our Secure and Private Investigation of the Kidney Exchange problem: SPIKE. It fulfills above requirements (see also the overview of the phases in Fig. 1).

First, we explain the matching phase, which analyzes the compatibility between donors and recipients using six biological factors presented in the "Background" section. Then, we continue with the determination of the number of potential exchange cycles given a cycle length. The third phase computes the probability of a successful transplantation based on the matching results for all potential exchange cycles. In the final phase, we output a robust set of *disjoint* exchange cycles, i.e., with a high probability for compatibility. The final result contains a combination of disjoint exchange cycles that maximizes the likelihood of as many transplantation as possible being successful. The weight of a cycle c is denoted by w_c where a higher value indicates a higher likelihood of a transplantation being successful. Thus, the weight of a set of disjoint cycles $\mathcal{C}$, i.e., the likelihood for as many transplantation being successful in the set, can be described as the sum of all cycles w_{c_i} for $i \in \{1, \ldots, |\mathcal{C}|\}$ in the set. The weight of a cycle is determined by the sum over all edge weights w_e in the cycle. Finally, the weight of an edge is determined by the sum over the results of all matching criterion which are multiplied by a weight which can be assigned by medical experts to highlight certain biomedical factors. Note that we write this computation as a dot product between a vector $\vec{p}(k,l)$ and $\vec{w}$ where $\vec{p}(k,l)$ contains the results of the matching between pair k and l in vector form and $\vec{w}$ the respective weights of each criterion. Equation (1) describes the previous conditions. To achieve the described result, we greedily select disjoint cycles in decreasing priority according to the weight of each individual cycle.

$$\max \sum_{i=1}^{|C|} \overbrace{\left(\sum_{j=1}^{|cLen|} \underbrace{\left(\vec{p}(k,l) \cdot \vec{w}\right)_j}_{:=w_{e_j}} \right)_i}^{:=w_{c_i}} \tag{1}$$

Note that our solution is a local optimum which is computed with a greedy algorithm while the solutions by Breuer et al. [9, 10] are globally optimal. We argue that a locally optimal solution is sufficient in our application scenario for two reasons: First, we assume that the locally optimal results are in close proximity of the global optimum, as real world data sets will likely show sparse compatibility and the additional medical compatibility factors considered by SPIKE will increase the solution quality. Second, the additional expert evaluation following the algorithmic matching will most likely introduce a much higher variance in the chosen solution. The empirical evaluation of those two claims are interesting points for further research requiring real-world kidney exchange data sets. The protocol presented by Breuer et al. [10] enables usage in a dynamic setting, i.e., a setting in which donor-recipient pairs are put together in a pool where pairs come and go over time. They run their matching protocol on a subset of the pairs of the pool and, afterwards, evaluate the resulting compatibility graph. By design, SPIKE enables usage in a dynamic setting, too, since each part of the protocol can be executed independently of the others parts as long as they receive the output of the previous parts. Such a dynamic setting can also be adapted to an outsourcing scenario. Each input party has their own pool of donor-recipient pairs where they can select a random subset of pairs and send them to the computing parties.

Table 4 `matchHLA`($\langle \mathsf{hla_d} \rangle^{\mathcal{B}}$: vector, $\langle \mathsf{ahla_r} \rangle^{\mathcal{B}}$: vector) → int

```
1: ⟨comp⟩^B ← [⟨0⟩^B]^|HLA|
2: for i = 0, ..., |HLA| − 1 do                          ▷ SIMD
3:     ⟨comp⟩^B[i] ← ⟨hla_d[i]⟩^B ∧ ⟨ahla_r[i]⟩^B
4: end for
5: ⟨combined⟩^B ← ORTREE(⟨comp⟩^B)[0]
6: return ¬⟨combined⟩^B
```

Notation

We use Boolean operators to concisely present our MPC protocols: $\wedge$ is `AND`, $\vee$ is `OR`, $\neg$ is `Not`, and $\oplus$ is `XOR`. 0/1 are False/True. $|x|$ indicates the length of a vector x, i.e., the number of elements. Non trivial variable names in protocols are written in sans serif, function names (and calls) `monospaced`. Branching, implemented with `MUX` (multiplexer) Gates, is displayed using ternary notation: condition **?** true statement **:** false statement.

Compatibility matching

The first phase of SPIKE is called *compatibility matching*. In this phase, we compare the pair-wise general compatibility and match quality of all donors and recipients with respect to human leukocyte antibodies and antigens, ABO blood group compatibility, age, sex, and weight. The output of this phase is a weighted compatibility graph, where the edge weights indicate the probability of compatibility for each pair.

We present the main protocols for the compatibility assessment in the following. The subprotocols for assessing the individual matching criteria HLA mismatches, ABO blood type, age, sex, and weight are given as Additional file 1: Tables S1–S6 in the Appendix.

The HLA crossmatch subprotocol is shown in Table 4. It tests whether the human leukocyte antigens of the donor are unsuitable to the human leukocyte antibodies of the recipient rendering them incompatible.

The subprotocol takes a vector with the antigens of a donor hla_d and a vector with the antibodies of the recipient ahla_r as input. The number of observed HLA, denoted by $|\mathsf{HLA}|$, is publicly known. A vector comp stores whether the recipient possesses an antibody against any of the donor's HLA (cf. Line 3). For enhanced efficiency, we parallelize this comparison as *Single Instruction, Multiple Data* (SIMD) operation, such that all HLA matches of one patient are computed in just one step. Afterwards, the overall compatibility (i.e., no antigen-antibody mismatch was found) is computed with `OR` gates in a tree structure, to reduce the (multiplicative) dephts of the circuit from $|\mathsf{HLA}|$ to $\log_2(|\mathsf{HLA}|)$. To prepare for further processing, we invert combined and return it as result of the HLA crossmatching in Line 6.

In Table 5, we present our MPC protocol that combines the results of the evaluated six medical criteria influencing the compatibility of a kidney donation into a weighted

Table 5 `computeCompatibilityGraph`($\langle \mathsf{pairs} \rangle^{\mathcal{B}}$: vector, $\langle \mathsf{w} \rangle^{\mathcal{A}}$: vector) → weighted adjacency matrix

```
1: ⟨compG⟩^A ← matrix ∈ {⟨0⟩^A}^(|pairs|×|pairs|)
2: for i = 0, ..., |pairs| − 1 do
3:     for j = 0, ..., |pairs| − 1 do
4:         d ← pairs[i].d                          ▷ Extract donor.
5:         r ← pairs[j].r                          ▷ Extract recipient.
6:         ⟨edge_w⟩^A ← ⟨1⟩^A +
             ⟨w⟩^A[0] · b2a(evalHLA(⟨d.hla⟩^B, ⟨r.hla⟩^B)) +
             ⟨w⟩^A[1] · b2a(evalABO(⟨d.bg⟩^B, ⟨r.bg⟩^B)) +
             ⟨w⟩^A[2] · b2a(evalAge(⟨d.a⟩^B, ⟨r.a⟩^B)) +
             ⟨w⟩^A[3] · b2a(evalSex(⟨d.sex⟩^B, ⟨r.sex⟩^B)) +
             ⟨w⟩^A[4] · b2a(evalWeight(⟨d.weight⟩^B, ⟨r.weight⟩^B))
7:         ⟨compG⟩^A[i][j] ←
             b2a(matchHLA(⟨d.hla⟩^B, ⟨r.ahla⟩^B) > ⟨0⟩^B  ?
             a2b(⟨edge_w⟩^A) : ⟨0⟩^B)
8:     end for
9: end for
10: return ⟨compG⟩^A
```

adjacency matrix indicating the donor-recipient compatibility, named compG.

It takes a vector pairs containing all possible pairs of donors and recipients and a vector w with a weight for each criteria (i.e., how much it influences the overall probability for good compatibility compared to the other factors) as input. Lines 4 to 6 additively combine the computed weighted probability of each compatibility criterion and assign it to the respective edge representing the donor of the i-th pair and the patient of the j-th pair, where $i \neq j$ and $i, j \in \{0, \ldots, |\mathsf{pairs}| - 1\}$. In Line 7, we additionally check whether the i-th donor and the j-th patient exhibit general immunological compatibility using the HLA crossmatch subprotocol (cf. Table 4). If this is the case, we store the result of the edge weight at the respective index, otherwise, we store the secret shared constant 0.

MPC Cost. The two sections in Table 4 evaluate $|\mathsf{HLA}|$ `AND` gates (as SIMD) and $\log_2(|\mathsf{HLA}|)$ `OR`[7] gates, respectively. Finally, we invert combined once. This results in a circuit depth of $\log_2(|\mathsf{HLA}|) + 1$ and a total number of `AND` gates of $2 \times |\mathsf{HLA}|$. Boolean sharing ($\mathcal{B}$) is used in this protocol, as Boolean operations are performed and the circuit depths is low, thanks to the SIMD vectorization [61].

To fully assess the matching quality (Table 5), all criteria have to be evaluated for each recipient, i.e., Table 4 and Additional file 1: Tables S1, S2, S4, and S6 are run $|\mathsf{pairs}|^2$ times. Then, in Table 5, we additionally evaluate five multiplications, five additions, one comparison, one `AND` gate, and one `MUX` gate. Due to the arithmetic operations in this protocol, the results of the compatibility evaluation protocols must be converted between $\mathcal{B}$ and $\mathcal{A}$.

[7] $A \vee B = 1 \oplus ((1 \oplus A) \wedge (1 \oplus B))$

Table 6 `determineNumberCycles`($\langle$compG$\rangle^{\mathcal{A}}$: matrix) $\rightarrow$ number of cycles

1: $\langle \text{compG} \rangle^{\mathcal{B}} \leftarrow \texttt{a2b}(\langle \text{compG} \rangle^{\mathcal{A}})$
2: $\langle \text{uG} \rangle^{\mathcal{A}} \leftarrow \texttt{removeWeights}(\langle \text{compG} \rangle^{\mathcal{B}})$
3: $\langle \text{cG} \rangle^{\mathcal{A}} \leftarrow \texttt{pow}(\langle \text{uG} \rangle^{\mathcal{A}}, \text{cLen})$
4: $\langle |\text{cycles}| \rangle^{\mathcal{A}} \leftarrow \langle 0 \rangle^{\mathcal{A}}$
5: **for** $i = 0, \ldots, |\text{pairs}| - 1$ **do**
6: $\quad \langle |\text{cycles}| \rangle^{\mathcal{A}} \leftarrow \langle |\text{cycles}| \rangle^{\mathcal{A}} + \langle cG \rangle^{\mathcal{A}}[i][i]$
7: **end for**
8: **return** $\langle |\text{cycles}| \rangle^{\mathcal{A}}$

Cycle computation

The second phase of SPIKE computes the number of possible kidney exchange cycles given a concrete input cycle length[8] from the compatible donors and recipients that were output by the compatibility matching. Our MPC protocol for this part is shown in Table 6.

Table 6 takes the secret shared weighted compatibility graph compG as input. The desired length of cycles cLen is public. We first compute the unweighted adjacency matrix in Line 2 (cf. Additional file 1: Table S7, in the Appendix). For the unweighted matrix, we compute the cLen-th power using a naïve implementation[9]. The entries in this resulting matrix indicate how many paths of length cLen start at vertex i and end at vertex j. For cycles, the entries are on the diagonal, as start- and end-vertex are identical. Following this thought, the sum of the entries of the diagonal is the total number of cycles with the given cycle length cLen. Note that this number contains duplicates, namely, "congruent" cycles that are the same, but were found via a different start/end vertex.[10] We remove the duplicates later in Additional file 1: Table S9 (described in the Appendix).

MPC Cost. Table 6 contains mostly arithmetic operations ($|\text{pairs}|^3$ multiplications and $(|\text{pairs}|^3 - |\text{pairs}|^2$ additions), however, the computation of the unweighted adjacency matrix is most efficiently performed in $\mathcal{B}$ $|\text{pairs}|^2$ comparisons and MUX gates). For that reason we convert compG from $\mathcal{A}$ to $\mathcal{B}$ (cf. Line 1) and back (in Additional file 1: S7).

Cycle evaluation

The third phase of SPIKE then identifies the most likely successful unique exchange cycles consisting of compatible pairs of donors and recipients and sorts them in descending order with respect to their weight.

[8] As discussed in the Related Work, we recommend 2 to 3 to foster robustness.

[9] Even though exhibiting a cubic runtime complexity, this part's performance is negligible compared to the following parts (cf. Fig. 4), hence, an optimization is not vital.

[10] Cycle (A, B, C) and cycle (B, C, A) are duplicates, but cycle (C, B, A) is not.

Table 7 `findCycles`($\langle$compG$\rangle^{\mathcal{Y}}$: matrix, cCycle: vector, $\langle$allCycles$\rangle^{\mathcal{Y}}$: vector, $\langle$weight$\rangle^{\mathcal{Y}}$: int, $\langle$valid$\rangle^{\mathcal{Y}}$: int) $\rightarrow$ vector of tuples

1: **if** |cCycle| == cLen **then**
2: $\quad \langle \text{weight} \rangle^{\mathcal{Y}} \leftarrow \langle \text{weight} \rangle^{\mathcal{Y}} + \langle \text{compG} \rangle^{\mathcal{Y}}[\text{cLen} - 1][0]$
3: $\quad \langle \text{valid} \rangle^{\mathcal{Y}} \leftarrow \langle \text{compG} \rangle^{\mathcal{Y}}[\text{cLen} - 1][0] > \langle 0 \rangle^{\mathcal{Y}}$ **?** $\langle \text{valid} \rangle^{\mathcal{Y}} + \langle 1 \rangle^{\mathcal{Y}}$ **:** $\langle \text{valid} \rangle^{\mathcal{Y}} + \langle 0 \rangle^{\mathcal{Y}}$
4: $\quad \langle \text{addC} \rangle^{\mathcal{Y}} \leftarrow \langle \text{cLen} \rangle^{\mathcal{Y}} == \langle \text{valid} \rangle^{\mathcal{Y}}$
5: $\quad \langle \text{cWeight} \rangle^{\mathcal{Y}} \leftarrow \langle \text{addC} \rangle^{\mathcal{Y}}$ **?** $\langle \text{weight} \rangle^{\mathcal{Y}}$ **:** $\langle 0 \rangle^{\mathcal{Y}}$
6: $\quad \langle \text{cycle} \rangle^{\mathcal{Y}} \leftarrow \langle \text{cCycle} \rangle^{\mathcal{Y}}$
7: $\quad \langle \text{allCycles} \rangle^{\mathcal{Y}}.\texttt{append}((\langle \text{cWeight} \rangle^{\mathcal{Y}}, \langle \text{cycle} \rangle^{\mathcal{Y}}))$
8: $\quad \texttt{revert}()$
9: **else**
10: $\quad$ **for** $i = 0, \ldots, |\text{pairs}| - 1$ **do**
11: $\quad\quad$ **if** cCycle.contains(i) **then**
12: $\quad\quad\quad \texttt{continue}$
13: $\quad\quad$ **else**
14: $\quad\quad\quad \langle \text{weight} \rangle^{\mathcal{Y}} \leftarrow \langle \text{weight} \rangle^{\mathcal{Y}} + \langle \text{compG} \rangle^{\mathcal{Y}}[-1][i]$
15: $\quad\quad\quad \langle \text{valid} \rangle^{\mathcal{Y}} \leftarrow \langle \text{compG} \rangle^{\mathcal{Y}}[-1][0] > \langle 0 \rangle^{\mathcal{Y}}$ **?** $\langle \text{valid} \rangle^{\mathcal{Y}} + \langle 1 \rangle^{\mathcal{Y}}$ **:** $\langle \text{valid} \rangle^{\mathcal{Y}} + \langle 0 \rangle^{\mathcal{Y}}$
16: $\quad\quad\quad$ cCycle.append(i)
17: $\quad\quad\quad \langle \text{allCycles} \rangle^{\mathcal{Y}} \leftarrow \texttt{findCycles}(\langle \text{compG} \rangle^{\mathcal{Y}}, \text{cCycle}, \langle \text{allCycles} \rangle^{\mathcal{Y}}, \langle \text{weight} \rangle^{\mathcal{Y}}, \langle \text{valid} \rangle^{\mathcal{Y}})$
18: $\quad\quad\quad$ cCycle.remove()
19: $\quad\quad\quad \texttt{revert}()$
20: $\quad\quad$ **end if**
21: $\quad$ **end for**
22: **end if**
23: **return** $\langle \text{allCycles} \rangle^{\mathcal{Y}}$

Our first subprotocol for this phase, shown in Table 7, finds all exchange cycles of the desired length (including duplicates) and computes the weight of each cycle. This weight is the sum of all included weighted edges. As mentioned before, the weight associated with an exchange cycle indicates the probability of the transplantation being successfully carried out, i.e., its robustness.

The subprotocol takes the secret shared compatibility graph compG output by Table 5, the currently analyzed exchange cycle cCycle, its secret shared weight weight, a secret shared counter valid, which tracks the number of edges in cCycle, and a vector of secret shared tuples allCycles, which will be consecutively filled with all possible exchange cycles and the corresponding sum of weights. In a recursive execution of Subprotocol 7, this vector is filled, as explained in detail in the following. The desired output cycle length cLen and the number of recipient-donor pairs |pairs| are public. Contrary to the protocols in [9, 10], the output number of cycles |cycles| found in Table 6 is revealed for efficiency reasons. We consider this leakage as acceptable since it leaks only a very high-level aggregate property, generally not allowing the inference of the compatibility graph's topology[11].

[11] Exceptions are fully connected and unconnected graphs, as well as for $|\text{cycles}| = 1$ at pathological graph topologies. The first topologies have no security implication whatsoever and the later can, e.g., be easily avoided by introducing a check ensuring that the output is only revealed when more cycles have been found.

Table 8 `evaluateCycles`($\langle$compG$\rangle^{\mathcal{Y}}$: matrix) $\rightarrow$ vector of tuples

```
1:  ⟨allCycles⟩^Y, cCycle ← ∅
2:  ⟨weight⟩^Y, ⟨valid⟩^Y ← ⟨0⟩^Y
3:  for i = 0, ..., |pairs| − 1 do
4:      cCycle.append(i)
5:      ⟨allCycles⟩^Y ← findCycles(⟨compG⟩^Y, cCycle,
            ⟨allCycles⟩^Y, ⟨weight⟩^Y, ⟨valid⟩^Y)
6:      cCycle.remove()
7:  end for
8:  |allCycles| ← totalCycles()
9:  ⟨sortedCycles⟩^Y ← kNNSort(⟨allCycles⟩^Y, |cycles|)
10: |unique| ← ⌊ |cycles| / cLen ⌋
11: ⟨filteredCycles⟩^Y ← removeDuplicates(⟨sortedCycles⟩^Y)
12: return ⟨filteredCycles⟩^Y
```

In the legal sense, the revealed number is considered non-sensitive as well, as it is an aggregated, anonymized datum.

Table 7 first checks if the currently analyzed exchange cycle cCycle already has the desired length cLen. If this is the case, the weight of the last edge is added to the respective sum of this cycle's weights in Line 2. Next, each valid cCycle is added to allCycles with its respective sum of weights. A cCycle is valid, if it is closed (cf. Lines 3–4). An invalid cycle is associated with weight zero (cf. Line 5). Note that a weight of zero does not contribute to the solution, hence a cycle with weight zero is never considered for a solution. In Line 8, the operations done in Lines 2–3 are reverted to restore the state of cCycle before the last edge was added, i.e., the weight of the last edge is subtracted from weight and valid is decreased by 0 (no edge) or 1 (edge).

Cycles that do not have the desired length yet are handled in Lines 10–21. For these exchange cycles, the subprotocol checks whether they are already part of cCycle, as each vertex may only appear at most once (cf. Line 11). If it is not included, the weight of the edge from the previous to the new vertex is added by increasing cCycle's weight and counter $\langle valid \rangle^{\mathcal{Y}}$, and the new vertex is added to cCycle (cf. Lines 14–16). Afterwards, Table 7 is recursively called again with the newly added vertex. Once the function returns, we revert the operations done before to be able to analyze the next cycle (cf. Lines 18–19).

The second subprotocol of the cycle evaluation (cf. Additional file 1: Table S9 in the Appendix) removes duplicates from the exchange cycles set. It extracts $\#\text{unique} = \lfloor \frac{\#\text{cycles}}{\text{cLen}} \rfloor$ cycles and returns the k cycles with the highest probability for a successful transplantation.

Table 8 combines the previously discussed subprotocols. It first calculates the sum of weights for each cycle with Table 7 (findCycles) and sorts the result using Additional file 1: Table S8 (kNNSort), such that only the k cycles with the largest weight are output. Those are all

Table 9 `evalSolution`($\langle$filteredCycles$\rangle^{\mathcal{Y}}$: vector of tuples) $\rightarrow$ tuple(int, vector of vectors)

```
1:  ⟨sets⟩^Y ← ∅
2:  ⟨weights⟩^Y ← ∅
3:  ⟨dummyC⟩^Y ← {⟨|pairs|⟩^Y}^cLen
4:  for i = 0, ..., |unique| − 1 do
5:      ⟨tempSet⟩^Y ← ∅
6:      ⟨tempSet⟩^Y.append(⟨filteredCycles⟩^Y[i][1])
7:      ⟨weight⟩^Y ← ⟨filteredCycles⟩^Y[i][0]
8:      counter ← 1
9:      for j = 0, ..., |unique| − 1 do
10:         if i == j then
11:             continue
12:         end if
13:         ⟨cCycle⟩^Y ← ⟨filteredCycles⟩^Y[j][1]
14:         ⟨disjoint⟩^Y ← disjointSet(⟨tempSet⟩^Y,
              ⟨cCycle⟩^Y)
15:         ⟨vertices⟩^Y ← ∅
16:         ⟨vertices⟩^Y.append(⟨disjoint⟩^Y ?
              ⟨cCycle⟩^Y : ⟨dummyC⟩^Y)
17:         ⟨weight⟩^Y ← ⟨disjoint⟩^Y ? ⟨weight⟩^Y : ⟨0⟩^Y
18:         ⟨tempSet⟩^Y.append(⟨vertices⟩^Y)
19:         counter ← counter + 1
20:     end for
21:     ⟨sets⟩^Y.append(⟨tempSet⟩^Y)
22:     ⟨weights⟩^Y.append(⟨weight⟩^Y)
23: end for
24: return findMaximumSet(⟨sets⟩^Y, ⟨weights⟩^Y)
```

valid cycles, possibly including duplicates. Afterwards, the protocol removes all duplicates within the k cycles.

MPC Cost. The complexity of Subprotocol 7 depends on the number of pairs |pairs|, cLen, and the number of possible cycles |allCycles|. It is most efficient in $\mathcal{Y}$, as the MUX gates are not independent, thus, creating a deep circuit of depth $\mathcal{O}(|\text{allCycles}| \times |\text{cycles}| \times \text{cLen})$. For removing duplicates and extracting the most robust k exchange circuits, we evaluate $\#\text{cycles} \times (\#\text{unique} + \sum_{i=0}^{\#\text{cycles}} (\text{cLen} \times (\text{cLen} - 1)))$ comparisons, $\#\text{cycles} \times \sum_{i=0}^{\#\text{cycles}} ((\text{cLen} \times (\text{cLen} - 1)))$ AND gates, $\#\text{cycles} \times \sum_{i=0}^{\#\text{cycles}} (\text{cLen} - 1)$ OR gates, $\#\text{cycles} \times \#\text{unique} \times (1 + \text{cLen}) + \#\text{cycles}$ MUX gates. This step is most efficient with $\mathcal{Y}$, as the circuit is very deep. Thus, the complete cycle evaluation routine is most efficient in $\mathcal{Y}$, as each of our subroutines is most efficient in $\mathcal{Y}$.

Solution evaluation

The fourth phase of SPIKE determines the final output, a set of *disjoint* exchange cycles exhibiting the highest probability for a successful transplantation. As a pair of donor and recipient can only be involved in one exchange cycle, the output sets must be vertex disjoint. Thus, the resulting set contains a combination of disjoint exchange cycles that greedily maximizes the number of exchanges with respect to the likelihood of the transplantation being

Table 10 Complexity assessment

Phase	Name	Protocol	Time complexity
Part 1	Compatibility matching	Table 4	$\mathcal{O}(\|\mathsf{HLA}\|)$
		Additional file 1: Table S1	$\mathcal{O}(\|\mathsf{HLA}\|)$
		Additional file 1: Table S2	$\mathcal{O}(1)$
		Additional file 1: Table S4	$\mathcal{O}(1)$
		Additional file 1: Table S5	$\mathcal{O}(1)$
		Additional file 1: Table S6	$\mathcal{O}(1)$
		Table 5	$\mathcal{O}(\|\mathsf{pairs}\|^2 \times \|\mathsf{HLA}\|)$
Part 2	Cycle computation	Additional file 1: Table S7	$\mathcal{O}(\|\mathsf{pairs}\|^2)$
		Table 6	$\mathcal{O}(\mathsf{cLen} \times \|\mathsf{pairs}\|^3)$
Part 3	Cycle evaluation	Additional file 1: Table S10	$\mathcal{O}(1)$
		Table 7	$\mathcal{O}(\|\mathsf{pairs}\|^{\mathsf{cLen}})$
		Additional file 1: Table S8	$\mathcal{O}(\|\mathsf{cyclesSet}\| \times \mathsf{k} \times \mathsf{cLen})$
		Additional file 1: Table S9	$\mathcal{O}(\|\mathsf{cycles}\|^2)$
		Table 8	$\mathcal{O}(\|\mathsf{pairs}\|^{\mathsf{cLen}})$
Part 4	Solution evaluation	Additional file 1: Table S11	$\mathcal{O}(\|\mathsf{cycles}\| \times \mathsf{cLen})$
		Additional file 1: Table S12	$\mathcal{O}(\mathsf{cycles}\|^2)$
		Table 9	$\mathcal{O}(\|\mathsf{cycles}\|^3 \times \mathsf{cLen}^2)$

successful. Note that we find a locally optimal solution, which might differ from the globally optimal solution[12]. The locally optimal solution is computed using a greedy algorithm. This last part of SPIKE is shown in Table 9.

Table 9 takes a secret shared vector of tuples $\mathsf{filteredCycles}$ with all valid unique cycles and their respective weights, the number of valid cycles $|\mathsf{unique}|$, and the cycle length cLen as input. The number of pairs $|\mathsf{pairs}|$ is a public variable as before.

It checks each valid cycle cCycle whether it is disjoint from all other previously analyzed cycles in $\mathsf{tempSet}$. The MPC subprotocol for testing the disjointness is given in Additional file 1: S11 in the Appendix. If it is disjoint, cCycle is added to the set of potential solutions (Lines 16–22). Finally, the set with the highest weight is returned. Details of the corresponding MPC protocol can be found in Additional file 1: Table S12 in the Appendix.

MPC Cost. In total, we evaluate $|\mathsf{unique}|^2$ ADD gates, $|\mathsf{unique}|^2 \times \mathsf{cLen}^2 + |\mathsf{unique}|$ comparisons, $4 \times |\mathsf{unique}|^2 + |\mathsf{unique}|$ MUX, and $|\mathsf{unique}|^2 \times \mathsf{cLen}^2$ OR gates. The solution evaluation is most efficient in $\mathcal{Y}$, as there are only few arithmetic operations and mostly comparisons.

Complexity assessment

In Table 10, the asymptotic complexities for the four phases of SPIKE are given.

The most important parameters of the first part, the Compatibility Matching shown in the first section of the table, are the number of HLA (cf. Background) $|\mathsf{HLA}|$ and the number of pairs $|\mathsf{pairs}|$. In the default configuration, $|\mathsf{HLA}|$ is 50. For the second phase, the dominant parameter is the number of pairs $|\mathsf{pairs}|$. In the third section of Table 10, the asymptotic complexity for the Cycle Evaluation is given. The relevant parameters here are the number of pairs $|\mathsf{pairs}|$, the total number of cycles $|\mathsf{allCycles}| = |\mathsf{pairs}|^{\mathsf{cLen}}$, the number of existing cycles $|\mathsf{cycles}|$, the number of unique cycles $|\mathsf{unique}| = \lfloor \frac{|\mathsf{cycles}|}{\mathsf{cLen}} \rfloor$, the length of cycles cLen, and the factor k (i.e, the number of cycles with highest probability for successful transplantation), and the number of elements in $\mathsf{cyclesSet}$, $|\mathsf{cyclesSet}|$ of Table 8. The most important parameters of the last phase, the Solution Evaluation, are the number of unique cycles $|\mathsf{cycles}|$, and the length of cycles cLen.

Overall, the asymptotic complexity of SPIKE is:

$$\mathcal{O}(|\mathsf{pairs}|^2 \times |\mathsf{HLA}| + \mathsf{cLen} \times |\mathsf{pairs}|^3 + |\mathsf{cycles}|^3 \times \mathsf{cLen}^2).$$

The most most important parameters are the number of pairs $|\mathsf{pairs}|$, the number of considered HLA $|\mathsf{HLA}|$, the length of cycles cLen, and the number of unique cycles $|\mathsf{cycles}|$.

[12] Calculation of a global solution is provably a $\mathcal{NP}$-hard problem [73].

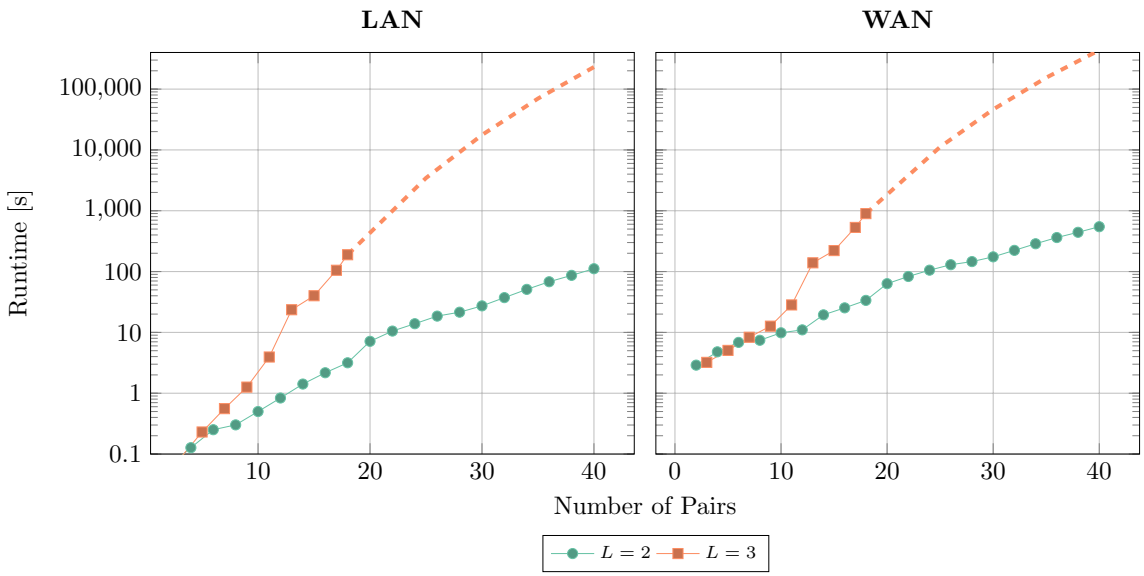


Fig. 3 Overall runtime of SPIKE for cycle lengths $L = 2$ and $L = 3$ in both network scenarios. The dashed line shows the extrapolated power function for $L = 3$

Results

All benchmarks were run on two servers equipped with Intel Core i9-7960X processors and 128 GB RAM. They are connected via 10Gb/s LAN with a median latency of 1.3ms. All benchmarks are averaged over 10 runs.

Network setups

To provide meaningful performance benchmarks for a variety of real-world settings, we envision two network settings for the privacy-preserving KEP protocol that we describe in the following. In addition, for the comparison to the works of Breuer at al. [9, 10], we replicated their network setting with 1Gb/s bandwidth and 1ms of latency.

LAN

The high-bandwidth, low latency network scenario, here referred to as *LAN*, is the most relevant real-world scenario for our application. In Germany, most (larger) medical institutions utilize high-bandwidth Internet connections. In the case of most university hospitals the German Research Network ("Deutsches Forschungsnetz" DFN[13]) provides dedicated, high bandwidth communication networks. Our *LAN* benchmarks are performed using a 10Gb/s connection with an average latency of 1.3ms.

WAN

One benefit of a MPC-based privacy-preserving KEP solution could be reduced legal and regulatory data protection requirements, due to the high security level of the computation itself. This would allow smaller, local hospitals and medical practices to directly participate in the kidney exchange. Those institutions might be connected via residential Internet access. For that scenario, we benchmarked SPIKE in a reduced-bandwidth, high latency network. A bandwidth restriction to 100Mb/s with added latency of 100ms was implemented using the `tc`[14] command to simulate the *WAN* network. The high latency was chosen to take packet loss due to unreliable connections into account.

Performance benchmarks

Figure 3 shows the total runtime of SPIKE for varying numbers of pairs, both network settings, and cycle lengths $L = 2$ and $L = 3$. The full results are in the Additional file 1: Tables S13–S20 in the Appendix.

During the evaluation of longer cycles ($L \geq 3$) RAM utilization proved itself to be a bottleneck for execution. For those scenarios, we benchmarked up to RAM exhaustion and extrapolated the runtimes according to the underlying power-law complexity. The extrapolation is shown with a dashed line. The sudden increase in

[13] https://dfn.de/

[14] https://man7.org/linux/man-pages/man8/tc.8.html

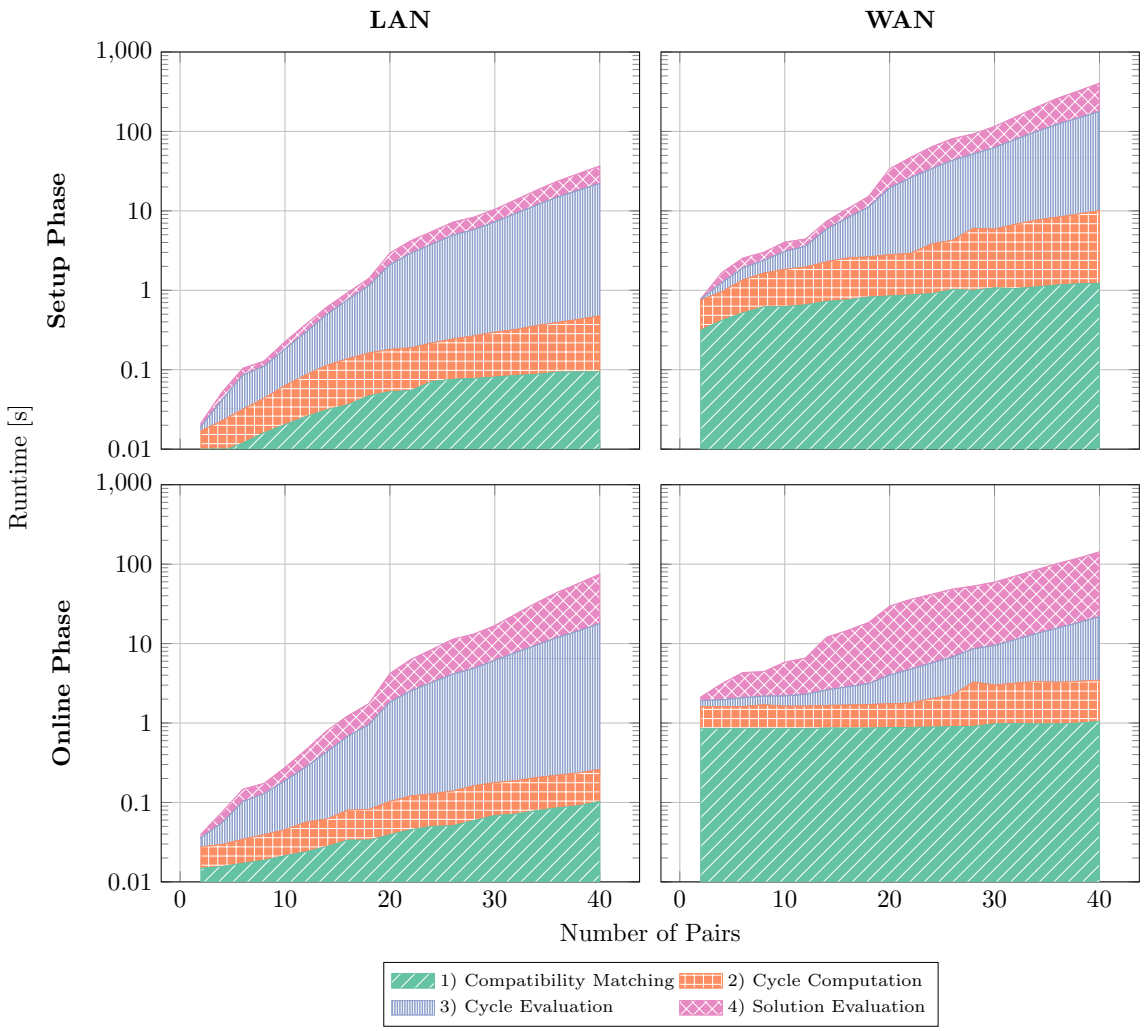


Fig. 4 Runtime of SPIKE for $L = 2$ separated by algorithmic parts, protocol phase, and network setting

runtime for $L = 3$ between 12 and 13 pairs occurs due to swapping.

As a general result, the expected polynomial relationship between the number of pairs and the overall runtime can be observed, reflected in the power-law development in the semilog graphs. For $L = 2$, we achieve a total runtime of under 4min for 40 pairs, thus, demonstrating real-world applicable performance. The WAN setting increases the overall runtime by less than an order of magnitude. Calculation times under 20min for 40 pairs in this setting render the participation feasible for physicians with residential Internet connections. To find a solution for larger cycle lengths, the exponent in the time complexity increases, increasing the runtimes significantly. But even then 25 pairs are computable in around 1 h. Extrapolated to data set sizes of 100 pairs, SPIKE is able to finish the calculation for cycle length $L = 2$ in just over 2 h[15].

Figure 4 shows the runtimes of the individual parts of the algorithm ($L = 2$). It is clearly visible, that the medical compatibility testing and graph creation, as well as the cycle computation quickly become negligible compared to the runtimes of cycle evaluation and the evaluation of the global solution. The duration of online and offline phases are in the same order of magnitude. By executing the phases separately, a 134% performance increase in the online execution can be achieved, compared to the accumulated runtime (cf. Fig. 3).

[15] Based on a power function $f(x) = a \cdot x^b + c$ fit with the parameters $a = 0.003563$, $b = 4.673$ and $c = 1005$ giving the runtime in milliseconds.

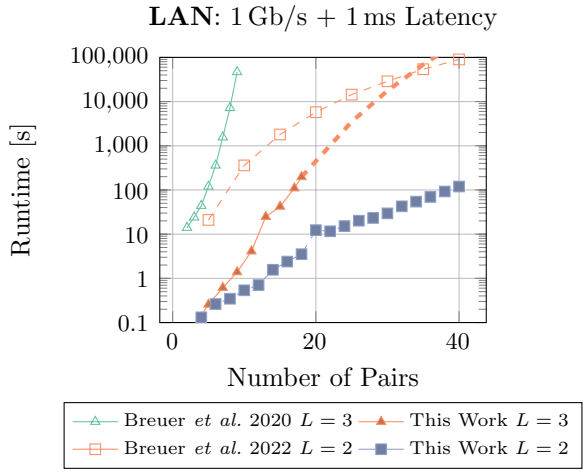


Fig. 5 Runtime comparison between this work with cycle lengths $L = 2$ and $L = 3$, and both Breuer *et al.* 2020 ($L = 3$) [9] and Breuer *et al.* 2022 ($L = 2$) [10]. All measurements use a LAN network setting with 1Gb/s bandwidth and 1ms latency. The dashed line shows the extrapolated power function for our algorithm at $L = 3$

Comparison to state-of-the-art

In Fig. 5, we compare the runtime of our implementation for $L = 2$ and $L = 3$ with two implementations from Breuer et al. [9, 10]. The first implementation [9] uses a Threshold Homomorphic Encryption scheme and enables to solve the privacy-preserving KEP with arbitrary cycle length, as in SPIKE. The maximum cycle size is set to $L = 3$ in their benchmarks. The second one [10] is based on three-party honest majority Shamir's Secret Sharing using the MP-SPDZ framework and limits its cycle length to $L = 2$. The performance data for both implementations is taken from the referenced publications.

Our implementation, as well as the MP-SPDZ based state-of-the-art [10], shows a polynomial-bound power-law graph. The Homomorphic Encryption-based implementation shows clearly an exponential runtime development, increasing rapidly. For 9 pairs, the maximum number of pairs benchmarked in the original publication [9], our implementation achieves a 29828× speedup. For $L = 2$, our implementation performs 414× better than the MP-SPDZ-based implementation [10].

To improve the medical quality of the donor-recipient matching, we implemented additional matching criteria, as described in the "Background" section. As we have seen in Fig. 4, the performance impact of the compatibility matching algorithm is negligible compared to the runtime of the remaining algorithmic parts. However, in Fig. 6 we compare the performance difference between the reduced set of medical matching criteria and the full set. For small number of pairs there is a transient phase, where the runtime of the full set rises faster. After this transient phase, both curves assume nearly the same slope. In the plots for the WAN network model, the latency-induced "baseline" runtime can be observed.

A comparison of communication size between SPIKE and [9, 10] for cycle lengths $L = 3$ and $L = 2$ is included in the appendix in Additional file 1: Tables S16 and S17. For $L = 2$ and 40 pairs, SPIKE requires 40× less communication than [10]. For $L = 3$ and 9 pairs (the maximum number of pairs evaluated by [9]), SPIKE require 104.1× less communication than [9].

Discussion

Security guarantees

Our privacy-preserving kidney exchange protocol, SPIKE, is implemented using the ABY [61] MPC framework, guaranteeing computational semi-honest security in a two-party setting. An adversary $\mathcal{A}$ can corrupt at most one of the two computing parties. $\mathcal{A}$ is assumed to follow the protocol specification and gets access to all messages of the corrupted party (sent and received), while trying to extract private information. This security model is standard in the privacy research community and protects against two security concerns: (1) inadvertent disclosure of sensitive data and (2) *full* data disclosure in case of a breach in one of the parties (in comparison to a centralized computation). The latter concern is a detriment of all centralized or trusted third party based approaches. This coupled with complex legal barriers are the driving forces behind the German Medical Informatics Initiative's[16,17] decision to promote decentralized data holding and processing. In the outsourcing scenario with two computation parties and an arbitrary number of data sources, both computation parties *must not* collude. However, an arbitrary number of data sources is allowed to collude or behave maliciously, without breaking the security guarantees. Note, that MPC only gives privacy guarantees for the computation, whereas maliciously formed inputs might lead to incorrect outputs. For a "holistic" data privacy perspective, please see [74].

While this adversarial model is not sufficient for all applications, e.g., computations with parties in different jurisdictions [62], it suits our setting, namely the joint computation among large, intra-national or intra-European medical institutions. Both semi-honest behaviour, as well as the non-collusion assumption, can be enforced by legal and regulatory means and build the predominant

[16] https://www.medizininformatik-initiative.de/

[17] The German Medical Informatics Initiative is the federal research initiative to enable medical data sharing and secondary use between all university medical centers in Germany.

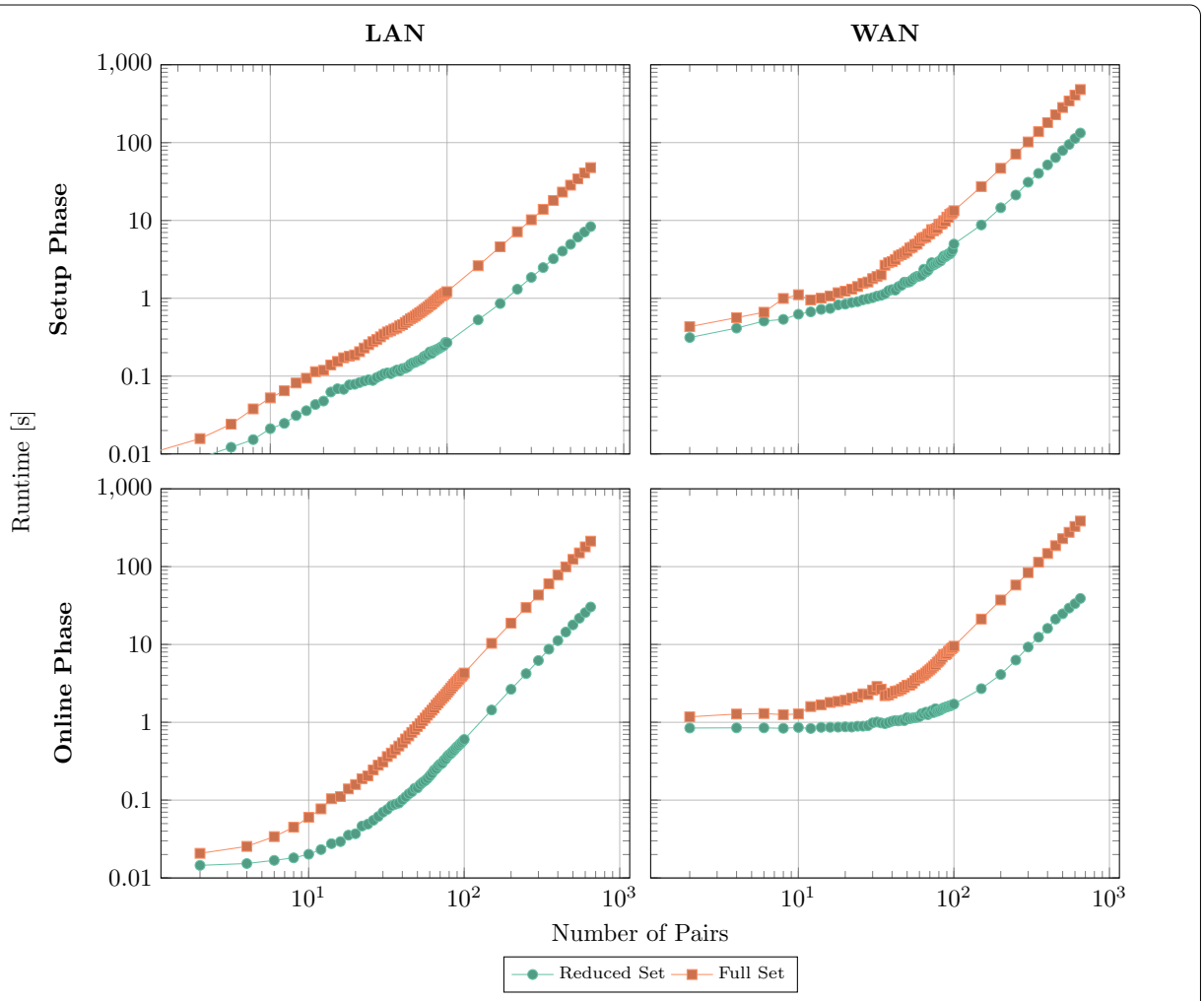


Fig. 6 Comparison of the compatibility matching (i.e., compatibility graph generation) performance between the reduced set of criteria (Breuer *et al.* 2020) [9] and the full set of this work for cycle length $L = 2$. The runtimes of the remaining algorithmic parts are independent of this choice

basis for data protection concepts in (German) federated medical research networks.

Furthermore, several real-world industry projects demonstrate that the non-collusion assumption of MPC is practical. For example, Mozilla Firefox starts to deploy MPC-based privacy-preserving collection for Telemetry data [75, 76] and Bosch is developing a MPC platform for smart homes and anonymous driving [77, 78]. Many more examples can be found in the MPC alliance[18] which is a consortium of industry peers working on MPC. Even in the German medical informatics realm employed MPC solutions are able to work in a (non-colluding) two-party setting [79].

For a full description of the cryptographic assumptions and guarantees inherited by the primitives used in ABY, we refer to the respective section in the Appendix and the original ABY publication [61].

While all data, including the association to the various data sources are considered to be private data and are protected by the aforementioned guarantees, we consider the *number* of donor-recipient pairs, as well as the maximum number of cycles in the graph, as public information. This choice has important performance impacts, however, if the numbers of pairs are to be considered private as well, the real numbers can be hidden by padding each input array to a fixed length with dummy entries.

Real-world deployability

This work introduces a protocol for finding a solution for the kidney exchange problem in a privacy-preserving fashion. As demonstrated in the performance benchmarks and the security discussion, it meets all initially determined requirements for a secure privacy-preserving

[18] https://www.mpcalliance.org/

solution to the KEP w.r.t privacy, efficiency, decentralization, and adaptability for medical experts.

Concretely, it enables real-world periodic batch-processing for a significantly larger number of donor-recipient pairs and a practical cycle length of $L = 2$ and $L = 3$ compared to previous work [9, 10], even in residential network settings. This allows even residential nephrology experts to participate in kidney exchanges, hence, providing a better medical care for their patients. However, SPIKE requires a significant amount of communication, thus, it is not yet ready for the usage of metered or cell data connections in a two-party computation protocol, which might be an interesting direction for future work. In contrast, SPIKE is already practical for an outsourcing scenario, where mobile clients secret share their data of 100 donor-recipient pairs among two non-colluding servers or cloud entities. But we also point out that larger sets of, e.g., 300 pairs are not practical yet due to the, although improved, but still limited scalability of our protocol. To run SPIKE on even larger datasets, two strategies are possible: (1) reducing the interval of calculation, hence, effectively reducing the participating pairs, or (2) partitioning on less sensitive features, such as "blood type" and running the computation on smaller data chunks in parallel. The first approach, however, results in a smaller set of pairs considered in the matching while the second change likely increases the number of mismatches that will not pass the final check by medical experts.

By using state-of-the-art provably secure cryptographic techniques, the privacy of sensitive medical information of donors and recipients is fully protected by clearly defined hardness assumptions of mathematical problems. Furthermore, by pursuing a completely decentralized approach without a trusted third party, the risk of data leakages in case of a data security incident at one participating facility is significantly reduced, compared to a breach in a central computation node or repository. This is especially important for quasi-identifying medical fields[19]. Often times, quasi-identifiers are not anonymizable, as they loose too much utility in the process. Hence, secure decentral storage and processing of non-anonymized data is increasingly important especially considering current efforts of simplified international data usage, such as the proposed European Health Data Space (EHDS) [80].

Allowing medical professionals to choose many parameters of the algorithm to adapt to new evidence-based guidelines or specific situational constraints ensures flexibility and maintainability for future application. The compatibility matching algorithm is configurable by choosing the considered HLA, as well as the weights of the chosen medical factors. This explicitly allows the deactivation of chosen comparisons. Due to the clear architecture boundaries in the open source implementation, additional checks and criteria can easily be included. Many hierarchically ordered optimization goals employed in current KEP solutions [7] can be included in SPIKE via a more involved weight calculation. One open research question is to quantify a possible transplantation success rate difference between globally optimal KEP solvers and our locally optimal solution. We argue, that the medical uncertainty, that can not easily be evaluated via algorithms, e.g., number and positions of renal arteries [81], might be larger than the uncertainty introduced by our local solution. To answer this highly relevant question a cross-examination of followed-up real world kidney-exchanges would be required and is left as future work.

While meeting all formal requirements, SPIKE falls short in two aspects: First, we observe a high memory consumption during the computation. This is expected, as this protocol was optimized for runtime performance. The reason is that hardware costs are typically not a prohibiting factor for meeting data protection regulations. Note that we use only standard hardware for our benchmarks. For a real-world deployment, it is realistic to assume a deployment on servers with significantly higher capacity. Thus, we argue that this aspect does not jeopardize the adoption in the intended use cases. However, improvements in this regard are still desirable. For example, developing internal batch processing of graph clusters and the employment of space-optimized data structures might be worthwhile opportunities for further research. An interesting direction for future work can be to explore the compatibility with recent advances in MPC-based graph analysis for breadth-first search [82] scaling linearly in the number of vertices. Second, the developed software components are research artifacts and fulfil a prototypical function. For real-world adoption the implementation of widespread medical standards, e.g., HL7 FHIR R4[20], audit- and authentication capabilities, integration in medical research pipelines, creation of deployment packages, and lastly full (legal) documentation must be pursued. This is, however, not in the scope of this work.

[19] Quasi-Identifier are groups of fields, that do not include the traditional fields of identifying data, such as names and birth dates. Nevertheless, the combination of fields in a quasi-identifier is rare enough to identify individual patients.

[20] https://www.hl7.org/fhir/R4/

Conclusion

In this work, we introduced SPIKE, the currently most efficient privacy-preserving Kidney Exchange Problem (KEP) protocol. Using provably secure cryptographic techniques, SPIKE provides highest data protection guarantees for patients' sensitive medical data without relying on a trusted third party, while allowing a decentralized computation of a locally optimal solution to the kidney exchange problem. In the absence of privacy-preserving Integer Linear Programming (ILP) solving algorithms, we implement approximate, adaptable medical compatibility matching algorithms, giving medical professionals the flexibility to accommodate updated guidelines and the specific situational constraints.

Our optimized protocols achieve a 30000× and 400× speedup compared to the current state-of-the-art [9, 10] for cycle lengths of $L = 3$ and $L = 2$, respectively. With a total runtime of under 4min for 40 pairs at $L = 2$ and around 1 h for 25 pairs at $L = 3$, we demonstrate sufficient performance for deploying it for some real-world applications.

However, we note that kidney exchange programs typically consider up to 300 pairs per run [83] which is not yet feasible for SPIKE since our protocol does not scale sufficiently well in the number of participating donor-recipient pairs leading to unsatisfactory runtimes beyond 170 pairs (for $L = 2$). Additionally, memory usage is another aspect that needs more future work. To summarize, SPIKE is not yet a routine solution ready for deployment for large scale kidney exchange programs, however, it offers the most efficient state-of-the-art solution to the problem. In this sense, it makes an important contribution towards moving into the direction of practical large-scale privacy-preserving solutions.

We also hope that the advancements in privacy protection and application performance will already allow more medical facilities to participate in kidney exchanges on a smaller scale, thus increasing the recipients' chances for timely and potentially live-saving surgery.

Abbreviations

MPC: Secure multi-party computation; HE: Homomorphic encryption; SSS: Shamir's secret sharing; $\mathcal{Y}$: Yao's garbled circuits; $\mathcal{B}$: Boolean secret sharing; $\mathcal{A}$: Arithmetic secret sharing; PPKE: Privacy-preserving kidney exchange; KEP: Kidney exchange problem; HLA: Human leukocyte antigens.

Supplementary Information

The online version contains supplementary material available at https://doi.org/10.1186/s12911-022-01994-4.

Additional file 1. Appendix: Supplementary tables.

Acknowledgements

Many thanks to Ulrich Zwirner for sharing his medical knowledge during our fruitful discussions. Furthermore, we thank the anonymous reviewers of BMC Medical Informatics and Decision Making for their constructive comments that helped us to further improve our writing.

Author Contributions

TB implemented the discussed software. TB and TK specified the medical requirements. HM and TK designed the research project and guided the protocol design. TB, TK, and HM designed the performance benchmarks which were conducted and analyzed by TB and TK. TS and KH led this research project. All authors contributed to the manuscript and substantively revised it. All authors read and approved the final manuscript.

Funding

Open Access funding enabled and organized by Projekt DEAL. This project received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement No.850990 PSOTI). It was co-funded by the Deutsche Forschungsgemeinschaft(DFG) – SFB1119 CROSSING/236615297 and GRK2050 Privacy & Trust/251805230, by the German Federal Ministry of Education and Research and the Hessen State Ministry for Higher Education, Research and the Arts within ATHENE, and by the German Federal Ministry of Education and Research within the HiGHmed project (#01ZZ1802G).

Availability of data and materials

Our code and the datasets analyzed are available here: https://encrypto.de/code/PPKE.

Declarations

Ethics approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

Competing interests

The authors declare that they have no competing interests.

Author details

[1]ENCRYPTO, Technical University of Darmstadt, Darmstadt, Germany. [2]Computational Biology and Simulation group, Technical University of Darmstadt, Darmstadt, Germany.

Received: 29 April 2022 Accepted: 13 September 2022
Published online: 22 September 2022

References

1. Murphy D, McCulloch CE, Lin F, Banerjee T, Bragg-Gresham JL, Eberhardt MS, Morgenstern H, Pavkov ME, Saran R, Powe NR, Hsu C-Y. For the centers for disease control and prevention chronic kidney disease surveillance team: trends in prevalence of chronic kidney disease in the United States. Ann Intern Med. 2016;165(7).
2. Thurlow JS, Joshi M, Yan G, Norris KC, Agodoa LY, Yuan CM, Nee R. Global epidemiology of end-stage kidney disease and disparities in kidney replacement therapy. Am J Nephrol. 2021;52(2):98.
3. Ibrahim HN, Foley R, Tan L, Rogers T, Bailey RF, Guo H, Gross CR, Matas AJ. Long-term consequences of kidney donation. N Engl J Med. 2009;360:459.
4. Eurotransplant: Annual Report 2020. https://www.eurotransplant.org/wp-content/uploads/2021/08/ETP_AR2020_opm_LR.pdf Accessed 2022-04-03
5. Nemati E, Einollahi B, Pezeshki ML, Porfarziani V, Fattahi MR. Does kidney transplantation with deceased or living donor affect graft survival? Nephro Urol Mon. 2014;6(4).

6. Ellison B. A systematic review of kidney paired donation: applying lessons from historic and contemporary case studies to improve the US Model. Wharton Research Scholars. 2014;107.
7. Biró P, van de Klundert J, Manlove D, Pettersson W, Andersson T, Bunapp L, Chromy P, Delgado P, Dworczak P, Haase B, Hemke A, Johnson R, Klimentova X, Kuypers D, Costa AN, Smeulders B, Spieksma F, Valentín MO, Viana A. Modelling and optimisation in European kidney exchange programmes. Eur J Oper Res. 2021;291(2):447–56.
8. Biró P, Burnapp L, Haase B, Hemke A, Johnson R, van de Klundert J, Manlove D. First handbook of the cost action ca15210: European network for collaboration on kidney exchange programmes (enckep). Brussels: European Cooperation in Science and Technology; 2017.
9. Breuer M, Meyer U, Wetzel S, Mühlfeld A. A privacy-preserving protocol for the kidney exchange problem. WPES. 2020.
10. Breuer M, Meyer U, Wetzel S. Privacy-preserving maximum matching on general graphs and its application to enable privacy-preserving kidney exchange. In: ACM Conference on Data and Application Security and Privacy (CODASPY) 2022.
11. Pansart L, Cambazard H, Catusse N, Stauffer G. kidney exchange problem: models and algorithms. In: HAL Archives-ouvertes. 2014.
12. Carvalho M, Klimentova X, Glorie K, Viana A, Constantino M. Robust models for the kidney exchange problem. Inf J Comput. 2020.
13. Ashby VB, Leichtman AB, Rees MA, Song PX-K, Bray M, Wang W, Kalbfleisch JD. A kidney graft survival calculator that accounts for mismatches in age, sex, hla, and body size. Clin J Am Soc Nephrol. 2017;12:1148–1160.
14. Abraham DJ, Blum A, Sandholm T. Clearing algorithms for barter exchange markets: enabling nationwide kidney exchange. In: ACM Conference on Electronic Commerce 2007. ACM
15. Dreier J, Kerschbaum F. practical privacy-preserving multiparty linear programming based on problem transformation. In: 2011 IEEE Third International Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third International Conference on Social Computing, 2011:916–924.
16. Catrina O, de Hoogh S. Secure multiparty linear programming using fixed-point arithmetic. In: Computer Security - ESORICS 2010, vol. 6345. Springer, Berlin, Heidelberg 2010.
17. Toft T. Solving linear programs using multiparty computation. in: financial cryptography and data security. Springer, Berlin, Heidelberg 2009.
18. Cornuéjols G. Valid inequalities for mixed integer linear programs. Math Programm. 2008;112(1):3–44.
19. Atamtürk A, Nemhauser GL, Savelsbergh MW. Conflict graphs in solving integer programming problems. Eur J Oper Res. 2000;121(1):40–55.
20. Savelsbergh MW. Preprocessing and probing techniques for mixed integer programming problems. ORSA J Comput. 1994;6(4):445–54.
21. Fouque P-A, Poupard G, Stern J. Sharing decryption in the context of voting or lotteries. In: International Conference on Financial Cryptography, 2000;90–104. Springer
22. Shamir A. How to share a secret. communications of the ACM. 1979;22(11).
23. Keller M. MP-SPDZ: A versatile framework for multi-party computation. In: CCS '20: 2020 ACM SIGSAC Conference on Computer and Communications Security 2020.
24. Pape U, Conradt D. Maximales matching in graphen. In: Ausgewählte Operations Research Software in FORTRAN 1980.
25. Sung YC. The HLA system: genetics, immunology, clinical testing, and clinical implications. Yonsei Med J. 2007;48:11.
26. Eurotransplant: Histocompatibility. In: Eurotransplant Manual Ver. 4.5, 2018. Chap. 10
27. Lefaucheur C, Loupy A, Hill GS, Andrade J, Nochy D, Antoine C, Gautreau C, Charron D, Glotz D, Suberbielle-Boissel C. Preexisting donor-specific HLA antibodies predict outcome in kidney transplantation. J Am Soc Nephrol. 2010;21:1398.
28. Ntokou ISA, Iniotaki AG, Kontou EN, Darema MN, Apostolaki MD, Kostakis AG, Boletis JN. Long-term follow up for anti-hla donor specific antibodies postrenal transplantation: high immunogenicity of HLA class II graft molecules. Transp Int. 2011;24:1084.
29. Santos C, Costa R, Malheiro J, Pedroso S, Almeida M, Martins LS, Dias L, Tafulo S, Henriques AC, Antonio C. Kidney transplantation across a positive crossmatch: a single-center experience. In: Transplantation Proceedings. 2014.
30. Eurotransplant: Kidney. In: Eurotransplant Manual Ver. 4.5, (2018). Chap. 4
31. Leeaphorn N, Pena JRA, Thamcharoen N, Khankin EV, Pavlakis M, Cardarelli F. HLA-DQ mismatching and kidney transplant outcomes. J Am Soc Nephrol. 2018;13:763.
32. Opelz G, Döhler B. Association of HLA mismatch with death with a functioning graft after kidney transplantation: a collaborative transplant study report. Am J Transp. 2012;12:3031.
33. Opelz G. Impact of HLA compatibility on survival of kidney transplants from unrelated live donors. Transplantation. 1997.
34. Lim WH, Chadban SJ, Clayton P, Budgeon CA, Murray K, Campbell SB, Cohney S, Russ GR, McDonald SP. human leukocyte antigen mismatches associated with increased risk of rejection, graft failure, and death independent of initial immunosuppression in renal transplant recipients. Clin Transp. 2012;26:E428.
35. Nguyen MC. Evaluation of hla typing data and transplant outcome in pediatric renal transplantation. PhD thesis, Medizinische Universität Wien 2021. https://repositorium.meduniwien.ac.at/obvumwhs/content/titleinfo/5894916/full.pdf
36. Blutspenden: Rund Ums Blut. https://www.blutspenden.de/rund-ums-blut/blutgruppen/ Accessed 2022-04-03
37. de Weerd AE, Betjes MGH. ABO-incompatible kidney transplant outcomes: a meta-analysis. Clin J Am Soc Nephrol. 2018;13:1234.
38. Waiser J, Schreiber M, Budde K, Fritsch L, Böhler T, Hause I, Neumayer H-H. Age-matching in renal transplantation. Nephrol Dial Transp. 2000;15:696.
39. Zhoua J-Y, Chenga J, Huanga H-F, Shen Y, Jiang Y, Chen J-H. The effect of donor-recipient sex mismatch on short- and long-term graft survival in kidney transplantation: a systematic review and meta-analysis. Clin Transp. 2013;27:964.
40. Miller AJ, Kiberd BA, Alwayn IP, Odutayo A, Tennankore KK. Donor-recipient weight and sex mismatch and the risk of graft loss in renal transplantation. Clin J Am Soc Nephrol. 2017;12:669.
41. El-Agroudy AE, Hassan NA, Bakir MA, Foda MA, Shokeir AA. Effect of donor/recipient body weight mismatch on recipient and graft outcome in living-donor kidney transplantation. Am J Nephrol. 2003;23:294.
42. Cho H, Wu DJ, Berger B. Secure genome-wide association analysis using multiparty computation. Nat Biotechnol. 2018;36(6):547–51.
43. Bonte C, Makri E, Ardeshirdavani A, Simm J, Moreau Y, Vercauteren F. Towards practical privacy-preserving genome-wide association study. BMC Bioinf. 2018;19(1):1–12.
44. Tkachenko O, Weinert C, Schneider T, Hamacher K. Large-scale privacy-preserving statistical computations for distributed genome-wide association studies. In: 13. ACM ASIA Conference on Computer and Communications Security (ASIACCS'18), pp. 221–235. ACM, Songdo, South Korea 2018. https://encrypto.de/papers/TWSH18.pdf
45. Schneider T, Tkachenko O. EPISODE: Efficient Privacy-preserving similar sequence queries on outsourced genomic databases. In: Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security, 2019;315–327.
46. Günther D, Holz M, Judkewitz B, Möllering H, Pinkas B, Schneider T. PEM: Privacy-preserving epidemiological modeling. Cryptology ePrint Archive 2020.
47. Barni M, Failla P, Kolesnikov V, Lazzeretti R, Sadeghi A-R, Schneider T. Secure evaluation of private linear branching programs with medical applications. In: 14. European Symposium on Research in Computer Security (ESORICS'09), pp. 424–439 2009.
48. Barni M, Failla P, Lazzeretti R, Sadeghi A-R, Schneider T. Privacy-preserving ECG classification with branching programs and neural networks. IEEE Transactions on Information Forensics and Security (TIFS), 452–468, 2011.
49. Yao AC-C. How to generate and exchange secrets. In: 27th Annual Symposium on Foundations of Computer Science (SFCS 1986) 1986.
50. Malkhi D, Nisan N, Pinkas B, Sella Y. Fairplay - A secure two-party computation system. In: 13. USENIX Security Symposium (USENIX Security'04) 2004.
51. Kolesnikov V, Schneider T. Improved garbled circuit: free XOR gates and applications. In: Automata, Languages and Programming. Lecture Notes in Computer Science 2008.
52. Zahur S, Rosulek M, Evans D. Two halves make a whole. In: Advances in Cryptology - EUROCRYPT 2015. Lecture Notes in Computer Science 2015.
53. Patra A, Schneider T, Suresh A, Yalame H. ABY2.00: improved mixed-protocol secure two-party computation. In: 30th USENIX Security Symposium (USENIX Security 21), 2021;2165–2182 .

54. Järvinen K, Leppäkoski H, Lohan E-S, Richter P, Schneider T, Tkachenko O, Yang Z. PILOT: practical privacy-preserving indoor localization using outsourcing. In: IEEE European Symposium on Security and Privacy (EuroS &P) 2019.
55. Goldreich O, Micali S, Wigderson A. How to play any mental game. In: Proceedings of the 19th Annual ACM Symposium on Theory of Computing. STOC '87 1987.
56. Kamara S, Raykova M. Secure outsourced computation in a multi-tenant cloud. In: IBM Workshop on Cryptography and Security in Clouds 2011.
57. Damgård I, Pastro V, Smart N, Zakarias S. Multiparty Computation from somewhat homomorphic encryption. In: Advances in Cryptology - CRYPTO 2012 2012.
58. Chaudhari H, Choudhury A, Patra A, Suresh A. Astra: High throughput 3pc over rings with application to secure prediction. In: ACM SIGSAC Conference on Cloud Computing Security Workshop (CCSW) 2019.
59. Chaudhari H, Rachuri R, Suresh A. Trident: Efficient 4pc framework for privacy preserving machine learning. 2021.
60. Braun L, Demmler D, Schneider T, Tkachenko O. Motion–a framework for mixed-protocol multi-party computation. ACM Transactions on Privacy and Security 2022.
61. Demmler D, Schneider T, Zohner M. ABY - A framework for efficient mixed-protocol secure two-party computation. In: Network and Distributed System Security Symposium(NDSS) 2015.
62. European Data Protection Board: Recommendations 01/2020 on measures that supplement transfer tools to ensure compliance with the EU level of protection of personal data 2021.
63. MI-I Taskforce Datenschutz: Übergreifendes Datenschutzkonzept der Medizininformatik-Initiative. https://www.medizininformatik-initiative.de/sites/default/files/2022-03/MII-Datenschutzkonzept_v1.0.pdf Accessed 03.07.2022
64. Lablans M, Schmidt E. Datenschutzkonzept der DKTK Clinical Communication Platform. https://dktk.dkfz.de/application/files/5016/2030/2474/20_11_23_Datenschutzkonzept_CCP-IT_inkl_Anlagen.pdf Accessed 03.07.2022
65. Wiesner S. Conjugate coding. ACM SIGACT News 1983;15(1).
66. Rabin MO. How to exchange secrets with oblivious transfer 1981.
67. Asharov G, Lindell Y, Schneider T, Zohner M. More efficient oblivious transfer extensions. J Cryptol. 2017;30:3.
68. Ben-Or M, Goldwasser S, Wigderson A. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In: Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing 1988.
69. Beaver D. Efficient multiparty protocols using circuit randomization. In: Annual International Cryptology Conference 1991
70. Rathee D, Schneider T, Shukla K. Improved multiplication triple generation over rings via RLWE-based AHE. In: International Conference on Cryptology and Network Security 2019.
71. Keller H, Möllering H, Schneider T, Yalame H. Balancing quality and efficiency in private clustering with affinity propagation. In: International Conference on Security and Cryptography (SECRYPT) 2021.
72. Braun L, Cammarota R, Schneider T. POSTER: A generic hybrid 2PC framework with application to private inference of unmodified neural networks (Extended Abstract). Privacy in Machine Learning Workshop (PriML@ NeurIPS'21) 2021
73. Biró P, Cechlárová K. Inapproximability of the kidney exchange problem. Inf Process Lett. 2007;101(5):199–202.
74. Desai T, Ritchie F, Welpton R. Five safes: designing data access for research. Technical report, University of the West of England 2016
75. Aas J, Geoghegan T. Introducing ISRG prio services for privacy respecting metrics. https://www.abetterinternet.org/post/introducing-prio-services/ Accessed 07.07.2022
76. Englehardt S. Next steps in privacy-preserving telemetry with prio. https://blog.mozilla.org/security/2019/06/06/next-steps-in-privacy-preserving-telemetry-with-prio/ Accessed 07.07.2022
77. Becker S, Trieflinger S. Bosch research launches carbyne stack open-source project for cloud-native secure multiparty computationd. https://www.bosch.com/stories/open-source-carbyne-stack/ Accessed 07.07.2022
78. Trieflinger S. Trustworthy computing - data sovereignty while connected. https://www.bosch.com/research/know-how/success-stories/trustworthy-computing-data-sovereignty-while-connected/ Accessed 07.07.2022
79. Kussel T, Brenner T, Tremper G, Schepers J, Lablans M, Hamacher K. Record linkage based patient intersection cardinality for rare disease studies using mainzelliste and secure multi-party computation. In Review 2022. https://www.researchsquare.com/article/rs-1486673/v1 Accessed 2022-07-08
80. Directorate-General for Health and Food Safety: Proposal for a regulation - the european health data space. https://health.ec.europa.eu/publications/proposal-regulation-european-health-data-space_en Accessed 08.07.2022
81. Bachul PJ, Osuch C, Chang E-S, Bętkowska-Prokop A, Pasternak A, Szura M, Matyja A, Walocha JA. Crossing anatomic barriers-transplantation of a kidney with 5 arteries, duplication of the pyelocalyceal system, and double ureter. Cell Transp. 2017;26(10):1669–72.
82. Araki T, Furukawa J, Ohara K, Pinkas B, Rosemarin H, Tsuchida H. Secure graph analysis at scale. In: Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security, pp. 610–629 (2021)
83. Biró P, Haase B, Andersson T, Ásgeirsson EI, Baltesová T, Boletis I, Bolotinha C, Bond G, Böhmig G, Burnapp L, Cechlárová K, Ciaccio PD, Fronek J, Hadaya K, Hemke A, Jacquelinet C, Johnson R, Kieszek R, Kuypers DR, Leishman R, Macher MA, Manlove D, Menoudakou G, Salonen M, Smeulders B, Sparacino V, Spieksma F, Valentín MO, Wilson N, van der Klundert J, Action EC. Building kidney exchange programmes. In: Europe - An Overview of Exchange Practice and Activities. Transplantation. 2019.

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Ready to submit your research? Choose BMC and benefit from:

- fast, convenient online submission
- thorough peer review by experienced researchers in your field
- rapid publication on acceptance
- support for research data, including large and complex data types
- gold Open Access which fosters wider collaboration and increased citations
- maximum visibility for your research: over 100M website views per year

At BMC, research is always in progress.

Learn more biomedcentral.com/submissions

Appendix

Additional MPC Subprotocols

This part of the Appendix contains additional protocols of our MPC-based PPKE solution, SPIKE, that are similar to the ones presented in the main part and, thus, referred to the appendix.

Additional Protocols for the Compatibility Matching

This subsection presents additional subprotocols for the compatibility matching phase.

Supplementary Table 1 $\texttt{evalHLA}(\langle \text{hla}_\text{d} \rangle^{\mathcal{B}}\text{: vector}, \langle \text{hla}_\text{r} \rangle^{\mathcal{B}}\text{: vector}) \rightarrow \text{int}$

1: $\langle \text{mm} \rangle^{\mathcal{B}} \leftarrow \{\langle 0 \rangle^{\mathcal{B}}\}^{|\text{HLA}|}$
2: **for** $i = 0, \ldots, |\text{HLA}| - 1$ **do**
3: $\quad \langle \text{mm} \rangle^{\mathcal{B}} \leftarrow \langle \text{hla}_\text{d} \rangle^{\mathcal{B}}[i] \oplus \langle \text{hla}_\text{r} \rangle^{\mathcal{B}}$ ▷ SIMD
4: **end for**
5: $\langle \text{sum} \rangle^{\mathcal{B}} \leftarrow \texttt{HammingW}(\{\langle 0 \rangle^{\mathcal{B}}\}^{|\text{HLA}|}, \langle \text{mm} \rangle^{\mathcal{B}})$
6: $\langle \text{c} \rangle^{\mathcal{B}} \leftarrow \langle \text{sum} \rangle^{\mathcal{B}} < \langle 5 \rangle^{\mathcal{B}}$
7: $\langle \text{b} \rangle^{\mathcal{B}} \leftarrow \langle \text{sum} \rangle^{\mathcal{B}} < \langle 3 \rangle^{\mathcal{B}}$
8: $\langle \text{a} \rangle^{\mathcal{B}} \leftarrow \langle \text{sum} \rangle^{\mathcal{B}} == \langle 0 \rangle^{\mathcal{B}}$
9: **return** $\langle \text{a} \rangle^{\mathcal{B}}$ **?**
$\langle \text{A} \rangle^{\mathcal{B}} (\langle \text{b} \rangle^{\mathcal{B}}$ **?** $\langle \text{B} \rangle^{\mathcal{B}}$ **:** $(\langle \text{c} \rangle^{\mathcal{B}}$ **?** $\langle \text{C} \rangle^{\mathcal{B}}$ **:** $\langle 0 \rangle^{\mathcal{B}}))$

HLA Antigen Comparison

In Supplementary Table 1, we compare the HLA antigens of the potential recipient and donor and determine the number of HLA mismatches. It takes two vectors hla_d and hla_r with the HLA antigens of the donor and recipient respectively as input. The number of |HLA| is public, as it is a fixed value. The vector mm indicates the HLA mismatches of the donor and the recipient. A mismatch occurs if either donor or recipient has a HLA antigen that the other does not have (cf. Line 3). For enhanced efficiency, we parallelize the comparison as SIMD operation, such that the vector mm is computed in a single step. Afterwards, the number of HLA mismatches is determined with a Hamming Weight Circuit (cf. Line 5). Based on the number of mismatches, the subprotocol outputs an indicator for the quality of the pairing w.r.t. the HLA antigens: Class A is an optimal fit with no mismatches, class B is a good fit, and class C is an acceptable fit with 3-4 mismatches (cf. Lines 6-9).

MPC Cost

Line 3 in Supplementary Table 1 evaluates $|\text{HLA}| \times$ XOR gates (as SIMD). Line 5 evaluates one Hamming Distance circuit. Lines 6-9 contain three comparison and three MUX gates. Thus, the circuit's multiplicative depth is 7, which is determined by the number of AND gates on the longest path. Naively, using Yao's Garbled Circuits ($\mathcal{Y}$) seems to be most efficient. However, considering that Table 4 is done in $\mathcal{B}$ sharing, the conversion cost outweigh the benefits of using $\mathcal{Y}$ instead of $\mathcal{B}$, thus, $\mathcal{B}$ is used here as well.

Supplementary Table 2 $\texttt{evalABO}(\langle \text{bg}_\text{d} \rangle^{\mathcal{B}}\text{: vector}, \langle \text{bg}_\text{r} \rangle^{\mathcal{B}}\text{: vector}) \rightarrow \text{int}$

1: $\langle \text{a} \rangle^{\mathcal{B}} \leftarrow \neg\Big((\langle \text{bg}_\text{r} \rangle^{\mathcal{B}}[0] \oplus \langle \text{bg}_\text{d} \rangle^{\mathcal{B}}[0]) \vee (\langle \text{bg}_\text{r} \rangle^{\mathcal{B}}[1] \oplus \langle \text{bg}_\text{d} \rangle^{\mathcal{B}}[1])\Big)$
2: $\langle \text{b} \rangle^{\mathcal{B}} \leftarrow (\langle \text{bg}_\text{r} \rangle^{\mathcal{B}}[1] \wedge \neg\langle \text{bg}_\text{d} \rangle^{\mathcal{B}}[0]) \vee (\langle \text{bg}_\text{r} \rangle^{\mathcal{B}}[0] \wedge \neg\langle \text{bg}_\text{d} \rangle^{\mathcal{B}}[1])$
3: $\langle \text{v} \rangle^{\mathcal{B}} \leftarrow \langle \text{a} \rangle^{\mathcal{B}} \vee \langle \text{b} \rangle^{\mathcal{B}}$
4: **return** $\langle \text{v} \rangle^{\mathcal{B}}$ **?** $\langle \text{best}_\text{age} \rangle^{\mathcal{B}}$ **:** $\langle 0 \rangle^{\mathcal{B}}$

ABO blood group comparison

Supplementary Table 2 contains the privacy-preserving evaluation of the compatibility of ABO blood groups of a donor and a recipient. It takes two two-bit vectors as input: $bg_d \in \{0,1\}^2$ is the blood group of the donor and $bg_r \in \{0,1\}^2$ is the blood group of the recipient. The blood group encoding is shown in Supplementary Table 3. Lines 1-2 ensure that the blood group of recipient and donor are compatible, i.e., they have to be either equal, $bg_r[1] > bg_d[0]$, or $bg_r[0] > bg_d[1]$ (cf. Table 2).

Supplementary Table 3 Encoding of the different blood groups.

Encoding	Blood Group
00	O
01	A
10	B
11	AB

MPC Cost

Here, we evaluate 14 XOR gates and five AND gates in total per donor/recipient pair. As XOR gates can be locally evaluated, they are "for free". Therefore, the AND gates and circuit depth determine, which MPC protocol is most efficient. $\mathcal{B}$ is slightly more efficient than $\mathcal{Y}$ since the circuit depth is smaller than the number of total AND gates.

Supplementary Table 4 $\texttt{evalAge}(\langle \text{a}_\text{d} \rangle^{\mathcal{B}}\text{: int}, \langle \text{a}_\text{r} \rangle^{\mathcal{B}}\text{: int}) \rightarrow \text{int}$

1: $\langle \text{eq} \rangle^{\mathcal{B}} \leftarrow \langle \text{a}_\text{d} \rangle^{\mathcal{B}} == \langle \text{a}_\text{r} \rangle^{\mathcal{B}}$
2: $\langle \text{yg} \rangle^{\mathcal{B}} \leftarrow \neg\langle \text{a}_\text{d} \rangle^{\mathcal{B}} \wedge \langle \text{a}_\text{r} \rangle^{\mathcal{B}}$
3: **return** $\langle \text{yg} \rangle^{\mathcal{B}}$ **?**
$(\langle \text{eq} \rangle^{\mathcal{B}}$ **?** $\langle \text{A} \rangle^{\mathcal{B}}$ **:** $\langle \text{B} \rangle^{\mathcal{B}})$ **:** $(\langle \text{eq} \rangle^{\mathcal{B}}$ **?** $\langle \text{A} \rangle^{\mathcal{B}}$ **:** $\langle 0 \rangle^{\mathcal{B}})$

Age Comparison

Supplementary Table 4 evaluates the compatibility of a donor and recipient based on their age group. It takes the age group of the donor $\langle a_d \rangle^B$ and the age group of the recipient $\langle a_r \rangle^B$ as input. Line 1 checks if they are in the same age group and Line 2 evaluates whether the donor is in a younger age group than the recipient. Afterwards, we compute the respective weight of this donor and recipient constellation. Similarly, as in Supplementary Table 1, class A indicates an optimal match, class B a good match, and Eq denotes that recipient and donor are in the same age group.

MPC Cost

Supplementary Table 4 contains one comparison, one inversion, one AND gate, and three MUX gates. As Line 1 and Line 2 are independent, similarly as the two MUX gates in Line 3, the circuit depth is 3. Thus, this subprotocol is slightly more efficient in $\mathcal{B}$ than in $\mathcal{Y}$.

Supplementary Table 5 $\texttt{evalSex}(\langle \text{s}_\text{d} \rangle^{\mathcal{B}}\text{: int}, \langle \text{s}_\text{r} \rangle^{\mathcal{B}}\text{: int}) \rightarrow \text{int}$

1: $\langle \text{eq} \rangle^{\mathcal{B}} \leftarrow \langle \text{s}_\text{d} \rangle^{\mathcal{B}} == \langle \text{s}_\text{r} \rangle^{\mathcal{B}}$
2: $\langle \text{fdmr} \rangle^{\mathcal{B}} \leftarrow \langle \text{s}_\text{d} \rangle^{\mathcal{B}} \wedge \neg\langle \text{s}_\text{r} \rangle^{\mathcal{B}}$
3: **return** $\langle \text{fdmr} \rangle^{\mathcal{B}}$ **?**
$(\langle \text{eq} \rangle^{\mathcal{B}}$ **?** $\langle \text{A} \rangle^{\mathcal{B}}$ **:** $\langle 0 \rangle^{\mathcal{B}})$ **:** $(\langle \text{eq} \rangle^{\mathcal{B}}$ **?** $\langle \text{A} \rangle^{\mathcal{B}}$ **:** $\langle \text{B} \rangle^{\mathcal{B}})$

Sex Comparison

Supplementary Table 5 evaluates the compatibility of a donor and recipient based on their sex. It takes two secret shares $\langle s_d \rangle^B$ and $\langle s_r \rangle^B$ as input, which represent the sex of the donor and recipient, respectively. In Line 1, the subprotocol determines if the pair shares the same sex. Line 2 checks whether the donor is female and the recipient male. As final step, the output weight of this donor and recipient constellation is computed, i.e., the optimal combination ("Class A") with equal sex receives the highest weight, while a female donor and a male recipient are assigned the lowest weight (0).

MPC Cost

Supplementary Table 5 evaluates one comparison, one inversion, one AND, and three MUX gates. As Line 1 and Line 2, as well as two of the MUX gates in Line 3, are independent, we have a circuit depth of 3. Thus, Supplementary Table 5 is slightly more efficient in $\mathcal{B}$ than in $\mathcal{Y}$.

Weight Comparison

Supplementary Table 6 evaluates the compatibility of a donor and recipient based on their weight. It takes two secret shares as input: $\langle w_d \rangle^B$ and

Supplementary Table 6 `evalWeight`($\langle \mathsf{w_d} \rangle^{\mathcal{B}}$: int, $\langle \mathsf{w_r} \rangle^{\mathcal{B}}$: int) $\rightarrow$ int

1: **return** $\langle \mathsf{w_d} \rangle^{\mathcal{B}} < \langle \mathsf{w_r} \rangle^{\mathcal{B}}$ **?** $\langle 0 \rangle^{\mathcal{B}}$ **:** $\langle \mathsf{A} \rangle^{\mathcal{B}}$

$\langle w_r \rangle^{\mathcal{B}}$, which represent the weight of the donor and recipient, respectively. If the donor weighs less than the recipient, it returns a secret shared 0, otherwise, it indicates a good fit (i.e., class "A" w.r.t. criteria weight).

MPC Cost
We evaluate only one comparison gate. As the evaluation of a single comparison is more efficient in $\mathcal{Y}$ than in $\mathcal{B}$ [1], $\mathcal{Y}$ would be more efficient. However, the conversion cost outweigh this benefit, which is why $\mathcal{B}$ is used for this subprotocol as in the previous comparison protocols.

Additional Protocols for the Cycle Computation

Supplementary Table 7 `removeWeights`($\langle \mathsf{compG} \rangle^{\mathcal{B}}$: matrix) $\rightarrow$ matrix

1: $\langle \mathsf{uG} \rangle^{\mathcal{A}} \leftarrow$ matrix $\in \langle 0 \rangle^{\mathcal{A}|\mathsf{pairs}|}$
2: **for** $i = 0, \ldots, |\mathsf{pairs}| - 1$ **do**
3: **for** $j = 0, \ldots, |\mathsf{pairs}| - 1$ **do**
4: $\langle \mathsf{uG} \rangle^{\mathcal{A}}[i][j] \leftarrow$ b2a($\langle \mathsf{compG} \rangle^{\mathcal{B}}[i][j] > \langle 0 \rangle^{\mathcal{B}}$ **?** $\langle 1 \rangle^{\mathcal{B}}$ **:** $\langle 0 \rangle^{\mathcal{B}}$
5: **end for**
6: **end for**
7: **return** $\langle \mathsf{uG} \rangle^{\mathcal{A}}$

Weight Removal
In Supplementary Table 7, we compute the unweighted compatibility graph, which is used for determining the number of cycles for the desired cycle length. It takes the weighted compatibility graph *compG* as input. The number of donor-recipient pairs |pairs| is public. In Line 4, we remove the edge weights: If it is greater than 0, it is set to 1, otherwise to 0. As preparation for later processing, a conversion to $\mathcal{A}$ is done.

MPC Cost
The subprotocol shown in Supplementary Table 7 evaluates $|\mathsf{pairs}|^2$ comparisons, `MUX` gates, and conversions. The comparisons and `MUX` gates are independent, which results in a circuit depth of 2. Due to the total number of `AND` gates, which is $2 \times |\mathsf{pairs}|$, this subprotocol is most efficient in $\mathcal{B}$.

kNN Sort Protocol
Our next MPC subprotocol shown in Supplementary Table 8 is a k-nearest neighbor sort (a slightly adapted version of the protocol in [2]) that identifies the k most robust cycles (i.e., with the highest likelihood to result in successful transplantations).
It takes a secret shared vector of tuples cyclesSet with exchange cycles and their respective weights and k as input. The length of cycles cLen is a public parameter. First, the subprotocol iterates over all cycles in |cyclesSet| to perform an insertion sort. Each cycle and the respective weight are added to sortedC and sortedW if its weight is one of the k highest weights (cf. Lines 11 to 27). Thus, the final sortedW and sortedC are sorted in decreasing order with respect to the weights of cycles.

MPC Cost
This subprotocol evaluates $|\mathsf{cyclesSet}| \times k$ comparisons and $|\mathsf{cyclesSet}| \times k \times (1 + \mathsf{cLen})$ `MUX` gates. It is most efficient in $\mathcal{Y}$ due to depth of the circuit determined by the number of `AND` gates.

Duplicate Removal
Supplementary Table 9 removes all duplicated exchange cycles and outputs the remaining $|\mathsf{unique}| = \lfloor \frac{|\mathsf{cycles}|}{\mathsf{cLen}} \rfloor$ cycles.
It takes a secret shared vector of tuples sortedCycles as input, which contains cycles and weights sorted according to the respective weights (i.e., the output by Supplementary Table 8). The number of existing cycles

Supplementary Table 8 `kNNSort`($\langle \mathsf{cyclesSet} \rangle^{\mathcal{Y}}$: vector of tuples, k: int) $\rightarrow$ vector of cycles

1: $\langle \mathsf{sortedW} \rangle^{\mathcal{Y}} \leftarrow \emptyset$
2: $\langle \mathsf{sortedC} \rangle^{\mathcal{Y}} \leftarrow \emptyset$
3: **for** $i = 0, \ldots, k$ **do**
4: $\langle \mathsf{sortedW} \rangle^{\mathcal{Y}}$.append($\langle 0 \rangle^{\mathcal{Y}}$)
5: $\langle \mathsf{vertices} \rangle^{\mathcal{Y}} \leftarrow \emptyset$
6: **for** $j = 0, \ldots,$ cLen $- 1$ **do**
7: $\langle \mathsf{vertices} \rangle^{\mathcal{Y}}$.append($|\langle \mathsf{pairs} \rangle^{\mathcal{Y}}|$)
8: **end for**
9: $\langle \mathsf{sortedC} \rangle^{\mathcal{Y}}$.append($\langle \mathsf{vertices} \rangle^{\mathcal{Y}}$)
10: **end for**
11: **for** $i = 0, \ldots, |\mathsf{cyclesSet}| - 1$ **do**
12: $\langle \mathsf{sortedW} \rangle^{\mathcal{Y}}[k] \leftarrow \langle \mathsf{cyclesSet} \rangle^{\mathcal{Y}}[i][0]$
13: $\langle \mathsf{sortedC} \rangle^{\mathcal{Y}}[k] \leftarrow \langle \mathsf{cyclesSet} \rangle^{\mathcal{Y}}[i][1]$
14: **for** $j = 0, \ldots, k - 1$ **do**
15: $\langle \mathsf{sel} \rangle^{\mathcal{Y}} \leftarrow \langle \mathsf{sortedW} \rangle^{\mathcal{Y}}[j] > \langle \mathsf{sortedW} \rangle^{\mathcal{Y}}[j - 1]$
16: $\langle \mathsf{tmp1} \rangle^{\mathcal{Y}} \leftarrow \langle \mathsf{sortedW} \rangle^{\mathcal{Y}}[j]$
17: $\langle \mathsf{tmp2} \rangle^{\mathcal{Y}} \leftarrow \langle \mathsf{sortedW} \rangle^{\mathcal{Y}}[j - 1]$
18: $\langle \mathsf{sortedW} \rangle^{\mathcal{Y}}[j] \leftarrow \langle \mathsf{sel} \rangle^{\mathcal{Y}}$ **?** $\langle \mathsf{tmp2} \rangle^{\mathcal{Y}}$ **:** $\langle \mathsf{tmp1} \rangle^{\mathcal{Y}}$
19: $\langle \mathsf{sortedW} \rangle^{\mathcal{Y}}[j - 1] \leftarrow \langle \mathsf{sel} \rangle^{\mathcal{Y}}$ **?** $\langle \mathsf{tmp1} \rangle^{\mathcal{Y}}$ **:** $\langle \mathsf{tmp2} \rangle^{\mathcal{Y}}$
20: **for** $l = 0, \ldots,$ cLen $- 1$ **do**
21: $\langle \mathsf{tmp1} \rangle^{\mathcal{Y}} \leftarrow \langle \mathsf{sortedC} \rangle^{\mathcal{Y}}[j][l]$
22: $\langle \mathsf{tmp2} \rangle^{\mathcal{Y}} \leftarrow \langle \mathsf{sortedC} \rangle^{\mathcal{Y}}[j - 1][l]$
23: $\langle \mathsf{sortedC} \rangle^{\mathcal{Y}}[j][l] \leftarrow \langle \mathsf{sel} \rangle^{\mathcal{Y}}$ **?** $\langle \mathsf{tmp2} \rangle^{\mathcal{Y}}$ **:** $\langle \mathsf{tmp1} \rangle^{\mathcal{Y}}$
24: $\langle \mathsf{sortedC} \rangle^{\mathcal{Y}}[j - 1][l] \leftarrow \langle \mathsf{sel} \rangle^{\mathcal{Y}}$ **?** $\langle \mathsf{tmp1} \rangle^{\mathcal{Y}}$ **:** $\langle \mathsf{tmp2} \rangle^{\mathcal{Y}}$
25: **end for**
26: **end for**
27: **end for**
28: $\langle \mathsf{result} \rangle^{\mathcal{Y}} \leftarrow \emptyset$
29: **for** $i = 0, \ldots, |\mathsf{cycles}| - 1$ **do**
30: $\langle \mathsf{result} \rangle^{\mathcal{Y}}$.append(tuple($\langle \mathsf{sortedW} \rangle^{\mathcal{Y}}[i]$, $\langle \mathsf{sortedC} \rangle^{\mathcal{Y}}[i]$))
31: **end for**
32: **return** $\langle \mathsf{result} \rangle^{\mathcal{Y}}$

|cycles|, the number of unique cycles |unique|, and the cycle length cLen are public parameters. For each cycle $c1$ in sortedCycles, it is checked if it is equal to any other cycle $c2$ (cf. Lines 6 to 13). If this is the case, its weight is set to 0 (cf. Line 15). To each equality, it is evaluated if the vertex of $c1$ at index l and the vertex of $c2$ at index $(l+k) \mod$ cLen are identical (cf. Line 9). With Supplementary Table 8, sortedCycles is sorted and only the |unique| cycles with the highest weight are returned. The number of unique cycles is |unique| $= \lfloor \frac{|\text{cycles}|}{\text{cLen}} \rfloor$.

MPC Cost

Supplementary Table 9 has |cycles| $\times \sum_{i=0}^{|\text{cycles}|}$ (cLen $\times$ (cLen $-$ 1)) comparisons and AND gates, |cycles| $\times \sum_{i=0}^{|\text{cycles}|}$ (cLen $-$ 1) OR gates, |cycles| MUX gates. Including Supplementary Table 8, this results in |cycles| $\times$ |unique| comparison and MUX gates, and an additional |cycles| $\times$ |unique| $\times$ (1 + cLen) MUX gates. This subprotocol is most efficient in $\mathcal{Y}$ due to the depth of the circuit created by AND gates.

Supplementary Table 9 `removeDuplicates`($\langle$sortedCycles$\rangle^{\mathcal{Y}}$: vector of tuples) $\rightarrow$ vector of cycles

1: **for** $i = 0, \ldots,$ |cycles| $- 1$ **do**
2: $\quad\langle$c1$\rangle^{\mathcal{Y}} \leftarrow \langle$sortedCycles$\rangle^{\mathcal{Y}}[i][1]$
3: $\quad\langle$combDup$\rangle^{\mathcal{Y}} \leftarrow \langle 0\rangle^{\mathcal{Y}}$
4: $\quad$**for** $j = 0 : i$ **do**
5: $\quad\quad\langle$c2$\rangle^{\mathcal{Y}} \leftarrow \langle$sortedCycles$\rangle^{\mathcal{Y}}[j][1]$
6: $\quad\quad$**for** $k = 1, \ldots,$ cLen $- 1$ **do**
7: $\quad\quad\quad\langle$duplicate$\rangle^{\mathcal{Y}} \leftarrow \langle 1\rangle^{\mathcal{Y}}$
8: $\quad\quad\quad$**for** $l = 0, \ldots,$ cLen $- 1$ **do**
9: $\quad\quad\quad\quad\langle$same$\rangle^{\mathcal{Y}} \leftarrow \langle$c1$\rangle^{\mathcal{Y}}[l] == \langle$c2$\rangle^{\mathcal{Y}}[(l+k) \mod$ cLen$]$
10: $\quad\quad\quad\quad\langle$duplicate$\rangle^{\mathcal{Y}} \leftarrow \langle$duplicate$\rangle^{\mathcal{Y}} \wedge \langle$same$\rangle^{\mathcal{Y}}$
11: $\quad\quad\quad$**end for**
12: $\quad\quad\quad\langle$combDup$\rangle^{\mathcal{Y}} \leftarrow \langle$combDup$\rangle^{\mathcal{Y}} \vee \langle$duplicate$\rangle^{\mathcal{Y}}$
13: $\quad\quad$**end for**
14: $\quad$**end for**
15: $\quad\langle$sortedCycles$\rangle^{\mathcal{Y}}[i][0] \leftarrow \langle$isDuplicate$\rangle^{\mathcal{Y}}$ **?** $\langle 0\rangle^{\mathcal{Y}}$ **:** $\langle$sortedCycles$\rangle^{\mathcal{Y}}[i][0]$
16: **end for**
17: **return** `kNNSort`($\langle$sortedCycles$\rangle^{\mathcal{Y}}$, |unique|)

Supplementary Table 10 #`TotalCycles`() $\rightarrow$ int

1: |allCycles| $\leftarrow$ |pairs|
2: **for** $i = 1, \ldots,$ cLen $- 1$ **do**
3: $\quad$|allCycles| $\leftarrow$ |allCycles| $\cdot$ (|pairs| $- i$)
4: **end for**
5: **return** |allCycles|

Total Number of Cycles

Supplementary Table 10 computes the maximum number of cycles that can exist in the compatibility graph. Each vertex must appear at most once in a cycle, which limits the number of possible cycles. As the numbers of pairs |pairs| and the cycle length cLen are public, computation can be done on plaintext.

Additional Protocols for the Solution Evaluation

Disjoint Cycles

Supplementary Table 11 computes whether a cycle `cCycle` does not join vertices with other cycles of a set of cycles `cycles`. It takes as input the set of secret shared cycles `cycles`, the secret shared cycle `cCycle`, and the number of cycles in `cycles` count. If another cycle shares a vertex with `cCycle`, `disJ` is set to 1 (cf. Line 10). In Line 12, we invert the result for further evaluation.

MPC Cost

In this subprotocol, we evaluate |cycles| $\times$ cLen (cf. Line 6). In Line 10, we evaluate $\log_2$(cLen) OR gates. At the end, we evaluate one XOR gate. As

Supplementary Table 11 `disjointSet`($\langle$cycles$\rangle^{\mathcal{B}}$: vector of tuples, $\langle$cCycle$\rangle^{\mathcal{B}}$: vector, $count$: int) $\rightarrow$ Boolean

1: $\langle$disJ$\rangle^{\mathcal{B}} \leftarrow \emptyset$
2: **for** $i = 0, \ldots,$ count $- 1$ **do**
3: $\quad\langle$c$\rangle^{\mathcal{B}} \leftarrow \langle$cycles$\rangle^{\mathcal{B}}[i][1]$
4: $\quad$**for** $j = 0, \ldots,$ cLen $- 1$ **do**
5: $\quad\quad$**for** $k = 0, \ldots,$ cLen $- 1$ **do**
6: $\quad\quad\quad\langle$tmp$\rangle^{\mathcal{B}} \leftarrow \langle$c$\rangle^{\mathcal{B}}[j] == \langle$cCycle$\rangle^{\mathcal{B}}[k]$
7: $\quad\quad\quad\langle$disJ$\rangle^{\mathcal{B}}$.append($\langle$tmp$\rangle^{\mathcal{B}}$)
8: $\quad\quad$**end for**
9: $\quad$**end for**
10: $\quad\langle$disJ$\rangle^{\mathcal{B}} \leftarrow$ `ORTREE`($\langle$disJ$\rangle^{\mathcal{B}}$)
11: **end for**
12: **return** $\neg\langle$disJ$\rangle^{\mathcal{B}}[0]$

the number of total AND gates is greater than the depth of the circuit, this subprotocol is most efficient in $\mathcal{B}$.

Supplementary Table 12 `findMaximumSet`($\langle$cyclesSets$\rangle^{\mathcal{Y}}$: vector of tuples, $\langle$cycleW$\rangle^{\mathcal{Y}}$: vector) $\rightarrow$ tuple

1: $\langle$weights$\rangle^{\mathcal{Y}} \leftarrow \emptyset$
2: $\langle$tmp$\rangle^{\mathcal{Y}} \leftarrow \emptyset$
3: **for** $i = 0, 1$ **do**
4: $\quad\langle$weights$\rangle^{\mathcal{Y}}$.append($\langle 0\rangle^{\mathcal{Y}}$)
5: $\quad\langle$sets$\rangle^{\mathcal{Y}} \leftarrow \emptyset$
6: $\quad$**for** $j = 0, \ldots$|unique| $- 1$ **do**
7: $\quad\quad\langle$vertices$\rangle^{\mathcal{Y}} \leftarrow \emptyset$
8: $\quad\quad$**for** $j = 0, \ldots$ cLen $- 1$ **do**
9: $\quad\quad\quad\langle$vertices$\rangle^{\mathcal{Y}}$.append($\langle$|pairs|$\rangle^{\mathcal{Y}}$)
10: $\quad\quad$**end for**
11: $\quad\quad\langle$tmp$\rangle^{\mathcal{Y}}$.append($\langle$vertices$\rangle^{\mathcal{Y}}$)
12: $\quad$**end for**
13: $\quad\langle$sets$\rangle^{\mathcal{Y}}$.append($\langle$tmp$\rangle^{\mathcal{Y}}$)
14: **end for**
15: **for** $i = 0, \ldots,$ |unique| $- 1$ **do**
16: $\quad\langle$weights$\rangle^{\mathcal{Y}}[1] \leftarrow \langle$cycleW$\rangle^{\mathcal{Y}}[i]$
17: $\quad\langle$sets$\rangle^{\mathcal{Y}}[1] \leftarrow \langle$cycleSets$\rangle^{\mathcal{Y}}[i]$
18: $\quad\langle$sel$\rangle^{\mathcal{Y}} \leftarrow \langle$weights$\rangle^{\mathcal{Y}}[1] > \langle$weights$\rangle^{\mathcal{Y}}[0]$
19: $\quad\langle$tmp1$\rangle^{\mathcal{Y}} \leftarrow \langle$weights$\rangle^{\mathcal{Y}}[1]$
20: $\quad\langle$tmp2$\rangle^{\mathcal{Y}} \leftarrow \langle$weights$\rangle^{\mathcal{Y}}[0]$
21: $\quad\langle$weights$\rangle^{\mathcal{Y}}[1] \leftarrow \langle$sel$\rangle^{\mathcal{Y}}$ **?** $\langle$tmp2$\rangle^{\mathcal{Y}}$ **:** $\langle$tmp1$\rangle^{\mathcal{Y}}$
22: $\quad\langle$weights$\rangle^{\mathcal{Y}}[0] \leftarrow \langle$sel$\rangle^{\mathcal{Y}}$ **?** $\langle$tmp1$\rangle^{\mathcal{Y}}$ **:** $\langle$tmp2$\rangle^{\mathcal{Y}}$
23: $\quad$**for** $j = 0, \ldots,$ |unique| $- 1$ **do**
24: $\quad\quad\langle$tmp1$\rangle^{\mathcal{Y}} \leftarrow \langle$sets$\rangle^{\mathcal{Y}}[1][j]$
25: $\quad\quad\langle$tmp2$\rangle^{\mathcal{Y}} \leftarrow \langle$sets$\rangle^{\mathcal{Y}}[0][j]$
26: $\quad\quad\langle$sets$\rangle^{\mathcal{Y}}[1][j] \leftarrow \langle$sel$\rangle^{\mathcal{Y}}$ **?** $\langle$tmp2$\rangle^{\mathcal{Y}}$ **:** $\langle$tmp1$\rangle^{\mathcal{Y}}$
27: $\quad\quad\langle$sets$\rangle^{\mathcal{Y}}[0][j] \leftarrow \langle$sel$\rangle^{\mathcal{Y}}$ **?** $\langle$tmp1$\rangle^{\mathcal{Y}}$ **:** $\langle$tmp2$\rangle^{\mathcal{Y}}$
28: $\quad$**end for**
29: **end for**
30: **return** ($\langle$weights$\rangle^{\mathcal{Y}}[0]$, $\langle$sets$\rangle^{\mathcal{Y}}[0]$)

Maximum Set

Supplementary Table 12 computes the set of cycles with the highest sum of weights, thus, the set of cycles with the highest probability for successful transplantations. Note that we do not compute the globally optimal solution, but a local optimum.

The subprotocol takes a secret shared vector of tuples cycles and a secret shared vector weights as input. cycles contains all sets of disjoint cycles and weights contains the respective weights of the each set. The number of pairs |pairs| and the number of unique cycles |unique| are public parameters. This subprotocol is a slight variation of Table 8 adapted to here used data structures.

The parameter k is fixed to 1 since we look for the set with the highest weight.

MPC Cost
This protocols evaluates |unique| comparison and $2|\text{unique}|^2 + 2|\text{unique}|$ `MUX` gates. Due to the large number of `MUX` gates in combination with the number of `AND` gates determining the depth of the circuit, it is most efficient in $\mathcal{Y}$ sharing.

Communication Improvement with ABY2.0
Implementing SPIKE in ABY2.0 [3] can further decrease the communication cost. As the respective protocols were implemented only very recently in MOTION2NX [4], we use ABY [1] in our benchmarks to show practicality and additionally discuss the improvements that can be achieved with ABY2.0 in the following.
ABY2.0's improvements for secure multiplication with two inputs decreases the communication of the first and second part, the compatibility matching and the cycle computation. The improvements to conversions between different sharing types additionally benefit the first and the second part, as these parts contain the most conversions between $\mathcal{B}$ and $\mathcal{A}$. Further, the optimizations for matrix multiplications are beneficial for the second part. Concretely, the communication of the protocol in Table 5 decreases from $3 \times \ell^2 + 24 \times \ell + 2 \times \ell \times \kappa$ bits to $23 \times \ell + \ell \times \kappa$ bits in every iteration of the inner loop (without considering the subprotocols). Similarly, the communication of the protocol in Table 6 can be reduced from $\ell \times (\frac{\ell}{2} + 2 \times \kappa + 4 \times |\text{pairs}|^3 + 1.5)$ bits to $\ell \times (\kappa + 2 \times |\text{pairs}|^3 + 3)$ bits, where ℓ is the bitlength and κ is the security parameter.

ABY Security Assumptions and Guarantees
The ABY MPC framework [1] provides mixed-abstraction building blocks for the creation of highly efficient hybrid-protocol MPC applications in a semi-honest adversary setting. Independent of the specific circuit design, a number of base-OTs are executed in the beginning to setup OT Extensions. The used base-OT primitive [5] guarantees security under the Computational Diffie-Hellman (CDH) hardness assumption. Being closely related to the discrete logarithm problem, this assumption is known to not be quantum resistant. The OT Extension [6, 7] primitive uses fixed-key AES modeled as a random permutation. While still considered secure in a semi-honest setting, theoretical attacks in the active security setting have been demonstrated [8]. Furthermore, ABY relies on the random oracle assumption, implemented by the SHA256 hash function. Similarly, Yao's Garbled Circuits [9] (denoted by $\mathcal{Y}$ in our protocols) directly rely on the random permutation assumptions, while Arithmetic and Boolean Sharing [10] (denoted by $\mathcal{A}/\mathcal{B}$ in our protocols) indirectly rely on those assumptions as a source of randomness. Those protocols can achieve information-theoretical security given a true correlated randomness source.

Detailed Benchmark Results
Supplementary Tables 13 to 15 show the detailed benchmark results for the setup and online phase in all three described network settings (A: LAN + $10\,\text{Gb/s}$, B: LAN + $1\,\text{Gb/s}$, C: WAN) and a cycle length of $L = 2$. Supplementary Tables 18 and 19 show the results for a cycle length of $L = 3$.
Supplementary Table 20, finally, compares the benchmark results of both reduced medical compatibility factor set and the full set. This benchmark was performed in the two network settings A and C, as above.

Supplementary Table 13 Comparison of the communication costs and setup and online runtimes of SPIKE for the three networking configurations A: LAN + $10\,\mathrm{Gb/s}$, B: LAN + $1\,\mathrm{Gb/s}$, C: WAN, and for cycle length $L = 2$. This table contains the aggregated total costs and the individual costs of Phase 1 (Compatibility Matching).

Pairs	Comm. [MiB]		Setup Phase [s]			Online Phase [s]		
Pairs	Setup	Online	A	B	C	A	B	C
Total								
2	0.1	0	0.021	0.021	0.78	0.04	0.039	2.1
4	1.1	0.1	0.052	0.051	1.7	0.075	0.08	3.1
6	3.4	0.3	0.1	0.11	2.5	0.15	0.15	4.3
8	5.6	0.4	0.13	0.17	3	0.17	0.18	4.4
10	12.7	0.8	0.22	0.24	4	0.28	0.29	5.8
12	19.5	1	0.37	0.34	4.4	0.46	0.37	6.6
14	55.8	2.3	0.61	0.68	7.4	0.8	0.88	12
16	95.4	3.4	0.94	1.1	11	1.2	1.3	15
18	159	5.1	1.4	1.6	15	1.8	1.9	18
20	412.1	11.8	2.9	4.9	34	4.2	7.4	30
22	617.8	16.6	4.2	5.2	47	6.3	6.4	36
24	823.3	21.1	5.5	6.7	64	8.4	8.5	42
26	1,104.8	27	7.2	8.7	81	11	11	49
28	1,281.6	30.2	8.3	10	93	13	13	53
30	1,608.3	36.5	10	13	120	17	17	59
32	2,202.9	48.3	14	19	150	24	24	71
34	2,999.7	63.8	18	22	200	33	33	85
36	3,971.7	82.2	24	26	260	44	43	100
38	5,036.2	101.8	29	35	320	57	57	120
40	6,394	126.6	37	45	400	75	75	140
Phase 1: Compatibility Matching								
2	0	0	0.0071	0.0065	0.31	0.015	0.015	0.85
4	0.1	0	0.0093	0.0087	0.42	0.016	0.015	0.85
6	0.2	0	0.012	0.013	0.52	0.017	0.017	0.85
8	0.4	0	0.016	0.016	0.62	0.019	0.018	0.84
10	0.6	0	0.02	0.021	0.62	0.021	0.021	0.85
12	0.8	0	0.026	0.025	0.65	0.024	0.024	0.85
14	1.2	0	0.031	0.032	0.72	0.028	0.028	0.86
16	1.5	0	0.036	0.038	0.75	0.034	0.031	0.86
18	1.9	0	0.047	0.045	0.82	0.033	0.033	0.86
20	2.4	0	0.053	0.054	0.85	0.039	0.039	0.88
22	2.9	0.1	0.055	0.065	0.87	0.045	0.046	0.88
24	3.4	0.1	0.071	0.073	0.9	0.05	0.049	0.89
26	4	0.1	0.075	0.083	1	0.051	0.056	0.9
28	4.6	0.1	0.077	0.085	1	0.059	0.06	0.91
30	5.3	0.1	0.081	0.088	1.1	0.068	0.067	0.97
32	6.1	0.1	0.084	0.09	1.1	0.071	0.069	0.98
34	6.8	0.1	0.087	0.092	1.1	0.079	0.083	0.97
36	7.7	0.1	0.093	0.099	1.2	0.085	0.089	0.97
38	8.6	0.2	0.093	0.11	1.2	0.091	0.098	0.99
40	9.5	0.2	0.094	0.11	1.2	0.1	0.1	1

Supplementary Table 14 Comparison of the communication costs and setup and online runtimes of SPIKE for the three networking configurations A: LAN + $10\,\mathrm{Gb/s}$, B: LAN + $1\,\mathrm{Gb/s}$, C: WAN, and for cycle length $L = 2$. This table contains individual costs of Phase 2 and 3 (Cycle Computation and Evaluation).

Pairs	Comm. [MiB]		Setup Phase [s]			Online Phase [s]		
Pairs	Setup	Online	A	B	C	A	B	C
Phase 2: Cycle Computation								
2	0	0	0.0099	0.0099	0.43	0.013	0.012	0.75
4	0.2	0	0.013	0.013	0.54	0.014	0.014	0.76
6	0.4	0.1	0.02	0.02	0.83	0.017	0.018	0.76
8	0.9	0.1	0.028	0.031	1	0.021	0.021	0.85
10	1.7	0.2	0.043	0.047	1.2	0.024	0.027	0.77
12	2.8	0.3	0.06	0.059	1.3	0.033	0.031	0.79
14	4.3	0.4	0.082	0.087	1.6	0.034	0.04	0.8
16	6.2	0.5	0.1	0.12	1.8	0.047	0.048	0.82
18	8.6	0.7	0.12	0.12	1.8	0.048	0.054	0.84
20	11.6	0.8	0.13	0.14	2	0.063	0.061	0.87
22	15.3	1	0.13	0.17	2	0.075	0.072	0.9
24	19.6	1.2	0.15	0.19	2.9	0.078	0.083	1.1
26	24.6	1.5	0.17	0.23	3.2	0.088	0.1	1.3
28	30.5	1.7	0.19	0.27	5	0.1	0.11	2.4
30	37.2	2	0.22	0.3	4.8	0.11	0.12	2
32	44.8	2.3	0.24	0.35	5.7	0.12	0.14	2.2
34	53.4	2.6	0.27	0.42	6.5	0.13	0.15	2.3
36	63	3	0.3	0.48	7.1	0.13	0.16	2.3
38	73.7	3.4	0.34	0.55	7.9	0.14	0.17	2.3
40	85.6	3.8	0.38	0.63	8.8	0.16	0.18	2.4
Phase 3: Cycle Evaluation								
2	0.1	0	0.0023	0.0027	0.022	0.0086	0.0082	0.3
4	0.7	0.1	0.019	0.02	0.29	0.026	0.03	0.35
6	2.2	0.1	0.054	0.061	0.56	0.068	0.07	0.47
8	3.8	0.2	0.066	0.1	0.74	0.089	0.096	0.48
10	8.6	0.4	0.12	0.13	1.2	0.14	0.16	0.56
12	13.4	0.5	0.21	0.2	1.6	0.22	0.2	0.66
14	35	0.9	0.38	0.41	3.6	0.37	0.43	0.94
16	57.3	1.3	0.62	0.66	5.6	0.6	0.69	1.2
18	90.2	1.8	0.98	1	8.5	0.87	0.95	1.4
20	181.2	3.3	1.9	2.2	17	1.7	1.9	2.3
22	255.2	4.4	2.7	2.9	23	2.4	2.5	3
24	332.8	5.3	3.6	3.8	30	3.1	3.1	3.7
26	431.8	6.5	4.7	4.9	39	4	4	4.5
28	514.4	7.2	5.5	5.7	46	4.7	4.7	5.3
30	635.1	8.4	6.9	7.3	57	6	5.9	6.4
32	815.8	10.4	8.9	12	73	7.5	9.7	8
34	1,037.4	12.8	11	12	92	9.4	9.3	10
36	1,292.4	15.4	15	14	110	12	10	12
38	1,567.8	18	18	19	140	14	14	15
40	1,894.4	21.2	22	23	170	18	18	18

Supplementary Table 15 Comparison of the communication costs and setup and online runtimes of SPIKE for the three networking configurations A: LAN + $10\,\mathrm{Gb/s}$, B: LAN + $1\,\mathrm{Gb/s}$, C: WAN and for cycle length $L = 2$. This table contains individual costs of Phase 4 (Solution Evaluation).

Pairs	Comm. [MiB]		Setup Phase [s]			Online Phase [s]		
Pairs	Setup	Online	A	B	C	A	B	C
Part 4: Solution Evaluation								
2	0	0	0.002	0.0016	0.0071	0.0038	0.0037	0.22
4	0.2	0	0.01	0.01	0.42	0.02	0.021	1.2
6	0.5	0.1	0.019	0.019	0.63	0.044	0.045	2.2
8	0.5	0.1	0.018	0.019	0.62	0.045	0.045	2.2
10	1.8	0.2	0.043	0.041	0.96	0.088	0.088	3.6
12	2.4	0.2	0.078	0.055	0.84	0.18	0.11	4.3
14	15.4	0.9	0.12	0.15	1.5	0.37	0.39	9.4
16	30.4	1.6	0.18	0.26	2.5	0.55	0.55	12
18	58.2	2.6	0.27	0.44	4	0.8	0.84	15
20	216.9	7.6	0.84	2.5	14	2.4	5.4	26
22	344.5	11.2	1.3	2	21	3.8	3.9	31
24	467.5	14.5	1.7	2.7	30	5.2	5.3	36
26	644.4	19	2.3	3.5	38	7.2	7.4	42
28	732.1	21.1	2.5	3.9	41	8.3	8.4	44
30	930.7	26	3.2	4.8	53	11	11	50
32	1,336.2	35.5	4.5	5.7	73	16	14	60
34	1,902.1	48.2	6.4	9.2	100	23	23	72
36	2,608.7	63.7	8.7	12	140	32	32	86
38	3,386.1	80.2	11	16	180	43	43	100
40	4,404.4	101.4	15	21	230	57	57	120

Supplementary Table 16 and Supplementary Table 17 compare the communication size for cycle lengths $L = 2$ and $L = 3$ of this work to [11] and [12], respectively.

Supplementary Table 16 Comparison of total communication cost for cycle length L=2.

Pairs	Communication [MiB]	
Pairs	This Work	Breuer et al. 2022
10	13.4	759
20	423.9	13,311.9
30	1,644.8	71,679.7
40	6,520.6	266,238.8

Supplementary Table 17 Comparison of total communication cost for cycle length L=3.

Pairs	Communication [MiB]	
Pairs	This Work	Breuer et al. 2020
3	0.6	0.4
5	4.3	40
7	20.5	200
9	54.1	5,632
15	2,107.3	–
18	9,644.8	–

Author details

References

1. Demmler, D., Schneider, T., Zohner, M.: ABY - A Framework for Efficient Mixed-Protocol Secure Two-Party Computation. Network and Distributed System Security Symposium(NDSS) (2015)
2. Järvinen, K., Leppäkoski, H., Lohan, E.-S., Richter, P., Schneider, T., Tkachenko, O., Yang, Z.: PILOT: Practical Privacy-Preserving Indoor Localization Using OuTsourcing. In: IEEE European Symposium on Security and Privacy (EuroS&P) (2019)
3. Patra, A., Schneider, T., Suresh, A., Yalame, H.: ABY2.00: Improved Mixed-Protocol Secure Two-Party Computation. In: 30th USENIX Security Symposium (USENIX Security 21), pp. 2165–2182 (2021)
4. Braun, L., Cammarota, R., Schneider, T.: POSTER: A Generic Hybrid 2PC Framework with Application to Private Inference of Unmodified Neural Networks (Extended Abstract). Privacy in Machine Learning Workshop (PriML@NeurIPS'21) (2021)
5. Naor, M., Pinkas, B., Pinkas, B.: Efficient Oblivious Transfer Protocols. In: Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms (2001)
6. Ishai, Y., Kilian, J., Nissim, K., Petrank, E.: Extending Oblivious Transfers Efficiently. In: Advances in Cryptology - CRYPTO 2003, vol. 2729 (2003)
7. Asharov, G., Lindell, Y., Schneider, T., Zohner, M.: More Efficient Oblivious Transfer Extensions. Journal of Cryptology **30**(3) (2017)
8. Guo, C., Katz, J., Wang, X., Yu, Y.: Efficient and Secure Multiparty Computation from Fixed-Key Block Ciphers. In: 2020 IEEE Symposium on Security and Privacy (SP), pp. 825–841 (2020). IEEE
9. Yao, A.C.-C.: How to Generate and Exchange Secrets. In: 27th Annual Symposium on Foundations of Computer Science (SFCS 1986) (1986)
10. Goldreich, O., Micali, S., Wigderson, A.: How to Play ANY Mental Game. In: Proceedings of the 19th Annual ACM Symposium on Theory of Computing. STOC '87 (1987)
11. Breuer, M., Meyer, U., Wetzel, S., Mühlfeld, A.: A Privacy-Preserving Protocol for the Kidney Exchange Problem. WPES (2020)
12. Breuer, M., Meyer, U., Wetzel, S.: Privacy-Preserving Maximum Matching on General Graphs and its Application to Enable Privacy-Preserving Kidney Exchange. In: ACM Conference on Data and Application Security and Privacy (CODASPY) (2022)

Supplementary Table 18 Comparison of the communication costs and setup and online runtimes of SPIKE for the three networking configurations A: LAN + 10 Gb/s, B: LAN + 1 Gb/s, C: WAN and for cycle length $L = 3$. This table contains the aggregated total costs and the individual costs of Phases 1 and 2 (Compatibility Matching and Cycle Computation).

Pairs	Comm. [MiB]		Setup Phase [s]			Online Phase [s]		
Pairs	Setup	Online	A	B	C	A	B	C
Total								
3	0.5	0.1	0.029	0.028	0.97	0.054	0.056	2.2
5	4	0.3	0.096	0.11	2.2	0.13	0.15	2.9
7	19.7	0.7	0.26	0.27	4.1	0.3	0.34	4.3
9	52.6	1.5	0.63	0.66	7.3	0.63	0.73	5.3
11	182.5	3.3	2	2.1	19	1.9	2	9.4
13	1,215.8	16.3	12	13	110	12	12	34
15	2,084.4	22.8	21	23	180	19	20	45
17	5,428.5	58.1	52	56	440	54	54	93
18	9,537.2	107.6	88	95	740	100	100	150
Phase 1: Compatibility Matching								
3	0.1	0	0.0079	0.0075	0.32	0.015	0.015	0.85
5	0.1	0	0.011	0.01	0.42	0.016	0.016	0.85
7	0.3	0	0.014	0.014	0.52	0.018	0.018	0.85
9	0.5	0	0.019	0.018	0.61	0.019	0.02	0.85
11	0.7	0	0.023	0.024	0.64	0.023	0.023	0.86
13	1	0	0.029	0.029	0.72	0.025	0.024	0.86
15	1.3	0	0.034	0.036	0.74	0.029	0.029	0.86
17	1.7	0	0.041	0.043	0.77	0.035	0.034	0.87
18	1.9	0	0.042	0.049	0.82	0.035	0.034	0.87
Phase 2: Cycle Computation								
3	0.1	0	0.013	0.012	0.54	0.014	0.013	0.76
5	0.4	0	0.018	0.019	0.83	0.016	0.016	0.76
7	1.1	0.1	0.029	0.031	1	0.019	0.019	0.77
9	2.2	0.2	0.052	0.053	1.3	0.024	0.023	0.77
11	3.8	0.3	0.072	0.079	1.5	0.026	0.029	0.78
13	6.2	0.4	0.1	0.11	2	0.032	0.034	0.84
15	9.3	0.5	0.1	0.13	1.8	0.036	0.04	0.81
17	13.4	0.7	0.12	0.15	2	0.049	0.049	0.86
18	15.8	0.8	0.13	0.16	2.2	0.047	0.054	0.91

Supplementary Table 19 Comparison of the communication costs and setup and online runtimes of SPIKE for the three networking configurations A: LAN + 10 Gb/s, B: LAN + 1 Gb/s, C: WAN and for cycle length $L = 3$. This table contains the individual costs of Phases 3 and 4 (Cycle and Solution Evaluation).

Pairs	Comm. [MiB]		Setup Phase [s]			Online Phase [s]		
Pairs	Setup	Online	A	B	C	A	B	C
Phase 3: Cycle Evaluation								
3	0.3	0	0.0061	0.0068	0.1	0.022	0.022	0.42
5	3.4	0.2	0.06	0.071	0.6	0.089	0.1	0.53
7	17.9	0.6	0.2	0.21	2	0.22	0.27	0.71
9	49.1	1.2	0.54	0.56	4.8	0.53	0.63	1.1
11	172.4	2.6	1.8	2	16	1.7	1.7	2.2
13	1,005.9	9.1	11	12	89	9.1	9.1	9.6
15	1,773.8	12.8	19	21	160	16	16	16
17	4,213.3	26.5	47	50	370	38	38	39
18	6,735.8	42	79	82	590	65	65	65
Phase 4: Solution Evaluation								
3	0	0	0.0024	0.0022	0.011	0.0038	0.0059	0.22
5	0.1	0	0.0083	0.0082	0.32	0.014	0.014	0.75
7	0.5	0.1	0.016	0.017	0.54	0.039	0.04	1.9
9	0.8	0.1	0.024	0.025	0.64	0.057	0.056	2.6
11	5.5	0.4	0.078	0.087	1	0.19	0.19	5.5
13	202.7	6.9	0.81	1.4	14	2.4	2.5	22
15	300	9.5	1.1	1.8	18	3.6	3.7	27
17	1,200.1	30.9	4.1	6	64	15	16	53
18	2,783.8	64.8	9.2	13	150	36	36	87

Supplementary Table 20 Comparison of the setup and online runtimes of SPIKE for the reduced medical factor compatibility matching and the full set in the two main networking configurations A: LAN + $10\,\mathrm{Gb/s}$, C: WAN.

Pairs	Comm. [MiB]		Setup Phase [s]		Online Phase [s]	
Pairs	Setup	Online	A	C	A	C
Reduced Medical Factor Set						
2	0.1	0	0.0084	0.34	0.045	3
50	14.9	0.3	0.14	1.7	0.26	3.4
100	59.8	1.1	0.29	4.4	0.81	4.4
150	134.7	2.5	0.55	8.5	1.9	5.8
200	239.5	4.4	0.91	15	3.8	7.7
250	374.4	6.9	1.4	23	6.4	11
300	539.2	9.9	2	31	9.4	14
350	734	13.4	2.5	41	14	20
400	958.8	17.5	3.2	53	18	26
450	1,213.6	22.1	4.2	65	25	32
500	1,498.3	27.3	5.3	80	31	37
550	1,813.1	33	6.3	96	38	48
600	2,157.8	39.3	7.2	110	45	56
650	2,532.5	46.1	9	130	53	64
Full Medical Factor Set						
2	0.1	0	0.013	0.88	0.047	3.4
50	44	11.8	0.51	4.6	1	5.2
100	177.1	47.1	1.3	14	4.7	12
150	399.2	105.9	2.8	29	12	24
200	710.5	188.3	5.1	48	22	41
250	1,110.9	294.3	7.6	71	35	64
300	1,600.4	423.8	12	100	51	92
350	2,179.1	576.8	14	140	66	120
400	2,846.8	753.4	18	180	86	160
450	3,603.7	953.5	23	230	110	200
500	4,449.6	1,177.2	28	280	140	250
550	5,384.7	1,424.4	35	340	170	300
600	6,408.9	1,695.2	41	410	200	350
650	7,522.2	1,989.5	48	480	240	420

E Privacy-Preserving Epidemiological Modeling on Mobile Graphs (In Submission to TOPS'23)

[GHJ+23] D. Günther, M. Holz, B. Judkewitz, H. Möllering, B. Pinkas, T. Schneider, A. Suresh. **"Privacy-preserving epidemiological modeling on mobile graphs"**. https://ia.cr/2020/1546. Code: https://zenodo.org/record/6599225. 2023. Appendix E.

Privacy-Preserving Epidemiological Modeling on Mobile Graphs

DANIEL GÜNTHER, MARCO HOLZ, ENCRYPTO, Technical University of Darmstadt, Germany
BENJAMIN JUDKEWITZ, Charité-Universitätsmedizin, Germany
HELEN MÖLLERING, ENCRYPTO, Technical University of Darmstadt, Germany
BENNY PINKAS, Bar-Ilan University, Israel
THOMAS SCHNEIDER, ENCRYPTO, Technical University of Darmstadt, Germany
AJITH SURESH, Technology Innovation Institute, Abu Dhabi

Since 2020, governments all over the world have used a variety of containment measures to control the spread of COVID-19, such as contact tracing, social distance regulations, and curfews. Epidemiological simulations are commonly used to assess the impact of those policies before they are implemented. Unfortunately, their predictive accuracy is hampered by the scarcity of relevant empirical data, specifically detailed social contact graphs. As this data is inherently privacy-critical, there is an urgent need for a method to perform powerful epidemiological simulations on real-world contact graphs without disclosing sensitive information.

In this work, we present RIPPLE, a privacy-preserving epidemiological modeling framework that enables the execution of standard epidemiological models for infectious disease on a population's most recent real contact graph while keeping all contact information privately and locally on the participants' devices. As underlying building block, we present PIR-SUM, a novel extension to private information retrieval that allows users to securely download the sum of a set of elements from a database rather than individual elements. We provide a proof-of-concept implementation of our protocols demonstrating that a 2-week simulation over a population of half a million can be finished in 7 minutes, with each participant communicating less than 50 KB of data.

CCS Concepts: • **Security and privacy** → **Privacy-preserving protocols**; Social network security and privacy; Information accountability and usage control; **Privacy protections**.

ACM Reference Format:
Daniel Günther, Marco Holz, Benjamin Judkewitz, Helen Möllering, Benny Pinkas, Thomas Schneider, and Ajith Suresh. 2018. Privacy-Preserving Epidemiological Modeling on Mobile Graphs. 1, 1 (July 2018), 36 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

The COVID-19 pandemic has profoundly affected people's daily lives, posing significant challenges such as increased mental illness, balancing childcare, homeschooling, and work, an increase in domestic abuse cases, and many more [91, 119, 128]. Governments all over the world have taken a variety of steps to restrict the spread of the virus to save human lives and keep the economic system working. Those range from closing institutions, such as schools, to country-wide

Authors' addresses: Daniel Günther, Marco Holz, guenther@encrypto.cs.tu-darmstadt.de, ENCRYPTO, Technical University of Darmstadt, Germany; Benjamin Judkewitz, benjamin.judkewitz@charite.de, Charité-Universitätsmedizin, Germany; Helen Möllering, moellering@encrypto.cs.tu-darmstadt.de, ENCRYPTO, Technical University of Darmstadt, Germany; Benny Pinkas, benny@pinkas.net, Bar-Ilan University, Israel; Thomas Schneider, schneider@encrypto.cs.tu-darmstadt.de, ENCRYPTO, Technical University of Darmstadt, Germany; Ajith Suresh, Ajith.Suresh@tii.ae, Technology Innovation Institute, Abu Dhabi.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2018 Association for Computing Machinery.
Manuscript submitted to ACM

lockdowns. Despite these courageous efforts, the global number of infections skyrocketed, and COVID-19 claimed far too many lives. Aside from highly lethal diseases like COVID-19, many other infectious diseases have emerged and have had a significant impact on human life over time. For example, since 2022 incidences of mpox (previously known as monkeypox) in Europe have increased to the point that quarantine measures have been implemented [23, 34, 56].

In the context of COVID-19, contact tracing apps are being used all over the world to notify contacts of potential infections [108, 113]. Unfortunately, contact tracing has a fundamental limitation: It only notifies contacts of an infected person *after* the infection has been detected, i.e., typically after a person develops symptoms, is tested, receives the test result, and can connect with previous contacts [85, 123]. Tupper et al. [123] report that in British Columbia in April 2021, this process ideally took five days, reducing new cases by only 8% compared to not using contact tracing. They conclude that contact tracing must be supplemented with multiple additional containment measures to effectively control disease spread.

In contrast, we consider epidemiological modeling, which allows predicting the spread of an infectious disease in the *future* and has received a lot of attention [59, 133]. Epidemiological modeling allows to assess the effectiveness of containment measures by mathematically modeling their impact on the spread, aiding governments in selecting effective strategies [120]. For example, Davis et al. [40] predicted in early 2020 that COVID-19 would infect 85% of the British population without any containment measures in place, causing a massive overload of the health system (13-80× the capacity of intensive care units). Their forecast also indicated that short-term interventions such as school closures, social distancing, and so on would not effectively reduce the number of cases. As a result, the British government decided to implement a lockdown in March 2020, effectively reducing transmissions and stabilising the health system [120].

With access to detailed information about a population's size, density, transportation, and health care system, epidemiological modeling could accurately forecast disease transmission in a variety of situations [2]. Especially precise, up-to-date information about movements and physical interactions in space and time is crucial for precisely forecasting transmission as well as the impact of various control measures before being implemented [76]. In practice, these simulations may quickly model a future disease's spread, calculate the projected number of infections when specific actions are taken, and divert the spread to specific areas.

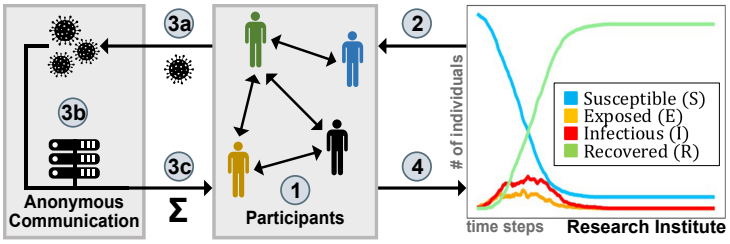
However, data on personal encounters in the real-world is very scarce and, thus the impact of containment measures can only be approximated so far [2, 53, 76]. This lack of data is primarily owing to the fact that encounter data has generally been acquired by surveys, which do not accurately reflect reality [48, 76], e.g., random encounters in public transport or shopping malls. Moreover, social interaction patterns change over time and sometimes even rapidly, as we have seen with social distancing measures, rendering collected contact information outdated. Hence, none of the existing data permits realistic simulations on the actual person-to-person social contact graph. Epidemiologists desire the full physical interaction graph of a population from a modeler's standpoint. Yet, strict data protection regulations such as in democratic states, honoring privacy rights, hinder accurate tracking of interpersonal contacts.

To address the issue of obtaining the most recent contact data while protecting individuals' privacy, we present RIPPLE, the first privacy-preserving framework for epidemiological modeling that allows precise simulations of disease spread based on current physical contact information while taking into account deployed control measures and without leaking any information about individuals' contacts. RIPPLE provides a privacy-preserving method for collecting real-time physical encounters and can compute arbitrary compartment-based epidemiological models[1] on the most recent contact graph of the previous days in a privacy-preserving manner. RIPPLE can be used to investigate the effect

[1] The implementation of concrete simulation functions is outside the scope of this work and referred to medical experts. More details on epidemiological modeling are given in §2.

of containment measures not only for COVID-19, but for *any* infectious diseases. We anticipate that our framework's privacy guarantee will encourage more people to participate, allowing epidemiologists to compute more accurate simulations that will eventually help to develop effective containment measures against diseases in the future.

Our Contributions. This paper introduces RIPPLE (cf. Fig. 1), a framework that expands the scope of privacy research from contact tracing to epidemiological modeling. While contact tracing only warns about potential infections in the *past*, epidemiological modeling can predict the spread of infectious diseases in the *future*. Anticipating the effects of various control measures allows for the development of informed epidemic containment strategies and political interventions prior to their implementation.



① Mobile app collects anonymous encounter tokens during interactions. ② Research Institute begins the simulation by providing initialization parameters. ③a Participants securely upload infection likelihood to servers. ③b Servers securely compute cumulative infection likelihood per participant. ③c Participants retrieve their cumulative infection likelihood. ④ The aggregate results (#S,#E,#I,#R) are sent to the Research Institute.

Fig. 1. Overview of RIPPLE Framework.

RIPPLE uses a fully decentralised system similar to the federated learning paradigm [93], fostering trust and widespread participation, and encouraging participants to contribute representative contact information. All participant data, such as encounter location, time, and distance, are kept locally on the participants' devices. Communication among participants occurs through anonymous channels facilitated by a group of semi-honest central servers.

RIPPLE offers two methods for privacy-preserving epidemiological modeling, each covering different use cases. The first method, $\text{RIPPLE}_{\text{TEE}}$, relies on the presence of a Trusted Execution Environment (TEE) on participants' mobile devices. The second method, $\text{RIPPLE}_{\text{PIR}}$, eliminates this assumption by utilising cryptographic primitives like Private Information Retrieval (PIR). Along the way, we develop a multi-server PIR extension enabling clients to retrieve the sum of a set of elements (in our case, infection likelihoods) from a database without learning anything about individual entries.

We assess the practicality of our methods by benchmarking core components using a proof of concept implementation. Our results show that, with adequate hardware, both protocols scale up to millions of participants. For instance, simulating 14 days with 1 million participants takes less than 30 minutes to complete. We summarize our contributions as follows:

(1) We present RIPPLE, the *first* privacy-preserving framework to perform epidemiological modeling on contact information stored on mobile devices. RIPPLE formalises the notion of *privacy-preserving* epidemiological modeling and defines privacy requirements.
(2) For epidemiological simulations using real-world contact data acquired with participants' mobile devices, we present two techniques – $\text{RIPPLE}_{\text{TEE}}$ and $\text{RIPPLE}_{\text{PIR}}$ – that combine anonymous communication techniques with either TEEs or PIR and anonymous credentials.

(3) We propose PIR-SUM, an extension to existing PIR schemes, that allows a client to download the sum of τ distinct database entries without learning the values of individual entries or revealing which entries were requested.
(4) We demonstrate the practicality of our framework by providing a detailed performance evaluation using our open source implementation of RIPPLE.

2 RELATED WORK & PRELIMINIARIES

This section discusses related works addressing privacy challenges in the context of infectious diseases as well as necessary background information on contact tracing and epidemiological modeling, including a clarification of the differences between the two. An overview of the (cryptographic) primitives used in this work is presented in §A.

2.1 Privacy-preserving Solutions in the Context of Infectious Diseases

CrowdNotifier [89] notifies visitors of (large) events about an infection risk when another visitor reported SARS-CoV-2 positive after the event, even if they have not been in close proximity of less than 2 meters. To protect user privacy, it follows a distributed approach where location and time information is stored encrypted on the user's device. Bampoulidis et al. [13] introduce a privacy-preserving two-party set intersection protocol that detects infection hotspots by intersecting infected patients, input by a health institute, with customer data from mobile network operators.

CoVault [42] is a data analytics platform based on secure multi-party computation techniques (MPC) and trusted execution environments (TEEs). The authors discuss the usage of CoVault for storing location and timing information of people usable by epidemiologists to analyse (unique) encounter frequencies or linkages among two disease outbreak clusters while preserving privacy.

Al-Turjman and David Deebak [4] integrate privacy-protecting health monitoring into a Medical Things device that monitors the health status (heart rate, oxygen saturation, temperature, etc.) of users in quarantine with moderate symptoms. Only in the event of an emergency is medical personnel notified. Pezzutto et al. [107] optimize the distribution of a limited set of tests to identify as many positive cases as possible, which are then isolated. Their system can be deployed in a decentralized, privacy-aware environment to identify individuals who are at high risk of infection. Barocchi et al. [14] develop a privacy-preserving architecture for indoor social distancing based on a privacy-preserving access control system. When users visit public facilities (e.g., a supermarket or an airport), their mobile devices display a route recommendation for the building that maximizes the distance to other people. Bozdemir et al. [19] suggest privacy-preserving trajectory clustering to identify typical movements of people allowing to detect forbidden gatherings when contact restrictions are in place.

Contact Tracing. A plethora of contact tracing systems has been introduced and deployed since the outbreak of the pandemic [3, 35, 113]. They either use people's location (GPS or telecommunication provider information) or measure proximity (via Bluetooth LE). Most systems can be categorized into centralized and decentralized designs [125]. In a centralized contact tracing system (e.g., [68, 118]), computations such as the generation of the tokens exchanged during physical encounters are done by a central party. This central party may also store some contact information depending on the concrete system design. In contrast, in decentralized approaches (e.g., [24, 108, 122]), computation and encounter information remain (almost completely) locally at the participants' devices.

Contact tracing focuses on determining contacts of infected people in the past. In contrast, epidemiological modeling, which we consider in this work, forecasts the spread of infectious diseases in the future. Thus, epidemiological modeling goes *beyond* established contact tracing systems. They share some technical similarities (specifically, the exchange of

encounter tokens), but on top of anonymously recording the contact graph, simulations have to be run on it. Similarly, presence tracing and hotspot detection are concerned with "flattening the curve" in relation to infections in the past. In contrast, epidemiological modeling is a tool for decision-makers to evaluate the efficacy of containment measures like social distancing in the future, allowing them to "get ahead of the wave".

2.2 Epidemiological Modeling

There are several options to model a disease mathematically. The popular compartment models [20, 21, 55, 61, 64, 66, 114, 133] capture the spread with a few continuous variables linked by simple differential equations. A prominent example is the SEIR model [45, 66, 74, 133] with four compartments to which people are assigned, namely, susceptible (S), exposed (E), infectious (I), and recovered (R). For each simulated time interval, the number of people assigned to each class is computed. While such models are useful for capturing macroscopic trends and also used in state-of-the-art epidemiological research, e.g., [32, 117], the basic approaches condense complex individual behaviour into few variables, thus, limiting the simulation's predictive power [92, 104]. Agent-based epidemiological models [54], on the other hand, initialise a large number of agents with a set of individual properties (e.g., location or age). These agents then interact according to a set of interaction rules (e.g., location-based or age-based) to simulate disease spread. The simulations are carried out in many time steps. Combining both directions, i.e., using agents in a compartment model, allows for a more realistic model of individual behaviour for forecasting disease transmission in a population. Many such simulations with varying parameters are run in parallel to simulate the effect of various policy interventions (e.g., reducing interactions between agents of a certain age, capping the maximum number of allowed contacts, or vaccinating a selected group of agents). The aggregated number of agents assigned to the same "infection class" (e.g., susceptible, exposed, infectious, and recovered for the SEIR model) is then computed for each simulation step.

A crucial question is how to model the agents' individual contact behaviour. Older models relied on survey-based contact matrices, which included information such as the average number of contacts in a given age range [76]. This is already a significant improvement over treating all people the same. However, aggregated network statistics cannot recreate the dynamics of a real complex network graph, as evidenced by the prevalence of super-spreaders with far more contacts than the average [80]. Thus, using the real-world contact graph between all individual members of the population would be ideal from an epidemiological standpoint.

Privacy-Preserving Epidemiological Modeling from Contact Tracing. If contact information collected through contact tracing apps was centralised, an up-to-date full contact graph could be constructed for epidemiological simulations. However, contact information is highly sensitive information that should not be shared. Contact information collected via mobile phones can reveal who, when, and whom people meet, which is by itself sensitive information and must be protected. Moreover, such information also enables to derive indications about the financial situation [16, 90, 116], personality [97], life-partners [6], and ethnicity [6]. One can think about many more examples: By knowing which medical experts are visited by a person, information about the health condition can be anticipated; contact with members of a religious minority as well as visits to places related to religion might reveal a religious orientation, etc. Thus, it would be ideal to enable precise epidemiological simulations without leaking any individual contact information.

One way to achieve privacy-preserving epidemiological modeling from contact tracing apps is to let each participant (i.e., each device using the contact tracing app) secretly share its contact information between a set of non-colluding servers, which can then jointly run simulations using techniques like secure multi-party computation (MPC), cf. §A. In fact, Araki et al. [9] show how to run graph algorithms on secret shared graphs via MPC efficiently. Even though such a

non-collusion assumption is common in the crypto community, the general public in some countries may have difficulty trusting a system in which all contact information is disclosed if the servers collude. In contrast, RIPPLE distributes trust among all participants in such a way that they can keep their own contact information local while simulating the spread of disease by sending messages to each other anonymously. Furthermore, only aggregated simulation results will be shared with a research institute, so no data directly relating to a single identity will be shared. This approach mimics the baseline idea of Federated Learning [93] and prominent contact tracing designs supported by Apple and Google.[2] The increased trust level of a distributed design fosters the crucial broad adoption of such a system in the population.

To the best of our knowledge, RIPPLE is the first framework that allows the execution of any agent-based compartment model on the distributed real contact graph while maintaining privacy.

3 THE RIPPLE FRAMEWORK

RIPPLE's primary goal is to facilitate the assessment of various combinations of potential containment measures proposed by epidemiologists and the government. Rather than implementing measures in real-life and analyzing their impact afterwards, our focus is on finding a balance between the benefits and drawbacks of these measures. Examples of such measures include mandating face masks in public places, limiting the size of gatherings, closing specific institutions or stores, and even implementing curfews and regional lockdowns.

Participants in RIPPLE collect personal encounter data anonymously and locally store it on their mobile devices such as cell phones, similar to privacy-preserving contact tracing apps. However, for epidemiological modeling, RIPPLE must also derive a contact graph without leaking sensitive personal information in order to compute simulations of disease spread, which may involve multiple sets of containment measures for some time period, such as two weeks. In almost every country, we can find a 6-hour period during the night when the majority of the population sleeps and mobile devices are idle, connected to the Internet via WiFi, and possibly charging, i.e., an ideal time window for running RIPPLE simulations. The results can then be analysed by medical experts to learn more about the disease or by political decision-makers to determine the most promising containment measures to implement.

To acquire representative and up-to-date physical encounter data, widespread public usage of RIPPLE would be ideal, similar to contact tracing apps. One way to encourage this is to piggyback RIPPLE on the official contact tracing applications used by several countries. Politicians, on the other hand, can motivate residents beyond the intrinsic incentive of supporting public health by coupling the use of RIPPLE with additional benefits such as discounted or free travel passes.

3.1 System and Threat Model

RIPPLE comprises of p participants, denoted collectively by $\mathcal{P}$, a research institute RI who is in charge of the epidemiological simulations, and a set of MPC servers C responsible for anonymous communication among the participants.

We assume that the research institute and MPC servers are semi-honest [60], which means they correctly follow protocol specifications while attempting to learn additional information. The semi-honest MPC servers are also used to establish an anonymous communication channel. We discuss the security of the anonymous communication channel in more detail in §B.3. A protocol is considered to be secure if nothing is leaked beyond what can be inferred from the output. Though the semi-honest security model is not the strongest security model, it provides a good trade-off between privacy and efficiency, which is why it is commonly used in the design of several practical privacy-preserving applications such as privacy-preserving machine learning [26, 94, 96, 105], genome/medical research [115, 121, 127], and localization services [71, 124]. It also protects against passive attacks by curious administrators and accidental data leakage.

[2] https://covid19.apple.com/contacttracing

Furthermore, it is quite often the first step toward developing protocols with stronger privacy guarantees [11, 87]. We believe this is a reasonable assumption in our setting because the research institute and the servers will be controlled/run by generally trusted entities such as governments or (public) medical research centres, potentially in collaboration with NGOs such as the EFF[3] or the CCC[4].

Given the importance of effective containment measures, we expect motivated participants to contribute to epidemiological modeling. However, assuming complete honesty from all millions of potential participants is unrealistic. Therefore, we also consider a client-malicious security model [25, 84] for the participants in $\mathcal{P}$, in which some of the participants are malicious and may deviate from the protocol to gather additional information about their encounters. Malicious behaviour can actively try to hamper or even destroy the correctness of the simulation. However, in the scope of this work, we concentrate on the aforementioned deviations for additional information gain, leaving the problem of developing efficient countermeasures against correctness attacks to future work. Tab. 1 summarises the notations used in this work.

	Parameter	Description
Entities	$\mathcal{P}$	Set of all participants; $\mathcal{P} = \{\mathcal{P}_1, \ldots, \mathcal{P}_p\}$
	RI	Research Institute
	C	Communication Servers $\{S_0, S_1, S_2\}$
Simulations	$\text{param}_{\text{sim}}$	simulation parameters defined by RI
	N_{sim}	# distinct simulations (executed in parallel)
	N_{step}	# steps per simulation
	$\text{class}_{\text{inf}}$	infection classes; $\text{class}_{\text{inf}} = \{\text{class}_{\text{inf}}^1, \ldots, \text{class}_{\text{inf}}^{N_{\text{inf}}}\}$
	I_i^s	$\mathcal{P}_i$'s infection class in simulation step $s \in [0, N_{\text{step}}]$
	$\mathcal{E}_i$	Encounter tokens of $\mathcal{P}_i$
	$E_i^{\max}$	#max. encounters by $\mathcal{P}_i$ in pre-defined time interval
	E^{avg}	average number of encounters
Protocols	κ	computational security parameter $\kappa = 128$
	r_e	Unique token for encounter $e \in [0, E^{\max}]$
	$\delta_i^{r_e}$	$\mathcal{P}_i$'s infection likelihood w.r.t. token r_e
	Δ_i	$\mathcal{P}_i$'s cumulative infection likelihood
	m_i^e	metadata of an encounter e by $\mathcal{P}_i$
	$(\text{pk}_i, \text{sk}_i)$	$\mathcal{P}_i$'s public/private key pair
	σ_i^e.	$\mathcal{P}_i$'s signature on message about encounter e

Table 1. Notations used in RIPPLE.

3.2 Phases of RIPPLE

RIPPLE consists of four phases shown in Fig. 1: i) Token Generation, ii) Simulation Initialization, iii) Simulation Execution, and iv) Result Aggregation. While RIPPLE can be applied to any compartment-based epidemiological modeling of infectious diseases (see §2.2), we will explain RIPPLE using the SEIR model [45, 74] and the COVID-19 virus as an example. For simplicity, we assume that each participant has installed an app that emulates RIPPLE on their mobile device, and they enter attributes like workplace, school, regular eateries, and cafes locally within the app (resp. the app could make suggestions for those based on the user's frequent locations).

Fig. 2 summarises the phases of the RIPPLE framework in the context of a single simulation setting. Multiple simulations can be executed in parallel. The concrete number of simulation runs with the same parameters or different parameters should be determined by epidemiologists. Note that simulations are run on collected data, e.g., from the last days, and not on real-time encounter information. This combines efficiency with maximally up-to-date encounter information.

① **- Token Generation:** During a physical encounter, participants exchange data via Bluetooth LE to collect anonymous encounter information (Fig. 3a), similar to contact tracing [65, 108, 122]. These tokens are stored locally on the devices of the users and do not reveal any sensitive information (i.e., identifying information) about the individuals involved. In addition to these tokens, the underlying application will collect additional information on the context of

[3] https://www.eff.org [4] https://www.ccc.de/en/

the encounter known as “metadata" for simulation purposes. This varies depending on the underlying instantiation of the protocol and can include details such as duration, proximity, time, and location. To generate the metadata, a set of standard labels (e.g., restaurants, bars, gyms) can be automatically assigned to a location derived from Google maps. The metadata can be used to include or exclude different encounters in the simulation phase, allowing the effect of containment measures to be modelled (e.g., restaurant closings by excluding all encounters that happened in restaurants). The token generation phase is not dependent on the simulation phase, so no simulation-dependent infection data is exchanged. The token generation phase is modelled as an ideal functionality $\mathcal{F}_{\mathsf{gen}}$ that will be instantiated later in §4.

Protocol RIPPLE

① - Token Generation

- $\mathcal{P}_i \in \mathcal{P}$ executes $\mathcal{F}_{\mathsf{gen}}$ all the time (on its mobile device), collecting encounter data of the form (r_e, m_e) with $e < E_i^{\max}$.

② - Simulation Initialization

- $\mathcal{P}_i \in \mathcal{P}$ receives $\mathsf{param}_{\mathsf{sim}}$ from RI and locally sets $I_i^1 = I_i^{\mathsf{init}}$.

③ - Simulation Execution

For each simulation step $s \in [N_{\mathsf{step}}]$, $\mathcal{P}_i \in \mathcal{P}$ execute the following:

- Filter out encounters using $\mathsf{param}_{\mathsf{sim}}$ to obtain encounter set $\mathcal{E}_i^s$.
- For each token $r_e \in \mathcal{E}_i^s$, compute the infection likelihood $\delta_i^{r_e}$ locally using the formula from RI.
- Invoke $\mathcal{F}_{\mathsf{esim}}$ with the input $\{\delta_i^{r_e}\}_{r_e \in \mathcal{E}_i^s}$ and obtain $\Delta_i^s = \sum_{r_e \in \mathcal{E}_i^s} \hat{\delta}_i^{r_e}$.
- Update the infection class I_i^s using Δ_i^s and the guidelines from RI.

④ - Result Aggregation

For each simulation step $s \in [N_{\mathsf{step}}]$, execute the following:

- $\mathcal{P}_i \in \mathcal{P}$ prepares $\{v_i^1, \ldots, v_i^{N_{\mathsf{inf}}}\}^s$ with $v_i^k = 1$ if $I_i^s = \mathsf{class}_{\mathsf{inf}}^k$ and $v_i^k = 0$ otherwise, for $k \in [N_{\mathsf{inf}}]$.
- Invoke $\mathcal{F}_{\mathsf{agg}}$ with inputs $\{v_i^1, \ldots, v_i^{N_{\mathsf{inf}}}\}^s$ to enable RI obtain the tuple $\{C_{\mathsf{inf}}^1, \ldots, C_{\mathsf{inf}}^{N_{\mathsf{inf}}}\}^s$, where $C_{\mathsf{inf}}^k = \sum_{\mathcal{P}_i \in \mathcal{P}} v_i^k$ for $k \in [N_{\mathsf{inf}}]$.

Fig. 2. RIPPLE Framework (for one simulation setting).

Running Example: Assume that a participant, Alice, takes the bus to pick up her daughter from school. There are several other people on this bus – for simplicity, we call them $\mathrm{Bob}_1, \ldots, \mathrm{Bob}_x$. As part of the token generation phase, Alice’s phone exchanges unique anonymous tokens with the devices of the different Bobs. Now, two weeks later, it is night, and the national research institute (RI) wants to run a simulation covering 14 days to see what effect closing all schools would have on the disease’s spread. To accomplish this, the RI notifies all registered participants’ applications to run a simulation using encounter data from the previous two weeks.

② - Simulation Initialization: The research institute RI initiates the simulation phase by sending a set of parameters, denoted by $\mathsf{param}_{\mathsf{sim}}$, to the participants in $\mathcal{P}$. The goal is to “spread" a fictitious infection across N_{sim} different simulation settings. To begin a simulation, each participant $\mathcal{P}_i$ is assigned to an infection class $I_i^{\mathsf{init}} \in \mathsf{class}_{\mathsf{inf}}$ (e.g., {S}usceptible, {E}xposed, {I}nfectious, {R}ecovered for the SEIR model) as specified in $\mathsf{param}_{\mathsf{sim}}$. For each individual simulation, $\mathsf{param}_{\mathsf{sim}}$ defines a set of containment measures, such as school closings and work from home, which the

participants will use as filters to carry out the simulation in the next stage.[5] In addition, RI publishes a formula to calculate the infection likelihood δ. The likelihood is determined by several parameters in the underlying modeling, such as encounter distance and time. For example, this likelihood might range from 0 (no chance of infection) to 100 (certain to get infected).

Running Example: Assume Alice is designated as infectious, while Bob_1 is designated as susceptible by RI. The other participants $\text{Bob}_2, \ldots, \text{Bob}_x$ are also assigned to an infection class (S, E, I, or R). To simulate containment measures, the RIPPLE-app now employs filters defined in $\text{param}_{\text{sim}}$. Using the information provided by the participants[6], the application may automatically filter out encounters that would not happen if a containment measure were in place, such as encounters in school while simulating school closings.

③ **- Simulation Execution:** Once the RI initialises the simulation, N_{step} simulation steps (steps ③a, ③b, ③c in Fig. 1) are performed for each of the N_{sim} simulation settings (e.g., $N_{\text{step}} = 14$ days). Without loss of generality, consider the first simulation step and let $N_{\text{sim}} = 1$. The simulation proceeds as follows:

1) Participant $\mathcal{P}_i \in \mathcal{P}$ filters out the relevant encounters based on the containment measures defined by RI. Let the corresponding encounter tokens be represented by the set $\mathcal{E}_i$.
2) For each token $r_e \in \mathcal{E}_i$, $\mathcal{P}_i$ computes the infection likelihood $\delta_i^{r_e}$ using the formula from RI, i.e., the probability that $\mathcal{P}_i$ infects the respective participant he met during the encounter with identifier token r_e.
3) Participants use the likelihood values δ obtained in the previous step to execute an ideal functionality called $\mathcal{F}_{\text{esim}}$, which allows them to communicate the δ values anonymously through a set of MPC servers $\mathcal{C}$. Furthermore, it allows each participant $\mathcal{P}_j$ to receive a cumulative infection likelihood, denoted by Δ_j, based on all of the encounters they had on the day being simulated, i.e., $\Delta_j = \sum_{r_e \in \mathcal{E}_j} \hat{\delta}_j^{r_e}$. In this case, $\hat{\delta}_j^{r_e}$ denotes the infection likelihood computed by participant $\mathcal{P}_f$ and communicated to $\mathcal{P}_j$ for an encounter between $\mathcal{P}_f$ and $\mathcal{P}_j$ with identifier token r_e. As will be discussed later in §3.3, $\mathcal{F}_{\text{esim}}$ must output the cumulative result rather than individual infection likelihoods because the latter can result in a breach of privacy.
4) Following the guidelines set by the RI, $\mathcal{P}_j$ updates its infection class I_j using the cumulative infection likelihood Δ_j acquired in the previous step.

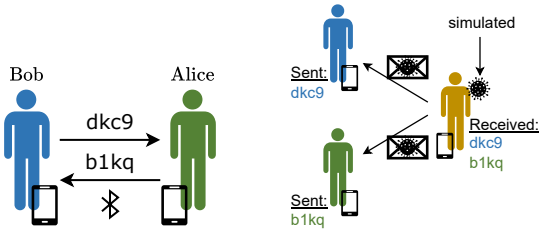
These steps above are repeated for each of the N_{step} simulation steps in order and across all the N_{sim} simulation settings.

Running Example: Let the simulated containment measure be the closure of schools. As Alice is simulated to be infectious, Alice's phone computes the infection likelihood for every single encounter it recorded on the day exactly two weeks ago (Day 1) *except* those that occurred at her daughter's school. Following that, Alice's phone combines the computed likelihood of each encounter with the corresponding unique encounter token to form tuples, which are then sent to the servers instantiating the anonymous communication channel. Using the encounter token as an address, the servers anonymously forward the likelihood to the person Alice has met, for example, Bob_1 (cf. Fig. 3b). Likewise, Bob_1 receives one message from each of the other participants he encountered and obtains the corresponding likelihood information. Bob_1 aggregates all likelihoods he obtained from his encounters on Day 1 and checks the aggregated result to a threshold defined by the RI to see if he has been infected in the simulation[7].

[5] Note, that potential alternatives (e.g., visiting a bar after restaurant closings) are not covered in this model. [6] This may also include location data obtained from the mobile app., e.g., Check In and Journal fields in the Corona-Warn contact tracing app. [7] Bob_1 obtains only the aggregated likelihood in the actual protocol.

④ - **Result Aggregation:** For a given simulation setting, each participant $\mathcal{P}_i \in \mathcal{P}$ will have its infection class I_i^s updated at the end of every simulation step $s \in [N_{\text{step}}]$. The goal of this phase is to allow RI to obtain the aggregated number of participants per class (e.g., #S, #E, #I, #R) for each simulated time step. For this, we rely on a *Secure Aggregation* functionality, denoted by $\mathcal{F}_{\text{agg}}$, which takes a N_{inf}-tuple of the form $\{v_i^1, \ldots, v_i^{N_{\text{inf}}}\}^s$ from each participant for every simulation step s and outputs the aggregate of this tuple over all the p participants to RI. In this case, v_i^k is an indicator variable for the k-th infection class, which is set to one if $I_i^s = \text{class}_{\text{inf}}^k$ and zero otherwise. Secure aggregation [50, 81, 86, 86] is a well-studied building block in cryptography these days, particularly in the context of federated learning, and there are numerous solutions proposed for various settings, such as using TEEs, a semi-trusted server aggregating ciphertexts under homomorphic encryption, or multiple non-colluding servers that aggregate secret shares. In this work, we consider $\mathcal{F}_{\text{agg}}$ to be a black box that can be instantiated using multiple existing solutions.

Running Example: All participants will know their updated infection class at the end of Day 1's simulation round, and they will prepare a 4-tuple of the form $\{v^S, v^E, v^I, v^R\}$ representing their updated infection class in the SEIR model. Participants will then engage in a secure aggregation protocol that determines the number of participants assigned to each infection class, which is then delivered to the RI. Then, the second simulation round begins, which replicates the procedure but this time using encounters from 13 days ago, i.e., Day 2. The RI obtains the aggregated number of participants for each of the simulated 14 days, i.e., a simulation of how the disease would spread if all schools had been closed in the previous 14 days (cf. graph in Fig. 1).



(a) Token Generation (b) Simulation

Fig. 3. Token Generation and Simulation phases in RIPPLE.

3.3 Privacy Requirements

Keeping the contact graph private requires that the participants remain unaware of any unconscious interactions. This means they cannot find out if they had unconscious contact with the same person more than once or how often they did. We remark that an insecure variant of RIPPLE, in which each participant $\mathcal{P}_i$ receives the infection likelihood $\hat{\delta}_i^e$ for all of its encounters $e \in E_i$ separately (instead of the aggregation of all), will not meet this requirement as described next.

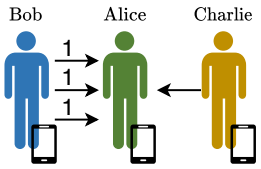


Fig. 4. Linking Identities Attack. Alice and Bob had several encounters, but Alice and Charlie only had one.

Linking Identities Attacks. To demonstrate this, observe that when running multiple simulations (with different simulation parameters $\text{param}_{\text{sim}}$) on the same day, participants will use the same encounter tokens and metadata from the token generation phase in each simulation. If a participant $\mathcal{P}_i$ (Alice) can see the infection likelihood $\hat{\delta}_i$ of each of her interactions separately, $\mathcal{P}_i$ can look for correlations between those likelihoods to see if another participant $\mathcal{P}_j$ (Bob) was encountered more than once. We call this a *Linking Identities Attack* and depict it in Fig. 4, where, for simplicity, the infection likelihood accepts just two values: 1 is a high infection likelihood and 0 is a low one.

Consider the following scenario to help clarify the issue: Alice and Bob work together in the same office. As a result, they have numerous conscious encounters during working hours. However, in their spare time, they may be unaware that they are in the same location (e.g., a club) and may not want the other to know. Even if they do not see each other, their phones constantly collect encounters. Assume the RI sent the participants a very simple infection likelihood formula that simply returns 0 (not infected) or 1 (infected). Furthermore, since the data is symmetric, both Alice and Bob have the same metadata (duration, distance, etc.) about their conscious and unconscious encounters. Let Bob be modelled as infectious in the first simulation. As a result, he will send a 1 for each (conscious and unconscious) encounter he had (including those with Alice). If multiple simulations are run on the same day (i.e., with the same encounters), Alice will notice that some encounters, specifically all conscious and unconscious encounters with Bob, always have the same infection likelihood: If Bob is not infectious, all will return a 0; if Bob is infectious, all will return a 1. Thus, even if Alice had unconscious encounters with Bob, she can detect the correlations between the encounters and, as a result, determine which unconscious encounters were most likely with Bob.

The more simulations she runs, the more confident she becomes because the infection state for multiple simulations is a unique fingerprint. Since every participant knows the formula, this attack can be extended to complex infection likelihood functions as well. While it may be more computationally expensive than the simple case, Alice is still able to identify correlations. This attack also works even if all of the encounters were unconscious. In such situations, Alice may not be able to trace related encounters to a single person (Bob), but she can infer that they were all with the same person (which is more than learning nothing). To avoid a Linking Identities attack, RIPPLE ensures that in a simulation phase, each participant receives just an aggregation of all infection likelihoods of their encounters. It cannot be avoided that participants can understand that when "getting infected" someone of their contacts must have been in contact with a (simulated) infectious participant. As this is only a simulated infection, we consider this leakage acceptable.

Sybil Attack. While the Linking Identities Attack is already possible for semi-honest adversaries, malicious participants may go even further to circumvent the aggregation mechanism that prevents access to individual infection likelihoods. They could, for example, construct many *sybils*, i.e., multiple identities using several mobile devices, to collect each encounter one by one and then conduct a Linking Identities Attack with the information.

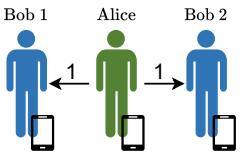


Fig. 5. Sybil Attack.

A registration system can be used to increase the costs of performing sybil attacks, i.e., to prevent an adversary from creating many identities. This assures that only legitimate users are allowed to join and participate in the simulation. In a closed ecosystem, such as a cpmpany, this can be achieved by letting each member receive exactly one token to participate in the simulation. On a larger scale at the national level, one can let each citizen receive a token linked to a digital ID card. In such authentication mechanisms, anonymous credentials (cf. §A) can be used to ensure anonymity.

Inference Attacks. Note that although RIPPLE mimics the spirit of Federated Learning (FL) [93], it is not susceptible to so-called inference attacks [52, 99] in the same sense as FL. First of all, RIPPLE only reveals the final output (to a research institute RI) and no individual updates/results that ease information extraction. However, the analysis results provided to RI (cf. §3.1) contain information about the spread of the modeled disease in a specific population (otherwise it would be meaningless to run the simulation). The ideal functionality does not cover leakage from the final output but protects privacy during the computation. Thus, anything that might be inferred from the output is not considered

in our security model. We argue that it is in the public interest to provide such aggregated information to the RI for deciding upon effective containment measures against infectious diseases.

4 INSTANTIATING $\mathcal{F}_{\text{esim}}$

In this section, we propose two instantiations of $\mathcal{F}_{\text{esim}}$ that cover different use cases and offer different trust-efficiency trade-offs. Our first design, RIPPLE$_{\text{TEE}}$ (§4.1), assumes the presence of trusted execution environments (TEEs) such as ARM TrustZone on the mobile devices of the participants. In our second design, RIPPLE$_{\text{PIR}}$ (§4.2), we eliminate this assumption and provide a software only solution using cryptographic techniques such as private information retrieval.

4.1 RIPPLE$_{\text{TEE}}$

The deployment of the entire operation in a single designated TEE would be a simple solution to achieving the ideal functionality $\mathcal{F}_{\text{esim}}$. However, given the massive amount of data that must be handled in a large-scale simulation with potentially millions of users, TEE resource limitations are a prohibitive factor. Furthermore, since the TEE would contain the entire population's contact graph, it would be a single point of failure and an appealing target for an attack on TEE's known vulnerabilities. RIPPLE$_{\text{TEE}}$ (Fig. 6), on the other hand, leverages the presence of TEEs in participants' mobile devices but in a decentralised manner, ensuring that each TEE handles only information related to the encounters made by the respective participant.

Before going into the details of RIPPLE$_{\text{TEE}}$, we will go over the $\mathcal{F}_{\text{anon}}$ functionality (cf. §B.3) that we will use in our instantiation. We define it as follows: $\mathcal{F}_{\text{anon}}$ allows two participants, $\mathcal{P}_i$ and $\mathcal{P}_j$, to send messages to each other anonymously via a set of communication servers $\mathcal{C}$. The set $\mathcal{C}$ consists of one server acting as an entry node ($\mathcal{N}_{\texttt{entry}}$), receiving messages from senders, and one server acting as an exit node ($\mathcal{N}_{\texttt{exit}}$), forwarding messages to receivers. In $\mathcal{F}_{\text{anon}}$, sender $\mathcal{P}_i$ does not learn to whom the message is sent, and receiver $\mathcal{P}_j$ does not learn who sent it. Similarly, the servers in $\mathcal{C}$ will be unable to link receiver and sender of a message. Anonymous communication (cf. §A) is an active research area, e.g., [1, 5, 51, 63], and $\mathcal{F}_{\text{anon}}$ in RIPPLE$_{\text{TEE}}$ can be instantiated using any of these efficient techniques.

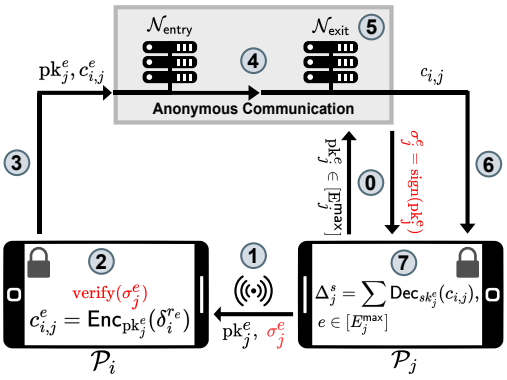


Fig. 6. RIPPLE$_{\text{TEE}}$ Overview. Messages in red denote additional steps needed for malicious participants.

4.1.1 *The RIPPLE$_{\text{TEE}}$ Protocol.*

Token Generation (steps ⓪ to ① in Fig. 6): During the pre-computation phase, the TEE of each participant $\mathcal{P}_i \in \mathcal{P}$ generates a list of fresh unique public/private keys $(\text{pk}_i^e, \text{sk}_i^e)$ for all possible encounters $e \in [E_i^{\max}]$. The keys can be pre-generated and stored, e.g., on the day before. The newly generated public keys are then sent by $\mathcal{P}_i$'s TEE to the exit node $\mathcal{N}_{\texttt{exit}}$ (step ⓪ in Fig. 6) to enable anonymous communication (cf. §B.3) via $\mathcal{F}_{\text{anon}}$ later in the protocol's simulation part.

During a physical encounter e, $\mathcal{P}_i$ and $\mathcal{P}_j$ exchange two unused public keys pk_i^e and pk_j^e (step ① in Fig. 6). Simultaneously, both participants compute and record metadata m_e, such as the time, location, and duration of the encounter, and store this information alongside the received public key.

Additional measures are required for malicious participants to ensure that the participants are exchanging public keys generated by the TEEs: After obtaining the new public keys from $\mathcal{P}_i$, the exit node $\mathcal{N}_{\texttt{exit}}$ signs them and returns the signatures to $\mathcal{P}_i$ after checking that it is connecting directly with a non-corrupted TEE (step ⓪ in Fig. 6). During a physical encounter, $\mathcal{P}_j$ will provide the corresponding signature, denoted by σ_j^e along with pk_j^e so that the receiver $\mathcal{P}_i$ can verify that the key was correctly generated by $\mathcal{P}_j$'s TEE (step ② in Fig. 6).

Simulation Execution. (steps ② to ⑦ in Fig. 6): All local computations, including infection likelihood calculation and infection class updates, will be performed within the participants' TEEs. In detail, for each encounter e involving participants $\mathcal{P}_i$ and $\mathcal{P}_j$, the following steps are executed:

- $\mathcal{P}_i$'s TEE computes $\delta_i^{r_e}$ and encrypts it using the public key pk_j^e of $\mathcal{P}_j$ obtained during the token generation phase. Let the ciphertext be $c_{i,j}^e = \text{Enc}_{\text{pk}_j^e}(\delta_i^{r_e})$ (step ② in Fig. 6).
- $\mathcal{P}_i$'s TEE establishes a secure channel with the entry node $\mathcal{N}_{\texttt{entry}}$ of C via remote attestation and uploads the tuple $(\text{pk}_j^e, c_{i,j}^e)$ (step ③ in Fig. 6).
- The tuple $(\text{pk}_j^e, c_{i,j}^e)$ traverses through the servers in C and reaches the exit node $\mathcal{N}_{\texttt{exit}}$ (step ④ in Fig. 6, instantiation details for the anonymous communication channel are given in §B.3).
- If the public key pk_i^e has already been used in this simulation step[8], $\mathcal{N}_{\texttt{exit}}$ discards the tuple (step ⑤ in Fig. 6).
- Otherwise, $\mathcal{N}_{\texttt{exit}}$ uses pk_j^e to identify the recipient $\mathcal{P}_j$ and sends the ciphertext $c_{i,j}^e$ to $\mathcal{P}_j$ (step ⑥ in Fig. 6).

After receiving the ciphertexts for all of the encounters, $\mathcal{P}_j$'s TEE decrypts them and aggregates the likelihoods to produce the desired output (step ⑦ in Fig. 6).

4.1.2 Security of RIPPLE$_{\texttt{TEE}}$. First, we consider the case of semi-honest participants. During the token generation phase, since the current architecture in most mobile devices does not allow direct communication with a TEE while working with Bluetooth LE interfaces, participant $\mathcal{P}_i$ can access both the sent and received public keys before they are processed in the TEE. However, unique keys are generated per encounter and do not reveal anything about an encounter's identities due to the security of the underlying $\mathcal{F}_{\text{gen}}$ functionality, which captures the goal of several contact tracing apps in use.

The $\mathcal{F}_{\text{anon}}$ functionality, which implements an anonymous communication channel utilising the servers in C, aids in achieving *contact graph privacy* by preventing participants from learning to/from whom they are sending/receiving messages. While the entry node learns who sends messages to it, it does not learn who receives them. Similarly, the exit node $\mathcal{N}_{\texttt{exit}}$ has no knowledge of the sender but learns the recipient using the public key. Regarding *confidentiality*, participants in RIPPLE$_{\texttt{TEE}}$ have no knowledge of the messages being communicated because they cannot access the content of the TEEs and the TEEs communicate directly to the anonymous channel. Furthermore, servers in C will not have access to the messages as they are encrypted.

For the case of malicious participants, they could send specifically crafted keys during the token generation phase instead of the ones created by their TEE. However, this will make the signature verification fail and the encounter will get discarded. Furthermore, a malicious participant may reuse public keys for multiple encounters. This manipulation, however, will be useless because the exit node $\mathcal{N}_{\texttt{exit}}$ checks that each key is only used once before forwarding messages to participants. During the simulation phase, all data and computation are handled directly inside the TEEs of the participants, so no manipulation is possible other than cutting the network connection, i.e., dropping out of the simulation, ensuring *correctness*. Dropouts occur naturally when working with mobile devices and have no effect on privacy guarantees.

[8] This step is not required for semi-honest participants.

4.2 RIPPLE$_{\text{PIR}}$

In the following, we show how to get rid of RIPPLE$_{\text{TEE}}$'s assumption of each participant having a TEE on their mobile devices. If we simply remove the TEE part of RIPPLE$_{\text{TEE}}$ and run the same protocol, decryption and aggregation of a participant's received infection likelihoods would be under their control. Thus, the individual infection likelihoods of all encounters would be known to them, leaking information about the contact graph (cf. §3.3). To get around this privacy issue, we need to find another way to aggregate the infection likelihoods so that only the sum, not individual values, can be derived by the participants.

Private Information Retrieval (PIR, cf. §A) is one promising solution for allowing participants to retrieve infection likelihoods sent to them anonymously. PIR enables the private download of an item from a public database D held by M servers without leaking any information to the servers, such as which item is queried or the content of the queried item. However, classical PIR is unsuitable for our needs because we need to retrieve the sum of τ items from the database rather than the individual ones. As a result, we introduce the ideal functionality $\mathcal{F}_{\text{pirsum}}$ (Fig. 7), which is similar to a conventional PIR functionality but returns the sum of τ queried locations of the database as a result. For the remainder of this section, we consider $\mathcal{F}_{\text{pirsum}}$ to be an ideal black-box and will discuss concrete instantiations in §5.

Functionality $\mathcal{F}_{\text{pirsum}}$

$\mathcal{F}_{\text{pirsum}}$ interacts with M servers, denoted by $\mathcal{C}$, and participant $\mathcal{P}_i \in \mathcal{P}$.

Input: $\mathcal{F}_{\text{pirsum}}$ receives τ indices denoted by $Q = \{q_1, \ldots, q_\tau\}$ from $\mathcal{P}_i$ and a database D from $\mathcal{C}$.

Output: $\mathcal{F}_{\text{pirsum}}$ sends $\sum_{j=1}^{\tau} \mathsf{D}[q_j]$ to $\mathcal{P}_i$ as the output.

Fig. 7. Ideal functionality for PIR-SUM (semi-honest).

4.2.1 *The RIPPLE$_{\text{PIR}}$ Protocol.*

Token Generation (step ① in Fig. 8): During a physical encounter e among participants $\mathcal{P}_i$ and $\mathcal{P}_j$, they generate and exchange unique κ-bit random tokens denoted by r_i^e and r_j^e. Both participants, like in RIPPLE$_{\text{TEE}}$, record the metadata m^e as well. Thus, at the end of a simulation step $\mathsf{s} \in [N_{\text{step}}]$ (e.g., a day), $\mathcal{P}_i$ holds a list of sent encounter tokens $E_i^{\mathsf{s}} = \{r_i^e\}_{e \in \mathcal{E}_i}$, where $\mathcal{E}_i$ is the complete (sent/received) set of encounters of $\mathcal{P}_i$, and a list of received tokens, denoted by $R_i^{\mathsf{s}} = \{r_j^e\}_{e \in \mathcal{E}_i}$. Looking ahead, these random tokens will be used as addresses for communicating the corresponding infection likelihood among the participants.

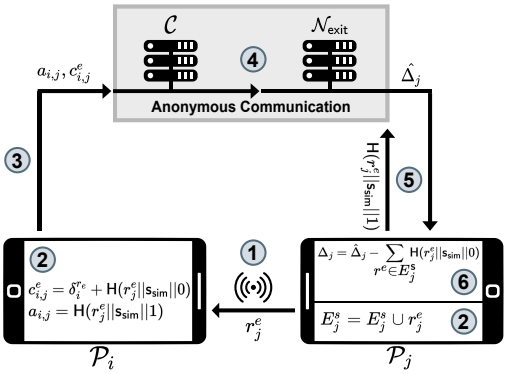


Fig. 8. RIPPLE$_{\text{PIR}}$ Overview.

Simulation Execution (steps ② to ⑥ in Fig. 8): For local computations like encounter filtering and infection likelihood calculation, the steps for an encounter e between $\mathcal{P}_i$ and $\mathcal{P}_j$ are:

- $\mathcal{P}_i$ blinds each $\delta_i^{r_e}$ computed with the corresponding random token r_j^e received from $\mathcal{P}_j$ and obtains the ciphertext $c_{i,j}^e = \delta_i^{r_e} + \mathsf{H}(r_j^e||\mathsf{s}_{\text{sim}}||0) \bmod 2^l$. Further, it computes the destination address for the ciphertext as $a_{i,j} = \mathsf{H}(r_j^e||\mathsf{s}_{\text{sim}}||1)$. Here, H() is a cryptographic hash function and $\mathsf{s}_{\text{sim}} \in [N_{\text{sim}}]$ denotes the current simulation setting. (step ② in Fig. 8) To prevent the exit node $\mathcal{N}_{\texttt{exit}}$ from linking messages from different simulations, s_{sim} is utilized in H() to generate unique (ciphertext, address) tuples for the same encounters across multiple simulation settings.

• $\mathcal{P}_i$ sends the tuple $(c^e_{i,j}, a_{i,j})$ anonymously to $\mathcal{N}_{\texttt{exit}}$ with the help of the servers in $\mathcal{C}$. $\mathcal{N}_{\texttt{exit}}$ discards all the tuples with the same address field $(a_{i,j})$. (step ③ to ④ in Fig. 8, instantiation details for the anonymous communication channel are given in §B.3)

As a server in $\mathcal{C}$, $\mathcal{N}_{\texttt{exit}}$ locally creates the database D for the current simulation step using all of the $(a_{i,j}, c^e_{i,j})$ tuples received (part of step ④ in Fig. 8). A naive solution of inserting $c^e_{i,j}$ using a simple hashing of the address $a_{i,j}$ will not provide an efficient solution in our case since we require only one message to be stored in each database entry to have an injective mapping between addresses and messages. This is required for the message receiver to precisely download the messages that were sent to them. Using simple hashing, this would translate to a large database size to ensure a negligible probability of collisions. Instead, in RIPPLE$_{\mathsf{PIR}}$, we use a novel variant of a garbled cuckoo table that we call arithmetic garbled cuckoo table (AGCT, cf. §4.2.3), with $a_{i,j}$ as the insertion key for the database.

Once the database D is created, $\mathcal{N}_{\texttt{exit}}$ sends it to the other servers in $\mathcal{C}$ based on the instantiation of $\mathcal{F}_{\mathsf{pirsum}}$ (cf. §5). Each $\mathcal{P}_j \in \mathcal{P}$ will then participate in an instance of $\mathcal{F}_{\mathsf{pirsum}}$ with the PIR servers $\mathcal{C}$ sharing a database D. $\mathcal{P}_j$ uses the addresses of all its sent encounters from E^{s}_j, namely $\mathsf{H}(r^e||\mathsf{s}_{\mathsf{sim}}||1)$, as the input to $\mathcal{F}_{\mathsf{pirsum}}$ and obtains the blinded cumulative infection likelihood, denoted by $\hat{\Delta}_j$, as the output (step ⑤ in Fig. 8). The cumulative infection likelihood, Δ_j, is then unblinded as $\Delta_j = \hat{\Delta}_j - \sum_{r^e \in E^{\mathsf{s}}_j} \mathsf{H}(r^e||\mathsf{s}_{\mathsf{sim}}||0) \bmod 2^l$ concluding the current simulation step (step ⑥ in Fig. 8).

4.2.2 Security of RIPPLE$_{\mathsf{PIR}}$. Except for the database constructions at exit node $\mathcal{N}_{\texttt{exit}}$ and the subsequent invocation of the $\mathcal{F}_{\mathsf{pirsum}}$ functionality for the cumulative infection likelihood computation, the security guarantees for semi-honest participants in RIPPLE$_{\mathsf{PIR}}$ are similar to those of RIPPLE$_{\mathsf{TEE}}$.

Unlike RIPPLE$_{\mathsf{TEE}}$, $\mathcal{N}_{\texttt{exit}}$ in RIPPLE$_{\mathsf{PIR}}$ cannot identify the message's destination from the address as it is only known by the receiving participant. Further, participants obtain their cumulative infection likelihood directly via the $\mathcal{F}_{\mathsf{pirsum}}$ functionality, ensuring that $\mathcal{N}_{\texttt{exit}}$ cannot infer the participant's encounter data and, thus, *contact graph privacy*.

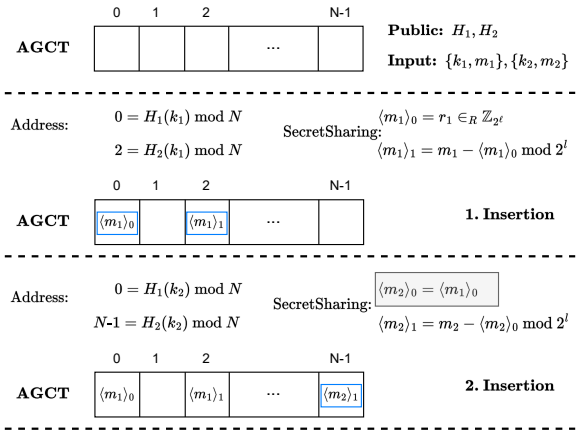


Fig. 9. Insertion into the Arithmetic Garbled Cuckoo Table (AGCT). H_1 and H_2 are two hash functions. $\{k_1, m_1\}$ and $\{k_2, m_2\}$ are key-value pairs where the key is used to determine the address of the data in the database.

Malicious participants in RIPPLE$_{\mathsf{PIR}}$, as opposed to RIPPLE$_{\mathsf{TEE}}$, can tamper with the protocol's correctness by providing incorrect inputs. However, as stated in the threat model in §3, we assume that malicious participants in our framework will not tamper with the correctness and will only try to learn additional information. A malicious participant could re-use the same encounter token for multiple encounters during token generation which would result in multiple tuples with the same address. However, as stated in the protocol, $\mathcal{N}_{\texttt{exit}}$ will discard all such tuples, effectively removing the malicious participant from the system. Another potential information leakage caused by a participant re-using encounter tokens is that the entry point of the anonymous communication channel will be able to deduce that multiple participants, say $\mathcal{P}_i$ and $\mathcal{P}_j$, had an encounter with the same participant. This is not an issue in our protocol because we

instantiate the $\mathcal{F}_{\mathsf{anon}}$ functionality using a 3-server oblivious shuffling scheme (cf. §B.3), where all the servers except $\mathcal{N}_{\texttt{exit}}$ will not see any messages in the clear, but only see secret shares.

4.2.3 Arithmetic Garbled Cuckoo Table (AGCT). We design a variant of garbled cuckoo tables ([109], cf. §A) that we term arithmetic garbled cuckoo table (AGCT) to reduce the size of the PIR database while ensuring a negligible collision probability. It uses arithmetic sharing instead of XOR-sharing to share database entries and the details are presented next.

Let's assume two key-message pairs $\{k_1, m_1\}$ and $\{k_2, m_2\}$[9] shall be added to database D with N bins and two hash function H_1 and H_2 to determine the insertion addresses. The insertion process works as follows:

1. Insertion of $\{k_1, m_1\}$:
   a) Compute $a_1 = \mathsf{H}_1(k_1) \bmod N$ and $a_2 = \mathsf{H}_2(k_1) \bmod N$.
   b) Check if bins a_1, a_2 are already occupied. Let's assume this is not the case.
   c) Compute the arithmetic sharing of the message m_1: $\langle m_1 \rangle_0 = r_1 \in_R \mathbb{Z}_{2^\ell}$ and $\langle m_1 \rangle_1 = m_1 - \langle m_1 \rangle_0 \bmod 2^\ell$.
   d) Insert $D[a_1] = \langle m_1 \rangle_0$, $D[a_2] = \langle m_1 \rangle_1$.
2. Insertion of $\{k_2, m_2\}$:
   a) Compute $b_1 = \mathsf{H}_1(k_2) \bmod N$ and $b_2 = \mathsf{H}_2(k_2) \bmod N$.
   b) Check if bins b_1 and b_2 are already occupied. Let's assume $b_1 = a_1$, i.e., the first bin is already occupied, but bin b_2 is free.
   c) Compute the arithmetic sharing m_2 with $\langle m_2 \rangle_0 = \langle m_1 \rangle_0$ as $b_1 = a_1$. Then, the other share is $\langle m_2 \rangle_1 = m_2 - \langle m_2 \rangle_0 \bmod 2^\ell$.
   d) Insert $D[b_1] = \langle m_2 \rangle_0$ and $D[b_2] = \langle m_2 \rangle_1$.

Double Collision: Now the question is how to handle the insertion of a database entry if both addresses determined by the two hash functions are already occupied. An easy solution is to pick different hash functions s.t. no double collision occurs for all n elements that shall be stored in the database. Alternatively, Pinkas et al. [109] demonstrate for a garbled cuckoo table how to extend the database by $d + \lambda$ bins, where d is the upper bound of double collisions and λ is an error parameter, such that double collisions occur with a negligible likelihood. For details, please refer to [109, §5].

5 PIR-SUM: INSTANTIATING $\mathcal{F}_{\mathsf{pirsum}}$

So far, the discussion has focused on RIPPLE as a generic framework composed of multiple ideal functionalities that could be efficiently instantiated using state-of-the-art privacy-enhancing technologies. In this section, we will concentrate on instantiating our novel $\mathcal{F}_{\mathsf{pirsum}}$ functionality (Fig. 7) using three semi-honest MPC servers. In particular, we have three servers S_0, S_1, and S_2, and we design the $\mathsf{PIR}_{\mathsf{sum}}$ protocol to instantiate the $\mathcal{F}_{\mathsf{pirsum}}$ functionality.

The problem statement in our context is formally defined as follows: Participant $\mathcal{P}_i \in \mathcal{P}$ has a set of τ indices denoted by $Q = \{q_1, \ldots, q_\tau\}$ and wants to retrieve $\mathsf{res} = \sum_{q \in Q} \mathsf{D}[q] \bmod 2^\ell$. In this case, D is a database with N elements of ℓ-bits each that is held in the clear by both the servers S_1 and S_2. The server S_0 aids in the computation performed by the servers S_1 and S_2. Furthermore, we assume a one-time setup (cf. §B.1) among the servers and $\mathcal{P}_i$ that establishes shared pseudorandom keys among them to facilitate non-interactive generation of random values and, thus, save communication [9, 26, 105].

[9] k corresponds to a key and m to a message in our application.

5.1 Overview of $\mathsf{PIR}_{\mathsf{sum}}$ protocol

At a high level, the idea is to use multiple instances of a standard 2-server PIR functionality [18, 33], denoted by $\mathcal{F}_{\mathsf{pir}}^{\mathsf{2S}}$, and combine the responses to get the sum of the desired blocks as the output. $\mathsf{D}^m = \mathsf{D} + m \bmod 2^\ell$ denotes a modified version of the database D in which every block is summed with the same ℓ-bit mask m, i.e., $\mathsf{D}^m[i] = \mathsf{D}[i] + m$ for $i \in [N]$. The protocol proceeds as follows:

- S_1 and S_2 non-interactively sample τ random mask values $\{m_1, \ldots, m_\tau\}$ such that $\sum_{j=1}^{\tau} m_j = 0$.[10]
- $\mathsf{S}_1, \mathsf{S}_2$, and $\mathcal{P}_i$ execute τ instances of $\mathcal{F}_{\mathsf{pir}}^{\mathsf{2S}}$ in parallel, with servers using D^{m_j} as the database and $\mathcal{P}_i$ using q_j as the query for the j-th instance for $j \in [\tau]$. The result obtained by $\mathcal{P}_i$ from the j-th $\mathcal{F}_{\mathsf{pir}}^{\mathsf{2S}}$ instance is denoted by res_j.
- $\mathcal{P}_i$ locally computes $\sum_{j=1}^{\tau} \mathsf{res}_j$ to obtain the desired result.

The details for instantiating $\mathcal{F}_{\mathsf{pir}}^{\mathsf{2S}}$ using the standard linear summation PIR approach [33] are provided in §C. The approach requires $\mathcal{P}_i$ to communicate $N \cdot \tau$ bits to the servers, which is further reduced in $\mathsf{RIPPLE}_{\mathsf{PIR}}$ as shown in §5.3.

Malicious participants. Recall from our threat model (cf. §3.1) that a malicious participant may deviate from the protocol to gain additional knowledge but does not try to harm correctness. For example, it could use the same query, say q_j, in all τ instances and retrieve only the block corresponding to q_j by dividing the result by τ. We use a simple verification scheme over the $\mathcal{F}_{\mathsf{pir}}^{\mathsf{2S}}$ functionality to prevent these manipulations. Its details are presented next.

For malicious participants, we want to ensure that $\mathcal{P}_i$ used a distinct vector $\vec{\mathsf{b}}$ (representing a PIR query q_j, cf. §C) during the τ parallel instances. One naive approach is to have S_1 and S_2 compute the bitwise-OR of all the τ bit query vectors $\vec{\mathsf{b}}_1, \ldots, \vec{\mathsf{b}}_\tau$, and then run a secure two-party computation protocol to compare the number of ones in the resultant vector to τ. We use the additional server S_0 to further optimize this step. S_1 and S_2 send randomly shuffled versions of their secret shared bit vectors to S_0, who reconstructs the shuffled vectors and performs the verification locally. This approach leaks no information to S_0 because it has no information about the underlying database D. The verification procedure is as follows:

- S_1 and S_2 non-interactively agree on a random permutation, denoted by π.
- S_u sends $\pi([\vec{\mathsf{b}}_j]_u)$ to S_0 for $j \in [\tau]$ and $u \in \{1, 2\}$.
- S_0 locally reconstructs $\pi(\vec{\mathsf{b}}_j) = \pi([\vec{\mathsf{b}}_j]_1) \oplus \pi([\vec{\mathsf{b}}_j]_2)$, for $j \in [\tau]$. If all the τ bit vectors are correctly formed and distinct, it sends `Accept` to S_1 and S_2. Else, it sends `abort`.

Note that the verification using $\mathcal{P}_0$ will incur a communication of $2\tau N$ bits among the servers. Furthermore, the above verification method can be applied to any instantiation of $\mathcal{F}_{\mathsf{pir}}^{\mathsf{2S}}$ that generates a boolean sharing of the query bit vector among the PIR servers and computes the response as described above, e.g., the PIR schemes of [17, 18, 33].

5.2 Instantiating $\mathcal{F}_{\mathsf{pirsum}}$

The formal protocol for $\mathsf{PIR}_{\mathsf{sum}}$ in the case of malicious participants is provided in Fig. 10 and is based on a variant of the standard 2-server PIR functionality $\mathcal{F}_{\mathsf{pir}}^{\mathsf{2S}}$ (as will be discussed in $\mathbf{HYB}_2$ below). In $\mathsf{PIR}_{\mathsf{sum}}$, the servers $\mathsf{S}_1, \mathsf{S}_2$ and the participant $\mathcal{P}_i$ run τ instances of $\mathcal{F}_{\mathsf{pir}}^{\mathsf{2S}}$ in parallel, one for each query $q \in \mathcal{Q}$. Following the execution, $\mathcal{P}_i$ receives $\mathsf{D}[q] + \mathsf{r}_q$ whereas S_u receives $\mathsf{r}_q, [q]_u$, for $u \in \{1, 2\}$ and $q \in \mathcal{Q}$. $\mathcal{P}_i$ then adds up the received messages to get a masked version of the desired output, i.e, $\sum_{q \in \mathcal{Q}} \mathsf{D}[q] + \mathsf{mask}_{\mathcal{Q}}$ with $\mathsf{mask}_{\mathcal{Q}} = \sum_{q \in \mathcal{Q}} \mathsf{r}_q$. $\mathsf{S}_1, \mathsf{S}_2$ compute $\mathsf{mask}_{\mathcal{Q}}$ in the same way.

[10] These masks are sampled for each participant.

Protocol $\mathsf{PIR}_{\mathsf{sum}}$

Input(s): i) $\mathsf{S}_1, \mathsf{S}_2 : \mathsf{D}; |\mathsf{D}| = N$, ii) $\mathcal{P}_i : Q = \{q_1, \ldots, q_\tau\}$, and iii) $\mathsf{S}_0 : \perp$.
Output: $\mathcal{P}_i$: res $= \sum_{q \in Q} \mathsf{D}[q]$ for distinct queries, else res $= \perp$.

Computation

1. For each $q \in Q$,
   a. $\mathsf{S}_1, \mathsf{S}_2$ and $\mathcal{P}_i$ invoke $\mathcal{F}_{\mathsf{pir}}^{\mathsf{2S}}$ (cf. $\mathbf{HYB}_2$ in proof of Lemma 5.1) with the inputs D, q.
   b. Let $\mathsf{r}_q, [q]_u$ denote the output of S_u, for $u \in \{1, 2\}$ and $\mathsf{D}[q] + \mathsf{r}_q$ denote the output of $\mathcal{P}_i$.
2. $\mathcal{P}_i$ computes res$' = \sum_{q \in Q}(\mathsf{D}[q] + \mathsf{r}_q)$, while $\mathsf{S}_1, \mathsf{S}_2$ computes $\mathsf{mask}_Q = \sum_{q \in Q} \mathsf{r}_q$.
3. $\mathsf{S}_1, \mathsf{S}_2$ and S_0 invokes $\mathcal{F}_{\mathsf{vrfy}}$ on the secret shares of queries, denoted by $\{[q]_u\}_{q \in Q, u \in \{1,2\}}$, to check the distinctness of the queries in Q.
4. If $\mathcal{F}_{\mathsf{vrfy}}$ returns `Accept`, $\mathsf{S}_1, \mathsf{S}_2$ sends mask_Q to $\mathcal{P}_i$, who computes res = res$' - \mathsf{mask}_Q$. Otherwise, `abort`.

Fig. 10. $\mathsf{PIR}_{\mathsf{sum}}$ Protocol.

The protocol could be completed by S_1 and S_2 sending mask_Q to $\mathcal{P}_i$, then $\mathcal{P}_i$ unmasking its value to obtain the desired output. However, before communicating the mask, the servers must ensure that all queries in Q are distinct, as shown in $\mathcal{F}_{\mathsf{pirsum}}$ (Fig. 11). For this, $\mathsf{S}_1, \mathsf{S}_2$ use their share of the queries $q \in Q$ and participate in a secure computation protocol with S_0. We capture this with an ideal functionality $\mathcal{F}_{\mathsf{vrfy}}$, which takes the secret shares of τ values from S_1 and S_2 and returns `Accept` to the servers if all of the underlying secrets are distinct. Otherwise, it returns `abort`.

5.2.1 Security of $\mathsf{PIR}_{\mathsf{sum}}$ *Protocol.* Fig. 11 presents the ideal functionality for $\mathsf{PIR}_{\mathsf{sum}}$ in the context of malicious participants. In this case, $\mathcal{F}_{\mathsf{pirsum}}$ first checks whether all the queries made by the participant $\mathcal{P}_i$ are distinct. If yes, the correct result is sent to $\mathcal{P}_i$; otherwise, $\perp$ is sent to $\mathcal{P}_i$.

Functionality $\mathcal{F}_{\mathsf{pirsum}}$

$\mathcal{F}_{\mathsf{pirsum}}$ interacts with servers in C, and participant $\mathcal{P}_i \in \mathcal{P}$.
Input: $\mathcal{F}_{\mathsf{pirsum}}$ receives τ indices denoted by $Q = \{q_1, \ldots, q_\tau\}$ from $\mathcal{P}_i$ and a database D from C.
Computation: $\mathcal{F}_{\mathsf{pirsum}}$ sets $y = \sum_{j=1}^{\tau} \mathsf{D}[q_j]$ if all the queries in Q are distinct. Else, it sets $y = \perp$.
Output: $\mathcal{F}_{\mathsf{pirsum}}$ sends y to $\mathcal{P}_i$.

Fig. 11. PIR-SUM functionality (malicious participants).

Lemma 5.1. *Protocol* $\mathsf{PIR}_{\mathsf{sum}}$ *(Fig. 10) securely realises the* $\mathcal{F}_{\mathsf{pirsum}}$ *ideal functionality (Fig. 11) for the case of malicious participants in the* $\{\mathcal{F}_{\mathsf{pir}}^{\mathsf{2S}}, \mathcal{F}_{\mathsf{vrfy}}\}$*-hybrid model.*

Proof. The proof follows with a hybrid argument based on the three hybrids $\mathbf{HYB}_0$, $\mathbf{HYB}_1$, and $\mathbf{HYB}_2$ discussed below. Furthermore, any secure three-party protocol can be used to instantiate $\mathcal{F}_{\mathsf{vrfy}}$ in RIPPLE.

We use a standard 2-server PIR functionality, denoted by $\mathcal{F}_{\mathsf{pir}}^{\mathsf{2S}}$, to instantiate $\mathcal{F}_{\mathsf{pirsum}}$. The guarantees of $\mathcal{F}_{\mathsf{pir}}^{\mathsf{2S}}$, however, are insufficient to meet the security requirements of $\mathcal{F}_{\mathsf{pirsum}}$, so we modify $\mathcal{F}_{\mathsf{pir}}^{\mathsf{2S}}$ as a sequence of hybrids, denoted by **HYB**: The modification is carried out in such a way that for a malicious participant $\mathcal{P}_i$, each hybrid is computationally indistinguishable from the one before it. $\mathcal{F}_{\mathsf{pir}}^{\mathsf{2S}}$ is equal to the first hybrid $\mathbf{HYB}_0$. We use the hybrid $\mathbf{HYB}_2$ instead of $\mathcal{F}_{\mathsf{pir}}^{\mathsf{2S}}$, and we omit introducing a different notation for the same for simplicity.

HYB$_0$: Let $\mathcal{F}_{\text{pir}}^{2S}$ denote a 2-server PIR ideal functionality for our case, with servers S_1 and S_2 acting as database holders and $\mathcal{P}_i$ acting as the client. For a database D held by S_1 and S_2 and a query q held by $\mathcal{P}_i$, $\mathcal{F}_{\text{pir}}^{2S}$ returns $D[q]$ to $\mathcal{P}_i$, but S_1 and S_2 receive nothing.

HYB$_1$: We modify $\mathcal{F}_{\text{pir}}^{2S}$ so that it returns $D[q] + r$ to $\mathcal{P}_i$, and S_1, S_2 receive r, where r is a random value from the domain of database block size, such that addition of r to the database blocks respects the underlying distribution. In other words, the modification can be thought of as the standard $\mathcal{F}_{\text{pir}}^{2S}$ being executed over a database $D^r = D + r$ rather than the actual database D. This modification leaks no additional information regarding the query to the servers because they will receive random masks that are independent of the query q. Furthermore, from the perspective of $\mathcal{P}_i$ with no prior knowledge of the database D, **HYB**$_1$ will be indistinguishable from **HYB**$_0$ because the values it sees in both cases are from the same distribution. As a result, **HYB**$_0 \approx$ **HYB**$_1$.

HYB$_2$: Looking ahead, in PIR_{sum}, the servers S_1, S_2 and the participant $\mathcal{P}_i$ run τ instances of $\mathcal{F}_{\text{pir}}^{2S}$ in parallel, one for each query $q \in \mathcal{Q}$. As shown in $\mathcal{F}_{\text{pirsum}}$ (Fig. 11), the servers must ensure that all of the queries in $\mathcal{Q}$ are distinct. For this, we modify $\mathcal{F}_{\text{pir}}^{2S}$ in **HYB**$_1$ to additionally output a secret share of the query q to each of S_1 and S_2. Because the servers S_1 and S_2 are assumed to be non-colluding in our setting, this modification will leak no information about the query q to either server. Since the output to $\mathcal{P}_i$ remains unchanged, **HYB**$_1 \approx$ **HYB**$_2$ from $\mathcal{P}_i$'s perspective. □

5.3 Reducing participant's communication

PIR_{sum} in $\text{RIPPLE}_{\text{PIR}}$ can be implemented using two approaches with different trade-offs to minimize participants' communication and computation. $\text{PIR}_{\text{sum}}^{\text{I}}$ (Fig. 12) prioritizes low communication over computation, while $\text{PIR}_{\text{sum}}^{\text{II}}$ (Fig. 13) reduces both the computational and communication overhead of the participant by involving an additional server $S_0 \in \mathcal{C}$.

5.3.1 $\text{PIR}_{\text{sum}}^{\text{I}}$ *(Fig. 12).* In this approach, we instantiate $\mathcal{F}_{\text{pir}}^{2S}$ using PIR techniques based on Function Secret Sharing (FSS) [17, 18, 36]. To retrieve the q-th block from the database, $\mathcal{P}_i$ uses FSS on a Distributed Point Function (DPF) [58] that evaluates to a 1 only when the input q is 1 and to 0 otherwise. $\mathcal{P}_i$ generates two DPF keys k_1 and k_2 that satisfy the above constraint and sends one key to each of the servers S_1 and S_2. The servers S_1 and S_2 can then locally expand their key share to obtain their share for the bit vector $\vec{b}$ and the rest of the procedure proceeds similarly to the naive linear summation method discussed in §5.1 (more details on Linear Summation PIR are given in §C). The key size for a database with N blocks using the optimised DPF construction in [18] is about $\lambda \log_2(N/\lambda)$ bits, where $\lambda = 128$ for an AES-based implementation. Fig. 12 provides the formal details of the $\text{PIR}_{\text{sum}}^{\text{I}}$ protocol.

Protocol $\text{PIR}_{\text{sum}}^{\text{I}}$

Input(s): i) $S_1, S_2 : D; |D| = N$, ii) $\mathcal{P}_i : \mathcal{Q} = \{q_1, \ldots, q_\tau\}$, and iii) $S_0 : \perp$.
Output: $\mathcal{P}_i$: res $= \sum_{q \in \mathcal{Q}} D[q]$

Computation S_1 and S_2 sample τ random mask values $\{m_1, \ldots, m_\tau\} \in \mathbb{Z}_{2^\ell}^\tau$ such that $\sum_{j=1}^{\tau} m_j = 0$. For each $q \in \mathcal{Q}$, execute:

1. S_1, S_2 locally compute $D^{m_q} = D + m_q$.
2. Execute DPF protocol [18] (verifiable DPF for malicious participants) with $\mathcal{P}_i$ as client with input q. Server S_u obtains $[\vec{b}_q]_u$ with $b_q^j = 1$ for $j = q$ and $b_q^j = 0$ for $j \neq q$, for $u \in \{1, 2\}$.

Verification Let $\{\vec{b}_{q_1}, \ldots, \vec{b}_{q_\tau}\}$ denote the bit vectors whose XOR-shares are generated during the preceding steps.

3. Servers verify correctness of q_j, $j \in [\tau]$, by executing the Ver algorithm of the verifiable DPF protocol [18] (cf. §B.4). It outputs `Accept` to S_1 and S_2 if q_j has exactly 1 one and $(N-1)$ zeroes. Else, it outputs `abort`.

4. S_u computes $[\vec{b}_c]_u = \oplus_{q\in Q}[\vec{b}_q]_u$, for $u \in \{1,2\}$.
5. S_1 and S_2 non-interactively agree on random permutation π.
6. S_u sends $\pi([\vec{b}_c]_u)$ to S_0, for $u \in \{1,2\}$.
7. S_0 locally reconstructs $\pi(\vec{b}_c) = \pi([\vec{b}_q]_1) \oplus \pi([\vec{b}_q]_2)$, sends `Accept` to S_1 and S_2, if $\pi(\vec{b}_c)$ has exactly τ ones, `abort` otherwise.

Output Transfer Send $\perp$ to $\mathcal{P}_i$ if verifiable DPF or S_0 generated `abort` during verification. Otherwise, proceed as follows:

8. S_u sends $[y_q]_u = \bigoplus_{j=1}^{N} [b_q^j]_u D^{m_q}[j]$ to $\mathcal{P}_i$, for $q \in Q, u \in \{1,2\}$.
9. $\mathcal{P}_i$ locally computes res $= \sum_{q\in Q}([y_q]_1 \oplus [y_q]_2)$.

Fig. 12. $\mathsf{PIR}^{\mathsf{I}}_{\mathsf{sum}}$ Protocol.

Security. For semi-honest participants, the security of protocol $\mathsf{PIR}^{\mathsf{I}}_{\mathsf{sum}}$ directly reduces to that of the 2-server PIR protocol in [18]. However, as mentioned in [18], a malicious participant could generate incorrect DPF keys, compromising the scheme's security and correctness. To prevent this type of misbehaviour, Boyle et al. [18] present a form of DPF called "verifiable DPF", which can assure the correctness of the DPF keys created by $\mathcal{P}_i$ at the cost of an increased *constant* amount of communication between the servers.

Protocol $\mathsf{PIR}^{\mathsf{II}}_{\mathsf{sum}}$

Input(s): i) S_1, S_2 : D; $|D| = N$, ii) $\mathcal{P}_i : Q = \{q_1, \ldots, q_\tau\}$, and iii) $S_0 : \perp$.
Output: $\mathcal{P}_i$: res $= \sum_{q\in Q} D[q]$

Computation S_1 and S_2 sample τ random mask values $\{m_1, \ldots, m_\tau\} \in \mathbb{Z}_{2^\ell}^{\tau}$ such that $\sum_{j=1}^{\tau} m_j = 0$. For each $q \in Q$, execute the following:

1. S_1, S_2 locally compute $D^{m_q} = D + m_q$, i.e., $D^{m_q}[j] = D[j] + m_q$, for $j \in [N]$.
2. $\mathcal{P}_i, S_1, S_2$ sample random $\theta_q \in [N]$.
3. $\mathcal{P}_i$ computes and sends $q' = q - \theta_q$ to S_0.
4. Servers execute DPF protocol [18] with S_0 as client with input q'. Server S_u obtains $[\vec{b}_{q'}]_u$ with $b^j_{q'} = 1$ for $j = q'$ and $b^j_{q'} = 0$ for $j \neq q'$, for $u \in \{1,2\}$.
5. S_u locally applies θ_u on $[\vec{b}_{q'}]_u$ to generate $[\vec{b}_q]_u$, for $u \in \{1,2\}$.

Verification Let $\{\vec{b}_{q_1}, \ldots, \vec{b}_{q_\tau}\}$ denote the bit vectors whose XOR-shares are generated during the preceding steps:

6. S_k computes $[\vec{b}_c]_k = \oplus_{q\in Q}[\vec{b}_q]_k$, for $u \in \{1,2\}$.
7. S_1 and S_2 non-interactively agree on random permutation π.
8. S_u sends $\pi([\vec{b}_c]_u)$ to S_0, for $u \in \{1,2\}$.
9. S_0 locally reconstructs $\pi(\vec{b}_c) = \pi([\vec{b}_q]_1) \oplus \pi([\vec{b}_q]_2)$. It sends `Accept` to S_1 and S_2, if $\pi(\vec{b}_c)$ has exactly τ ones. Else, it sends `abort`.

Output Transfer Send $\perp$ to $\mathcal{P}_i$ if S_0 generated `abort` during verification. Otherwise, proceed as follows:

10. S_u sends $[y_q]_u = \bigoplus_{j=1}^{N} [b_q^j]_u D^{m_q}[j]$ to $\mathcal{P}_i$, for $q \in Q, u \in \{1,2\}$.
11. $\mathcal{P}_i$ locally computes res $= \sum_{q\in Q}([y_q]_1 \oplus [y_q]_2)$.

Fig. 13. $\mathsf{PIR}^{\mathsf{II}}_{\mathsf{sum}}$ Protocol.

While using verifiable DPFs in $\mathsf{PIR}^{\mathsf{I}}_{\mathsf{sum}}$ ensures that the τ bit vectors generated by $\mathcal{P}_i$ are valid, it does not ensure that the bit vectors $\vec{\mathsf{b}}_1, \ldots, \vec{\mathsf{b}}_\tau$ correspond to τ distinct locations in the database D. However, we leverage the correctness guarantee of verifiable DPFs to reduce the communication cost for verification, as discussed in §5.1, §B.4, and §C. In detail, all τ bit vectors $\vec{\mathsf{b}}_1, \ldots, \vec{\mathsf{b}}_\tau$, i.e., the PIR queries, that are available in a secret-shared form among S_1 and S_2 are now guaranteed to have exactly one 1 in them, with the remaining bit positions being 0. To ensure distinctness, S_1 and S_2 XOR all their respective τ shares locally to obtain the secret-share of a single vector $\vec{\mathsf{b}}_c = \oplus_{k=1}^{\tau} \vec{\mathsf{b}}_k$. The problem now boils down to determining whether or not $\vec{\mathsf{b}}_c$ has exactly τ bit positions set to 1. This can be accomplished by servers S_1 and S_2 agreeing on a random permutation π and reconstructing $\pi(\vec{\mathsf{b}}_c)$ to S_0 and allowing S_0 to perform the check, as in the naive approach (cf. §5.1).

Computation Complexity (#AES operations). In $\mathsf{PIR}^{\mathsf{I}}_{\mathsf{sum}}$, the participant $\mathcal{P}_i$ must perform $4 \cdot \log_2(N/\lambda)$ AES operations as part of the key generation algorithm for each of the τ instances of $\mathcal{F}^{\mathsf{2S}}_{\mathsf{pir}}$ over a database of size N, where $\lambda = 128$ for an AES-based implementation. Similarly, S_1 and S_2 must perform $\log_2(N/\lambda)$ AES operations for each of the N DPF evaluations. We refer to Table 1 in [18] for more specifics.

5.3.2 $\mathsf{PIR}^{\mathsf{II}}_{\mathsf{sum}}$ *(Fig. 13).* In this approach, we use the server S_0 to reduce the computation and communication of the participant $\mathcal{P}_i$ in $\mathsf{PIR}^{\mathsf{I}}_{\mathsf{sum}}$. The idea is that S_0 plays the role of $\mathcal{P}_i$ for the PIR protocol in $\mathsf{PIR}^{\mathsf{I}}_{\mathsf{sum}}$. However, $\mathcal{P}_i$ cannot send its query q to S_0 in clear because it would violate privacy. As a result, $\mathcal{P}_i$ selects random values $q', \theta_q \in [N]$ such that $q = q' + \theta_q$. In this case, q' is a *shifted version* of the index q, and θ is a *shift correction* for q. $\mathcal{P}_i$ sends q' to S_0 and θ_q to both S_1 and S_2. The remainder of the computation until output retrieval will now take place solely among the servers.

The servers run a DPF instance [18] with S_0 acting as the client and input query q'. At the end of the computation, S_1 and S_2 obtain the bit vector $\vec{\mathsf{b}}_{q'}$, which corresponds to q'. However, as discussed in $\mathsf{PIR}^{\mathsf{I}}_{\mathsf{sum}}$, the servers require an XOR sharing corresponding to the actual query q in order to continue the computation. S_1 and S_2 do this by using the shift correction value θ_q received from $\mathcal{P}_i$. Both S_1 and S_2 will perform a right cyclic shift of their $\vec{\mathsf{b}}_{q'}$ shares by θ_q positions. A negative value for θ_q indicates a cyclic shift to the left.

It is easy to see that the XOR shares obtained after the cyclic shift correspond to the bit vector $\vec{\mathsf{b}}_q$. To further optimise $\mathcal{P}_i$'s communication, $\mathcal{P}_i$ and servers $\mathsf{S}_1, \mathsf{S}_2$ non-interactively generate a random shift correction values θ_q using the shared-key setup (cf. §B.1), and only the corresponding q' values are sent to S_0. The rest of the protocol is similar to $\mathsf{PIR}^{\mathsf{I}}_{\mathsf{sum}}$, and the formal protocol is shown in Fig. 13. In terms of malicious participants, $\mathsf{PIR}^{\mathsf{II}}_{\mathsf{sum}}$ has an advantage over $\mathsf{PIR}^{\mathsf{I}}_{\mathsf{sum}}$ as there is no need to use a verifiable DPF to protect against malicious $\mathcal{P}_i$, because the semi-honest server S_0 generates the DPF key instead of $\mathcal{P}_i$.

Improving Verification Costs in $\mathsf{PIR}^{\mathsf{II}}_{\mathsf{sum}}$. A large amount of communication is used in both $\mathsf{PIR}_{\mathsf{sum}}$ protocols to protect against malicious participants. More specifically, in Step 8 of Fig. 13 (resp., Step 8 of Fig. 12), $2N$ bits are sent to S_0 to ensure the distinctness of the queries made by the participant $\mathcal{P}_i$. We note that allowing a small amount of leakage to S_0 could improve this communication and is discussed next.

Consider the following modification to the $\mathsf{PIR}^{\mathsf{II}}_{\mathsf{sum}}$ protocol. Instead of sampling θ_q for each query $q \in Q$ (cf. Step 2 in Fig. 13), $\mathcal{P}_i$, S_1, and S_2 sample only one random shift value θ and use it for all τ instances. Since the queries must be distinct, $\mathcal{P}_i$ is forced to send distinct q' values to S_0 in Step 3 of Fig. 13. If not, S_0 can send `abort` to S_1 and S_2 at this

step, eliminating the need for communication-intensive verification. The relative distance between the queried indices would be leaked to S_0 as a result of this optimization. In concrete terms, if we use the same θ value for any two queries $q_m, q_j \in Q$, then $q_m - q_n = q'_m - q'_n$. Because S_0 sees all q' values in the clear, it can deduce the relative positioning of $\mathcal{P}_i$'s actual queries. However, since S_0 has no information about the underlying database D, this leakage may be acceptable for some applications.

5.3.3 Summary of communication costs. Tab. 2 summarises the communication cost for our two $\mathsf{PIR}_{\mathsf{sum}}$ approaches for instantiating $\mathcal{F}_{\mathsf{pirsum}}$ over a database of size N with τ PIR queries per client.

Stage	$\mathsf{PIR}^{\mathsf{I}}_{\mathsf{sum}}$	$\mathsf{PIR}^{\mathsf{II}}_{\mathsf{sum}}$
$\mathcal{P}_i$ to servers in C	$2\tau(\lambda+2)\log_2(N/\lambda)+4\tau\lambda$	$\tau \log_2 N$
Server to server	0	$2\tau(\lambda+2)\log_2(N/\lambda)+4\tau\lambda$
Servers in C to $\mathcal{P}_i$	$\tau \cdot 2\ell$	$\tau \cdot 2\ell$
+ Verification (mal.)	$2N+2+\delta$	$2N+2$

Table 2. Summary of communication costs in bits between participants $\mathcal{P}_i$ and a server $S_j \in C$ for $\mathsf{PIR}_{\mathsf{sum}}$. λ denotes the AES key size ($\lambda = 128$ in [17]), ℓ denotes the block size in bits ($\ell = 128$ in this work), and δ denotes the constant involved in the verifiable DPF approach enabling malicious security [18] (cf. §C).

6 EVALUATION

In this section, we evaluate and compare the computation and communication efficiency of our two RIPPLE protocols presented in §4. A fully-fledged implementation, similar to existing contact tracing apps, would necessitate collaboration with industry partners to develop a real-world scalable system for national deployment. Instead, we provide a proof-of-concept implementation and micro benchmark results for all major building blocks.[11] Moreover, we do not measure the speed of the communication link between the participants and the servers. We focus on the simulation phase for benchmarking, which is separate from the token generation phase. The simulations can ideally be done overnight while mobile phones are charging and have access to a high-bandwidth WiFi connection. According to studies [129, 131], sleeping habits in various countries provide a time window of several hours each night that can be used for this purpose.

Setup and Parameters. We run the benchmarks on the server-side with three servers (two for FSS-PIR and one as a helper server as discussed in §5.3) with Intel Core i9-7960X CPUs@2.8 GHz and 128 GB RAM connected with 10 Gbit/s LAN and 0.1 s RTT. The client is a Samsung Galaxy S10+ with an Exynos 9820@2.73 GHz and 8GB RAM. As Android does not allow third-party developers to implement applications for Android's TEE Trusty [7], we use hardware-backed crypto operations already implemented by Android instead. We use the code of [73] to instantiate FSS-PIR. We implement the AGCT in C++ and follow previous work on cuckoo hashing [112] by using tabulation hashing for the hash functions.

We instantiate our protocols in RIPPLE with $\kappa = 128$ bit security. We use RSA-2048 as the encryption scheme in $\mathsf{RIPPLE}_{\mathsf{TEE}}$ since Android offers a hardware-backed implementation. We omit the overhead of remote attestation for the sake of simplicity. For $\mathsf{RIPPLE}_{\mathsf{PIR}}$, we use the FSS-PIR scheme of [18, 73] as the baseline and the addresses are hashed with SHA-256 and trimmed to $40 - 1 + \log_2(\mathsf{p} \cdot E^{\mathsf{avg}})$ bits, where p is the number of participants and E^{avg} represents the average number of encounters per participant per simulation step. We set $E^{\mathsf{avg}} = 100$ while benchmarking based on numbers provided by research on epidemiological modeling [43, 98]. To avoid cycles when inserting n messages into the AGCT (cf. §4.2.3), we set its size to $10n$. This can be further improved as discussed in §4.2.3 [109, 111, 112]. A typical simulation step corresponds to one day, such that 14 simulation steps can simulate two weeks.

[11] Note that we are not attempting to create the most efficient instantiation. More optimizations will undoubtedly improve efficiency, and our protocols can be heavily parallelized with a large number of servers. Instead, our goal here is to demonstrate the viability of RIPPLE protocols for large-scale deployment.

Entities	Protocol	Population (p)									
		1K	10K	50K	100K	500K	1M	2M	5M	10M	20M
Participants in $\mathcal{P}$ (in KB)	$\text{RIPPLE}_{\text{TEE}}$ (§4.1)	16.00	16.00	16.00	16.00	16.00	16.00	16.00	16.00	16.00	16.00
	$\text{RIPPLE}_{\text{PIR}}$: $\text{PIR}^{\text{I}}_{\text{sum}}$ (§5.3.1)	51.63	62.42	69.97	73.22	80.77	84.02	87.27	91.56	94.81	98.06
	$\text{RIPPLE}_{\text{PIR}}$: $\text{PIR}^{\text{II}}_{\text{sum}}$ (§5.3.2)	3.45	3.49	3.52	3.53	3.56	3.57	3.59	3.60	3.62	3.63
Servers in C (in GB)	$\text{RIPPLE}_{\text{TEE}}$ (§4.1)	0.02	0.19	0.96	1.92	9.60	19.20	38.40	96.00	192.00	384.00
	$\text{RIPPLE}_{\text{PIR}}$ (§5)	0.01	0.10	0.48	0.96	4.80	9.60	19.20	48.00	96.00	192.00

Table 3. Communication costs per simulation step in our RIPPLE instantiations.

6.1 Communication Complexity

In this section, we look at the communication costs that our protocols incur. To analyse the scalability of our protocols, we consider p participants ranging from thousand (1K) to twenty million (20M). Tab. 3 summarises the communication costs of each participant as well as the communication servers (C) for one simulation step in a specific simulation. One simulation step includes all protocol steps, beginning with participants locally computing their infection likelihood δ and ending with them obtaining their cumulative infection likelihood Δ for that step.

6.1.1 Participant Communication. As shown in Tab. 3, a participant in $\text{RIPPLE}_{\text{TEE}}$ requires just 16KB of total communication in every simulation step, and this is independent of the population size. This is because each participant will only send and receive infection likelihood messages related to its encounters. While the value in the table corresponds to an average of 100 encounters ($E^{\text{avg}} = 100$), we depict the participants' communication in Fig. 14 with varied average number of encounters E^{avg} ranging from 10 to 500 for a population of 10M. Note that a 2-week simulation with $E^{\text{avg}} = 500$ can be completed by a participant in $\text{RIPPLE}_{\text{TEE}}$ with roughly 1MB communication.

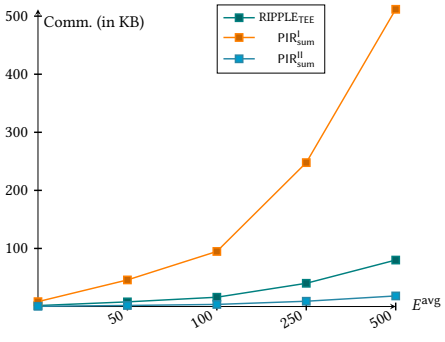


Fig. 14. Participant's communication with varying E^{avg} for a population of p =10M.

Unlike $\text{RIPPLE}_{\text{TEE}}$, participant communication in both $\text{PIR}^{\text{I}}_{\text{sum}}$ and $\text{PIR}^{\text{II}}_{\text{sum}}$ increases for larger populations as the corresponding database size increases. The communication, however, is only sub-linear in the database size[12].

In particular, the participant's communication in $\text{PIR}^{\text{I}}_{\text{sum}}$ ranges from 51.63KB to 98.06KB, with the higher cost over $\text{RIPPLE}_{\text{TEE}}$ attributed to the size of DPF keys used in the underlying FSS-PIR scheme [18], as discussed in §5. The communication in $\text{PIR}^{\text{II}}_{\text{sum}}$, on the other hand, is about 3.5KB for all participant sizes we consider. This reduced communication is due to the optimization in $\text{PIR}^{\text{II}}_{\text{sum}}$, which offloads the DPF key generation task to the helper server S_0 (cf. §5.3.2). A participant in $\text{PIR}^{\text{I}}_{\text{sum}}$ send approximately 7MB of data for a 2-week simulation for a 10M population with $E^{\text{avg}} = 500$, whereas it is only 0.25MB in the case of $\text{PIR}^{\text{II}}_{\text{sum}}$.

Tab. 4 provides the communication cost for a participant for multiple population sizes in

Population p	Protocol	E^{avg}				
		10	50	100	250	500
100K	$\text{RIPPLE}_{\text{TEE}}$ (§4.1)	1.60	8.00	16.00	40.00	80.00
	$\text{RIPPLE}_{\text{PIR}}$: $\text{PIR}^{\text{I}}_{\text{sum}}$ (§5.3.1)	6.24	34.99	73.22	193.79	403.83
	$\text{RIPPLE}_{\text{PIR}}$: $\text{PIR}^{\text{II}}_{\text{sum}}$ (§5.3.2)	0.35	1.76	3.53	8.87	17.81
1M	$\text{RIPPLE}_{\text{TEE}}$ (§4.1)	1.60	8.00	16.00	40.00	80.00
	$\text{RIPPLE}_{\text{PIR}}$: $\text{PIR}^{\text{I}}_{\text{sum}}$ (§5.3.1)	7.32	40.38	84.02	220.78	457.81
	$\text{RIPPLE}_{\text{PIR}}$: $\text{PIR}^{\text{II}}_{\text{sum}}$ (§5.3.2)	0.35	1.78	3.57	8.98	18.01
10M	$\text{RIPPLE}_{\text{TEE}}$ (§4.1)	1.60	8.00	16.00	40.00	80.00
	$\text{RIPPLE}_{\text{PIR}}$: $\text{PIR}^{\text{I}}_{\text{sum}}$ (§5.3.1)	8.40	45.78	94.81	247.77	511.79
	$\text{RIPPLE}_{\text{PIR}}$: $\text{PIR}^{\text{II}}_{\text{sum}}$ (§5.3.2)	0.36	1.80	3.62	9.08	18.22

Table 4. Communication (in KB)/participant/simulation step for varying average numbers of encounters E^{avg} and population sizes p

[12] DB size of $10n$, where n is the number of messages, and communication costs of $\text{RIPPLE}_{\text{PIR}}$ can be reduced by optimizing the database size by extending the database by only $d + \lambda$ bins, where d is the upper bound of double collisions and λ is an error parameter (cf. §4.2.3 and [109]).

$\text{RIPPLE}_{\text{TEE}}$, $\text{PIR}^{\text{I}}_{\text{sum}}$, and $\text{PIR}^{\text{II}}_{\text{sum}}$, while varying the average number of encounters E^{avg} per simulation step from 10 to 500. The communication cost in $\text{RIPPLE}_{\text{TEE}}$ is independent of the population size and grows linearly in E^{avg}. A similar trend can be seen in $\text{RIPPLE}_{\text{PIR}}$ with the exception that the cost increases sublinearly with the population size due to the use of FSS-based PIR scheme in $\text{RIPPLE}_{\text{PIR}}$.

6.1.2 Server Communication. The servers' communication is primarily attributed to the anonymous communication channel that they have established, which provides unlinkability and, thus, privacy to the messages of the participants. As discussed in §B.3, in order to communicate M messages through the channel, the servers must communicate $2M$ messages in $\text{RIPPLE}_{\text{TEE}}$, and $3M$ messages in $\text{RIPPLE}_{\text{PIR}}$. When it comes to concrete values, however, the server communication in $\text{RIPPLE}_{\text{PIR}}$ is half that of $\text{RIPPLE}_{\text{TEE}}$, as shown in Tab. 3. This is due to the larger message size in $\text{RIPPLE}_{\text{TEE}}$ as a result of the use of public-key encryption.

For a population of 10M, the servers in $\text{RIPPLE}_{\text{TEE}}$ must communicate 192GB of data among themselves, whereas $\text{RIPPLE}_{\text{PIR}}$ requires 96GB. Setting the proper bit length for the address field in the messages can further reduce communication. For example, a population of 20M with $E^{\text{avg}} = 100$ can be accommodated in a 70-bit address field. Using this optimization will result in an additional 23 % reduction in communication at the servers, as shown in Tab. 5. Fig. 15 captures these observations better, and Tab. 5 and Tab. 4 in the next subsection provide a detailed analysis of the concrete communication costs.

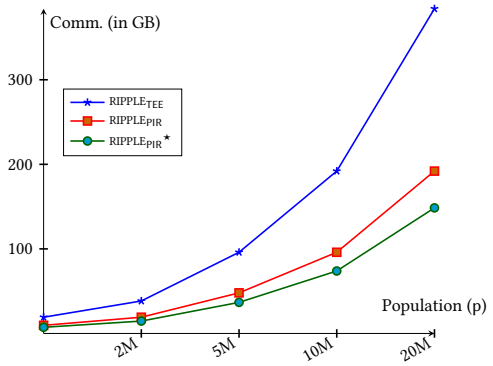


Fig. 15. Communication costs for servers per simulation step for varying population. ⋆ denotes the results for optimized bit addresses in $\text{RIPPLE}_{\text{PIR}}$ (cf. Tab. 5).

6.1.3 Communication Micro Benchmarks. Tab. 5 details the communication costs per simulation step at various stages in our instantiations of RIPPLE. We find that a participant's communication costs are very low compared to the overall costs. In $\text{RIPPLE}_{\text{TEE}}$, a participant communicates at most 268 KB and incurs a runtime of 92 seconds over a two-week simulation over a population of one million. In $\text{PIR}^{\text{II}}_{\text{sum}}$, the cost is reduced to 100 KB and 40 seconds of runtime. Communication increases to 1.2 MB in $\text{PIR}^{\text{I}}_{\text{sum}}$ due to the participant's handling of DPF keys.

Finally, Tab. 5 does not include costs for verification against malicious participants since they can be eliminated using server S_0 (cf. §5.3.2) or sketching algorithms similar to those in [18].

6.2 Computation Complexity

This section focuses on the runtime, which includes time for computation and communication between entities. Tab. 6 summarizes the computation time with respect to a participant $\mathcal{P}_i$ for a two-week simulation over a population of

Stages of RIPPLE	Protocol	Population (p)									
		1K	10K	50K	100K	500K	1M	2M	5M	10M	20M
Message Generation by $\mathcal{P}_i \in \mathcal{P}$ (in KB)	$\text{RIPPLE}_{\text{TEE}}$ (§4.1)[a]	12.80	12.80	12.80	12.80	12.80	12.80	12.80	12.80	12.80	12.80
	$\text{RIPPLE}_{\text{PIR}}$: ● (§4.2)	3.20	3.20	3.20	3.20	3.20	3.20	3.20	3.20	3.20	3.20
	$\text{RIPPLE}_{\text{PIR}}$: ◐ (§4.2)	2.30	2.34	2.38	2.39	2.41	2.43	2.44	2.45	2.46	2.48
Secure Shuffle by C (in GB)	$\text{RIPPLE}_{\text{TEE}}$ (§4.1)	0.02	0.19	0.96	1.92	9.60	19.20	38.40	96.00	192.00	384.00
	$\text{RIPPLE}_{\text{PIR}}$ - ● (§4.2)	0.01	0.10	0.48	0.96	4.80	9.60	19.20	48.00	96.00	192.00
	$\text{RIPPLE}_{\text{PIR}}$ - ◐ (§4.2)	0.01	0.07	0.36	0.72	3.62	7.28	14.63	36.75	73.88	148.50
Output Computation by $\mathcal{P}_i \in \mathcal{P}$ (in KB)[b]	$\text{RIPPLE}_{\text{TEE}}$ (§4.1)	6.40	6.40	6.40	6.40	6.40	6.40	6.40	6.40	6.40	6.40
	$\text{PIR}^{\text{I}}_{\text{sum}}$ - ● (§5.3.1)	51.36	62.42	69.97	73.22	80.77	84.02	87.27	91.56	94.81	98.06
	$\text{PIR}^{\text{I}}_{\text{sum}}$ - ◐ (§5.3.1)	26.48	32.64	37.69	39.82	44.77	47.05	49.38	52.33	54.77	57.26
	$\text{PIR}^{\text{II}}_{\text{sum}}$ (§5.3.2)	3.45	3.49	3.52	3.53	3.56	3.57	3.59	3.60	3.62	3.63

● - 128-bit address for $\text{RIPPLE}_{\text{PIR}}$ and ◐ - $40 - 1 + \log_2(\text{p} \cdot E^{\text{avg}})$ bit address for $\text{RIPPLE}_{\text{PIR}}$.
[a]Includes registration of public keys with the exit node $\mathcal{N}_{\texttt{exit}}$. [b]includes message download, decryption/PIR queries, summation.

Table 5. Detailed communication costs per simulation step in RIPPLE.

half a million. The longer computation time in $\text{RIPPLE}_{\text{TEE}}$, as shown in Tab. 6, is due to the public key encryption and decryption that occurs within the mobile device's TEE. This cost, however, is independent of population size and scales linearly with the average number of encounters, denoted by E^{avg}. In particular, for a 14-day simulation with a population of half a million, $\mathcal{P}_i$ in $\text{RIPPLE}_{\text{TEE}}$ needs approximately 43.7 seconds to perform the encryption and decryption tasks and may require additional time for the remote attestation procedure, which is not covered in our benchmarks. $\mathcal{P}_i$'s computation time in $\text{RIPPLE}_{\text{PIR}}$, on the other hand, is significantly lower and is at most 5 milliseconds for $\text{PIR}^{\text{II}}_{\text{sum}}$, while it increases to around 165 milliseconds for $\text{PIR}^{\text{I}}_{\text{sum}}$. The increased computation time in $\text{PIR}^{\text{I}}_{\text{sum}}$ is due to DPF key generation, which scales sub-linearly with population size.

	Per Simulation Step			Per Simulation ($N_{\text{step}} = 14$)		
	Message Generation (in ms)	PIR Queries (in ms)	Output Computation (in ms)	Message Generation (in sec)	PIR Queries (in sec)	Output Computation (in sec)
$\text{RIPPLE}_{\text{TEE}}$	80.00	-	3040.00	1.12	-	42.56
$\text{PIR}^{\text{I}}_{\text{sum}}$	0.30	11.73	4.8e-2	4.26e-3	0.16	6.72e-4
$\text{PIR}^{\text{II}}_{\text{sum}}$	0.30	3.0e-3	4.8e-2	4.26e-3	4.2e-5	6.72e-4

Table 6. Average participant computation times per simulation step distributed across various tasks. Values are obtained using a mobile for a population of p = 500K with $E^{\text{avg}} = 100$.

In Fig. 16, we plot the overall runtime of our two instantiations in RIPPLE for a full simulation of 2 weeks over various populations ranging from 1K to 500K. After a population of 100K, the runtime of $\text{RIPPLE}_{\text{PIR}}$ begins to exceed that of $\text{RIPPLE}_{\text{TEE}}$ due to an increase in database size, which results in longer data transfer times. More details regarding computation time are presented in Tab. 7. Note that the runtimes in Fig. 16 include runtime for computation and communication of the secure shuffle among the servers for anonymous communication and among servers and clients for the PIR in $\text{RIPPLE}_{\text{PIR}}$.

Stages of RIPPLE	Protocol	Population (p)					
		1K	10K	50K	100K	500K	1M
Message Generation by $\mathcal{P}_i \in \mathcal{P}$ (in sec)	$\text{RIPPLE}_{\text{TEE}}$ (§4.1)	1.12	1.12	1.12	1.12	1.12	1.12
	$\text{RIPPLE}_{\text{PIR}}$: (§4.2)	4.26e-3	4.26e-3	4.26e-3	4.26e-3	4.26e-3	4.26e-3
Secure Shuffle by $\mathcal{C}$ (in sec)	$\text{RIPPLE}_{\text{TEE}}$ (§4.1)	0.70	5.20	25.38	60.77	211.47	493.33⋆[a]
	$\text{RIPPLE}_{\text{PIR}}$ (§4.2)	0.78	6.65	32.36	71.17	386.68	1542.30⋆
Output Computation[b] (in sec)	$\text{RIPPLE}_{\text{TEE}}$ (§4.1)	44.66	44.66	44.66	44.66	44.66	44.66
	$\text{PIR}^{\text{I}}_{\text{sum}}$ (§5.3.1)	32.31	32.33	32.34	32.35	32.36	32.37
	$\text{PIR}^{\text{II}}_{\text{sum}}$ (§5.3.2)	32.20	32.20	32.20	32.20	32.20	32.20

[a]⋆ denotes system crash due to memory. [b]includes message download, decryption/PIR queries, summation.

Table 7. Detailed computation costs per simulation ($N_{\text{step}} = 14$, i.e., 14 days) in RIPPLE.

6.2.1 Computation Micro Benchmarks. Tab. 7 contains the computation costs per simulation at the different stages of our instantiations of RIPPLE's. As visible, data transfer time as part of anonymous communication through servers accounts for the majority of computation time and begins to affect overall performance as the population grows. Our system crashed due to memory constraints after a population of 500K while running the experiments, which is due to the fact that our implementation requires to store the whole PIR database in the memory and its size increases linearly with the number of participants. This will not be the case in a real-world deployment of powerful servers, which are equipped with more internal memory and additionally can store parts of the database on hard disks. Similar as w.r.t. communication, participants' computation costs are very low in comparison to the overall costs.

6.2.2 Battery Usage. The token generation phase in RIPPLE consumes the most amount of mobile battery as this phase is active throughout the day. This usage could be optimized by mobile OS providers like Apple and Google, as discussed by Vaudenay et al. [126] and Avitabile et al. [12] in the context of contact tracing apps. Their technology enables an app to run in the background, thus, significantly improving battery life, which is otherwise not possible for a standard third-party mobile application. Additionally, RIPPLE could offer users the choice to only participate in simulations while charging in order to not cause any unwanted battery drain.

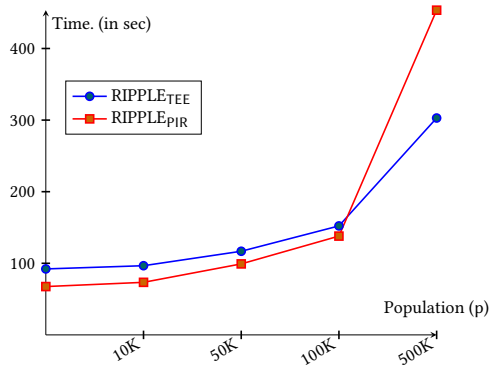


Fig. 16. Runtime per simulation in RIPPLE (14 days).

6.2.3 Comparison to Related Work. Note that no experimental comparison to related work is (and can be) done, as RIPPLE is the first distributed privacy-preserving epidemiological modeling system. Established contact tracing apps, such as the SwissCovid[13], the German Corona-Warn-App[14], or the Australian COVIDSafe[15] only record contacts for notifying contacts of infected people. Concretely, contact tracing basically relates to RIPPLE's token generation phase, while the other three phases (simulation initialization,

[13] https://github.com/SwissCovid [14] https://www.coronawarn.app/en/ [15] https://www.health.gov.au/resources/apps-and-tools/covidsafe-app

simulation execution, and result aggregation, cf. §3.2) are not covered by any contact tracing system. Crucially, the main contribution of our work is how to realize the simulation execution, which has never been done before. Hence, due to differences in the fundamental functionalities, no meaningful comparison between the systems is possible.

6.2.4 Code availability. Available at DOI: 10.5281/zenodo.6599225.

Summmary. Our benchmarking using the proof-of-concept implementation demonstrated the RIPPLE framework's viability for real-world adaptation. One of the key benefits of our approaches is that participants have very little work to do. The system's efficiency can be further improved with appropriate hardware and optimized implementations.

ACKNOWLEDGMENTS

This project received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement No. 850990 PSOTI). It was co-funded by the Deutsche Forschungsgemeinschaft (DFG) within SFB 1119 CROSSING/236615297 and GRK 2050 Privacy & Trust/251805230.

REFERENCES

[1] Ittai Abraham, Benny Pinkas, and Avishay Yanai. 2020. Blinder: MPC Based Scalable and Robust Anonymous Committed Broadcast. In *ACM CCS*.
[2] David Adam. 2020. Special report: The simulations driving the world's response to COVID-19. *Nature* (2020).
[3] Nadeem Ahmed, Regio A Michelin, Wanli Xue, Sushmita Ruj, Robert Malaney, Salil S Kanhere, Aruna Seneviratne, Wen Hu, Helge Janicke, and Sanjay K Jha. 2020. A Survey of COVID-19 Contact Tracing Apps. *IEEE Access* (2020).
[4] Fadi Al-Turjman and Bakkiam David Deebak. 2020. Privacy-Aware Energy-Efficient Framework Using the Internet of Medical Things for COVID-19. *IEEE Internet Things Mag.* 3, 3 (2020).
[5] Nikolaos Alexopoulos, Aggelos Kiayias, Riivo Talviste, and Thomas Zacharias. 2017. MCMix: Anonymous Messaging via Secure Multiparty Computation. In *USENIX Security*.
[6] Yaniv Altshuler, Nadav Aharony, Micky Fire, Yuval Elovici, and Alex Pentland. 2012. Incremental Learning with Accuracy Prediction of Social and Individual Properties from Mobile-Phone Data. In *International Conference on Privacy, Security, Risk and Trust and 2012 International Conference on Social Computing*.
[7] Android. 2020. Third-party Trusty applications. https://source.android.com/security/trusty.
[8] Sebastian Angel, Hao Chen, Kim Laine, and Srinath Setty. 2018. PIR with Compressed Queries and Amortized Query Processing. In *IEEE S&P*.
[9] Toshinori Araki, Jun Furukawa, Kazuma Ohara, Benny Pinkas, Hanan Rosemarin, and Hikaru Tsuchida. 2021. Secure Graph Analysis at Scale. In *ACM CCS*.
[10] ARM. 2009. ARM security technology building a secure system using TrustZone technology. https://developer.arm.com/documentation/genc009492/c.
[11] Yonatan Aumann and Yehuda Lindell. 2010. Security Against Covert Adversaries: Efficient Protocols for Realistic Adversaries. *Journal of Cryptology* (2010).
[12] Gennaro Avitabile, Vincenzo Botta, Vincenzo Iovino, and Ivan Visconti. 2020. Towards Defeating Mass Surveillance and SARS-CoV-2: The Pronto-C2 Fully Decentralized Automatic Contact Tracing System. https://eprint.iacr.org/2020/493
[13] Alexandros Bampoulidis, Alessandro Bruni, Lukas Helminger, Daniel Kales, Christian Rechberger, and Roman Walch. 2022. Privately Connecting Mobility to Infectious Diseases via Applied Cryptography. *PETs* (2022).
[14] Paolo Barsocchi, Antonello Calabrò, Antonino Crivello, Said Daoudagh, Francesco Furfari, Michele Girolami, and Eda Marchetti. 2021. COVID-19 & privacy: Enhancing of indoor localization architectures towards effective social distancing. *Array* 9 (2021).
[15] Sebastian P. Bayerl, Tommaso Frassetto, Patrick Jauernig, Korbinian Riedhammer, Ahmad-Reza Sadeghi, Thomas Schneider, Emmanuel Stapf, and Christian Weinert. 2020. Offline Model Guard: Secure and Private ML on Mobile Devices. *DATE* (2020).
[16] Joshua Blumenstock, Gabriel Cadamuro, and Robert On. 2015. Predicting poverty and wealth from mobile phone metadata. *Science* (2015).
[17] Dan Boneh, Elette Boyle, Henry Corrigan-Gibbs, Niv Gilboa, and Yuval Ishai. 2021. Lightweight Techniques for Private Heavy Hitters. In *IEEE S&P*.
[18] Elette Boyle, Niv Gilboa, and Yuval Ishai. 2016. Function Secret Sharing: Improvements and Extensions. In *ACM CCS*.
[19] Beyza Bozdemir, Sébastien Canard, Orhan Ermis, Helen Möllering, Melek Önen, and Thomas Schneider. 2021. Privacy-preserving Density-based Clustering. In *ASIACCS*.
[20] Fred Brauer. 2008. Compartmental models in epidemiology. In *Mathematical Epidemiology*.
[21] Fred Brauer, Carlos Castillo-Chavez, and Zhilan Feng. 2019. Simple Compartmental Models for Disease Transmission. In *Mathematical Models in Epidemiology*.

[22] Megha Byali, Harsh Chaudhari, Arpita Patra, and Ajith Suresh. 2020. FLASH: Fast and Robust Framework for Privacy-preserving Machine Learning. *PETS* (2020).

[23] Clea Caulcutt. 2022. Belgium introduces quarantine for monkeypox cases. *Politico* (2022). https://www.politico.eu/article/belgium-introduce-quarantine-monkeypox-case/.

[24] Justin Chan, Dean Foster, Shyam Gollakota, Eric Horvitz, Joseph Jaeger, Sham Kakade, Tadayoshi Kohno, John Langford, Jonathan Larson, Puneet Sharma, Sudheesh Singanamalla, Jacob Sunshine, and Stefano Tessaro. 2020. PACT: Privacy Sensitive Protocols and Mechanisms for Mobile Contact Tracing. https://arxiv.org/pdf/2004.03544.pdf.

[25] Nishanth Chandran, Divya Gupta, Sai Lakshmi Bhavana Obbattu, and Akash Shah. 2022. SIMC: ML Inference Secure Against Malicious Clients at Semi-Honest Cost. In *USENIX Security*.

[26] Harsh Chaudhari, Ashish Choudhury, Arpita Patra, and Ajith Suresh. 2019. ASTRA: High Throughput 3PC over Rings with Application to Secure Prediction. In *ACM CCSW@CCS*.

[27] Harsh Chaudhari, Rahul Rachuri, and Ajith Suresh. 2020. Trident: Efficient 4PC Framework for Privacy Preserving Machine Learning. In *NDSS*.

[28] David Chaum. 1985. Security without Identification: Transaction Systems to Make Big Brother Obsolete. *Commun. ACM* (1985).

[29] David Chaum. 1988. The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability. *Journal of Cryptology* (1988).

[30] David L Chaum. 1981. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Commun. ACM* (1981).

[31] Guoxing Chen, Yinqian Zhang, and Ten-Hwang Lai. 2019. OPERA: Open Remote Attestation for Intel's Secure Enclaves. In *ACM CCS*.

[32] Yi-Cheng Chen, Ping-En Lu, Cheng-Shang Chang, and Tzu-Hsuan Liu. 2020. A Time-Dependent SIR Model for COVID-19 With Undetectable Infected Persons. *Transactions on Network Science and Engineering* (2020).

[33] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. 1995. Private Information Retrieval. In *FOCS*.

[34] Adam Durbin Christy Cooney. 2022. High-risk monkeypox contacts advised to isolate. *BBC* (2022). https://www.bbc.com/news/uk-61546480.

[35] Matteo Ciucci and Frédéric Gouardères. 2020. National COVID-19 contact tracing apps. *EPRS: European Parliamentary Research Service* (2020).

[36] Henry Corrigan-Gibbs, Dan Boneh, and David Mazières. 2015. Riposte: An Anonymous Messaging System Handling Millions of Users. In *IEEE S&P*.

[37] Henry Corrigan-Gibbs and Dmitry Kogan. 2020. Private Information Retrieval with Sublinear Online Time. In *EUROCRYPT*.

[38] Ivan Damgård, Marcel Keller, Enrique Larraia, Valerio Pastro, Peter Scholl, and Nigel P. Smart. 2013. Practical Covertly Secure MPC for Dishonest Majority – Or: Breaking the SPDZ Limits. In *ESORICS*.

[39] George Danezis and Len Sassaman. 2003. Heartbeat Traffic to Counter (n-1) Attacks: Red-Green-Black Mixes *(WPES'03)*.

[40] Nicholas G Davies, Adam J Kucharski, Rosalind M Eggo, Amy Gimma, W John Edmunds, Thibaut Jombart, Kathleen O'Reilly, Akira Endo, Joel Hellewell, Emily S Nightingale, et al. 2020. Effects of non-pharmaceutical interventions on COVID-19 cases, deaths, and demand for hospital services in the UK: a modelling study. *The Lancet Public Health* (2020).

[41] Leo de Castro and Anitgoni Polychroniadou. 2022. Lightweight, Maliciously Secure Verifiable Function Secret Sharing. In *EUROCRYPT*.

[42] Roberta De Viti, Isaac Sheff, Noemi Glaeser, Baltasar Dinis, Rodrigo Rodrigues, Jonathan Katz, Bobby Bhattacharjee, Anwar Hithnawi, Deepak Garg, et al. 2022. CoVault: A Secure Analytics Platform. (2022). https://arxiv.org/pdf/2208.03784.pdf.

[43] Sara Y Del Valle, James M Hyman, Herbert W Hethcote, and Stephen G Eubank. 2007. Mixing patterns between age groups in social networks. *Social Networks* (2007).

[44] Daniel Demmler, Thomas Schneider, and Michael Zohner. 2015. ABY - A Framework for Efficient Mixed-Protocol Secure Two-Party Computation. In *NDSS*.

[45] Odo Diekmann, Hans Heesterbeek, and Tom Britton. 2012. *Mathematical Tools for Understanding Infectious Disease Dynamics*. Princeton University Press.

[46] Changyu Dong, Liqun Chen, and Zikai Wen. 2013. When private set intersection meets big data: an efficient and scalable protocol. In *ACM CCS*.

[47] Wenliang Du. 2001. A study of several specific secure two party computation problems. *USA: Purdue University* (2001).

[48] W John Edmunds, CJ O'callaghan, and DJ Nokes. 1997. Who mixes with whom? A method to determine the contact patterns of adults that may lead to the spread of airborne infections. *Proceedings of the Royal Society of London. Series B: Biological Sciences* (1997).

[49] J. Ekberg, K. Kostiainen, and N. Asokan. 2014. The Untapped Potential of Trusted Execution Environments on Mobile Devices. In *IEEE S&P*.

[50] Z. Erkin, J. R. Troncoso-pastoriza, R. L. Lagendijk, and F. Perez-Gonzalez. 2013. Privacy-Preserving Data Aggregation in Smart Metering Systems: An Overview. In *Signal Processing Magazine*.

[51] Saba Eskandarian and Dan Boneh. 2022. Clarion: Anonymous Communication from Multiparty Shuffling Protocols. In *NDSS*.

[52] Hossein Fereidooni, Samuel Marchal, Markus Miettinen, Azalia Mirhoseini, Helen Möllering, Thien Duc Nguyen, Phillip Rieger, Ahmad-Reza Sadeghi, Thomas Schneider, Hossein Yalame, et al. 2021. SAFELearn: secure aggregation for private federated learning. In *IEEE Security and Privacy Workshops (SPW)*.

[53] Neil Ferguson. 2005. What would happen if a flu pandemic arose in Asia? *Nature* (2005).

[54] Neil M Ferguson, Derek AT Cummings, Christophe Fraser, James C Cajka, Philip C Cooley, and Donald S Burke. 2006. Strategies for mitigating an influenza pandemic. *Nature* (2006).

[55] Jesús Fernández-Villaverde and Charles I Jones. 2022. Estimating and Simulating a SIRD Model of COVID-19 for Many Countries, States, and Citie. *Journal of Economic Dynamics and Control* (2022).

[56] European Centre for Disease Prevention and Control. 2022. Epidemiological update: Monkeypox outbreak. (2022). https://www.ecdc.europa.eu/en/news-events/epidemiological-update-monkeypox-outbreak.
[57] Craig Gentry and Shai Halevi. 2019. Compressible FHE with Applications to PIR. In *TCC*.
[58] Niv Gilboa and Yuval Ishai. 2014. Distributed Point Functions and Their Applications. In *EUROCRYPT*.
[59] Giulia Giordano, Franco Blanchini, Raffaele Bruno, Patrizio Colaneri, Alessandro Di Filippo, Angela Di Matteo, and Marta Colaneri. 2020. Modelling the COVID-19 epidemic and implementation of population-wide interventions in Italy. *Nature Medicine* (2020).
[60] Oded Goldreich. 2009. *Foundations of Cryptography: Volume 2, Basic Applications.* Cambridge University Press.
[61] Alison Gray, David Greenhalgh, Liangjian Hu, Xuerong Mao, and Jiafeng Pan. 2011. A Stochastic Differential Equation SIS Epidemic Model. *SIAM J. Appl. Math.* (2011).
[62] Daniel Günther, Maurice Heymann, Benny Pinkas, and Thomas Schneider. 2022. GPU-accelerated PIR with Client-Independent Preprocessing for Large-Scale Applications. In *USENIX Security*.
[63] Thomas Haines and Johannes Müller. 2020. SoK: Techniques for Verifiable Mix Nets. In *CSF*.
[64] Tiberiu Harko, Francisco SN Lobo, and MK3197716 Mak. 2014. Exact analytical solutions of the Susceptible-Infected-Recovered (SIR) epidemic model and of the SIR model with equal death and birth rates. *Appl. Math. Comput.* (2014).
[65] Gary F Hatke, Monica Montanari, Swaroop Appadwedula, Michael Wentz, John Meklenburg, Louise Ivers, Jennifer Watson, and Paul Fiore. 2020. Using Bluetooth Low Energy (BLE) Signal Strength Estimation to Facilitate Contact Tracing for COVID-19. https://arxiv.org/ftp/arxiv/papers/2006/2006.15711.pdf.
[66] Shaobo He, Yuexi Peng, and Kehui Sun. 2020. SEIR modeling of the COVID-19 and its dynamics. *Nonlinear Dynamics* (2020).
[67] Yan Huang, David Evans, and Jonathan Katz. 2012. Private Set Intersection: Are Garbled Circuits Better than Custom Protocols?. In *NDSS*.
[68] Inria and Fraunhofer AISEC. 2020. *ROBust and privacy-presERving proximity Tracing protocol.* https://github.com/ROBERT-proximity-tracing/documents.
[69] Intel. 2014. Intel® Software Guard Extensions Programming Reference. https://software.intel.com/sites/default/files/managed/48/88/329298-002.pdf.
[70] Intel. unk. Attestation Service for Intel Software Guard Extensions. https://api.trustedservices.intel.com/documents/sgx-attestation-api-spec.pdf.
[71] Kimmo Järvinen, Helena Leppäkoski, Elena-Simona Lohan, Philipp Richter, Thomas Schneider, Oleksandr Tkachenko, and Zheng Yang. 2019. PILOT: Practical Privacy-Preserving Indoor Localization Using OuTsourcing. In *EuroS&P*.
[72] P. Jauernig, A. Sadeghi, and E. Stapf. 2020. Trusted Execution Environments: Properties, Applications, and Challenges. In *IEEE S&P*.
[73] Daniel Kales, Olamide Omolola, and Sebastian Ramacher. 2019. Revisiting User Privacy for Certificate Transparency. In *EuroS&P*.
[74] W. O. Kermack and A. G. McKendrick. 1991. Contributions to the mathematical theory of epidemics—I. In *Bulletin of Mathematical Biology*.
[75] Adam Kirsch, Michael Mitzenmacher, and Udi Wieder. 2010. More Robust Hashing: Cuckoo Hashing with a Stash. *Journal on Computing* (2010).
[76] Petra Klepac, Adam J Kucharski, Andrew JK Conlan, Stephen Kissler, Maria L Tang, Hannah Fry, and Julia R Gog. 2020. Contacts in context: large-scale setting-specific social mixing matrices from the BBC Pandemic project. *MedRxiv* (2020). https://www.medrxiv.org/content/10.1101/2020.02.16.20023754v2.full.pdf.
[77] Victor I. Kolobov, Elette Boyle, Niv Gilboa, and Yuval Ishai. 2022. Programmable Distributed Point Functions. In *CRYPTO*.
[78] Nishat Koti, Mahak Pancholi, Arpita Patra, and Ajith Suresh. 2021. SWIFT: Super-fast and Robust Privacy-Preserving Machine Learning. In *USENIX Security*.
[79] Nishat Koti, Arpita Patra, Rahul Rachuri, and Ajith Suresh. 2022. Tetrad: Actively Secure 4PC for Secure Training and Inference. In *NDSS*.
[80] Kai Kupferschmidt. 2020. Case clustering emerges as key pandemic puzzle. https://www.science.org/doi/full/10.1126/science.368.6493.808.
[81] Klaus Kursawe, George Danezis, and Markulf Kohlweiss. 2011. Privacy-friendly aggregation for the smart-grid. In *PETS*.
[82] Eyal Kushilevitz and Rafail Ostrovsky. 1997. Replication is NOT Needed: SINGLE Database, Computationally-Private Information Retrieval. In *FOCS*.
[83] Sven Laur, Jan Willemson, and Bingsheng Zhang. 2011. Round-Efficient Oblivious Database Manipulation. In *International Conference on Information Security*.
[84] Ryan Lehmkuhl, Pratyush Mishra, Akshayaram Srinivasan, and Raluca Ada Popa. 2021. MUSE: Secure Inference Resilient to Malicious Clients. In *USENIX Security*.
[85] Dyani Lewis. 2020. Where Covid contact-tracing went wrong. *Nature* (2020).
[86] F. Li, B. Luo, and P. Liu. 2010. Secure Information Aggregation for Smart Grids Using Homomorphic Encryption. In *International Conference on Smart Grid Communications*.
[87] Yehuda Lindell and Benny Pinkas. 2007. An Efficient Protocol for Secure Two-Party Computation in the Presence of Malicious Adversaries. In *EUROCRYPT*.
[88] Yehuda Lindell, Benny Pinkas, Nigel P Smart, and Avishay Yanai. 2015. Efficient Constant Round Multi-Party Computation Combining BMR and SPDZ. In *CRYPTO*.
[89] Wouter Lueks, Seda F. Gürses, Michael Veale, Edouard Bugnion, Marcel Salathé, Kenneth G. Paterson, and Carmela Troncoso. 2021. CrowdNotifier: Decentralized Privacy-Preserving Presence Tracing. *PETs* (2021).
[90] Shaojun Luo, Flaviano Morone, Carlos Sarraute, Matías Travizano, and Hernán A Makse. 2017. Inferring personal economic status from social network location. *Nature Communications* (2017).

[91] Dominika Maison, Diana Jaworska, Dominika Adamczyk, and Daria Affeltowicz. 2021. The challenges arising from the COVID-19 pandemic and the way people deal with them. A qualitative longitudinal study. *PloS One* (2021).
[92] Robert M. May and Alun L. Lloyd. 2001. Infection dynamics on scale-free networks. *Physical Review E* (2001).
[93] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *International Conference on Artificial Intelligence and Statistics*.
[94] Pratyush Mishra, Ryan Lehmkuhl, Akshayaram Srinivasan, Wenting Zheng, and Raluca Ada Popa. 2020. Delphi: A Cryptographic Inference Service for Neural Networks. In *USENIX Security*.
[95] Payman Mohassel and Saeed Sadeghian. 2013. How to Hide Circuits in MPC an Efficient Framework for Private Function Evaluation. In *EUROCRYPT*.
[96] Payman Mohassel and Yupeng Zhang. 2017. SecureML: A System for Scalable Privacy-Preserving Machine Learning. In *IEEE S&P*.
[97] Yves-Alexandre de Montjoye, Jordi Quoidbach, Florent Robic, and Alex Sandy Pentland. 2013. Predicting Personality Using Novel Mobile Phone-Based Metrics. In *International Conference on Social Computing, Behavioral-Cultural Modeling, and Prediction*.
[98] Joël Mossong, Niel Hens, Mark Jit, Philippe Beutels, Kari Auranen, Rafael Mikolajczyk, Marco Massari, Stefania Salmaso, Gianpaolo Scalia Tomba, Jacco Wallinga, et al. 2008. Social Contacts and Mixing Patterns Relevant to the Spread of Infectious Diseases. *PLoS Medicine* (2008).
[99] Milad Nasr, Reza Shokri, and Amir Houmansadr. 2019. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *IEEE S&P*.
[100] B. Ngabonziza, D. Martin, A. Bailey, H. Cho, and S. Martin. 2016. TrustZone Explained: Architectural Features and Use Cases. In *International Conference on Collaboration and Internet Computing*.
[101] Olga Ohrimenko, Felix Schuster, Cédric Fournet, Aastha Mehta, Kapil Vaswani, and Manuel Costa. 2016. Oblivious Multi-Party Machine Learning on Trusted Processors. In *USENIX Security*.
[102] Rasmus Pagh and Flemming Friche Rodler. 2004. Cuckoo Hashing. *Journal of Algorithms* (2004).
[103] Christian Paquin and Greg Zaveruch. 2013. *U-Prove Cryptographic Specification V1.1 (Revision 3)*. http://www.microsoft.com/uprove.
[104] Romualdo Pastor-Satorras and Alessandro Vespignani. 2002. Immunization of complex networks. *Physical Review E* (2002).
[105] Arpita Patra, Thomas Schneider, Ajith Suresh, and Hossein Yalame. 2021. ABY2.0: Improved Mixed-Protocol Secure Two-Party Computation. In *USENIX Security*.
[106] Arpita Patra and Ajith Suresh. 2020. BLAZE: Blazing Fast Privacy-Preserving Machine Learning. In *NDSS*.
[107] Matthias Pezzutto, Nicolás Bono Rosselló, Luca Schenato, and Emanuele Garone. 2021. Smart Testing and Selective Quarantine for the Control of Epidemics. *Annual Review of Control, Robotics, and Autonomous Systems* 51 (2021), 540–550.
[108] Benny Pinkas and Eyal Ronen. 2021. Hashomer–Privacy-Preserving Bluetooth Based Contact Tracing Scheme for Hamagen. *Real World Crypto and NDSS Corona-Def Workshop* (2021).
[109] Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai. 2020. PSI from PaXoS Fast, Malicious Private Set Intersection. In *EUROCRYPT*.
[110] Benny Pinkas, Thomas Schneider, Gil Segev, and Michael Zohner. 2015. Phasing: Private Set Intersection Using Permutation-based Hashing. In *USENIX Security*.
[111] Benny Pinkas, Thomas Schneider, Christian Weinert, and Udi Wieder. 2018. Efficient Circuit-Based PSI via Cuckoo Hashing. In *EUROCRYPT*.
[112] Benny Pinkas, Thomas Schneider, and Michael Zohner. 2018. Scalable Private Set Intersection based on OT Extension. *TOPS* (2018).
[113] Leonie Reichert, Samuel Brack, and Björn Scheuermann. 2021. Poster: Privacy-Preserving Contact Tracing of COVID-19 Patients. In *IEEE S&P*.
[114] Reinhard Schlickeiser and Martin Kröger. 2021. Analytical Modeling of the Temporal Evolution of Epidemics Outbreaks Accounting for Vaccinations. (2021).
[115] Thomas Schneider and Oleksandr Tkachenko. 2019. EPISODE: Efficient Privacy-PreservIng Similar Sequence Queries on Outsourced Genomic DatabasEs. In *ASIACCS*.
[116] Vivek K Singh, Laura Freeman, Bruno Lepri, and Alex Sandy Pentland. 2013. Predicting Spending Behavior Using Socio-mobile Features. In *International Conference on Social Computing*.
[117] Michael Small and Chi K Tse. 2005. Small World and Scale free Model of Transmission of SARS. In *International Journal of Bifurcation and Chaos*.
[118] Hallam Stevens and Monamie Bhadra Haines. 2020. TraceTogether: Pandemic Response, Democracy, and Technology. https://www.tracetogether.gov.sg.
[119] Amanda Taub. 2020. A New Covid-19 Crisis: Domestic Abuse Rises Worldwide. *The New York Times* (2020).
[120] Robin N. Thompson. 2020. Epidemiological models are important tools for guiding COVID-19 interventions. *BMC Medicine* 18, 1 (2020), 152.
[121] Oleksandr Tkachenko, Christian Weinert, Thomas Schneider, and Kay Hamacher. 2018. Large-Scale Privacy-Preserving Statistical Computations for Distributed Genome-Wide Association Studies. In *ASIACCS*.
[122] Carmela Troncoso, Mathias Payer, Jean-Pierre Hubaux, Marcel Salathé, James R. Larus, Wouter Lueks, Theresa Stadler, Apostolos Pyrgelis, Daniele Antonioli, Ludovic Barman, Sylvain Chatel, Kenneth G. Paterson, Srdjan Capkun, David A. Basin, Jan Beutel, Dennis Jackson, Marc Roeschlin, Patrick Leu, Bart Preneel, Nigel P. Smart, Aysajan Abidin, Seda Gurses, Michael Veale, Cas Cremers, Michael Backes, Nils Ole Tippenhauer, Reuben Binns, Ciro Cattuto, Alain Barrat, Dario Fiore, Manuel Barbosa, Rui Oliveira, and José Pereira. 2020. Decentralized Privacy-Preserving Proximity Tracing. *IEEE Data Engineering Bulletin* (2020).
[123] Paul Tupper, Sarah P. Otto, and Caroline Colijn. 2021. Fundamental Limitations of Contact Tracing for COVID-19. *FACETS* (2021).
[124] Christopher van der Beets, Raine Nieminen, and Thomas Schneider. 2022. FAPRIL: Towards Faster Privacy-Preserving Fingerprint-Based Localization. In *SECRYPT*.

[125] Serge Vaudenay. 2020. Centralized or Decentralized? The Contact Tracing Dilemma. Cryptology ePrint Archive, Report 2020/531. https://ia.cr/2020/531.
[126] Serge Vaudenay and Martin Vuagnoux. 2020. *Analysis of SwissCovid.* Technical Report.
[127] Meilof Veeningen, Supriyo Chatterjea, Anna Zsófia Horváth, Gerald Spindler, Eric Boersma, Peter van der SPEK, Onno Van Der Galiën, Job Gutteling, Wessel Kraaij, and Thijs Veugen. 2018. Enabling Analytics on Sensitive Medical Data with Secure Multi-Party Computation. In *Medical Informatics Europe.*
[128] Nina Vindegaard and Michael Eriksen Benros. 2020. COVID-19 pandemic and mental health consequences: Systematic review of the current evidence. *Brain, Behavior, and Immunity* (2020).
[129] Olivia J Walch, Amy Cochran, and Daniel B Forger. 2016. A global quantification of "normal" sleep schedules using smartphone data. *Science Advances* (2016).
[130] Guan Wang, Tongbo Luo, Michael T Goodrich, Wenliang Du, and Zutao Zhu. 2010. Bureaucratic protocols for secure two-party sorting, selection, and permuting. In *ASIACCS.*
[131] Victoria Woollaston. 2015. *Sleeping habits of the world revealed: The US wakes up grumpy, China has the best quality shut-eye and South Africa gets up the earliest.* https://www.dailymail.co.uk/sciencetech/article-3042230/Sleeping-habits-world-revealed-wakes-grumpy-China-best-quality-shut-eye-South-Africa-wakes-earliest.html.
[132] Andrew Chi-Chih Yao. 1986. How to Generate and Exchange Secrets. In *FOCS.*
[133] Tijana Šušteršič, Andjela Blagojević, Danijela Cvetković, Aleksandar Cvetković, Ivan Lorencin, Sandi Baressi Šegota, Dragan Milovanović, Dejan Baskić, Zlatan Car, and Nenad Filipović. 2021. Epidemiological Predictive Modeling of COVID-19 Infection: Development, Testing, and Implementation on the Population of the Benelux Union. *Frontiers in Public Health* 9 (2021).

A CRYPTOGRAPHIC PRIMITIVES USED

In the following, we provide an overview about the (cryptographic) primitives and other techniques used in this work.

Anonymous Communication. To simulate the transmission of the modelled disease, RIPPLE requires anonymous messaging between participants. Mix-nets [30] and protocols based on the dining cryptographer (DC) problem [29] were the first approaches to anonymous messaging. A fundamental technique underlying mix-nets is the execution of an oblivious shuffling algorithm that provides unlinkability between the messages before and after the shuffle. In a mix-net, so-called mix servers jointly perform the oblivious shuffling so that no single mix server is able to reconstruct the permutation performed on the input data. Past research established a wide variety of oblivious shuffle protocols based on garbled circuits [47, 67, 130], homomorphic encryption [67], distributed point functions [1], switching networks [95], permutation matrices [83, §4.1], sorting algorithms [83, §4.2], and re-sharing [83, §4.3+4.4]. Recently, the works of [9] and [51] proposed efficient oblivious shuffling schemes using a small number of mix net servers.

Trusted Execution Environment (TEE). RIPPLE$_{\mathsf{TEE}}$ (§4.1) requires a TEE on the mobile devices of participants. TEEs are hardware-assisted environments that provide secure storage and execution of code on sensitive data which is isolated from the normal execution environment. Data stored in a TEE is secure even if the operating system is compromised, i.e., it offers confidentiality, integrity, and access control [49, 72]. Widely adopted TEEs are Intel SGX [69] and ARM TrustZone [10] (often used on mobile platforms [100]). Using TEEs for private computation has been extensively investigated, e.g., [15, 101]. A process called remote attestation allows external parties to verify that its private data sent via a secure channel is received and processed inside the TEE using the intended code [31, 70].

Private Information Retrieval (PIR). The first computational single-server PIR (cPIR) scheme was introduced by Kushilevitz and Ostrovsky [82]. Recent cPIR schemes [8, 57] use homomorphic encryption (HE). However, single-server PIR suffers from significant computation overhead since compute intensive HE operations have to be computed on each of the database block for each PIR request. In contrast, multi-server PIR relies on a non-collusion assumption between multiple PIR servers and uses only XOR operations [17, 18, 33, 36, 37] making it significantly more efficient than cPIR.

Cuckoo Hashing. In RIPPLE$_{\mathsf{PIR}}$ (§4.2), messages of participants have to be stored in a database D. To do so, a hash function H can be used to map an element x into bins of the database: $D[H(x)] = x$. However, as we show in §4.2,

$\text{RIPPLE}_{\text{PIR}}$ requires that at most one element is stored in every database location which renders simple hashing impracticable [110]. Cuckoo hashing uses h hash functions $H_1, \ldots, H_h$ to map elements into bins. It ensures that each bin contains exactly one element. If a collision occurs, i.e., if a new element is to be added into an already occupied bin, the old element is removed to make space for the new one. The evicted element, then, is placed into a new bin using another of the h hash functions. If the insertion fails for a certain number of trials, the element is inserted into a special bin called stash which is allowed to hold more than one element. Pinkas et al. [110] show that for $h = 2$ hash functions and $n = 2^{20}$ elements inserted to $2.4n$ bins, a stash size of 3 is sufficient to have a negligible error probability.

Garbled Cuckoo Table (GCT). As $\text{RIPPLE}_{\text{PIR}}$ uses key-value pairs for the insertion into the database, a combination of garbled Bloom filters [46] with cuckoo hashing [75, 102], called Garbled Cuckoo Table [109], is needed. Instead of storing x elements in one bin as in an ordinary cuckoo table, in a GCT, h XOR shares of x are stored at the h locations determined by inputting k into all h hash functions. E.g., with $h = 2$, if one of these two locations is already in use, the XOR share for the other (free) location is set to be the XOR of x and the data stored in the used location. In §4.2.3, we introduce a variant of GCT called *arithmethic garbled cuckoo table* (AGCT) that uses arithmetic sharing over the ring $\mathbb{Z}_{2^\ell}$ instead of XOR sharing. For a database with $2.4n$ entries where n is the number of elements inserted, Pinkas et al. [109] show that the number of cycles is maximally $\log_2 n$ with high probability.

Secure Multi-Party Computation (MPC). MPC [132] allows a set of mutually distrusting parties to jointly compute an arbitrary function on their private inputs without leaking anything but the output. In the last years, MPC techniques in various security models have been introduced, extensively studied, and improved, e.g., in [38, 44, 88]. These advancements significantly enhance the efficiency of MPC making it more and more practical for real-world applications. Due to the practical efficiency it can provide, various works [9, 22, 27, 78, 79, 106] have recently concentrated on MPC for a small number of parties, especially in the three and four party honest majority setting tolerating one corruption. In RIPPLE, we employ MPC techniques across three servers to enable an anonymous communication channel (cf. §B.3) and to develop efficient PIR_{sum} protocols (cf. §5).

Anonymous Credentials. To protect against sybil attacks (cf. §3.3), i.e., to hinder an adversary from creating multiple identities that can collect encounter information to detect correlations among unconscious encounters, we suggest to use anonymous credentials such that only registered participants can join RIPPLE. In this manner, the registration process can, for example, be linked to a passport. Such a registration system increases the cost to create (fake) identities. Chaum [28] introduced anonymous credentials where a client holds the credentials of several unlinkable pseudonyms. The client can then prove that it possesses the credentials of pseudonyms without the service provider being able to link different pseudonyms to the same identity. Additionally, anonymous credentials allow to certify specific properties like the age. Several instantiations for anonymous credentials have been proposed, e.g., Microsoft U-Prove [103].

B BUILDING BLOCKS IN RIPPLE

This section contains details about the building blocks used in the RIPPLE framework, such as shared-key setup, collision-resistant hash functions, anonymous communication channels, and Distributed Point Functions.

B.1 Shared-Key Setup

Let $F : \{0,1\}^\kappa \times \{0,1\}^\kappa \rightarrow X$ be a secure pseudo-random function (PRF), with co-domain X being $\mathbb{Z}_{2^\ell}$ and $C' = C \cup \{\mathcal{P}_i\}$ for a participant $\mathcal{P}_i \in \mathcal{P}$. The following PRF keys are established among the parties in C' in RIPPLE:

- k_{ij} among every $P_i, P_j \in C'$ and $i \neq j$.

- k_{ijk} among every $P_i, P_j, P_k \in C'$ and $i \neq j \neq k$.
- $k_{C'}$ among all the parties in C'.

To sample a random value $r_{ij} \in_R \mathbb{Z}_{2^\ell}$ non-interactively, each of P_i and P_j can invoke $F_{k_{ij}}(id_{ij})$. In this case, id_{ij} is a counter that P_i and P_j maintain and update after each PRF invocation. The appropriate sampling keys are implied by the context and are, thus, omitted.

B.2 Collision Resistant Hash Function

A family of hash functions $\{\mathsf{H} : \mathcal{K} \times \mathcal{L} \rightarrow \mathcal{Y}\}$ is said to be *collision resistant* if, for all probabilistic polynomial-time adversaries $\mathcal{A}$, given the description of H_k, where $k \in_R \mathcal{K}$, there exists a negligible function $negl()$ such that $\Pr[(x, x') \leftarrow \mathcal{A}(k) : (x \neq x') \wedge \mathsf{H}_k(x) = \mathsf{H}_k(x')] = negl(\kappa)$, where $x, x' \in_R \{0,1\}^m$ and $m = \text{poly}(\kappa)$.

B.3 Anonymous Communication Channel

This section describes how to instantiate the $\mathcal{F}_{\text{anon}}$ functionality used by RIPPLE for anonymous communication, as discussed in §4. We start with the protocol for the case of RIPPLE$_{\text{PIR}}$ and then show how to optimize it for the use in the RIPPLE$_{\text{TEE}}$ protocol. Recall from §4.2 that in RIPPLE$_{\text{PIR}}$, participants in $\mathcal{P}$ upload a set of messages from which a database D must be constructed at the end by S_1 and S_2. The anonymous communication is required to ensure that neither S_1 nor S_2 can link the source of the message even after receiving all messages in clear, which may not be in the same order. To tackle this problem, we use an approach based on oblivious shuffling inspired by [9, 51], which is formalised next.

Problem Statement. Consider the vector $\vec{m} = \{m_1, \ldots, m_\tau\}$ of τ messages with $m_j \in \mathbb{Z}_{2^\ell}$ for $j \in [\tau]$. We want servers S_1 and S_2 to obtain $\pi(\vec{m})$, where $\pi()$ denotes a random permutation that neither S_1 nor S_2 knows. Furthermore, an attacker with access to a portion of the network and, hence, the ability to monitor network data should not be able to gain any information about the permutation $\pi()$.

In RIPPLE$_{\text{PIR}}$, the vector $\vec{m}$ corresponds to the infection likelihood messages of the form $(a_{i,j}, c^e_{i,j})$ that each participant $\mathcal{P}_i \in \mathcal{P}$ sends over the network (cf. §4.2). W.l.o.g., we let $\mathcal{P}_i$ have the complete $\vec{m}$ with them. The protocol makes use of the third server S_0 in our setting and proceeds as follows:

1. $\mathcal{P}_i$ generates an additive sharing of $\vec{m}$ among S_0 and S_1:
   a) $\mathcal{P}_i, \mathsf{S}_0$ sample random $\langle\vec{m}\rangle_1 \in_R \mathbb{Z}^\tau_{2^\ell}$.
   b) $\mathcal{P}_i$ computes and sends $\langle\vec{m}\rangle_2 = \vec{m} - \langle\vec{m}\rangle_1$ to S_1.
2. S_0 and S_1 agree on a random permutation π_{01} and locally apply π_{01} to their shares. Let $\pi_{01}(\vec{m}) = \pi_{01}(\langle\vec{m}\rangle_1) + \pi_{01}(\langle\vec{m}\rangle_2)$.
3. $\mathsf{S}_0, \mathsf{S}_1$ perform a *re-sharing* of $\pi_{01}(\vec{m})$, denoted by $\vec{m_{01}}$, by jointly sampling a random $\vec{r_{01}} \in_R \mathbb{Z}^\tau_{2^\ell}$ and setting $\langle\vec{m_{01}}\rangle_1 = \pi_{01}(\langle\vec{m}\rangle_1) + \vec{r_{01}}$ and $\langle\vec{m_{01}}\rangle_2 = \pi_{01}(\langle\vec{m}\rangle_2) - \vec{r_{01}}$.
4. S_1 sends $\langle\vec{m_{01}}\rangle_2$ to S_2. Now, $(\langle\vec{m_{01}}\rangle_1, \langle\vec{m_{01}}\rangle_2)$ forms an additive sharing of $\vec{m_{01}}$ among S_0 and S_2.
5. S_0 and S_2 agree on a random permutation π_{02} and apply π_{02} to their shares. Let $\pi_{02}(\vec{m_{01}}) = \pi_{02}(\langle\vec{m_{01}}\rangle_1) + \pi_{02}(\langle\vec{m_{01}}\rangle_2)$.
6. S_0 sends $\pi_{02}(\langle\vec{m_{01}}\rangle_1)$ to S_2, who reconstructs $\pi_{02}(\vec{m_{01}})$.
7. S_2 generates an *additive-sharing* of $\pi_{02}(\vec{m_{01}})$, denoted by $\vec{m_{02}}$, among S_1 and S_2, by jointly sampling $\langle\vec{m_{02}}\rangle_1 \in_R \mathbb{Z}^\tau_{2^\ell}$ with S_1 and locally setting $\langle\vec{m_{02}}\rangle_2 = \pi_{02}(\vec{m_{01}}) - \langle\vec{m_{02}}\rangle_1$.
8. S_2 sends $\langle\vec{m_{02}}\rangle_2$ to S_1, who locally compute the output as $\vec{m_{02}} = \langle\vec{m_{02}}\rangle_1 + \langle\vec{m_{02}}\rangle_2$.

Anonymous Communication in RIPPLE$_{\mathsf{TEE}}$. As discussed in §4.1, the server S_2 is only required to have the complete set of messages in the clear but in an unknown random order. As a result, in the case of RIPPLE$_{\mathsf{TEE}}$, only the first permutation (π_{01} in Step 2) is sufficient and steps 5-8 are no longer required. Furthermore, in addition to the communication by S_1 in step 4, S_0 sends its share of $\vec{m_{01}}$ to S_2, who can then reconstruct $\vec{m_{01}} = \pi_{01}(\vec{m})$.

Security Guarantees. As discussed in §3.1, we assume that the MPC servers $S_i, i \in [2]$, that also instantiate the anonymous communication channel are semi-honest. We claim that the protocol described above will produce a random permutation of the vector $\vec{m}$ that neither S_1 nor S_2 is aware of. To see this, note that $\vec{m_{02}} = \pi_{02}(\vec{m_{01}}) = \pi_{02}(\pi_{01}(\vec{m}))$ and both S_1 and S_2 know only one of the two permutations π_{01} and π_{02}, but not both. Furthermore, the re-sharing performed in step 3 and the generation of additive shares in step 6 above ensures that an attacker observing the traffic cannot relate messages sent and received.

As we also consider a client-malicious security model [25, 84], where some clients might deviate from the protocol to gain additional information, we also have to take into consideration how the clients could manipulate the communication to break anonymity. For RIPPLE$_{\mathsf{TEE}}$, this is trivial: The TEE ensures that clients' messages are correctly generated and uploaded. For RIPPLE$_{\mathsf{PIR}}$, a malicious client could manipulate how many messages it uploads. However, messages with addresses that are already used will be dropped by the exit servers, i.e., effectively removing the malicious client from the system. A receiver will never fetch messages with unknown, random addresses. Furthermore, the servers use secure communication channels and even send freshly re-shared shares. Hence, considering the discussed aspects/assumptions, classical attacks on anonymous communication such as flooding [39] are not relevant for our model.

B.4 Distributed Point Functions (DPF)

Consider a point function $P_{\alpha,\beta} : \mathbb{Z}_{2^\ell} \rightarrow \mathbb{Z}_{2^{\ell'}}$ such that for all $\alpha \in \mathbb{Z}_{2^\ell}$ and $\beta \in \mathbb{Z}_{2^{\ell'}}$, $P_{\alpha,\beta}(\alpha) = \beta$ and $P_{\alpha,\beta}(\alpha') = 0$ for all $\alpha' \neq \alpha$. That is, when evaluated at any input other than α, the point function $P_{\alpha,\beta}$ returns 0 and when evaluated at α it returns β.

An (s, t)-distributed point function (DPF) [36, 58] distributes a point function $P_{\alpha,\beta}$ among s servers in such a way that no coalition of at most t servers learns anything about α or β given their t shares of the function. We use $(2, 1)$-DPFs in RIPPLE to optimize the communication of PIR-based protocols, as discussed in §5.3. Formally, a $(2, 1)$-DPF comprises of the following two functionalities:

- $\mathsf{Gen}(\alpha, \beta) \rightarrow (k_1, k_2)$. Output two DPF keys k_1 and k_2, given $\alpha \in \mathbb{Z}_{2^\ell}$ and $\beta \in \mathbb{Z}_{2^{\ell'}}$.
- $\mathsf{Eval}(k, \alpha') \rightarrow \beta'$. Return $\beta' \in \mathbb{Z}_{2^{\ell'}}$, given key k generated using Gen, and an index $\alpha' \in \mathbb{Z}_{2^\ell}$.

A $(2, 1)$-DPF is said to be *correct* if for all $\alpha, x \in \mathbb{Z}_{2^\ell}$, $\beta \in \mathbb{Z}_{2^{\ell'}}$, and $(k_1, k_2) \leftarrow \mathsf{Gen}(\alpha, \beta)$, it holds that

$$\mathsf{Eval}(k_1, x) + \mathsf{Eval}(k_2, x) = (x = \alpha)\ ?\ \beta : 0.$$

A $(2, 1)$-DPF is said to be *private* if neither of the keys k_1 and k_2 leaks any information about α or β. That is, there exists a polynomial time algorithm that can generate a computationally indistinguishable view of an adversary $\mathcal{A}$ holding DPF key k_u for $u \in \{1, 2\}$, when given the key k_u.

As mentioned in [18, 36], a malicious participant could manipulate the Gen algorithm to generate incorrect DPF keys that do not correspond to any point function. While [36] used an external non-colluding auditor to circumvent this issue in the two server setting, [18] formalised this issue and proposed an enhanced version of DPF called Verifiable DPFs. In addition to the standard DPF, a verifiable DPF has an additional function called Ver that can be used to ensure

the correctness of the DPF keys. In contrast to Eval, Ver in a $(2,1)$-verifiable DPF is an interactive protocol between the two servers, with the algorithm returning a single bit indicating whether the input DPF keys k_1 and k_2 are valid.

A verifiable DPF is said to be *correct* if for all $\alpha \in \mathbb{Z}_{2^\ell}$, $\beta \in \mathbb{Z}_{2^{\ell'}}$, keys $(k_1, k_2) \leftarrow \mathsf{Gen}(\alpha, \beta)$, the verify protocol Ver outputs 1 with probability 1. Ver should ensure that no additional information about α or β is disclosed to the party in possession of one of the DPF keys. Furthermore, the probability that Ver outputs 1 to at least one of the two servers for a given invalid key pair (k_1', k_2') is negligible in the security parameter κ.

Recent results in the area of (verifiable) DPFs [41, 77] might be an interesting direction for future work to further enhance the efficiency of our $\text{RIPPLE}_{\text{PIR}}$ construction.

Communication Complexity. Using the protocol of Boyle et. al. [18], a $(2,1)$-DPF protocol for a point function with domain size N has key size $(\lambda+2)\cdot\log(N/\lambda)+2\cdot\lambda$ bits, where $\lambda = 128$ for an AES based implementation. The additional cost in the case of verifiable DPF is for executing the Ver function, which has a constant number of elements in [18]. Furthermore, as stated in [18], the presence of additional non-colluding servers can improve the efficiency of Ver, and we use S_0 in the case of $\mathsf{PIR}^{\mathsf{I}}_{\mathsf{sum}}$, as discussed in §5.3.1. We refer to [18] for more details.

C PIR-SUM PROTOCOL DETAILS

This section provides additional details of our $\mathsf{PIR}_{\mathsf{sum}}$ protocols introduced in §5.1. We begin by recalling the security guarantees of a 2-server PIR for our setting [33, 62]. Informally in a two-server PIR protocol, where the database D is held by two non-colluding servers S_1 and S_2, a single server $\mathsf{S}_u \in \{\mathsf{S}_1, \mathsf{S}_2\}$ should not learn any information about the client's query. The security requirement is formally captured in Definition C.1.

Definition C.1. (Security of 2-server PIR) A PIR scheme with two non-colluding servers is called secure if each of the servers does not learn any information about the query indices.

Let $view(\mathsf{S}_u, \mathcal{Q})$ denote the view of server $\mathsf{S}_u \in \{\mathsf{S}_1, \mathsf{S}_2\}$ with respect to a list of queries, denoted by $\mathcal{Q}$. We require that for any database D, and for any two τ-length list of queries $\mathcal{Q} = (q_1, \ldots, q_\tau)$ and $\mathcal{Q}' = (q_1', \ldots, q_\tau')$, no algorithm whose run time is polynomial in τ and in computational parameter κ can distinguish the view of the servers S_1 and S_2, between the case of participant $\mathcal{P}_i$ using the queries in $\mathcal{Q}$ $(\{view(\mathsf{S}_u, \mathcal{Q})\}_{u\in\{1,2\}})$, and the case of it using $\mathcal{Q}'$ $(\{view(\mathsf{S}_u, \mathcal{Q}')\}_{u\in\{1,2\}})$.

Linear Summation PIR for $\mathcal{F}^{\mathsf{2S}}_{\mathsf{pir}}$ with optimized Communication. This section describes Chor et al.'s 2-server linear summation PIR protocol [33], as well as how to optimize communication using DPF techniques discussed in Appendix B.4. To retrieve the q-th block from database D of size N, the linear summation PIR proceeds as follows:

- Participant $\mathcal{P}_i$ prepares an N-bit string $\vec{\mathsf{b}}_q = \{\mathsf{b}_q^1, \ldots, \mathsf{b}_q^N\}$ with $\mathsf{b}_q^j = 1$ for $j = q$ and $\mathsf{b}_q^j = 0$ and $j \neq q$, for $j \in [N]$.
- $\mathcal{P}_i$ generates a Boolean sharing of $\vec{\mathsf{b}}_q$ among S_1 and S_2, i.e., $\mathcal{P}_i$ and S_1 non-interactively sample the random $[\vec{\mathsf{b}}_q]_1 \in_R \{0,1\}^N$ and $\mathcal{P}_i$ sends $[\vec{\mathsf{b}}_q]_2 = \vec{\mathsf{b}}_q \oplus [\vec{\mathsf{b}}_q]_1$ to S_2.
- S_u, for $u \in \{1,2\}$, sends $[y]_u = \bigoplus_{j=1}^{N} [\mathsf{b}_q^j]_u \mathsf{D}[j]$ to $\mathcal{P}_i$.
- $\mathcal{P}_i$ locally computes $\mathsf{D}[q] = [y]_1 \oplus [y]_2$.

The linear summation PIR described above requires communication of $N + 2\ell$ bits, where ℓ denotes the size of each data block in D.

Optimizing Communication using DPFs. Several works in the literature [18, 36, 58, 62] have used DPFs (cf. Appendix B.4) as a primitive to improve the communication in multi-server PIR. The idea is to use a DPF to allow the servers S_1 and S_2 to obtain the XOR shares of an N-bit string $\vec{\mathsf{b}}$ that has a zero in all positions except the one representing the

query q. Because DPF keys are much smaller in size than the actual database size, this method aids in the elimination of N-bit communication from $\mathcal{P}_i$ to the servers, as in the aforementioned linear summation PIR.

To query the q-th block from a database D of size N,

- Participant $\mathcal{P}_i$ executes the key generation algorithm with input q to obtain two DPF keys, i.e., $(k_1, k_2) \leftarrow \mathsf{Gen}(q, 1)$.
- $\mathcal{P}_i$ sends k_u to S_u, for $u \in \{1, 2\}$.
- S_u, for $u \in \{1, 2\}$, performs a DPF evaluation at each of the positions $j \in [N]$ using key k_u and obtains the XOR share corresponding to bit vector $\vec{\mathsf{b}}_q$.
  - S_u expands the DPF keys as $[\mathsf{b}_q^j]_u \leftarrow \mathsf{Eval}(k_u, j)$ for $j \in [N]$.
- S_u, for $u \in \{1, 2\}$, sends $[y]_u = \bigoplus_{j=1}^{N} [\mathsf{b}_q^j]_u \mathsf{D}[j]$ to $\mathcal{P}_i$.
- $\mathcal{P}_i$ locally computes $\mathsf{D}[q] = [y]_1 \oplus [y]_2$.

For the case of semi-honest participants, we use the DPF protocol of [18] and the key size is $O(\lambda \cdot \log(N/\lambda))$ bits, where $\lambda = 128$ is related to AES implementation in [18].

To prevent a malicious participant from sending incorrect or malformed keys to the servers, we use the verifiable DPF construction proposed in [18] for the case of malicious participants. This results only in a constant communication overhead over the semi-honest case. Furthermore, as noted in [18], we use the additional server S_0 for a better instantiation of the verifiable DPF, removing the need for interaction with the participant $\mathcal{P}_i$ for verification. We provide more information in Appendix B.4 and refer the reader to [18] for all details.