TECHNISCHE
UNIVERSITÄT
DARMSTADT

Leakage and Fault Resilience of Cryptographic Implementations

Vom Fachbereich Informatik der TU Darmstadt genehmigte

Dissertation

zur Erlangung des akademischen Grades
Doctor rerum naturalium (Dr. rer. nat.)

von

Maximilian Orlt

Darmstadt, 2024

Gutachter:	Prof. Sebastian Faust, Ph.D.
	Prof. François-Xavier Standaert, Ph.D.
Datum der Einreichung:	21.12.2023

Author: Maximilian Orlt
Title: Leakage and Fault Resilience of Cryptographic Implementations
Ort: Darmstadt, Technische Universität Darmstadt
Datum der mündlichen Prüfung: 05.02.2024
Veröffentlichungsjahr der Dissertation auf TUprints: 2024

Erklärung zur Dissertation

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

Maximilian Orlt

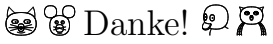
Acknowledgments

Studying is like a journey. While the initial phase may resemble a well-organized expedition, pursuing a Ph.D. transforms into an unpredictable adventure. I am infinitely grateful for the support and friendship I received from the community along this way. I would like to take this opportunity and simply say: Thank you!

I am very grateful to my supervisor, Sebastian Faust, for believing in me and always supporting me throughout my Ph.D. adventure. Thank you for all the good advice during the countless enjoyable discussions! Besides my supervisor, I would like to thank my further committee members: Amir Moradi, Ahmad-Reza Sadeghi, Thomas Schneider, and François-Xavier Standaert.

On my way to the Ph.D. I had the pleasure to meet many great researchers. I would like to thank my co-authors, including all researchers not mentioned in the thesis, as our research was separate from this thesis. Further, I want to thank Sebastian Berndt, Sebastian Faust, Julia Hesse, François-Xavier Standaert for their mentoring support in the beginning of my Ph.D studies when I was sometimes a bit lost and did not immediately see the next research steps. Apart from the mentoring support, I thank Dorothee Nikolaus and Jacqueline Wacker, for guiding me so many times in the bureaucracy jungle at TU Darmstadt. Especially, the retreat you organized in Sardinia was amazing! Luckily, I was able to travel to many conferences, and I enjoyed any of these travels primarily because of all the wonderful company. Apart from our many fruitful research discussions, I also have many great memories, such as the bicycle trip over the Golden Gate Bridge, the trip to Santa Barbara, the walks in Lyon, and the hiking in Malta and Amalfi. Thank you, Abdel, Alex, Benni, David, Elena, Loïc, Marc, Olga, Patrick, Paula, Poulami, Rune, Siavash! I especially want to thank Andreas for the many after-work dinners and the unconditional support "im Büro an der Forschungsfront." My Ph.D. journey became much more pleasant with you, Danke LiKo!

There are people in my life that I have always been able to count on: My family has supported me unconditionally my whole life. Danke, dass ihr schon immer für mich da seid! Last but not least, I want to thank someone very special in my live. Thank you, Céline, for your never-ending support, infinite understanding, and for having an "Owlways Monkey Place" for me!



List of Own Publications

Publications of this Thesis

[36] S. Berndt, T. Eisenbarth, S. Faust, M. Gourjon, M. Orlt, and O. Seker. "Combined Fault and Leakage Resilience: Composability, Constructions and Compiler". In: *Advances in Cryptology - CRYPTO 2023 - 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part III*. 2023, pp. 377–409.

[39] F. Berti, S. Faust, and M. Orlt. "Provable Secure Parallel Gadgets". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 4 (2023), pp. 420–459.

[57] G. Cassiers, S. Faust, M. Orlt, and F. Standaert. "Towards Tight Random Probing Security". In: *Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part III*. 2021, pp. 185–214.

[90] A. Erwig, J. Hesse, M. Orlt, and S. Riahi. "Fuzzy Asymmetric Password-Authenticated Key Exchange". In: *Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part II*. 2020, pp. 761–784.

[94] S. Faust, J. Krämer, M. Orlt, and P. Struck. "On the Related-Key Attack Security of Authenticated Encryption Schemes". In: *Security and Cryptography for Networks - 13th International Conference, SCN 2022, Amalfi, Italy, September 12-14, 2022, Proceedings*. 2022, pp. 362–386.

Further Publications

[2] A. Abromeit, F. Bache, L. A. Becker, M. Gourjon, T. Güneysu, S. Jorn, A. Moradi, M. Orlt, and F. Schellenberg. "Automated Masking of Software Implementations on Industrial Microcontrollers". In: *Design, Automation & Test in Europe Conference & Exhibition, DATE 2021, Grenoble, France, February 1-5, 2021*. 2021, pp. 1006–1011.

[19] G. Barthe, M. Gourjon, B. Grégoire, M. Orlt, C. Paglialonga, and L. Porth. "Masking in Fine-Grained Leakage Models: Construction, Implementation and Verification". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2 (2021), pp. 189–228.

[95] S. Faust, L. Masure, E. Micheli, M. Orlt, and F. Standaert. “Connecting Leakage-Resilient Secret Sharing to Practice: Scaling Trends and Physical Dependencies of Prime Field Masking”. In: *Advances in Cryptology - EUROCRYPT 2024 - 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26-30, 2024, Proceedings, Part IV*. 2024, pp. 316–344.

Abstract

Modern cryptography is the science to develop cryptographic primitives for secure communication in the presence of an adversary. The security of these primitives is usually proven in the black box model, which restricts the adversary to manipulate only the inputs of the primitives and to observe their outputs. However, Paul Kocher demonstrated at CRYPTO'96 that real-world implementations of such primitives can be attacked not only via their input-output behavior but also via so-called side-channels. The latter allow the adversary to learn information about inner computations, e.g., by measuring the runtime or power consumption of the device that executes the primitive. Shortly after, at EUROCRYPT'97, Boneh et al. even advanced the research on implementation attacks by injecting faults, manipulating inner computations using methods such as laser beams or electromagnetic pulses. Their attacks illustrate the vulnerability of implementations that are provable secure in the black box model. As a consequence, Ishai et al. formalized those attacks and investigated provably secure countermeasures for side-channel and fault attacks at CRYPTO'03 and EUROCRYPT'06, respectively. Following their new research direction, this thesis addresses leakage and fault resilience of cryptographic implementations.

As a first step, this thesis presents methods to protect arbitrary computations against side-channel attacks, focusing on masked circuits in the random probing model. We propose two different approaches to prove security, namely the so called Probe Distribution Table (PDT) and Dependency Graph (DG). While the PDT significantly improves concrete security results of state-of-the-art constructions, the DG allows for an asymptotic security analysis which is particularly efficient for affine circuits.

As a second step, this thesis introduces two fault-resilient cryptographic primitives to address faults in secret values such as keys and passwords. The first primitive, fuzzy asymmetric password-authenticated key exchange (faPAKE), deals with typos in passwords, allowing two parties to generate a common key based on partial password knowledge. The second primitive is a deterministic authenticated encryption scheme which is provably secure against fault attacks, and which allows for secure communication in the presence of malicious faults on the secret key.

Finally, the thesis presents two secure compilers that combine fault and leakage

resilience, addressing scenarios where an adversary can simultaneously probe and fault the internal computation. Given that standalone fault and leakage resilience is not sufficient to ensure combined resilience, we introduce a new security property, the so-called fault-invariance, that allows proofs to ensure security against combined attacks.

My Contribution

This thesis is based on five scientific publications co-authored with my supervisor Sebastian Faust, and many other excellent researchers: Sebastian Berndt, Francesco Berti, Gaëtan Cassiers, Thomas Eisenbarth, Andreas Erwig, Marc Gourjon, Julia Hesse, Juliane Krämer, Siavash Riahi, Okan Seker, François-Xavier Standaert, and Patrick Struck. I am deeply grateful to all of my co-authors for their invaluable contributions and stimulating research discussions. Often it becomes rather difficult to segregate the contributions of individual researchers, and our papers are no exception. Nonetheless, to fulfill the formal requirements of the thesis, I have outlined my specific contributions to each of the five papers below.

Chapter 3. The paper [57] introduces a new tool to verify the leakage resilience of circuits in the random probing model. Sebastian Faust and François-Xavier Standaert supervised this work, when we discussed the requirements for tighter security verifications in this leakage model. Gaëtan Cassiers contributed the Monte-Carlo technique for efficient security bounds and the work-intensive implementation part of the tool. I mainly contributed the new security notion, the so-called Probe Distribution Table (PDT), which allows us to compute the security of composed circuits. More precisely, I proved that one can analyze the security of a large circuit using the PDTs of its subcircuits.

Further, the chapter presents the results of [39]. Most of the observations of this paper result from countless discussions with Francesco Berti and Sebastian Faust. Francesco Berti focused on the algebraic background, and the security analysis of the multiplication gadget. My focus was the dependency graph that allows to describe the dependencies of intermediate values of circuits. The graph allows for a simpler security analysis and provides generic security results for any number of shares. Further, I gave a composition technique that allows to construct the dependency graph of circuits from the dependency graphs of its subcircuits.

Chapter 4. In [90], Julia Hesse has supervised Andreas Erwig, Siavash Riahi, and me in our first research project. As a team, Andreas Erwig, Siavash Riahi, and I investigated password-authenticated key exchange protocols that are fuzzy

(allowing small password deviation) and asymmetric (avoiding clear password files on server side.) We formulated our new notion in the Universal Composability framework (UC), and constructed two secure protocols. In addition, Andreas Erwig, Siavash Riahi and I did the UC security proofs under the guidance of Julia Hesse.

Moreover, [94] analyzes authenticated encryption schemes against related-key attacks (RKA). Sebastian Faust and Juliane Krämer jointly supervised this project and we went through several research discussions to find the right security requirements to correctly define the security model for authenticated encryption schemes secure against related-key attacks. The initial project idea was contributed by Patrick. Together, Patrick and I analyzed generic constructions for authenticated encryption schemes, giving positive and negative security results. First, we investigated the RKA security of the encryption scheme and the message authentication code. Then, we analyzed how the security of the authenticated encryption reduces to the related-key attack security of the underlying encryption and message authentication code. Moreover, I constructed the N-Star scheme, proving that it satisfies our strongest security notion.

Chapter 5 This chapter is based on [36], a joint work with Sebastian Berndt, Marc Gourjon, Okan Seker, Thomas Eisenbarth, and Sebastian Faust. Initially, we started this project to improve the multiplication gadget for polynomial sharing. We developed a multiplication gadget that is optimal in the number of shares.

In addition, Sebastian Berndt, Sebastian Faust, and I continued to research combined security where the adversary can do both, faults and probes simultaneously. Sebastian Berndt mainly contributed to the fault resilience of internal computations, and together we investigated a formal property that ensures fault detection. Further, I defined an adapted (S)NI property called fault resilient (S)NI which allows for leakage resilient compositions even in the presence of faults. For the security proof, I extracted a new property, called shift-invariance, that implies probing security in the presence of faults.

Each chapter of this thesis includes parts taken verbatim from the corresponding papers listed above with some adjustments.

Contents

1. Introduction

Cryptography (Ancient Greek kryptós, "hidden"; and graphein, "to write") is the science of *secure communication* in an *adversarial environment*. In this context, an adversarial environment refers to the scenario where an attacker tries to undermine the communication between two or more parties. The most apparent strategy of such an attacker is to simply eavesdrop on the information exchanged between the parties. However, various other malicious attacks are also noteworthy, with prominent examples being an adversary who actively alters another party's message or sends a message in its name. The overarching objective of cryptography is to enhance *security* by reducing an attacker's opportunities to sabotage such communication.

When two parties securely exchange messages, they usually transform the message using *cryptographic primitives* to hide the actual message (confidentiality), to know the sender (authenticity), and to be convinced that the message was not changed by an attacker (integrity). Auguste Kerckhoffs, a Dutch linguist and cryptographer [55], gave the following fundamental principle for such cryptographic primitives.

> *"It should not require secrecy,*
> *and it should not be a problem if it falls into enemy hands."*
>
> -Auguste Kerckhoffs, [134]

This approach is well known as *Kerckhoffs' principle*, and it is interpreted in such a way that "it", i.e., the cryptographic primitive, must be secure even if the "enemy", i.e., the adversary, knows the functionality of the used primitive (e.g., [131], Chap. 1). Following this principle, it was generally assumed that an adversary is familiar with the design of a cryptographic primitive and the security argumentation of a primitive was typically only based on the absence of known vulnerabilities. Classical cryptography therefore relied mostly on artistic creativity to thwart potential vulnerabilities, fostering a dynamic competition between designing of secure primitives and breaking existing ones. However, this strategy was lacking a fundamental theoretical security analysis of a primitive, as security relied only on the assumption that the adversary has not *yet* broken the used primitive. While in the beginning cryptography was only used by small groups, e.g., the government

or secret organizations, it became more important than ever with the digitization in the 20^{th} century. Rapidly, it became an integral part of everyone's life, which is exemplified by its role in enabling contactless payments, online shopping, or smart devices in general. Motivated by its importance in today's world, the field of cryptography started to receive wide attention by academics and practitioners, which eventually resulted in it being introduced as a formal scientific discipline, e.g., [76]. In contrast to classical cryptography, this new scientific discipline, often referred to as modern cryptography, no longer relies on the absence of discovered vulnerabilities to establish security; instead, it is based on rigorous mathematical theory, often described as *provable security*.

Provable Security. The formal treatment of security requires the following important ingredients: An *adversarial model*, that precisely defines the security needs for the adversarial environment, a carefully chosen cryptographic primitive to enable a secure communication method, and a *security proof* which shows that breaking the security of the cryptographic primitive is "hard". The typical *hardness assumptions* for such security proofs are well-known mathematical problems such as the complexity of factoring large integers (e.g., [131], Chap. 8). According to Kerckhoffs' principle, the cryptographic primitive is publicly known but initiated with a secret key. Its security is then based on the secrecy of this key and on an underlying hard problem.

Here, a generally accepted security model within modern cryptography is the *black box* model. It grants the adversary access to the inputs and outputs of a cryptographic primitive without disclosing the primitive's secret key, and a security proof typically shows that a scheme is secure as long as the underlying mathematical problem cannot be solved. While the black box model has significantly contributed to the formal investigation of security in modern cryptography, it has gradually become apparent that this model may be overly idealized. In the black box model, the adversary manipulates the inputs and observes the outputs of a cryptographic primitive, assuming that the algorithm operates within a fully trusted environment [173]. This interpretation of Kerckhoffs' principle takes an optimistic view, as it implies that the adversary only gains knowledge of the input-output behavior of a cryptographic implementation "if it falls into enemy hands." Unfortunately, this assumption turned out to be too naive, because the model does not consider the case where the adversary attacks the inner working of the implementation.

Implementation Attacks. Since the late 1990's, it has been known that real-world cryptographic implementations are not only attacked via their input-output behavior as modeled by the black box model, but also via *implementation attacks*. A prominent example of an implementation attack was presented in a pivotal work by Paul Kocher [138], at CRYPTO 1996. In his work, Kocher demonstrated an attack that exploits the fact that the running time of a device depends on the underlying secret key. This physical effect leaks enough information about the key to break schemes that were previously proven secure in the idealized black box model. Kocher's attack is a prominent example of a so-called *side-channel attack*. Following Kocher's work, a diverse range of side-channel attacks were developed in the subsequent years, exploiting physical leakages such as power consumption [139], electromagnetic radiation [104], cache accesses [48], or acoustic signals [107].

Almost one year later at EUROCRYPT 1997, Boneh, Demillo, and Lipton [47] presented a new cryptanalytic attack that also bypassed the security of the black box model. Their attack essentially exploited computational errors, which allowed to recover the cryptographic keys. Due to Kerckhoff's principle, security depends only on the secrecy of the key, and therefore, key recovery attacks are the worst-case scenario. Shortly after, at CRYPTO 1997, Biham and Shamir [43] enhanced this attack to encompass a larger class of cryptographic primitives, showing that implementation attacks can be more than only passive side-channel attacks. The adversary can also perform active *fault attacks* to manipulate internal calculations through techniques such as clock glitches [77], voltage glitches [176], electromagnetic pulses [73, 79], and laser beams [43, 167].

These results highlight that active and passive implementation attacks can lead to catastrophic security flaws in supposedly secure implementations. The critical observation is that those attacks only exploit the physical behavior of the devices on which the cryptographic schemes are implemented and as the famous cryptographer Adi Shamir said at the Turing Award Lecture 2002 "Cryptography is typically bypassed, not penetrated." [165]

Beyond the Black Box. Roughly three years after Kocher presented his side-channel attack at CRYPTO 1996, Daemen and Rijmen proposed a countermeasure against side-channel attacks where a secret bit value is transformed into two values, one carrying the original bit value and the other its complement [72]. It started a creative art of various ad-hoc countermeasures, e.g., [112, 147], and led to a competition of designing unbroken methods and breaking existing ones [60, 67]. Similar to the beginnings of cryptography as a science in general, it took some time before implementation attacks were formalized, establishing a novel scientific subfield of modern cryptography.

Concretely, seven years later at CRYPTO 2003, Ishai, Sahai, and Wagner (ISW: Private Circuit I) [126] introduced the first adversarial model aimed at capturing side-channel attacks. They modeled the implementation of cryptographic primitives as a circuit, where every operation, such as addition and multiplication, was represented as a gate, interconnected by wires to assess the primitive's functionality. This computational model, unlike the black box model, offered insights into the internal computation of an implementation and proved valuable for modeling implementation attacks.

For instance, the *t-threshold probing model* also introduced in the Private Circuit I paper allowed adversaries to gain knowledge of t different wires' values. This model represents the first step towards the formalization of implementation attacks, and describes an idealized side-channel attack, where the adversary can measure up to t different intermediate values of an implementation.

During EUROCRYPT 2006, Ishai, Prabhakaran, Sahai, and Wagner (IPSW: Private Circuit II) [125] additionally demonstrated that the circuit model could also serve as a basis for modeling fault attacks. Their adversary had the capability to manipulate the values of an unlimited number of wires within the circuit. However, they constrained the adversary to specific fault attacks, initially permitting the manipulation of wire values to zero. Similar to the probing model, the adversary's actions were bounded by the number of faults often referred to as the *e-threshold fault model*, permitting a broader range of fault types, also called fault class. Such a fault class typically consists of functions to add a constant to the wire's value or fix it by a constant. For further details about the probing and fault models, we refer to Chapter 2.[1] Both models initiated a new scientific branch of modern cryptography and there are various model adaptions to describe implementation attacks more realistically. One prominent example is the *p-random probing model* [126] also used in our work. In this model, each wire probabilistically reveals its value with a given probability p, and with a complementary probability of $1 - p$, it leaks nothing about the value. Both models, the probing and the fault model enabled the study of *leakage and fault-resistant* constructions.

Leakage and Fault Resilience. A prevalent approach involves the use of a *compiler* that takes an unprotected circuit as input and produces a protected version as output. Such protected circuits do not use the plain secrets for internal computations, and instead, they transform the secrets via so-called *encodings*. For example, in a leakage-resilient circuit, sensitive data s is encoded via a function

[1] Numerous other fascinating research findings exist. Due to space constraints, we provide a concise introduction here and provide a detailed discussion about the extensive work in the respective chapters.

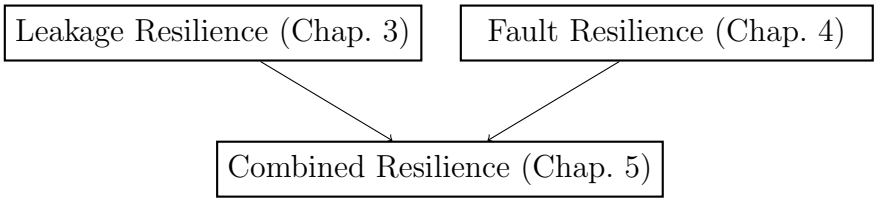


Figure 1.1.: Content overview of this thesis

Enc to share the value s into $n > t$ shares $s_0, \dots s_{n-1} \leftarrow \mathsf{Enc}(s)$, ensuring that learning t shares s_i does not reveal any information about the secret s. This procedure is often called *masking* [147], and circuits using such encodings are often known as *masked circuits*. Similarly to masking, a circuit can be protected against faults using an encoding $s_0, \dots s_{n-1} \leftarrow \mathsf{Enc}(s)$ such that e faulted values s_i in $s_0, \dots s_{n-1}$ can be detected or even corrected. Such an encoding is often referred to as *error detection/correction codes*, e.g. [151, 156]. While the emerging research branch explored compilers designed to be secure against both types of attacks, countermeasures against combined attacks have only recently gained attention. On the one hand this fact is surprising as combined attacks give the attacker much more power than the sum total of both individual attacks. The idea of such attacks is, for example, that the adversary can explicitly fault the countermeasures against side-channel attacks, and therefore increase vulnerabilities to gain efficient side-channel leakage. On the other hand it turned out that countermeasures against those combined attacks are rather challenging for the same reason. Following the groundbreaking work of Ishai et al. [125, 126], this thesis focuses on resilience against implementation attacks, and we overview our contribution in the following section.

1.1. Overview

After providing a concise overview of the historical evolution of cryptography in the context of implementation attacks, we now outline the core objectives of this thesis: *Leakage and Fault Resilience of Cryptographic Implementations*. This work yields a threefold contribution, encompassing leakage resilience, fault resilience, and their intricate combination. First, it considers side-channel attacks (Chapter 3), examining masking as a countermeasure in the random probing model. Subsequently, in Chapter 4, the thesis explores fault attacks, investigating effective strategies for secure key generation and encryption in the presence of faulted secrets such as passwords and keys. Finally, Chapter 5 combines both side-channel and fault attacks,

considering them concurrently to provide a comprehensive proof technique and a provably secure compiler which enhances the security of cryptographic primitives against combined attacks.

Leakage Resilience. Chapter 3 proposes and analyzes general methods to protect arbitrary computations against side-channel attacks. Therefore, we present two alternative approaches to prove the security of masked circuits in the random probing model, and a secure compiler to transform any circuit into a masked one. More precisely, the first proof technique (Section 3.1.1) is based on the findings in [57] and proposes a novel method for proofs that significantly improved the latest security results in the random probing model but is limited by its runtime complexity. In particular, this technique allowed us to prove that the masking countermeasures proposed by the very first paper ISW: Private Circuit I [126] with $n = 6$ shares and a realistic leakage probability of $p \geq 10^{-4}$ ensures about the same security as the latest securely proven construction in 2021 of Belaïd et al. [26] using $n = 27$ shares.

The second approach (Section 3.1.2) is based on the results in [39] and allows us to prove the security of masked circuits with an arbitrarily large number of shares n. We used this method to prove the random probing security of our proposed compiler. Our compiler is particularly efficient for affine circuits, that only use linear operations such as additions. Here, our compiler allows a constant leakage probability $p = O(1)$. In detail, our countermeasure for affine circuits achieves $O(p^n)$-security with linear complexity $O(n)$. This improves the compiler of Belaïd et al. [26] using a complexity $O(n^{2.4})$, and Dziembowski et al. [85] only achieving $O(\sqrt{p}^n)$. Affine circuits are particularly noteworthy because many new primitives minimize the use of non-linear operations while significantly increasing affine operations. Nevertheless, we also give security results for more general circuits.

Disclaimer: At CHES 2023 [39], we argued that we achieve security for leakage probability $p = O(1/\sqrt{n})$. It turned out that this is not the case for arbitrary circuits. For this reason, we updated our paper on the IACR Cryptology ePrint Archive [38] with a more fine grained analysis that proves our result for $p = O(1/n^2)$ in general, and $p = O(1)$ for affine circuits.

Fault Resilience. Chapter 4 proposes two fault-resilient cryptographic primitives designed to tolerate specific fault types. These faults can be categorized into two groups: passive faults, resulting from faulty interactions between users and devices, and active faults, where adversaries intentionally introduce faults to compromise cryptographic primitives. As a first step, this chapter only focuses

on faults in secrets, such as keys and passwords, in a black-box manner, with Chapter 5 extending the consideration to internal computation using a circuit as a computational model.

The first primitive, fuzzy asymmetric password-authenticated key exchange (faPAKE, Section 4.1), addresses passive faults in passwords. It allows for key generation between two parties based on an asymmetrically used password, where one party must know the complete password, while the other only needs partial information. The asymmetric approach protects against attacks targeting data theft from servers and the fault-resilience of this protocol ensures a correctly generated key up to a specified fault threshold of the password. The latter property is especially relevant in scenarios involving biometric-based passwords because biometric measurements usually have a certain degree of inaccuracy.

The second primitive, (Section 4.2) based on [94], addresses secure communication in the presence of active faults on the secret key of a cryptographic primitive. Assuming a secure key has already been generated (e.g., with faPAKE), the next step involves ensuring secure communication using this key. A crucial primitive for this is authenticated encryption as it fulfills the three main security requirements discussed in the introduction: confidentiality, authenticity, and integrity. For this reason, the chapter's focus is on the fault resilience of deterministic authenticated encryption schemes proposed by [149] following the three well-known paradigms *Encrypt-and-MAC* (E&M), *Encrypt-then-MAC* (EtM), and *MAC-then-Encrypt* (MtE). These schemes consist of two underlying primitives, namely a simple encryption to hide the message and massage authentication code (MAC) to authenticate the sender. Consequently, such schemes use two cryptographic keys K_e and K_m for the underlying encryption and MAC, respectively. For the fault attack, we distinguish two different scenarios. In the first case, we assume imprecise faults where the adversary can only fault both underlying keys simultaneously and cannot revert a change, whereas in the second case the adversary can do precise faults, that might change only one of the underlying keys. For imprecise faults, we show that all three constructions, i.e., E&M, EtM, and MtE, are fault resilient if both underlying primitives (MAC and encryption) are fault resilient. For the precise faults, we first show that E&M and EtM are generally insecure by presenting two attacks. Then we can show that although the MtE paradigm is also insecure for some specific choices of underlying primitives, for other instantiations it is provably secure against precise faults.

Combined Resilience. Finally, Chapter 5 considers fault and leakage resilience simultaneously. In this section, we introduce two novel compilers that allow for security against both faults and probes, effectively addressing a scenario in which

an adversary can simultaneously tamper with and probe the internal values of a computation. Given that our work rigorously demonstrates that standalone fault and leakage resilience does not suffice to ensure combined resilience, we introduce security properties specifically tailored to prove the combined resilience of our two proposed compilers. These compilers are provably secure against up to e faults in the t-probing and *t/2-region probing model* [10, 126] only using $n = d + e + 1$ shares.

The latter model is a slightly adapted threshold probing model, where the circuit is divided into multiple regions and in each region the adversary can probe up to $t/2$ wires. This model has a better security implication on the random probing model, and therefore on the real-world security as the number of probes in the region probing model increases with the circuit size. In particular, our second compiler is also secure against combined attacks in the random probing model with $O((n^2p)^n)$ security in general and for affine circuits the compiler even achieves $O((np)^n)$.

Conclusion. Our work is an important step to bridge the gap between theoretically secure cryptographic primitives and the practical security of their real-world implementations. Nevertheless, our contribution is still from a theoretical point of view, and based on our findings Chapter 6 proposes some exciting future directions to improve the results of this thesis. In the following (Chapter 2), we give the required basics needed in this work.

2. Preliminaries

Modern cryptography relies on mathematical proofs using probability theory, algebra and number theory. While a detailed introduction to these concepts, such as finite fields [102][1], matrices [122], random variables [124], and polynomials [15], is beyond the scope of this thesis, we will selectively give the notions frequently used in our work. Subsequently, we introduce the foundational principles for private circuits such as their computational models, leakage models, and fault models. Further, we describe the Universal Composability model used in Section 4.1.

Notations. Let X and Y be two random variables defined over a set F. We distinguish between two cases: (i) they are *independent* [101], with $Pr[X = x, Y = y] = Pr[X = x] \cdot Pr[Y = y]$ for all $x, y \in F$, or they are *dependent*, with $Pr[X = x, Y = y] \neq Pr[X = x]Pr[Y = y]$ for at least one $x, y \in F$. The *statistical distance* (or statistical difference [110]) between two random variables X and Y is a measure of how close their distributions are:

$$\Delta(X;Y) := \frac{1}{2}\sum_{x \in F} |\Pr[X = x] - \Pr[Y = x]|$$

This metric satisfies $0 \leq \Delta(X;Y) \leq 1$, and $\Delta(X;Y) = 0$ implies that X and Y have the same distribution.

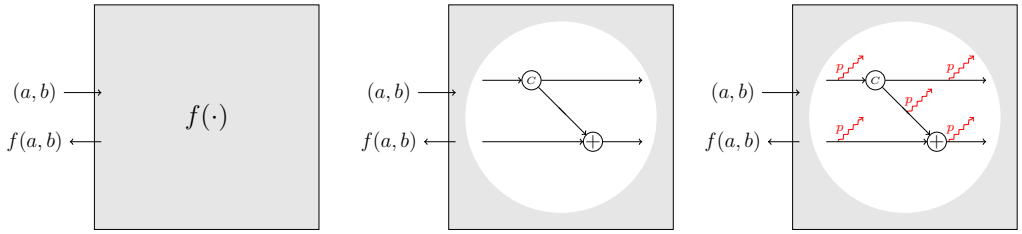
Matrix operations are used to compose two real matrices $A = (a_{i,j})_{i \in [m], j \in [n]}$ and $B = (b_{i,j})_{i \in [k], j \in [l]}$ with index set $[n] := \{0, 1, \ldots, n-1\}$. Specifically, we use *matrix transposition* [142] $(a_{i,j})^T_{i \in [m], j \in [n]} := (a_{j,i})_{i \in [m], j \in [n]}$, matrix multiplication $A \cdot B = (c_{i,j})_{i \in [m], j \in [n]}$ with $k = n$ such that $c_{i,j} = \sum_{l=0}^{n} a_{i,l} \cdot b_{l,j}$, and the *Kronecker product* [174][2] $A \otimes B = (a_{i,j}B)_{i \in [m], j \in [n]}$.

An important matrix is the *Vandermonde matrix* [4][3]. Its i^{th} row has the form $(\alpha_i^0, \alpha_i^1, \alpha_i^2, ..., \alpha_i^d)$ with pairwise different α_i, i.e., $\alpha_i = \alpha_j \Rightarrow i = j$. This matrix, denoted as $V_{n,d} := V_{n,d}[\alpha_0, \ldots, \alpha_{n-1}]$, performs a matrix-vector multiplication with

[1] In honor of its discoverer Évariste Galois, it is sometimes referred to as the *Galois field*.

[2] The operation is named after the German mathematician Leopold Kronecker (1823–1891)

[3] It is a special class of matrices [130], named after Alexandre-Théophile Vandermonde (1735-1796), a french mathematician.



(a) Black box model (b) Computational model (c) Leakage model

Figure 2.1.: Function $f(a, b) = (a, a + b)$ described in the different models

$(v_0, v_1 \dots v_d)$, representing n function values of the polynomial $f(x) = \sum_{i=0}^{d} v_i x^i$:

$$V_{n,d} \cdot (v_0, v_1, \dots, v_d)^T = (f(\alpha_0), f(\alpha_2), ..., f(\alpha_{n-1}))^T$$

It is well-known that this matrix is a bijection [171] when $d = n-1$, and the inverse Vandermonde matrix V^{-1} can interpolate the coefficients v_i of the polynomial from a linear transformation of the function values $f(\alpha_i)$. We write Enc and Dec to refer to encoding and decoding algorithms. Namely, Enc is an algorithm that encodes a value x with $\tilde{x} \leftarrow \mathsf{Enc}(x)$, and Dec is the decoding such that it holds $x = \mathsf{Dec}(\tilde{x})$. Similarly, $\widetilde{\mathsf{Enc}}$ and $\widetilde{\mathsf{Dec}}$ refer to a decryption algorithm. At a high-level the difference between encryption and encoding algorithms is that an encryption uses a secret key to hide a message. For more details we refer to the book of Katz and Lindell "Introduction to Modern Cryptography" [131]. Encoding algorithms do not use keys to hide a message and provide other properties, e.g., error detection [156]. This chapter presents to example encodings frequently used in this work. In the next section, we provide a concise explanation of the circuit concept briefly mentioned in the introduction.

Computational Model. To describe the potential leakage of a cryptographic primitive, Ishai, Sahai, and Wagner [126] modeled the implementation of a primitive as a circuit. The advantage of the representation as a circuit is that it defines all possible intermediate values explicitly because each intermediate value is carried by at least one wire. More precisely, the circuit is a labeled and directed acyclic graph, where each node describes a gate that can have input and output wires represented as incoming and outgoing edges. With fan-in and fan-out we refer to the number of input and output wires of gates, respectively. The gates

Gate	Name	Fan-in	Fan-out	Functionality
⊕	*Addition*	2	$1^{(*)}$	Adds two input variables.
⊖	*Subtraction*	2	$1^{(*)}$	Subtracts the values.
⊙	*Multiplication*	2	$1^{(*)}$	Multiplies two input variables.
@	*Constant*	0	$1^{(*)}$	Outputs the constant value a.
ℝ	*Random*	0	$1^{(*)}$	Outputs a uniform random variable.
ℂ	*Copy*	1	$2^{(*)}$	Copies the input variable.

Table 2.1.: Example gates of a circuit. $^{(*)}$ Sometimes gates are defined with arbitrary fan-out to avoid copy gates

describe operations such as addition and multiplication[4], and Table 2.1 provides all relevant gates for this work. If an output of a gate is the input of the following gate, both gates are connected with a directed edge describing the wire that transmits the output value to the next gate. The labels of edges represent the values carried by the wires, and the circuit defines all intermediate values that might occur during the computation.

In Figure 2.1b, we give an example circuit computing the functionality

$$(a, a + b) \leftarrow f(a, b)$$

of the black box depicted in Figure 2.1a. Using the circuit as the underlying computational model, Ishai et al. proposed multiple leakage models. In the following we describe the model used in this work and discuss some alternative models.

2.1. Leakage Models and Masking[5]

As the leakage is heavily influenced by the inner workings of cryptographic implementations and the black box model only describes the input-output behavior, it is crucial to use a computational model beyond the black box model depicted in Figure 2.1a. For this reason, we use the circuit in Figure 2.1b as a computational model to define the leakage model shown in Figure 2.1c.

Leakage Model. In the literature leakage models bound the leakage to restrict the attacker. Without such limitations, the attacker would have unrestricted access to the circuit, and there would be no chance to prevent the attacker from simply

[4]These operations are defined over a finite field $(+, \cdot, F)$ [102].

[5]This section is based on our works [36, 39] (cf. App B&E) and parts of this section were taken verbatim.

learning all intermediate values in the circuit. This, in turn, would break the scheme's security as the adversary could learn all secrets.[14, 126].

As described by Chari et al. [60], such restrictions are appropriate for two reasons. Firstly, it is reasonable to hope that side-channel measurements incorporate a degree of noise induced by environmental factors, such as fluctuations in temperature and voltage within the laboratory setting. Secondly, the computational processes of an implementation may generate additional noise caused by the characteristics of the device itself, including the material and structural composition of the chip. Such factors can lead to noise sources, such as coupling effects between calculations in physical proximity.

The most famous leakage model is the *t-threshold probing model* [126]. In this model the adversary can probe up to t wires of the circuit to learn their values during the computation. This model was found not to cover the inherent characteristics of leakage resulting from calculations on the hardware, such as those caused by physical defaults [91][6]. That is why several approaches adapt the threshold model[45, 91] to model the leakage more realistically. They considered natural hardware effects such as glitches [144, 145], transitions [13, 66] and couplings [64], and modeled them as extended-probes. It turned out that the physical effect of leakage with all their natural behavior is strongly related to its hardware and might differ a lot with the choice of the device. Therefore, [19] analyzed the leakage that a single probe can measure in a real-world setup. According to their observations, they extended the probes in the threshold model.

Although these models are handy for initial security analyses, it has been found that they do not model scenarios in which an adversary exploits the fact that the leakage increases with the circuit size. For example, *horizontal attacks* [63, 169] combine the information of multiple leakages to reduce the noise of the implementations.

Prouff and Rivain [153] modeled the environmental and computational noise described above with the so-called *noisy leakage model*. Literally, they assume that every wire leaks, but the leakage is only a noisy version of the values carried by the wires. For example, the noise could be sampled from a random distribution such as the Gaussian distribution. Their model defines the noise as a probabilistic leakage function restricted with the Euclidean Norm (or statistical distance used by Duc, Dziembowski, and Faust in [78]). The model was accepted as a theoretical benchmark to define leakage as close to reality as possible. The advantage is that the noisy leakage model describes large classes of known side-channel attacks, but

[6] As mentioned in [91], there are several attacks exploiting defaults, such as glitches, transition-based leakages, and potential couplings.

the disadvantage is that the security proofs become rather complex. There is usually a trade-off between modeling leakage as naturally as possible and simplifying it for security proofs.

That is why Duc et al. [78] compared the noisy leakage model with the *t-region probing model* and *p-random probing model*, both initially introduced by Ishai et al. [126]. In the t-region probing model, the circuit is split into multiple regions, and the adversary can learn the values of up to t wires in each region of the circuit. The favorable aspect of this model is that it is convenient to manipulate in proofs and it implies security in the noisy leakage model with only a loss of the field[7] and region size. Due to the model's simplicity, we adapted it to analyze security against combined attacks where the adversary can simultaneously fault and probe the circuit (Section 2.2 and 5). Given the expected noise in real-world leakage, adversaries cannot consistently attain precise probes, as modeled in the optimistic region probing model.

Consequently, we consider the p-random probing model in Chapter 3 to analyze our leakage resilient constructions (without faults). This model takes into consideration an "all-or-nothing" distribution, where each wire leaks with probability p, and the adversary learns nothing of the wire's value with probability $1 - p$. In Figure 2.1c, we give an example circuit leaking each wire with probability p. The research community followed Duc et al. and explored the security in the random probing model [10, 24, 26, 85] and our work follows their approach as well.

Duc et al. [78] prove that security in the random probing model also implies security in the noisy model but with a security loss of the field size. Since the noisy model is considered as the most realistic one, the research community investigated many related leakage models that are easy to handle for security proofs but still relatively close to the noisy model.

To get a better security implication to the noisy model, and to get rid of the security loss of the field size in particular, Dziembowski, Faust, and Skorski [83] proposed the *p-average random probing model*. This model is a modified version of the random probing model and allows a more powerful class of leakage functions. For example, the adversary is allowed to choose leakage functions where only the average leakage probability is p. In other words, for all n possible values x_i carried by the wire, their leakage probability p_i can be different as long as it is on average $p = \frac{1}{n}\sum_{i=1}^{n} p_i$. This model allows a much larger class of leakage function in contrast to the random probing model where p_i is the same for each x_i. Additionally, the leakage function of the average random probing model always outputs the internal randomness used to decide whether a value leaks. This function does not

[7] It refers to the field of the circuit defined in the computational model.

follow the all-or-nothing paradigm in the random probing model, as the adversary even learns information about the inner computation when the leakage function does not leak the wire's value. Prest et al. [152] alternatively address the field size issue by introducing an alternative metric for the noisy model. Rather than modifying the probing model, where the actual proof takes place, they adjust the metric within the noisy model to establish a strong correlation between security in the random probing model and their new noisy model. This novel metric, the *(Average) Related Error*, represents a worst-case scenario that better suits the all-or-nothing paradigm observed in the random probing model. Following the findings of Prest et al. and Duc et al., the research community has dedicated significant attention to exploring security in the random probing model [10, 24, 26, 85]. Alternatively, Masure et al. [146] and Cassiers et al. [58] recently started investigating low-noise scenarios for specific leakage functions such as the Hamming weight. This research direction is particularly interesting for low-end devices with low noise levels (e.g., [95, 146]). However, our work assumes some noise, as discussed by Chari et al. [60], and also analyzes the security in the random probing model. We provide the basic idea for leakage countermeasures in the following paragraph.

Masking. An important countermeasure against side-channel attacks is *masking*, introduced by Thomas S. Messerges [147]. The basic idea of a masking scheme is to encode the computation such that leakage from the encoded values does not reveal relevant information: A value, e.g., a secret s, is encoded by n random values so-called shares

$$(s_i)_{i\in[n]} \leftarrow \mathsf{Enc}(s)$$

such that any strict subset of the shares is uniformly random and independent of the secret s. There are two widely used masking schemes: *Polynomial masking* [33, 154, 166] and *additive masking* [147][8]. The first one is also called Shamir's secret sharing [166][9] or Reed-Solomon code [156][10]. To encode a secret s, we construct a random polynomial f with degree $n-1$ such that $f(0) = s$ is the secret. For the shares $(s_i)_{i\in[n]}$ we need n pairwise different support points $\alpha_0, \ldots, \alpha_{n-1} \neq 0$, and

[8] Due to space constraints, we omit Inner Product masking. For more details we refer to [82].

[9] In 1979, Adi Shamir [166] originally proposed the encoding as (k, n)-*threshold scheme*. Afterwards, [33] applied the encoding to multi-party computation, and consequentially, [154] to private circuits.

[10] In 1960, Irving S. Reed und Gustave Solomon have already developed a similar encoding. More precisely, Shamir's secret sharing is a special sub class of Reed-Solomon codes [136]. However, they considered error correction and not the privacy property proven by Shamir. In Section 2.2, we also use this error correction property for polynomial masking.

the i^{th} share s_i is defined as $f(\alpha_i)$. As discussed at the beginning of this chapter, the inverse Vandermonde matrix and the n shares $s_i = f(\alpha_i)$ can be used to compute the polynomial f, and therefore decode the secret value $s = f(0)$ as well. For privacy, Shamir [166] proved that any strict subset of $n-1$ shares is uniform random and independent of the secret. For more details we refer to Section 2.2 of this chapter. The maximum sharing number of polynomial masking is limited by the field size $|F|$, due to the pairwise different support points $\alpha_i \in F$.

Alternatively, the maximum sharing number of additive masking is not limited by algebraic restrictions. This encoding generates the shares $s_0, \ldots, s_{n-2}$ uniformly at random and sets $s_{n-1} = s - (\sum_{i=0}^{n-2} s_i)$. This encoding is called *arithmetic masking* [147], if the sum above is an arithmetic sum, and if the sum is a bit-wise xor, the masking is called *Boolean masking* [147]. The decoding is well defined with

$$\mathsf{Dec}((s_i)_{i\in[n]}) = \sum_{i=0}^{n-1} s_i = s \ ,$$

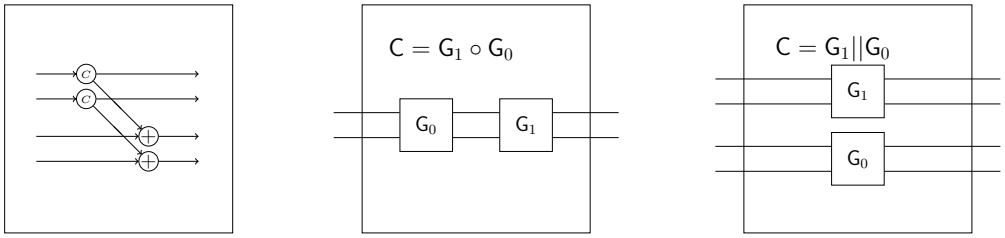
but any strict subset of the shares is uniformly random and independent of s.

Masked Circuits. Instead of directly computing on the secrets, sensitive computation can be protected with *masked circuits* [126] that only compute on the shares to make the leakage of a circuit (almost) independent of its decoded secrets. Ishai et al. [126] provide a generic solution to transform a circuit into a masked one. Initially, they only considered Boolean masking but Rivain and Prouff [159] observed that their security proofs only use the field structure and no specific properties of Boolean masking. Consequently, the results of Ishai et al. [126] hold for computation over any finite field. Their basic idea is a *compiler* that takes as input an unprotected circuit, and replaces each gate with sub circuits called *gadgets*. Let f be a gate (e.g., Table 2.1) with l inputs and k outputs then f is replaced by a gadget G_f with the same functionality $((y_i^0)_{i\in[n]}, \ldots, (y_i^{k-1})_{i\in[n]}) \leftarrow \mathsf{G}_f(\mathsf{Enc}(x^0), \ldots, \mathsf{Enc}(x^{l-1}))$ such that

$$f(x^0, ..., x^{l-1}) = (\mathsf{Dec}(y_i^0)_{i\in[n]}, \ldots, \mathsf{Dec}(y_i^{k-1})_{i\in[n]})$$

for all $(x^0, ..., x^{l-1}) \in F^l$.

Ishai et al. [126] provide gadgets for the addition and the multiplication. Their addition gadget adds share-wise two input sharings $(a_i)_{i\in[n]} \leftarrow \mathsf{Enc}(a)$, $(b_i)_{i\in[n]} \leftarrow \mathsf{Enc}(b)$, and outputs $(c_i)_{i\in[n]}$ with $c_i = a_i + b_i$. Thanks to the linearity of the encoding, correctness follows immediately as $\mathsf{Dec}((c_i)_{i\in[n]}) = a + b$. The multiplication gadget of two inputs, $(a_i)_{i\in[n]}$, $(b_i)_{i\in[n]}$, is more complex. First, it computes the



(a) Masked Circuit (b) Sequential Composition (c) Parallel Composition

Figure 2.2.: Sequential $\mathsf{C} = \mathsf{G}_1 \circ \mathsf{G}_0$ and parallel composition $\mathsf{C} = \mathsf{G}_1 || \mathsf{G}_0$..

Kronecker product $a_i \cdot b_j$ of all shares. This operation results in n^2 intermediate values $a_i b_j$ with $\sum_{i,j=0}^{n-1} a_i b_j = a \cdot b$. To get a valid masking with n additive shares instead of n^2, the gadget compresses the intermediate values to a random encoding by appropriately adding up these values. For security reasons the intermediate sums are blinded by injecting fresh randomness. Formally, the gadget outputs $(c_i)_{i\in[n]}$ with $c_i = a_i b_i + \sum_{j\in[n]\setminus\{i\}}^{n-1}(a_i b_j + r_{i,j})$ and uniform random values $r_{i,j}$ with $r_{i,j} = -r_{j,i}$. Note that the condition $r_{i,j} = -r_{j,i}$ is required for correctness and it follows

$$\sum_{i=0}^{n-1} c_i = \sum_{i,j=0}^{n-1} a_i b_j = a \cdot b \; .$$

Both gadgets are widely used, and a compiler that uses both gadgets is called *ISW-based* compiler. As pointed out in [159] and [69], the security of the multiplication gadgets is based on the assumption that the randomness of the encodings of both inputs are independent. This is realized using refresh gadgets that take as input a masked value and re-randomizes the encoding such that its decoding is the same. Such a refresh gadget can be the ISW multiplication gadget with a fixed encoding of one as the first input $(a_i)_{i\in[n]} = (1, 0 \ldots, 0)$ [18]. Then, for any second input $(b_i)_{i\in[n]}$, the multiplication gadget outputs an encoding $(c_i)_{i\in[n]}$ such that $\mathsf{Enc}((c_i)_{i\in[n]}) = \mathsf{Enc}((b_i)_{i\in[n]})$. Hence, a compiler additionally inserts further randomness to ensure secure composition of multiplication gadgets that get dependent inputs. Barthe et al. [18] present such a compiler which is provably secure in the threshold probing model. In our work, we present an alternative ISW-based compiler secure in the random probing model.

Leakage resilience. A masked circuit is leakage resilient if its leakage reveals almost[11] nothing about its inputs and outputs. Often, this is following a simulation-

[11] With "almost" we refer to the leakage simulator of the following security experiment that is only statistical close to the leakage.

based strategy, where a so-called simulator S creates a leakage distribution that is statistically close to the leakage from the real execution of the circuit evaluation. The simulated leakage is independent of the secrets if S does not need the decoding of the circuit's inputs. Formally, we can define the leakage resilience in the random probing model with a security experiment Leak [85]: Let C be a masked circuit with k input encodings $(x_i)^j_{i\in[n]} \leftarrow \mathsf{Enc}(x^j)$ for $j \in [k]$ with secret values $x^1, \ldots, x^k$ such that for each secret x^j any subset of $t < n$ shares $(x_i)^j_{i\in[n]}$ is uniform random and independent of its secret. Then, the leakage experiment $\mathsf{Leak}(\mathsf{C}, \mathbf{x}, p)$ with leakage probability $p \in [0, 1]$ is defined as follows:

- Each wire of C is added to a set, noted as $\mathcal{L}_p(\mathsf{C})$, with probability p.
- Each secret x^j is encoded $(x_i)^j_{i\in[n]} \leftarrow \mathsf{Enc}(x^j)$.
- Output: $(\mathcal{L}_p(\mathsf{C}), A_{|\mathcal{L}_p(\mathsf{C})})$, where A is the set of the values carried by the wires $\mathcal{L}_p(\mathsf{C})$ of C with inputs $(x_i)^1_{i\in[n]}, \ldots, (x_i)^k_{i\in[n]}$.

The circuit C is (p, ϵ)-*private* if there is a simulation algorithm S that outputs a random variable without knowing the secrets x^j such that the random variable is ϵ-close to the actual output $\mathsf{Leak}(\mathsf{C}, \mathbf{x}, p)$. Alternatively, the circuit has a *security level* p' defining the success probability for an attack to reconstruct one of the secrets x^j with $(\mathcal{L}_p(\mathsf{C}), A_{|\mathcal{L}_p(\mathsf{C})})$. If we have two different security proofs for the same circuit, the proof with better security claims (i.e. smaller statistical distance) is called *tighter*. Further, a proof is tight if there is no proof with a smaller distance.

2.2. Combined Model and Error Detection[12]

Active faults can be categorized into two distinct types: memory and computational ones. Memory faults alone are sufficient for key recovery attacks. That is why, Chapter 4 investigates such attacks by using the black box approach. Moreover, Chapter 5 extends our analysis to encompass computational faults. As in Private Circuit II [125], we use the circuit as the underlying computational model. If the faults in this model are additionally considered as probes, their model also provides results for a combined model where the adversary can fault and probe simultaneously, cf. [75]. However, for our combined model, we allow significantly more probes.

[12] This section is based on our work [36] (cf. App E) and parts of this section were taken verbatim.

Combined Model. We consider each gadget as a region and adapt the region-probing model (cf. Section 2.1) such that the adversary can probe t wires in each gadget and fault e wires of the circuit. We model such faults as functions. An adversary with a set of fault functions, also called *fault class* $\mathcal{F}$, and access to wires of a circuit can change the wire's value during the computation according to faults $f \in \mathcal{F}$. More precisely, a faulted wire gets a value x from its output gate and transforms the value with a function f such that the following gate gets as input the faulted value $f(x, u)$. The fault f is in the fault class $f \in \mathcal{F}$ and u are the values already revealed by probes up to this computational step [36]. We call such a fault *adaptive* as it depends on the observed leakage u. For the combined attack, we modified the security game of Dhooghe and Nikova [75] to allow for adaptive faults and region probes. Formally, a (t, e)-attacker $\mathcal{A}$ with respect to a fault class $\mathcal{F}$ and a target circuit C does the following [36]: $\mathcal{A}$ is given as input the circuit C and outputs

(1) e wires in C to fault with fault functions in $\mathcal{F}$,

(2) t wires of each gadget in C to probe its value, and

(3) two possible circuit inputs x_0, x_1.

Then, the experiment chooses $b \xleftarrow{\$} \{0, 1\}$ uniformly at random, and runs $y'_b \leftarrow \mathsf{C}'(x_b)$ where C' represents the circuit C with the e faults chosen by the attacker. Finally, upon receiving the t wire values that $\mathcal{A}$ requested to probe, the attacker outputs a bit b'. The circuit C is ϵ-secure if for any (t, e)-attacker $\mathcal{A}$, it holds

$$\Pr[b = b'] = 1/2 \text{ and } \Pr[y'_b \in \{\perp, y_b\}] \geq 1 - \epsilon \ ,$$

where $y_b \leftarrow \mathsf{C}(x_b)$ is the output of a non-faulted run of C on x_b.

Error Correction. Additionally to the leakage resilience of polynomial masking shown by Shamir [166], Reed and Salomon [156] proved that this encoding has also error detection and correction properties. As briefly explained in Section 2.1, a secret s is shared with a random polynomial $f \in F[x]$ of degree d and $f(0) = s$. Now, the i-th share is $s_i = f(\alpha_i)$ with pairwise different support points $\alpha_0, \ldots, \alpha_{n-1} \neq 0$. The secret can be determined with any subset of $d + 1$ shares due to polynomial interpolation with the Vandermonde matrix, and further shares are redundant points and allow error detection or correction [156]. Consequently, if an attacker modifies less than $(n-d)/2$ shares, more than half of all possible subsets with $d+1$ different shares are correct, and therefore decode to the correct key. Furthermore,

error detection is possible if at least $d+1$ shares are correct and still decode to the correct value. This does not allow error correction, but detection up to $n-d$ faults by the attacker. Section 5.2 gives a detailed discussion about alternative error correction codes such as duplication. Next we show how to protect circuits with polynomial masking.

Countermeasure. As polynomial masking with degree d polynomials provides fault resilience due to redundancy, *and* leakage resilience due to their d-wise independence, it is a good candidate to protect a circuit against combined attacks. Furthermore, polynomial masking has the same linearity property as the additive masking presented in Section 2.1. Similarly to additive masking, this property allows to construct gadgets for polynomial masking to refresh, add, and multiply encodings [33, 154, 166]. More precisely, polynomial masking hides a secret a in a random polynomial f with $f(0) = a$ and degree d. Because of its linearity, the addition of two polynomials f, g with degree d and $f(0) = a$, $g(0) = b$ results again in a degree t polynomial $h = g + h$ with $h(0) = a + b$. Therefore, the addition gadget of two polynomial sharings $(a_i)_{i\in[n]}$, $(b_i)_{i\in[n]}$ is a simple share-wise addition. To re-randomize polynomial sharings $(a_i)_{i\in[n]}$, the refresh gadget simply generates an encoding of zero $(e_i)_{i\in[n]} \xleftarrow{\$} \mathsf{Enc}(0)$, and share-wise adds both encodings. The multiplication, on the other hand, is slightly more complex. That is, because the product of two polynomials f, g with degree d and $f(0) = a$, $g(0) = b$ indeed yields a polynomial $h = g \cdot h$ with $h(0) = a \cdot b$, however, the degree of h is $2t$ instead of t. Therefore, after computing the polynomial h, the multiplication gadget, e.g.[154], must reduce its degree from $2d$ to d (Figure 5.1b). This degree reduction can be done by using the inverse Vandermonde matrix, which leads to similar re-randomization challenges as in the multiplication gadget of additive masking. In summary, multiplication proceeds in two steps: (1) computation of a share-wise multiplication **SWMult**, and (2) degree reduction of polynomial h via a sub gadget **d-Red**. Due to the intermediate degree $2d$ polynomial h, the method requires $n = 2d + e + 1$ shares to protect the gadget against d probes and e faults. The above techniques allow to construct gadgets for polynomial masking and indeed, polynomial masking has become a well-known countermeasure against side-channel attacks [70, 111, 160]. Furthermore, it is also an error detection code and provides simple protection against faults as shown in Chapter 4.

For our combined resilience, we use polynomial masking to construct a compiler that is simultaneously secure against faults and leakage.

2.3. Universal Composability and Generic Algorithms

Section 4.1 proves the security in the Universal Composability model (UC).

Universal Composability. The security analysis of password-based key exchange protocols (PAKE) [32], e.g., [34, 53, 54, 108, 133], are performed within Canetti's *Universal Composability* (UC) framework [52], This is because of its powerful composability theorem that ensures protocol resilience across diverse environments, including scenarios involving arbitrary parallel executions. In this framework, the protocol operates in the presence of two entities [123]: the *environment* and the *adversary*.[13] In PAKE, the environment provides inputs to the involved parties (client, server, and adversary) and monitors their outputs. Then, the adversary may corrupt the client or server to obtain complete control over a party and access its internal state. And in some cases the adversary can even compromise the server, gaining insight into the servers internal state, often referred to as asymmetric PAKE (aPAKE), e.g., [31, 50, 109, 128].

The UCmodel covers the required security properties as *ideal functionalities*, defining the desired behavior of the protocol, encompassing both client and server actions, and specifying the adversary's capabilities. To assess security, we compare the *real protocol*, namely the actual implemented one, with the *ideal protocol* executing the ideal functionality with *dummy parties*, i.e., client and server, directly forwarding inputs and outputs. Therefore, the dummy parties forward inputs from the environment to the ideal functionality, and vice versa.

Now, a protocol achieves *UC security* if it realizes the ideal protocol, meaning no environment can distinguish between the two protocols in polynomial time with noticable success probability. Formally, they are indistinguishable except with negligible probability (e.g., [131], Chap. 3). Notably, this holds true even when the environment has complete control over the adversary, running all attacks defined by the ideal functionality. For instance, the Canetti et al. definition of the ideal functionality for PAKE [54] gets the client's and server's secret passwords as inputs and generates two random keys as outputs. The functionality ensures the equality of both keys if and only if the passwords match.

When considering protocol extensions, two alternative investigations were considered. Firstly, assuming an adversary can steal data from the server (aPAKE), e.g., [31, 50, 109, 128], requires extending the functionality with properties enabling the environment to select an adversary for querying additional server information. Alternatively, allowing clients to use passwords with a defined fault

[13] The following text describes both parties only for our specific setting. For a more general explanation we would like to refer to more formal explanations, e.g., [123].

threshold (fPAKE). e.g., [81], requires a corresponding extension of the functionality to include this aspect. In Section 4.1, we combine both properties to develop a protocol that provides both features simultaneously. Such UC poofs often require idealized assumptions, e.g., generic algorithms such as used in the random oracle model, generic group model, and ideal cipher model [121]. Our work in Section 4.1 is not an exception.

Generic Algorithms. A famous idealization is the *random oracle* [30]. This oracle describes a theoretical black box that outputs a value chosen uniformly at random from its output set for each unique query. This idealization is frequently employed to model hash functions in cryptographic proofs.[14]. A similar model, that additionally preserves a group structure, is the *Generic Group Model* (GGM). This oracle executes the group operations for all parties of the protocol, and consequently, even the adversary has to query the oracle to compute a group operation. Formally, the GGM outputs a random group element identifier id_a for any queried exponent a. Further, it also answers group operation queries $id_a \cdot id_b$. If there were queries (a, ida), (b, idb), (c, idc) with $c = a + b$, it outputs idc, otherwise idc is an independent random value. The same holds for queries $(id_a)^b$ with idc if there exist entries (a, ida), (c, idc) with $c = ab$. In contrast to a specifically initiated group structure, the advantage of the generic group model is that a simulator, e.g. the one used in the UC framework, is in charge of all group operations. For example, this allows the simulator to generate an output of a group operation before knowing the operation's input. Another idealized primitive used in Section 4.1 is an *ideal cipher* [29]. This oracle is a random permutation for each key. In other words it outputs an uniform random value $c_{m,k}$ for each encryption query (k, m) with key k and plaintext m by ensuring injectivity over m for each key k. Consequently, it also outputs the correct plaintext m for the corresponding decryption query $(k, c_{c,m})$. The idealized cipher supports the same advantage as mentioned for the GGM, i.e., the simulator is in charge of the cipher's input-output behavior. For example, it could generate a ciphertext before it knows its plaintext.

[14] This assumption contradicts a bit the definition of hash functions as they are deterministic but as Bellare and Rogaway wrote: "Although standard hash functions are too structured to make good random oracles [...], one doesn't have to look much further[...]" [30].

3. Leakage Resilience

In the context of masked circuits, the general idea of leakage resilience is to show that the circuit's leakage is independent of the internal secrets, i.e., the leakage is independent of the decodings s of the masked circuit's inputs. As described in Section 2.1, the security level and the (p, ϵ)-privacy are two established notions to describe this resilience. It is easy to compute the security level of a circuit with n wires carrying a single additive masking $(s_i)_{i\in[n]} \leftarrow \mathsf{Enc}(s)$, i.e., it is easy to analyze the experiment Leak described in Section 2.1. In the random probing model, each wire of the circuit leaks its carried value with probability p. As mentioned in Section 2.1, any strict subset of the shares $(s_i)_{i\in[n]}$ is independent of the secret s. Hence, the probability that the leakage is independent of the secret is easy to compute in the p-random probing model: if at least one wire does not leak its carried share, the leakage is independent of the decoding, and this happens with probability $1 - p^n$. Said differently, the circuit has a security level p^n.

This analysis becomes more complex if we consider gadgets or even larger circuits, such as the compiler's outputs described in Section 2.1. If the simulator can always simulate the circuit's leakage, the simulator is *perfect*, and the circuit has perfect security. Most of the time, however, the circuits are not perfectly secure, and we are interested in the security level or the (p, ϵ)-privacy of such a circuit. A good upper bound for the security level of a gadget is a simulator that only simulates the leakage if it can simulate the leakage perfectly; otherwise, the simulator aborts with some probability p'. Consequently, this abort probability p' is a good upper bound for the security level, as the leakage is independent of the encodings with probability $1 - p'$. A meaningless simulator is one that continuously aborts ($p' = 1$), and this gives the trivial upper bound 1 for the security level. Usually, masked circuits have a security level in $p' \in [p^n, 1]$ as the lower bound is the simple probability that a single encoding leaks its decoding, namely p^n.

Challenges. The main objectives of our work in [57] and [39] are to design efficient circuits with small security level, and to develop proof techniques that are simple and as tight as possible. Efficiency in terms of circuit complexity and randomness cost is crucial to ensure practicality of the gadgets application, as randomness is costly, and a low circuit complexity leads to fast and efficient im-

plementations. We propose an optimized ISW-based compiler in Section 3.1.2, and additionally suggest two different proof techniques to examine its leakage resilience in Section 3.1.1 and Section 3.1.2. Those proofs are used to prove the security level or (p, ϵ)-privacy of masked circuits, and in particular the masked circuits generated by a compiler described in Section 2.1. As those masked circuits consist of composed gadgets, e.g., multiplication, addition, and refresh gadgets, the initial step in the proof involves examining the security of each individual gadget. Given the fact that the composition of provably secure gadgets does not imply security of the resulting circuit e.g. [18, 69], such a proof approach requires security theorems as well. The idea of such theorems is to guarantee secure compositions of multiple securely masked sub-circuits. For instance, those theorems, combined with the security proofs of the individual gadgets, allow us to prove the overall security of its composition, e.g., the outputs of our compiler. Another goal of such theorems is tightness to accurately determine the security, and efficiency to verify larger classes of gadgets and circuits. Especially in contrast to the threshold probing model, the straightforward security analysis of a circuit in the random probing model requires checking the leakages of all possible wire combinations, and hence, the proof complexity is 2^n for a circuit of size n.[1]

3.1. Contribution

This chapter introduces two approaches for analyzing the random probing security of masked circuits, and specifically those generated by the widely used ISW-based compilers. Therefore, Section 3.1.2 presents the Probe Distribution Table (PDT) of [57], which allows us to investigate circuits with up to six shares. We conducted a case study on a masked AES S-box. Section 3.1.2 gives the alternative technique, which helps to study the security of circuits with any number of shares. For this purpose, the leakage analysis is reduced to a graph problem, the so-called *Dependency Graph (DG)* of [39]. This technique allows us to study an adapted ISW-based compiler that we designed to have low latency and randomness cost.

3.1.1. Probe Distribution Table

The *Probe Distribution Table* (PDT) of [57] defines a matrix for analyzing composed circuits' leakage resilience, e.g., security level. Concretely, we give composi-

[1] For example, a circuit with n wires has 2^n different leakage combinations if each wire independently leaks its value with probability $p > 0$. Contrarily to the random model, the t threshold model only considers $n!/(n-t)!$ different leakage combinations as it only consideres leakages of at most t wires.

tion theorems that allow us to analyze the leakage resilience of composed circuits computing a simple matrix multiplication or the Kronecker product of the sub-circuits' PDTs. As discussed in the beginning of this chapter, this feature enables a gadget-wise proof of masked circuits. This section shows that this composition technique results in almost tight security results, and allows us to analyze larger circuits such as the masked AES S-box with up to six shares.

Concrete Results. Figure 3.1 shows the security level of a masked circuit computing x^3 for an input sharing $(x_i)_{i\in[n]}$. In particular, this computation is done by a share-wise[2] squaring $z_i \leftarrow x_i^2$. Then, we refresh $(z_i)_{i\in[n]}$ and multiply it again with $(x_i)_{i\in[n]}$ using the ISW multiplication. The first plot uses the PDT composition approach, and the second one evaluates the complete circuit without any composition techniques. It is easy to see that the tightness loss of our composition method is relative low. The benefit of the PDT is that the verification is only exponential in the gadget size and linear in the number of gadgets used by the masked circuit. This allows us to also verify larger circuits such as the full AES S-box, also depicted in Figure 3.1.

Proof Idea. For the PDT, the wires of a gadget G are grouped by output wires O, input wires I, and intermediate wires W that are neither input nor output wires. The PDT definition uses perfect leakage simulators simulating the leakage of O and I with the help of input shares $I' \subset I$. This also means that the simulator does not abort and also perfectly simulates the leakage if it needs all shares. Since we analyze the security in the p-random probing model, we assume that each value of the intermediate wires of the gadget leaks with probability p, noted as $\mathcal{L}_p(\mathsf{G}) \subset W$. Consequently, its leakage simulator requires different input shares for each possible leakage combination $\mathcal{L}_p(\mathsf{G})$, and hence, the set of required input shares I' is randomized by the internal leakage $\mathcal{L}_p(\mathsf{G})$. For a better understanding let $O' \subset O$ be an arbitrary subset of leaked output wires. Then, the *probe distribution* of the gadget G with respect to O' gives the probabilities that the simulator needs exactly I' to simulate the internal leakage $\mathcal{L}_p(\mathsf{G})$ randomized by p and O'. Only considering the individual gadget security we can compute its security level by adding all probabilities of the probe distribution where I' includes at least one input sharing that allows to compute its secret decoding.

However, we are interested in composition results as well, and this is where the PDT comes into account. Considering a sequential composition of two gadget G_1

[2]We have considered characteristic two fields for our example, and in those fields the square operation is linear. Hence, the gadget is a share-wise operation as well. Note that this is only an example and the tightness of the proof does not depend on the field characteristic.

and G_2, such that the output of G_1 becomes the input of G_2, we can simulate the leakage of both with the individual gadgets leakage simulator S_1 and S_2, respectively. Starting with S_2, the leakage simulator needs some input shares for a perfect simulation, and since the input sharings are the output sharings of G_1 we can simulate the required shares of S_2 using the simulator S_1. In other words, we the simulators are composed similarly to the gadgets. This strategy illustrates that the simulation of the set of simulated output wires depends on the next gadget, and not on the actual one. In other words we need the probe distribution for all possible output wire sets O', and this is why we the PDT is a matrix where each column represents a probe distribution with respect to an O' such that all possible output combinations are represented $O' \subset O$. In particular, for each possible leakage of output wires O', we compute the probability for each subset of input wires I' that the leakage simulator exactly needs this set for a perfect simulation. Since we compute the distribution for each possible leakage of output values, we can use the PDTs to compute the security of composed gadgets. In [57], we have proven that the PDT of parallel-composed gadgets is a simple Kronecker product of the PDTs of the underlying gadgets, and the approximated PDT of two sequentially-composed gadgets is a simple matrix multiplication of the gadgets' PDTs.

Theorem (Informal). *The leakage resilience of gadgets can be defined as a matrix, and the resilience of composed gadgets results from simple matrix operations.*

- [57], Theorems 1 & 2.

The theorem allows for a verification of the full circuit that is not exponential in the circuit size but only exponential in the gadget size. Although this is already a huge improvement, we further improved the verification complexity with statistical approximations. Therefore, the paper provides the tool *Sampled Testing of the RAndom Probing Security* (STRAPS) to analyze the gadgets using Monte-Carlo approximations. Considering our composition results, we can see that the parallel composition is yet to be made tight. In the next section, we evaluate an alternative approach that allows also to analyze circuits with a larger number of shares, and that is tight in terms of parallel composed refresh gadgets, and almost tight for affine circuits.

3.1.2. Dependency Graph[3]

Similar to the PDT approach in Section 3.1.1, the *Dependency Graph* (DG) of [39] describes the independence of leakage and secret values. In contrast to the first

[3]Parts of this section were taken verbatim from our work [39] (cf. App B).

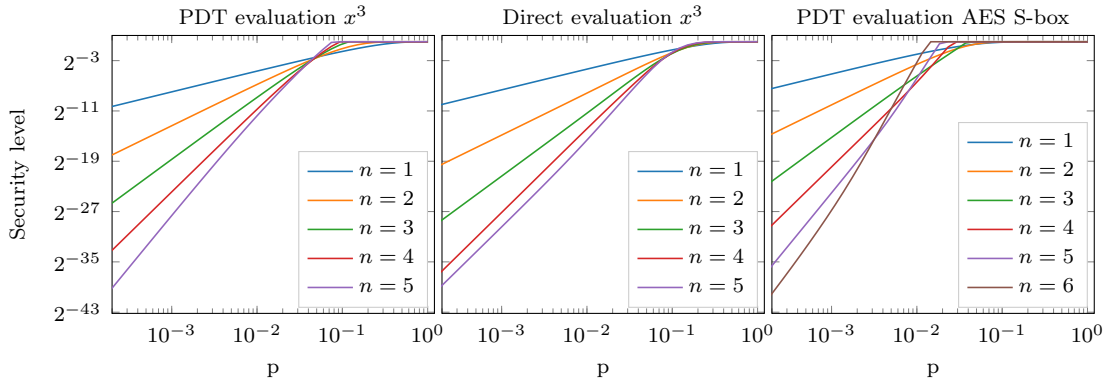


Figure 3.1.: The security upper bounds of masked circuits. The first two plots give the security of a cubing operation. The left plot comes from PDT composition while the middle plot is a direct security evaluation of the full circuit. The right plot shows the security of the full AES S-Box. (Figures of [57])

approach, the DG reduces the gadget-wise simulation of parallel compositions, leading to non-tight approximations for the security. The DG enables a security proof without the need of simulators for the refresh gadgets. A further feature is that the DG allows security proofs for an arbitrary large number of shares and can verify the security of our optimized ISW-based compiler.

Concrete Results. We propose a slightly modified ISW-based compiler C [39] to reduce the latency of the multiplication from $O(n)$ to $O(\log(n))$. Our compiler inserts randomness between each multiplication and addition by essentially using a simple refresh gadget. This refresh gadget takes as an input $(x_i)_{i\in[n]}$ and generates n random values $r_0, \ldots, r_{n-1}$ and adds them to the input in two steps. First, it computes $b_i \leftarrow x_i + r_i$, then, it subtracts r_{i-1} from b_i such that $y_i \leftarrow b_i - r_{(i-1 \mod n)}$ for all $i \in [n]$, and finally it outputs $(y_i)_{i\in[n]}$. With the dependency graph, we were able to asses the leakage resilience of this compiler.

Theorem (Informal). *Circuits masked with our compiler* $\widehat{\mathsf{C}} \leftarrow C(\mathsf{C})$ *are* (p, ϵ)*-private with* $\epsilon = |\mathsf{C}|(12p)^n$ *if* C *is affine; otherwise,* $\epsilon = |\mathsf{C}|8[1 - (1 - \sqrt{3p})^{8n}]^n)$.

- [39], Theorem 4.

This allows a leakage probability of $p = O(1)$ for affine Circuits and $p = O(n^{-2})$ for general circuits (Fixed Theorem 5, [38]). Here, we refer to the fixed version in Chapter B or on eprint [38] as it differs from the original claim for general circuits with $p = O(\sqrt{n})$ ([39], Flawed Theorem 5).

Disclaimer: As underlined in Section 1.1 our general claim of Theorem 5 at CHES 2023 [39] had to be updated on the IACR Cryptology ePrint Archive [38]. First, we had a bug in the proof that achieved a optimistic security for leakage probability $p = O(1/\sqrt{n})$. Then fixing the bug, it turned out that this is not the case for arbitrary circuits. Consequently, we updated our paper with a more fine grained analysis that proves our result for $p = O(1/n^2)$ in general, and $p = O(1)$ for affine circuits.

In the following paragraph we give the proof idea of the compiler and a high level definition of the DG used in the proof.

Proof Idea. As already mentioned before, the intermediate values of a circuit can be defined as a set of random variables T. Due to the intermediate computations, there are mutual dependencies between the random variables. The dependency graph represents those dependencies of the random variables in T caused by the intermediate computation of a masked circuit. Each edge is labeled with intermediate values of the circuit such that if any subset S of intermediate variables build a subgraph with no loops, the set $S \subset T$ consists of random variables that are independent of the decoding of the circuit's input and output. Hence, we only need to compute the probability that a sub graph of the dependency graph generates such loops if S is the randomized set of leakage.

We constructed dependency graphs for each gadget used by the compiler given above, and gave two properties that allow compositions of dependency graphs: *gluing* and *hollow*. Gluing defines how to compose the dependency graphs of two gadgets if the gadgets are composed. If an input and output wire become the same wire because of its composition, we merge the corresponding two edges of both dependency graphs to become one edge. The hollow structure further defines how the edges of a dependency graph are structured such that gluing, is always possible, and the composed graph is always a dependency graph. Formally, hollow indicates that the edges of the input and output shares are arranged in a circle with the same structure. Due to the unified structure, we can always glue the corresponding edges together without changing the arrangements of the edges. A further property of the hollow structure is that the edges are arranged in such a way that even after the composition of two graphs it holds that secret dependent random values always describe a loop in the graph. The graph composition is useful for two reasons. First, we get dependency graphs for any masked circuit generated by our compiler C. Second, we can reduce the security proof of leakage resilience to problems in graph theory. In particular, we only need to compute the probability that the leakage describe loops in the graph.

Theorem (Informal). *The leakage resilience of gadgets can be defined as a graph, so-called dependency graph, and the resilience of composed gadgets results from gluing their dependency graphs, a common construction in topology (adjunction space).*

- [39], Definition 4, Proposition 8 & 10.

Both composition results, the PDT and the DG are suitable for security proofs of ISW-based constructions in the random probing model. In the following we discuss alternative approaches, and discuss how they are related to our work.

3.2. Related Work[4]

The work gives security results for the random probing model and proposes a provable secure compiler with low randomness cost and latency. In addition to our results, this area also considers many other interesting research directions to develop proof techniques and compilers.

Related Proof Techniques. There are many different approaches [10, 24, 25, 26, 46] for analyzing security in the random probing model. They also investigate composability based on gadget-wise simulations. In particular, they introduce definitions based on counting the number of input and output wires of gadgets that are required to simulate the leakage.

Similar to the DGs, the work of [24, 25, 26] analyze compilers for generic number of shares. Their approach is suitable for special gadget following the modular approach originally proposed by Ananth et al. [9]. However, their approach leads to circuits with a costly randomness complexity of at least $O(n^{2.4})$ for affine gadgets.

Related Compiler. Our compiler improves the compiler [85] presented at Asiacrypt 2019. We modified their compiler such that the multiplication has a latency $O(\log(n))$ and the latency of the refresh gadget is constant instead of $O(n)$. Their compiler is leakage resilient for a leakage probability $p = O(1/n^2)$ while our compiler tolerates $p = O(1)$ for affine circuits, and only requires $p = O(1/n^2)$ in the non-affine areas.

At EUROCRYPT 2016, Andrychowicz et al. [10] presented a first compiler with constant leakage probability $p = O(1)$ using expander graphs. This is more of a feasibility result since expander graphs require a high number of shares. Two years later, Goudarzi et al. [113] gave a compiler for polynomial masking requiring

[4]Parts of this section were taken verbatim from our work [39] (cf. App B).

noise $p = O(1/\log(n))$. They presented an NTT-based secure multiplication with complexity $O(n\log(n))$. The compiler was further improved in [114] to allow more general fields $\mathbb{F}$ and complexity $\Theta(n\log(n))$. They use the additive FFT algorithm proposed by Gao and Mateer in 2010 [105] to avoid the limitations of the classical NTT. Despite the benefits of polynomial masking, the field size still restricts the number of shares $n < |\mathbb{F}|$ due to the share-wise different support points of polynomial masking. Especially for affine circuits, our compiler is more efficient. For example, our refresh gadget has linear complexity and does not use multiplication gates, while [114] requires $n\log(n)/2$ multiplications. Although our general compiler requires more noise $O(1/n^2)$, it is more secure for affine circuits. It tolerates a noise of $O(1)$ instead of $O(1/\log(n))$. Regarding non-affine circuits, [114] has better complexities with respect to efficiency and security owing to their NTT-based multiplication. However, our compiler uses a slightly modified ISW multiplication such that it has a latency $O(\log(n))$.

To allow security for a constant leakage probability p, Ananth et al. [9] proposed a *modular approach* how to compose a secure compiler multiple times. Instead of providing gadgets for an arbitrary number of shares n, they use gadgets for a fix number of shares x. Then they show, how the security increases if they use the compiler recursively such that they take the output of the compiler and compile it again. Applying this method arbitrarily k times, the resulting circuit masked into $n = x^k$ shares. Finally, this approach was further improved in several follow up works [24, 25, 26], but nonetheless this approach leads to relatively costly circuits with randomness complexity of at least $O(n^{2.4})$ for affine and non-affine circuits, while our compiler only requires $O(n)$ and $O(n^2)$, respectively. In detail, our countermeasure for affine circuits achieves $O(p^n)$-security with linear complexity $O(n)$. This also improves the compiler of Dziembowski et al. [85] only achieving $O(\sqrt{p}^n)$-security. The use of an ISW-based compiler was motivated by our other work. This work also used the widely used ISW multiplication for reasonable share number ($2 \geq n \geq 32$) and noise parameters $p \geq 10^{-4}$ [57]. It shows that the AES S-box masked with an ISW-based compiler using $n = 6$ shares and a realistic leakage probability of $p \geq 10^{-4}$ ensures about the same security as the securely proven construction of Belaïd et al. [26] using $n = 27$ shares.

4. Fault Resilient Implementations

This chapter is based on two publications [90, 94]. Section 4.1 focuses on fuzzy passwords caused by faulty interactions between user and device [90]. This work considers protocols that allows to establish a cryptographic key between a user, so-called client, and a server from a client-provided low-entropy password. These protocols should even work if the client inputs a fuzzy password, i.e., a password that is faulty up to a certain extent. The fuzzy password could be a client's iris scan or fingerprint that may differ slightly each time due to inaccurate scans. Section 4.2 focuses on key faults stored at the device (active faults in the storage) [94]. In more detail, we analyze authenticated communication where the adversary can get multiple ciphertext/plaintext pairs with different related-keys caused by key faults. For more general faults where the adversary can fault intermediate computational values, we refer to Chapter 5.

4.1. Fuzzy Asymmetric Password-Authenticated Key Exchange

Password-Authenticated Key Exchange (PAKE) protocols [29, 32, 34, 49, 53, 108, 132, 133] provide secure communication channels using low-entropy passwords. In particular, *Fuzzy Asymmetric Password-Authenticated Key Exchange* (faPAKE) combines the properties of *Asymmetric Password-Authenticated Key Exchange* (aPAKE) [31, 50, 109, 128] and *Fuzzy Password-Authenticated Key Exchange* (fPAKE) [81].

More precisely, fPAKE allows clients and (trusted) servers to securely generate a common key even if the password of the client is faulty. With trusted, we mean that we allow the server to know the password to verify whether the client uses a valid password. Alternatively, aPAKE allows clients and untrusted servers to securely generate a common key if the client has the correct (non faulty) password. With asymmetric we mean that we also protect the client's password, even if the secret data of the server is stolen – the server is untrusted. For this reason, the server does not store the entire password but partial information that is sufficient to verify whether the client has the correct password or not but does not immediately reveal

the password. In the non-fuzzy case this is often done via hash functions. Both classes of protocols are well studied in the literature. Despite that, our work is the first one that combines both concepts and constructs a framework faPAKE to allow the client to also use fuzzy passwords *without* storing the full password on the server side. In the fuzzy setting, however, it is not sufficient to store a single hash of the password on server side, as the hash value is not sufficient to verify fuzzy passwords.

Challenges. Our constructions are provably secure in the Universal Composability (UC) model [52]. As mentioned in Chapter 2, this model is used to prove the security of modular compositions to build provable secure systems out of multiple UC-secure protocols. In order to prove the security of a cryptographic primitive in the UC model we typically proceed as follows: We first describe the ideal functionality of a cryptographic primitive, representing the ideal behavior that the primitive aims to achieve in the real world. The ideal behavior includes the input-output behavior and, if required, the attacks an adversary might execute on the primitive. Then, to prove the actual security of a concrete cryptographic implementation, we must show that the behavior of the implementation is computationally indistinguishable from the ideal functionality. Since our construction is the first provable UC-secure faPAKE primitive, our contribution was twofold. First, we had to define the ideal functionality for such primitives to provide provable security, and second, we had to construct a protocol that fulfills the challenging combination of being asymmetric to hide password information on server's side, and similarly being fuzzy to allow the server also accept fuzzy passwords. Next, we give the basic idea to combine both properties.

Password Correction. The high-level idea of our PAKE protocol is to generate a secret sharing that decodes to a random key, and for each correct password bit, the client receives a correct share of the key. As in Chapter 3, we could use additive masking to hide the random key such that each bit of the password has a corresponding share. Now, for each correct bit of the password, the server outputs the corresponding share. Otherwise, the server outputs a uniform random value. Due to the properties of additive masking, it is easy to see that the client can decode the correct key if the password is valid, but if at least one bit is wrong, the server's output is independent of the secret shared key. The high-level idea of our protocol employs the same strategy to conceal the key. Additionally, we permit the client to discover the key if a certain threshold, denoted as t, of incorrect bits is not exceeded. We avoid this with polynomial masking (Chapter 2)

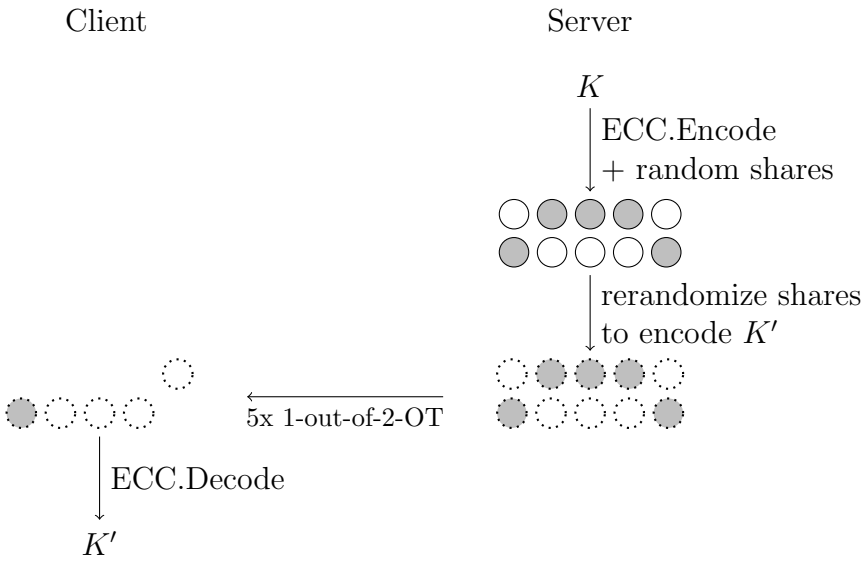


Figure 4.1.: faPAKE Protocol (Figure of [90])

that combines the privacy property of Shamir's secret sharing [166] and the error correction property of Reed-Salomon codes [156]. Again, a value v is shared with a random polynomial $f \in F[x]$ of degree d such that the i-th share v_i is $f(\alpha_i)$ with pairwise different support points $\alpha_0, \ldots, \alpha_{n-1} \neq 0$, and the secret is the constant term of the polynomial $f(0) = v$. The secret can be determined with any subset of $d+1$ shares due to polynomial interpolation. If an adversary faults less than $(n-d)/2$ shares, the redundant function values allow for an error correction. Hence, if we use polynomial masking, the shares are independent of the key if the password is wrong (more than $n-d$ bits are wrong), otherwise the client can reconstruct the key if the fuzzy password is close to the correct one (less than $(n-d)/2$ bits of the password are wrong).

Concrete Results. Our contribution is twofold, we give formal security requirements and secure faPAKE protocols. The security requirements are defined via an ideal functionality that combines the functionalities of fPAKE [81] and aPAKE [109]. The functionality defines the input-output behaviour, and outputs the same random key for the server and the client if the fuzzy password of the client is close to the correct one. Further, the functionality has additional interfaces to cover adversarial scenarios, such as a adversary simply guessing passwords or stealing data of the server. The second szenario is often referred to as untrusted server.

Our functionality covers multiple realistic attacks. For example, we cover password-

guessing 'TestPWD', where the adversary is bound to one guess per protocol run. Since we assume an untrusted server, we cover similar security requirements as in aPAKE. We also allows the adversary to compromise the server and to run dictionary attacks on the password file. This attack is covered by an interface of the functionality named StealPwdFile. The interface allows an attacker an unbounded number of password guesses via the OfflineTestPwd interface. Alternatively, the adversary can simulate the server once the password file is stolen. This issue is defined via the Impersonate interface. Compared to the functionality of the non-fuzzy version, our interfaces above also allow fuzzy matching, i.e., a password guess is considered valid not only upon an exact match but also if the guess is sufficiently close. In particular, the functionality is parametrized with a success threshold t that enables a successful key exchange with up to t faults in the password, and a security threshold s that does not allow the attacker to exchange a key successfully with more than s faults in the password. For example, polynomial masking with degree d and $n = d + 2t + 1$ allows t faults in the passwords, and still allows for a secure error correction. However, it only allows the adversary to learn up to d shares without learning the secret. Hence, it has a security parameter $s = 2t$. However, we need at least $n = d + 2t + 1$ shares to correct up to t faults. This gives the distance of the thresholds for correctness and security $s - t = t$.[1] Having defined the ideal functionality, we propose two secure constructions. The first one uses existing aPAKE protocols and the second one uses error correcting codes as discussed above.

The first construction is a trivial construction, where the servers store all hashes of valid but fuzzy versions of the password. Then faPAKE runs an aPAKE protocol multiple times with all possible hashes. If the fuzzy password of the client is close to the correct one, one of the multiple aPAKE runs gets the correct match of the fuzzy password and its hash and generates a correct key that the client and server can agree on. Unfortunately, this protocol is problematic for three reasons. First, it requires extremely huge password files as the file grows asymptotically. Second, the communication overhead strongly depends on the size of the password. Third, we can show that the protocol has weaker security properties. Notwithstanding the previous points, it is practical for applications where it is sufficient to allow only a few faults in the password. For example, it can correct the capitalization of

[1] The specified distance corresponds precisely to the discrepancy arising from successful error correction and detection. Our protocol needs the correction property for ensuring correctness, yet error detection can be accomplished with a smaller number of shares. While detection alone may not suffice for decoding to the correct value, it has the potential to get partial information about the value. Consequently, we cannot guarantee security if the adversary learns more than $d + t$ shares.

the first letter of passwords as done in Facebook's authentication protocol [148].

Our second protocol circumvents such problems with the error correcting property of Shamir's secret sharing. The fuzziness of passwords is measured with respect to their Hamming distance and the protocol runs for each password bit a 1-out-of-2 *Oblivious Transfer* (OT) [62]. For each correct password bit the client learns a correct share, and otherwise, it only gets an independent random value which is indistinguishable from the correct share. The protocol is inspired by a fuzzy symmetric PAKE protocol [81]. The oblivious transfer prevents the server from learning the client's password, and the error correction allows the client to get the correct cryptographic key even if it uses a slightly faulted password. The high-level idea is depicted in Figure 4.1: During the initial registration of its n bit password $(b_i)_{i\in[n]}$ password, the client chooses a uniform random key K and secret shares it with Shamir's secret sharing $(k_i)_{i\in[n]} \leftarrow \mathsf{Enc}(K)$. Then, the server generates a table with two rows and n columns

$$(s_{0,0}, s_{1,0}), (s_{0,1}, s_{1,1}), \ldots (s_{0,l-1}, s_{1,l-1})$$

such that $s_{b_i,i} = k_i$ for all $i \in [n]$, and the other entries are independent random values indistinguishable from the shares. Afterwards, the server deletes the password, and hence, it does not know which entry is a share and which is only a random value. The table and the key constitute the password file of the server, and during the Key Exchange phase, the client only learns $s_{b_i,i}$ if the bit of his password was correct. Otherwise, it only learns $s_{1-b_i,i}$, i.e., the independently chosen share for this bit position. To avoid that the server leaks the full password file by only two queries (e.g., $pw = 0^n$ and 1^n), the shares have to be re-randomized after each query, or in other words, the server only uses randomized versions of the shares for each password query. This is analyzed in the Generic Group Model (GGM) which ensures that a share and an independent random value are indistinguishable. The reason for this is that a simple output of the GGM is uniform random. The only dependence between the oracle requests appear when the outputs are related via the internal group structure. In detail, the shares are hidden in the exponent of a group generator g, and for each key exchange all shares are randomized by a uniform random exponent k'. Due to the linearity of polynomial masking, the shares decode to the new randomized key $K' = K^{k'}$. Hence, the server can randomize the password file without knowing the password but still knows the new resulting random key K'. However, we must ensure that the server does not learn the password during the key exchange. With this in mind we initiated the protocol with 1-out-of-2 oblivious transfer. At a high level, 1-out-of-2 oblivious transfer enables the server to send one of both secrets for one password position, while the client

can choose which secret it wants to receive. OT guarantees two security properties. First, the server does not learn the client's choice, and second, the client does not learn the other secret. The first property prevents the server from learning the password choice of the client, and the second prevents the client from learning the password file of the server.

Further, the protocol uses the ideal cipher model, to encrypt the randomized shares in the beginning of the key exchange. This allows the client to check whether the server really has the password file, since it commits to it with the encryption. This protects the protocol from an attacker that simulates the server without knowing the password file.

Theorem (Informal). *There is a UC secure* faPAKE *protocol using an UC secure cipher, an UC secure oblivious transfer, and a generic group.*

- [90], Theorem 4.1.

4.2. On the Related-Key Attack Security of Authenticated Encryption Schemes

In the preceding section, we demonstrated a method for generating secret keys in the presence of faulty inputs. In contrast to the previous section which focused on the secure generation of a secret key via faPAKE, this section focuses on secure communication in the presence of key-related faults. To achieve this, authenticated encryption plays a pivotal role as it combines the three security properties mentioned in the introduction: (1) confidentiality to hide the actual message, (2) authenticity to be sure of the sender's identity, and (3) integrity to be convinced that the message was not changed by an attacker.

Authenticated encryption, particularly in the form of *Authenticated Encryption with Associated Data* (AEAD), is an important primitive that is widely used today, e.g., in internet protocols like TLS 1.3 [157]. This significance was underscored by its inclusion in the CAESAR competition [37] and the NIST standardization process on lightweight cryptography [150]. Notably, the NIST standardization process prioritizes nonce-based schemes [161], which offer a balance of security and efficiency.

Nonce-based schemes dispense with the need for costly randomness and, instead, utilize a simple nonce, like a straightforward counter. To ensure security, these schemes require only that nonces are not reused with the same key [161]. Consequently, in our work, we are interested to examine the fault resilience of such

schemes, and in particular, assessing the security of the *N1* scheme (Encrypt-and-MAC), *N2* scheme (Encrypt-then-MAC), and *N3* scheme (MAC-then-Encrypt), initially proposed by Namprempre, Rogaway, and Shrimpton [149] at EUROCRYPT'14.

All three schemes are given in Figure 4.2. They consist of an encryption component and a MAC (Message Authentication Code), each initialized with a randomly generated key (K_e and K_m, respectively). Subsequently, they receive a nonce (N), associated data (A), and a message (M) as input and transform this information into a ciphertext (C). Namprempre et al. [149] basically show that all three schemes are secure nonce-based AEAD schemes if the underlying encryption and MAC are secure, and the MAC algorithm is additionally a pseudo random function (PRF). To evaluate their security in the context of an adversary tampering with the key, we model this scenario using a black-box approach, often referred to as *Related Key Attacks*.

Related Key Attacks describe an adversary actively tampering with the key[2]. In a seminal work by Bellare and Kohno [28], these attacks were formally defined to analyze the resilience of PRFs against key tampering, also referred to as *Related-Key Attack* (RKA) *security*. Their model considers a set of fault functions Φ such that the adversary can arbitrarily fault the key K to $K' \leftarrow \phi(K)$ for any $\phi \in \Phi$. In essence, they require that key tampering on a cryptographic primitive, such as a PRF, is indistinguishable from selecting a new key uniformly at random. In other words, a fault in the key provides the same advantage as obtaining a new primitive with an entirely random key. Further, they also show the limitations of RKA security and prove that it is impossible to achieve RKA security without limiting the function set Φ. Building on their security requirements, several additional primitives were developed to ensure RKA security including RKA secure encryption schemes [11, 16], and MACs [42, 172]. The practical relevance of RKA secure cryptographic primitives has been demonstrated through various attacks documented in the literature [44, 80, 127, 140]. Surprisingly, despite this, RKA security has not been explored for authenticated encryption schemes with associated data. As a result, the initial challenge we encountered was to establish security notions that satisfy all the necessary requirements.

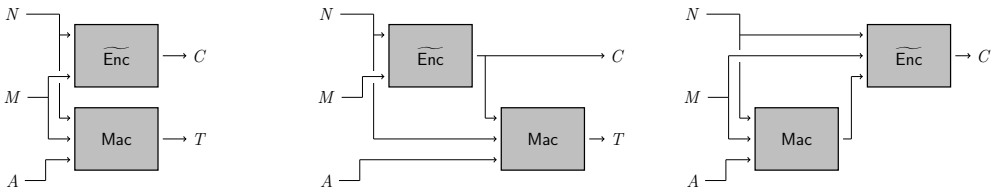
Security Model. As described above, the security of nonce-based schemes relies on the fact that each decryption query with the same key uses a fresh nonce. Since we consider key faults we must expect that the adversary might fault the

[2]Another prominent example of related key attacks involves badly randomized keys leading to the generation of multiple cryptographic primitives using different yet strongly dependent keys.

key *and* the nonce simultaneously, e.g., set the counter of the nonce always to zero when it faults the key. To model this scenario, we adapted the RKA security notion for nonce-based schemes and allow the adversary to use the same nonce for differently faulted keys. Then, our work [94] proposes two RKA security notions for nonce-based AEAD schemes.

First, a weaker notion, denoted by **rka-AE**, is suitable for nonce-based AEAD schemes that are constructed out of an encryption and a MAC just like the N schemes [149]. This notion exploits the special key structure $K = (K_e, K_m)$ of the AEAD scheme consisting of two subkeys K_e and K_m for the encryption and MAC, respectively. The weaker notion assumes that faults on K always affect both underlying keys K_e and K_m, and that the adversary can never make a precise fault to make a previous fault undo. In other words, the adversary is not allowed to produce differently faulted keys $K' = (K'_e, K'_m)$ and $K'' = (K''_e, K''_m)$ such that one of the underlying subkeys is the same, i.e., it has to hold $K'_e \neq K''_e$ *and* $K'_m \neq K''_m$. This weaker notion considers the fact that faults are not always accurate. A fault in the key can lead to additional, unexpected faults, making it unlikely that two faulted subkeys would be identical or one of the underlying keys is not affected by a fault.

Second, our stronger security notion, denoted by **s-rka-AE**, applies for all nonce-based AEAD schemes. The essential difference to the previous notion is that the weaker makes an assumption on the key structure, whereas the stronger notion does not make such an assumption This more robust security notion can assume precise faults affecting only a single area of the key. Hence in our case, it also considers faults where only one underlying key K_e, K_m of $K = (K_e, K_m)$ is faulted, e.g., $K' = (K'_e, K'_m)$ and $K'' = (K''_e, K''_m)$ with $K'_e = K''_e$ *and* $K'_m \neq K''_m$.



(a) *N1* scheme. (b) *N2* scheme. (c) *N3* scheme.

Figure 4.2.: The AEAD schemes of [149]. (Figures of [94])

Concrete Results. Our work [94] investigates the nonce-based N schemes (Figure 4.2) to construct RKA secure AEAD from encryption schemes and MACs. Essentially, we analyze the question whether RKA security of the underlying encryption and MAC scheme implies RKA security of the N schemes as well. In

other words, let the encryption scheme be secure against the class of fault functions Φ_e and the MAC be secure against Φ_m, then we examine whether the N schemes are fault resilient against fault functions resulting from the Cartesian product $\Phi_e \times \Phi_m$. The result of our security analysis strongly depends on our security model. First, we consider the weaker model **rka-AE** and prove that RKA security of the underlying primitives implies security of the N schemes.

Theorem (Informal). *The N schemes depicted in Figure 4.2 are **rka-AE**-secure against fault functions $\Phi_e \times \Phi_m$ if* Mac *and* $\widetilde{\mathsf{Enc}}$ *are RKA secure against fault functions Φ_e and Φ_m, respectively.*

- [94], Theorems 8, 10 & 12.

As Namprempre et al. [149] proved the general security of the N schemes under the additional condition that Mac is a PRF, our theorem requires Mac to be an RKA secure PRF and an RKA secure MAC. Further, we also show that the schemes are not secure if we drop the restrictions in **rka-AE** by presenting concrete attacks in the stronger model **s-rka-AE** ([94], Thm. 9, 11&13). However, we were able to construct an alternative scheme that fulfills even the stronger security requirements. For this reason we adapt the *N3* scheme such that its underlying encryption is a pseudorandom permutation. In contrast to the *N3* scheme the underlying encryption scheme does not use the nonce, and only encrypts the output of the MAC scheme and the message. The resulting scheme N^* is depicted in Figure 4.3 using a RKA secure PRP $\widetilde{\mathsf{Enc}}'$.

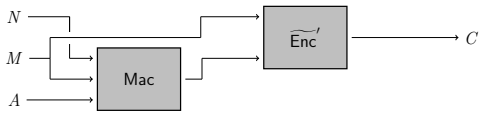


Figure 4.3.: The AEAD scheme N^*. (Figure of [94])

Theorem (Informal). *The N^* scheme depicted in Figure 4.3 is **s-rka-AE**-secure against fault functions $\Phi_e \times \Phi_m$ if* Mac *and* $\widetilde{\mathsf{Enc}}'$ *are RKA secure against fault functions Φ_e and Φ_m, respectively.*

- [94], Theorems 14.

Hence, the theorem shows that additionally requiring an RKA secure PRP implies RKA security for the MAC-then-Encrypt paradigm. At a high-level, the attack on the *N3* scheme exploited the vulnerability where two ciphertexts of the same message have partial equality when encrypted with the same K_e but different

K_m. Thankfully, a RKA secure PRP ensures RKA security for the MAC-then-Encrypt paradigm. Here, the vulnerability of the *N3* scheme is contradicted by the PRP property, as the outputs of the PRP are uniformly random even when the inputs are partially equal.

4.3. Related Work

Chapter 4 is about fuzzy passwords and related-key attacks. We first give a detailed overview of the state of the art of PAKE protocols. Then, we give an overview about related-key attacks.

Fuzzy Asymmetric Password Authenticated Key Exchange. As stated before, the security requirements defined in our work are inspired by [81] and [109]. Our protocol is the first faPAKE protocol combining both properties of aPAKE [31, 50, 109, 128] and fPAKE [81]. Since our publication in 2020, there were several follow-up works. The UC security in [88] was weakened such that only the client has to authenticate. This assumption allows the client to generate the key much faster but with the disadvantage that the identity of the server is not authenticated. Further, biometric based asymmetric protocols were proposed and implemented [21, 22]. For example, [170] proposes an asymmetric fuzzy encapsulation mechanism, that has less overhead and does not require that both parties are online at the same time. This protocol has a speed up of factor 2000 compared to ours.

Related-Key Attacks. Knudsen and Biham, both worked on countermeasures against such attacks. About 10 years later Bellare and Kohno [28] gave the first formalization of RKA security to provide provable secure pseudorandom functions and pseudorandom permutations. Further, they also show the limitations of RKA security and proved that it is impossible to achieve RKA security without limiting the function set. Bellare and Cash [27], and Abdalla et al. [1] designed RKA-secure pseudorandom functions for a relatively large class of RKA functions. Following this work, the RKA security was intensively studied by the research community, e.g., Feistel networks [16], encryption schemes [11], and MACs [42, 172]. The formalisim in [28], was further investigated by Harris [120] and Albrecht et al. [5]. They gave a generic attack against encryption schemes with related-key deriving functions that can depend on the cryptographic primitive, and Vaudenay [168] analyzed the practical relevance of those attacks. However, the analyzes of RKA security for authenticated encryption schemes is rather rare. Lu et al. [143] studied probabilistic authenticated encryption schemes for affine related-key deriving

functions. Unfortunately, Han et al. [119] shows that their security proof does not fulfill their security assumptions where they claim that the adversary cannot generate a forged cipher text for a related-key. Their essential observation is that their reduction proof does not allow to solve the DDH problem (the mathematical hard problem) using the adversary of their scheme. Consequently, its primitives security is not provable secure under the assumption of a hard problem Further, in contrast to their construction, our primitive is more general, and does not require expensive randomness.

Non-malleable Codes. While Section 4.2 introduces fault-resilient primitives, an alternative strategy involves taking an unprotected primitive and employing a "fault-resilient encoding" only on the key. Non-malleable codes, introduced by Dziembowski, Pietrzak, and Wichs [87], are one such encoding method. Their encoding ensures that faults on the encoded key do not compromise the key itself. In detail, Dziembowski et al. propose an encoding where the impact of bit-wise independent faults on the decoding is independent of the underlying decoded value, e.g. the key. Further, the faulted encoding decodes either to the correct value again (error correction), an invalid value (error detection), or is uniform random (secret independent). Giving the fact that they show that this property requires a restriction of fault functions, they provide a non malleable encoding secure only against bit-wise independent tampering. Subsequently, extensions were introduced to accommodate more substantial fault functions such as linear tampering [59], poly space tampering [97], split-state tapering [61] bounded-depth tampering [51], space-bounded tampering [92].

In our context, we have to assume that the attacker repeatedly tampers with the device, i.e. the key. This is described with continuous non-malleable codes [96] that are secure even if the adversary has multiple rounds to fault the key.

While non-malleable codes effectively protect against key faults, they introduce distinctive security considerations. Notably, the encoding of the key requires decoding each time the cryptographic primitive needs the key. This introduces a potential vulnerability, as an attacker may attempt to fault the key after the decoding process. Additionally, it may result in increased space complexity, as the decoding might need more space, and extended runtime, as the key must be decoded for each operation that uses the key.[3]

[3] In contrast to non-malleable codes, RKA security allows for more use cases such as bad randomized key. For more details we refer to our work [94].

5. Combined Resilience

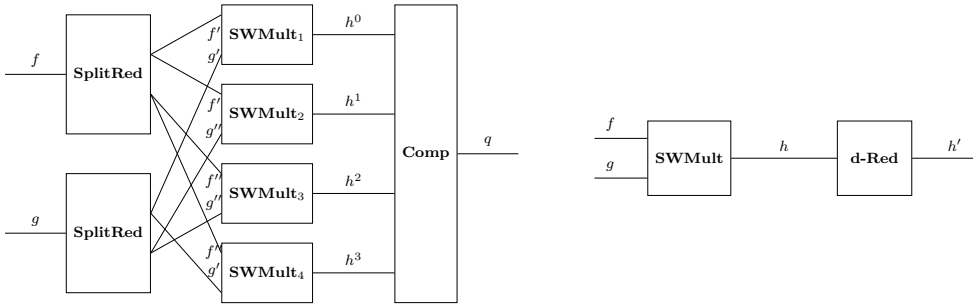
This part is based on [36] and analyzes masked circuits in the context of combined resilience, namely the simultanous resilence against leakage and faults. To be more specific, this section allows the adversary to simultaneously fault and probe the wires of a masked circuit. In contrast to Chapter 4, which only considers faulted inputs such as keys and passwords, this chapter also takes faults on intermediate values into account. That is why we use the circuit model as the underlying computational model again as we did in Chapter 3. However, as defined in Section 2.2, we additionally allow the adversary insert some faults into the computation. In simpler terms, the adversary can change the output value of a gate in the circuit such that the following gate gets a faulted value as input.

Challenges. The security verification of combined attacks often leads to extensive proofs with many case distinctions, since such proofs typically verify the probe security of all possible fault combinations. We need composability results, especially for verifying large circuits as in Chapter 3. Another challenge is the trade-off between the redundant computation for error detection and the increasing leakage caused by redundancy. A low number of redundant shares is significant for resilience against advanced leakage attacks such as horizontal attacks.

5.1. Contribution

This chapter presents the results of [36] which targets two main challenges. First, it gives two compilers provably secure against up to e faults in the t-probing and $t/2$-region probing models, respectively. Second, the paper defines fault-invariance that implies leakage resilience if the adversary can additionally inject faults. This property simplifies the previously tedious proof techniques by avoiding a brute-force analysis that investigates the resilience for all possible fault combinations.

Concrete Results. Our first compiler is simultaneously provable secure in the t-probing model and fault resilient against up to e faults using the optimal number of shares $n = t+e+1$. In order to achieve this, we replaced the multiplication gadget



(a) Our multiplication gadget [36] (b) Multiplication gadget [163].

Figure 5.1.: Structures of our multiplication gadget defined in [36] and the multiplication gadget used in [163]. (Figure of [36])

described above (Figure 5.1b), by a gadget that only requires $n = t + e + 1$ shares instead of $n = 2t + e + 1$. For this reason, our new multiplication transforms the polynomials f and g of the input encodings in such a way that the degree reduction is not necessary after the share-wise multiplication Figure 5.1a. First, the gadget shares the polynomial f (and g) with secret $f(0) = a$ (or $g(0) = b$) and degree t into two polynomials f', f'' (or g', g'') with degree $n - 1$ via sub gadgets **SplitRed** such that $f' + f''$ (or $g' + g''$) has degree $t/2$ and $f'(0) + f''(0) = a$ (or $g'(0) + g''(0) = b$). Then, it computes four share-wise multiplications $h^0 = f' \cdot g'$, $h^1 = f' \cdot g''$, $h^2 = f'' \cdot g''$, and $h^3 = f'' \cdot g'$ via the sub gadgets **SWMult**. Finally, it generates an encoding of zero, i.e., a polynomial x with $x(0) = 0$, and share wise adds $x + h^0 + h^1 + h^2 + h^3$ via **Comp**. The final output q is a sharing of a degree t polynomial with constant term $q(0) = a \cdot b$ because $q = x + (f' + f'')(g' + g'')$, and $q(0) = x(0) + (f'(0) + f''(0))(g'(0) + g''(0)) = 0 + a \cdot b$. With this gadget we get a secure compiler against additive faults with an optimal number of shares.

Theorem (Informal). *There is a leakage and fault resilient compiler secure against t probes and e additive faults using a minimum number of shares $n = t + e + 1$.*

- [36], Theorem 8, 9 & 10.

The compiler also provides security against an adversary using arbitrary faults if the faults are additionally counted as probes. This means that our compiler is also secure against e arbitrary wire faults and $t - e$ probes. This follows from the fact that a fault where a wire value is set to a constant value is worse than a simple probe because the adversary does not only "learn" the value of the wire, but also chooses its value. The same results hold for our second compiler. The second

compiler is extended to be secure in the $t/2$-region probing model by injecting additional refresh gadgets between each gadget.

Theorem (Informal). *There is a leakage and fault resilient compiler secure against $t/2$ probes in each gadget and e additive faults using an n sharing with $n = t + e + 1$.*

- [36], Theorem 8 & 9.

In the following step, this chapter introduces a new property, called *fault invariance*, which allows for efficient proof techniques for resilience against combined attacks.

Proof Idea. In contrast to leakage resilience, where we consider the inputs required to simulate the leakage, fault resilience analyses the outputs of a faulty circuit. For this reason, we define fault robustness, which describes how faults are forwarded to the gadgets' outputs so that they can be detected. If fault-robust gadgets are faulted and get faulted inputs, their outputs will either be a valid encoding of the correct values or an invalid encoding. Since fault robustness considers faulted inputs that might come from previous computations, the composition of fault-robust gadgets is again fault-robust. Hence, we can prove fault resilience by proving the fault robustness of each gadget. Our paper also shows that it is not sufficient to separately investigate leakage and fault resilience to prove security against combined attacks: Faults might increase the leakage of a circuit so that the leakage of a faulted circuit reveals more secret information then the one without faults [36]. The basic idea are, specifically-targeted faults, which have the potential to compromise the essential randomness required to hide the internal secret. In our work, we illustrated this vulnerability through an example where a single fault disrupted the internal randomness of a masked circuit. Consequently, the circuit became deterministic and did not hide the internal computation anymore. To address this problem, we provide a property called fault invariance that combines fault and leakage resilience. This property shows that the leakage simulator does not need more input values for the simulation if the simulated circuit is additionally faulted. More precisely, a gadget is fault invariant if all intermediate faults can be expressed as faults on the inputs or outputs of the gadgets. For example, let $f(x_0, x_1, \cdots)$ be an arbitrary intermediate value of an unfaulted gadget with input values $x_0, x_1 \cdots$ and f' the corresponding value of an arbitrarily faulted version of the gadget with fault functions in $\mathcal{F}$. If the gadget is fault invariant with respect to $\mathcal{F}$, we can describe the intermediate value f' with fault function

$f, f_0, f_1, \cdots \in \mathcal{F}$ such that it holds

$$f' = f(f(f_0(x_0), f_1(x_1), \cdots)) \ .$$

5.2. Related Work

Computational Faults. Section 5.1 combined the leakage and fault resilience. The leakage resilience of circuits was already addressed in Chapter 3. Since Chapter 4 only considered faults on secrets, such as keys and passwords, we first briefly summarize the fault resilience of intermediate faults in the computation. Then, we discuss the work of combined security in more detail. In [125], Ishai et al. considered fault attacks and proposed gadgets secure against attacks where the adversary can set the values to zero via faults. By introducing a fault threshold t, they have extended this approach and effectively allowed the adversary to arbitrarily fault up to t wires. An alternative restriction to allow arbitrary adaptive wire independent faults was introduced by Faust, Pietrzak, and Venturi [98]. They proposed a fault model where each wire can be faulted but only with a given success probability δ. They give a compiler that transforms a circuit into a tamper-resistant circuit, given that the original circuit can tolerate a logarithmic amount of leakage. Kiayias and Tselekounis [135] proposed a model where the adversary can fault the gates instead of the wires. They show that this model is strictly stronger as the gate tampering can be simulated by wire tampering but not vice versa. Additionally they give some impossibility results for both models. Alternatively, Genkin et al. [106] protect circuits against an adversary that can inject arbitrarily many additive faults. Similarly, Dachman-Soled and Kalai [71] protect circuits from faults affecting up to a $1/poly(k)$ fraction of the wires, where k is a security parameter that is independent of the size of the (unprotected) circuit. Other interesting approaches are Impeccable Circuits [3, 155, 164] that allow for error detection or correction, and Trojan resilient and Testable Circuits [12, 84] that protect the computation against malicious hardware.

Combined Resilience As a first step, Gammel and Mangard [103] investigated linear encodings that are secure against an adversary simultaneously running probing and fault attacks. Liu and Lysyanskaya [141] protected circuits against split-state leakage and tampering attacks that independently attacks separate parts of the hardware. Later, Schneider, Moradi, and Güneysu [162] and De Cnudde and Nikova [65] combined error-detection codes with leakage-resilient circuits. As highlighted in Chapter 4, polynomial sharing is a reasonable error detection code

because it provides both fault and leakage resilience. The use of polynomial sharings was originally inspired by the multiparty computation protocol of Ben-Or, Goldwasser, and Wigderson [33]. Seker et al. [163]. Their construction is secure against additive faults and requires $2d + e + 1$ shares. Dhooghe and Nikova [74][1] used the same error detection code and proposed a construction only requiring $d + e + 1$ shares. However, their construction has randomness complexity of $O(n^3)$ instead of $O(n^2)$.

Duplicated Masking.[2] In order to protect against combined attacks, an alternative approach is duplicated masking [75, 99, 100]. Duplicated masking replicates the masked computation proposed in Chapter 3. In particular, the computation of the masked circuit is executed $e + 1$ times in parallel to detect up to e faults, and the duplicated outputs of the computation are checked for equality to detect faults. Consequently, the duplication approach is vulnerable to so-called horizontal attacks [20] where the adversary can exploit the fact that multiple computations share the same randomness or secret key material due to the duplicated computation. Further, many modern primitives, e.g., [6, 7, 8, 41, 116, 117, 118] have a low number of non-linear operations at a cost of a significant number of affine operations. Here, it is noteworthy that affine operations that can be masked traditionally with a linear complexity, the overhead become significantly more expensive as the duplication leads to an quadratic overhead.

[1] This result is only in version 20190603:070457 on IACR Cryptol. ePrint Arch. `https://eprint.iacr.org`

[2] Parts of this paragraph were taken verbatim from our work [36] (cf. App E).

6. Conclusion

This thesis has addressed the crucial issue of protecting cryptographic primitives against implementation attacks, encompassing both side-channel and fault attacks. In Chapter 3, we introduced methods to protect computation against side-channel attacks, including a secure compiler, and two different proof techniques to analyze the security of cryptographic implementations. While the first approach demonstrated almost tight security results, it faced limitations due to its proof complexity. In contrast, the second proof method eliminated the complexity constraints and allows for security proofs for an arbitrary number of shares n, and yields asymptomatic security results which are particularly practical for affine circuits. Chapter 4 presented two fault-resilient cryptographic primitives. The first primitive generates secret keys from a password, ensuring correctness up to a specified fault threshold in the password. This fault tolerance is particularly important for biometric-based passwords, as these are inherently inaccurate. The second primitive offers authenticated encryption that is resilient against key faults. Its security property prevents the attacker from gaining any knowledge about the secret key, even if the adversary can additionally fault the key. In Chapter 5, we combined fault and leakage resilience to develop a secure compiler that simultaneously withstands fault and probing attacks. Given that individual fault and leakage resilience does not guarantee combined resilience, we introduced new security properties that provide this combined security. Finally, we present a compiler to transform unprotected computation into those resilient against combined attacks.

This work contributes to bridging the gap between theoretically secure cryptographic primitives and the practical security of their real-world implementations. However, it is essential to acknowledge that our contribution remains theoretical. For example, in our work, security is generally defined independently of the specific hardware a primitive is executed on. Barthe et al. [19], shows that the hardware heavily influences the amount of leakage, and from this standpoint, there exist numerous exciting directions to enhance and extend our findings. Next, we discuss the following primary directions: (1) bridging theory and practice, (2) improving the theoretical claims presented in this thesis, and exploring more practical directions such as (3) implementing verification tools for polynomial masking, or (4) a general practical exploration in the context of combined resilience.

(1) Bridging Theory And Practice. A hardware-specific analysis of our designs would be an obvious next step towards enhancing the relation between theoretical frameworks and practical applications. This includes two interesting first steps, namely investigating an *implemented compiler* to automatically generate provably secure implementations of cryptographic primitives, and its investigation in a *real laboratory* to examine the implementation's actual resilience against implementation attacks. This approach would require intensive research with multiple iterations, where the security model used to prove the security of the implemented compiler has to be adapted each time a security flaw in the generated implementation was detected in the laboratory. Especially for combined resilience, this leads to exciting research directions, that have recently started. For example considering only leakage, we could adapt the leakage models used in this thesis accordingly and tailor the countermeasures to specific hardware. We have already introduced a preliminary step in [2], which proposes a customized compiler for Arm Cortex M0+ microcontrollers, and in [19], which proposes a customized verification tool for those microcontrollers. Following this approach, further improvements were given by generalizing such leakages [175].

(2) Theoretical Improvements. Our findings also raise interesting theoretical questions. For example, a first step for future exploration would involve improving the randomness and computational complexity of our compiler. This is important on a theoretical level, as it helps to establish lower bounds on the efficiency of countermeasures, but it is also important on a practical level, as it improves the applicability of these countermeasures. Initial steps in this direction have been taken, particularly in the threshold model for ISW-based compilers, as demonstrated in [23], where it was shown that there is a multiplication gadget with complexity of $O(n \log(n))$. An alternative approach in [68] demonstrated that randomness complexity can be reduced through intelligent reuse of randomness in this model. Additionally, [115] revealed that achieving quasi-constant randomness complexity in relation to circuit size is possible. Nonetheless, further investigation is imperative to also determine the lower bounds in the random probing model, or ideally, in the combined model. Since these bounds are currently applicable only to standalone leakage resilience, an even more interesting research question is to investigate the required bounds for combined resilience. For example, given that a tight number of shares is already used in our compiler [36], extending such a bound to include results on runtime and randomness complexity is an exciting research question.

(3) Combined Verification Tools. The security proofs of our gadgets, especially the ones for combined resilience, often follow similar techniques. Unfortunately, these techniques are currently done only by expensive manual handwritten methods. Additionally, the verification tools recently developed to analyze the security of duplicated masking [75, 99, 100] are not suitable for our gadgets using polynomial masking. This limitation arises from a dual challenge. On the one hand, the fault resilience definition in the existing tool VERICA [158] is tailored to the duplication approach and requires adaptation to polynomial masking, as elaborated in [36]. On the other hand, prevalent leakage verification tools [24, 25, 57] incorporate a specific proof approximation, the so-called *optimistic sampling rule* [17]. The idea of this rule is that a probe is independent of the secrets if it consists of an added random value that does not affect the other probes. Adhering strictly to this rule predominantly results in false positives when applied to gadgets using polynomial masking, which would incorrectly identify a secure gadget as insecure. As suggested in [25], the application of Gaussian elimination methods can address this issue. Alternatively, SILVER's approach [137] mitigates these false positives at the expense of complex statistical independence tests, limiting its applicability in the context of random probing security and the combined attack model.

(4) Practical Exploration. A next possible step involves improving our compiler's security against real-world leakage and faults, mirroring the approach taken for standalone leakage as demonstrated in our work [2]. The initial phase requires the concrete implementation of our compiler and the execution of security experiments under laboratory conditions. This task also includes the development and implementation of cryptographic primitives explicitly designed to efficiently resist combined attacks (e.g., [40, 86, 129]). Especially in the context of post-quantum cryptography, many new primitives are worth investigating for their security against real-world attacks.

7. Bibliography

This thesis was linguistically revised with the help of ChatGPT[1], Grammarly[2], and DeepL[3].

[1] M. Abdalla, F. Benhamouda, A. Passelègue, and K. G. Paterson. "Related-Key Security for Pseudorandom Functions Beyond the Linear Barrier". In: *J. Cryptol.* 4 (2018), pp. 917–964.

[2] A. Abromeit, F. Bache, L. A. Becker, M. Gourjon, T. Güneysu, S. Jorn, A. Moradi, M. Orlt, and F. Schellenberg. "Automated Masking of Software Implementations on Industrial Microcontrollers". In: *Design, Automation & Test in Europe Conference & Exhibition, DATE 2021, Grenoble, France, February 1-5, 2021*. 2021, pp. 1006–1011.

[3] A. Aghaie, A. Moradi, S. Rasoolzadeh, A. R. Shahmirzadi, F. Schellenberg, and T. Schneider. "Impeccable Circuits". In: *IEEE Trans. Computers* 3 (2020), pp. 361–376.

[4] C. Aitken. *Determinants and Matrices*. 9. 1956.

[5] M. R. Albrecht, P. Farshim, K. G. Paterson, and G. J. Watson. "On Cipher-Dependent Related-Key Attacks in the Ideal-Cipher Model". In: *Fast Software Encryption - 18th International Workshop, FSE 2011, Lyngby, Denmark, February 13-16, 2011, Revised Selected Papers*. 2011, pp. 128–145.

[6] M. R. Albrecht, L. Grassi, C. Rechberger, A. Roy, and T. Tiessen. "MiMC: Efficient Encryption and Cryptographic Hashing with Minimal Multiplicative Complexity". In: *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I*. 2016, pp. 191–219.

[1] `https://chat.openai.com`
[2] `https://www.grammarly.com/`
[3] `https://www.deepl.com`

[7] M. R. Albrecht, C. Rechberger, T. Schneider, T. Tiessen, and M. Zohner. "Ciphers for MPC and FHE". In: *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I.* 2015, pp. 430–454.

[8] A. Aly, T. Ashur, E. Ben-Sasson, S. Dhooghe, and A. Szepieniec. "Design of Symmetric-Key Primitives for Advanced Cryptographic Protocols". In: *IACR Trans. Symmetric Cryptol.* 3 (2020), pp. 1–45.

[9] P. Ananth, Y. Ishai, and A. Sahai. "Private Circuits: A Modular Approach". In: *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part III.* 2018, pp. 427–455.

[10] M. Andrychowicz, S. Dziembowski, and S. Faust. "Circuit Compilers with O(1/\log (n)) Leakage Rate". In: *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II.* 2016, pp. 586–615.

[11] B. Applebaum, D. Harnik, and Y. Ishai. "Semantic Security under Related-Key Attacks and Applications". In: *Innovations in Computer Science - ICS 2011, Tsinghua University, Beijing, China, January 7-9, 2011. Proceedings.* 2011, pp. 45–60.

[12] M. A. Baig, S. Chakraborty, S. Dziembowski, M. Galazka, T. Lizurej, and K. Pietrzak. "Efficiently Testable Circuits". In: *14th Innovations in Theoretical Computer Science Conference, ITCS 2023, January 10-13, 2023, MIT, Cambridge, Massachusetts, USA.* 2023, 10:1–10:23.

[13] J. Balasch, B. Gierlichs, V. Grosso, O. Reparaz, and F. Standaert. "On the Cost of Lazy Engineering for Masked Software Implementations". In: *Smart Card Research and Advanced Applications - 13th International Conference, CARDIS 2014, Paris, France, November 5-7, 2014. Revised Selected Papers.* 2014, pp. 64–81.

[14] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. P. Vadhan, and K. Yang. "On the (im)possibility of obfuscating programs". In: *J. ACM* 2 (2012), 6:1–6:48.

[15] E. J Barbeau. *Polynomials.* 1989.

[16] M. Barbosa and P. Farshim. "The Related-Key Analysis of Feistel Constructions". In: *Fast Software Encryption - 21st International Workshop, FSE 2014, London, UK, March 3-5, 2014. Revised Selected Papers.* 2014, pp. 265–284.

[17] G. Barthe, S. Belaïd, G. Cassiers, P. Fouque, B. Grégoire, and F. Standaert. "maskVerif: Automated Verification of Higher-Order Masking in Presence of Physical Defaults". In: *Computer Security - ESORICS 2019 - 24th European Symposium on Research in Computer Security, Luxembourg, September 23-27, 2019, Proceedings, Part I*. 2019, pp. 300–318.

[18] G. Barthe, S. Belaïd, F. Dupressoir, P. Fouque, B. Grégoire, P. Strub, and R. Zucchini. "Strong Non-Interference and Type-Directed Higher-Order Masking". In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*. 2016, pp. 116–129.

[19] G. Barthe, M. Gourjon, B. Grégoire, M. Orlt, C. Paglialonga, and L. Porth. "Masking in Fine-Grained Leakage Models: Construction, Implementation and Verification". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2 (2021), pp. 189–228.

[20] A. Battistello, J. Coron, E. Prouff, and R. Zeitoun. "Horizontal Side-Channel Attacks and Countermeasures on the ISW Masking Scheme". In: *Cryptographic Hardware and Embedded Systems - CHES 2016 - 18th International Conference, Santa Barbara, CA, USA, August 17-19, 2016, Proceedings*. 2016, pp. 23–39.

[21] P. Bauspieß, T. Silde, A. Tullot, A. Costache, C. Rathgeb, J. Kolberg, and C. Busch. "Improved Biometrics-Authenticated Key Exchange". In: *IACR Cryptol. ePrint Arch.* (2022), p. 1408.

[22] P. Bauspieß, T. Silde, M. Poljuha, A. Tullot, A. Costache, C. Rathgeb, J. Kolberg, and C. Busch. *BRAKE: Biometric Resilient Authenticated Key Exchange*. Cryptology ePrint Archive, Paper 2022/1408. `https://eprint.iacr.org/2022/1408`. 2022.

[23] S. Belaïd, F. Benhamouda, A. Passelègue, E. Prouff, A. Thillard, and D. Vergnaud. "Randomness Complexity of Private Circuits for Multiplication". In: *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II*. 2016, pp. 616–648.

[24] S. Belaïd, J. Coron, E. Prouff, M. Rivain, and A. R. Taleb. "Random Probing Security: Verification, Composition, Expansion and New Constructions". In: *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part I*. 2020, pp. 339–368.

[25] S. Belaïd, D. Mercadier, M. Rivain, and A. R. Taleb. "IronMask: Versatile Verification of Masking Security". In: *43rd IEEE Symposium on Security and Privacy, SP 2022, San Francisco, CA, USA, May 22-26, 2022*. 2022, pp. 142–160.

[26] S. Belaïd, M. Rivain, and A. R. Taleb. "On the Power of Expansion: More Efficient Constructions in the Random Probing Model". In: *Advances in Cryptology - EUROCRYPT 2021 - 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17-21, 2021, Proceedings, Part II*. 2021, pp. 313–343.

[27] M. Bellare and D. Cash. "Pseudorandom Functions and Permutations Provably Secure against Related-Key Attacks". In: *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*. 2010, pp. 666–684.

[28] M. Bellare and T. Kohno. "A Theoretical Treatment of Related-Key Attacks: RKA-PRPs, RKA-PRFs, and Applications". In: *Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003, Proceedings*. 2003, pp. 491–506.

[29] M. Bellare, D. Pointcheval, and P. Rogaway. "Authenticated Key Exchange Secure against Dictionary Attacks". In: *Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14-18, 2000, Proceeding*. 2000, pp. 139–155.

[30] M. Bellare and P. Rogaway. "Random Oracles are Practical: A Paradigm for Designing Efficient Protocols". In: *CCS '93, Proceedings of the 1st ACM Conference on Computer and Communications Security, Fairfax, Virginia, USA, November 3-5, 1993*. 1993, pp. 62–73.

[31] S. M. Bellovin and M. Merritt. "Augmented Encrypted Key Exchange: A Password-Based Protocol Secure against Dictionary Attacks and Password File Compromise". In: *CCS '93, Proceedings of the 1st ACM Conference on Computer and Communications Security, Fairfax, Virginia, USA, November 3-5, 1993*. 1993, pp. 244–250.

[32] S. M. Bellovin and M. Merritt. "Encrypted key exchange: password-based protocols secure against dictionary attacks". In: *1992 IEEE Computer Society Symposium on Research in Security and Privacy, Oakland, CA, USA, May 4-6, 1992*. 1992, pp. 72–84.

[33] M. Ben-Or, S. Goldwasser, and A. Wigderson. "Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation (Extended Abstract)". In: *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*. 1988, pp. 1–10.

[34] F. Benhamouda, O. Blazy, C. Chevalier, D. Pointcheval, and D. Vergnaud. "New Techniques for SPHFs and Efficient One-Round PAKE Protocols". In: *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*. 2013, pp. 449–475.

[35] S. Berndt, T. Eisenbarth, S. Faust, M. Gourjon, M. Orlt, and O. Seker. "Combined Fault and Leakage Resilience: Composability, Constructions and Compiler". In: *IACR Cryptol. ePrint Arch.* (2023), p. 1143.

[36] S. Berndt, T. Eisenbarth, S. Faust, M. Gourjon, M. Orlt, and O. Seker. "Combined Fault and Leakage Resilience: Composability, Constructions and Compiler". In: *Advances in Cryptology - CRYPTO 2023 - 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part III*. 2023, pp. 377–409.

[37] D. J. Bernstein. *CAESAR: Competition for Authenticated Encryption: Security, Applicability, and Robustness*. 2004.

[38] F. Berti, S. Faust, and M. Orlt. "Long Paper: Provable Secure Parallel Gadgets". In: *IACR Cryptol. ePrint Arch.* (2023), p. 1182.

[39] F. Berti, S. Faust, and M. Orlt. "Provable Secure Parallel Gadgets". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 4 (2023), pp. 420–459.

[40] F. Berti, C. Guo, T. Peters, Y. Shen, and F. Standaert. "Secure Message Authentication in the Presence of Leakage and Faults". In: *IACR Trans. Symmetric Cryptol.* 1 (2023), pp. 288–315.

[41] G. Bertoni, J. Daemen, M. Peeters, and G. V. Assche. "Keccak". In: *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*. 2013, pp. 313–314.

[42] R. Bhattacharyya and A. Roy. "Secure Message Authentication Against Related-Key Attack". In: *Fast Software Encryption - 20th International Workshop, FSE 2013, Singapore, March 11-13, 2013. Revised Selected Papers*. 2013, pp. 305–324.

[43] E. Biham and A. Shamir. "Differential Fault Analysis of Secret Key Cryptosystems". In: *Advances in Cryptology - CRYPTO '97, 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 1997, Proceedings*. 1997, pp. 513–525.

[44] A. Biryukov, D. Khovratovich, and I. Nikolic. "Distinguisher and Related-Key Attack on the Full AES-256". In: *Advances in Cryptology - CRYPTO 2009, 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings*. 2009, pp. 231–249.

[45] R. Bloem, H. Groß, R. Iusupov, B. Könighofer, S. Mangard, and J. Winter. "Formal Verification of Masked Hardware Implementations in the Presence of Glitches". In: *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part II*. 2018, pp. 321–353.

[46] A. Bogdanov, Y. Ishai, and A. Srinivasan. "Unconditionally Secure Computation Against Low-Complexity Leakage". In: *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part II*. 2019, pp. 387–416.

[47] D. Boneh, R. A. DeMillo, and R. J. Lipton. "On the Importance of Checking Cryptographic Protocols for Faults (Extended Abstract)". In: *Advances in Cryptology - EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 11-15, 1997, Proceeding*. 1997, pp. 37–51.

[48] J. Bonneau and I. Mironov. "Cache-Collision Timing Attacks Against AES". In: *Cryptographic Hardware and Embedded Systems - CHES 2006, 8th International Workshop, Yokohama, Japan, October 10-13, 2006, Proceedings*. 2006, pp. 201–215.

[49] V. Boyko, P. D. MacKenzie, and S. Patel. "Provably Secure Password-Authenticated Key Exchange Using Diffie-Hellman". In: *Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14-18, 2000, Proceeding*. 2000, pp. 156–171.

[50] T. Bradley, S. Jarecki, and J. Xu. "Strong Asymmetric PAKE Based on Trapdoor CKEM". In: *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part III*. 2019, pp. 798–825.

[51] G. Brian, S. Faust, E. Micheli, and D. Venturi. "Continuously Non-malleable Codes Against Bounded-Depth Tampering". In: *Advances in Cryptology - ASIACRYPT 2022 - 28th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, December 5-9, 2022, Proceedings, Part IV*. 2022, pp. 384–413.

[52] R. Canetti. "Universally Composable Security: A New Paradigm for Cryptographic Protocols". In: *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*. 2001, pp. 136–145.

[53] R. Canetti, D. Dachman-Soled, V. Vaikuntanathan, and H. Wee. "Efficient Password Authenticated Key Exchange via Oblivious Transfer". In: *Public Key Cryptography - PKC 2012 - 15th International Conference on Practice and Theory in Public Key Cryptography, Darmstadt, Germany, May 21-23, 2012. Proceedings*. 2012, pp. 449–466.

[54] R. Canetti, S. Halevi, J. Katz, Y. Lindell, and P. D. MacKenzie. "Universally Composable Password-Based Key Exchange". In: *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*. 2005, pp. 404–421.

[55] J. Caraco, R. Géraud-Stewart, and D. Naccache. "Kerckhoffs' Legacy". In: *IACR Cryptol. ePrint Arch.* (2020), p. 556.

[56] G. Cassiers, S. Faust, M. Orlt, and F. Standaert. "Towards Tight Random Probing Security". In: *IACR Cryptol. ePrint Arch.* (2021), p. 880.

[57] G. Cassiers, S. Faust, M. Orlt, and F. Standaert. "Towards Tight Random Probing Security". In: *Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part III*. 2021, pp. 185–214.

[58] G. Cassiers, L. Masure, C. Momin, T. Moos, and F. Standaert. "Prime-Field Masking in Hardware and its Soundness against Low-Noise SCA Attacks". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2 (2023), pp. 482–518.

[59] H. Chabanne, G. D. Cohen, and A. Patey. "Secure network coding and non-malleable codes: Protection against linear tampering". In: *Proceedings of the 2012 IEEE International Symposium on Information Theory, ISIT 2012, Cambridge, MA, USA, July 1-6, 2012*. 2012, pp. 2546–2550.

[60] S. Chari, C. S. Jutla, J. R. Rao, and P. Rohatgi. "Towards Sound Approaches to Counteract Power-Analysis Attacks". In: *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*. 1999, pp. 398–412.

[61] E. Chattopadhyay and D. Zuckerman. "Non-malleable Codes against Constant Split-State Tampering". In: *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*. 2014, pp. 306–315.

[62] T. Chou and C. Orlandi. "The Simplest Protocol for Oblivious Transfer". In: *Progress in Cryptology - LATINCRYPT 2015 - 4th International Conference on Cryptology and Information Security in Latin America, Guadalajara, Mexico, August 23-26, 2015, Proceedings*. 2015, pp. 40–58.

[63] C. Clavier, B. Feix, G. Gagnerot, M. Roussellet, and V. Verneuil. "Horizontal Correlation Analysis on Exponentiation". In: *Information and Communications Security - 12th International Conference, ICICS 2010, Barcelona, Spain, December 15-17, 2010. Proceedings*. 2010, pp. 46–61.

[64] T. D. Cnudde, B. Bilgin, B. Gierlichs, V. Nikov, S. Nikova, and V. Rijmen. "Does Coupling Affect the Security of Masked Implementations?" In: *Constructive Side-Channel Analysis and Secure Design - 8th International Workshop, COSADE 2017, Paris, France, April 13-14, 2017, Revised Selected Papers*. 2017, pp. 1–18.

[65] T. D. Cnudde and S. Nikova. "More Efficient Private Circuits II through Threshold Implementations". In: *2016 Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC 2016, Santa Barbara, CA, USA, August 16, 2016*. 2016, pp. 114–124.

[66] J. Coron, C. Giraud, E. Prouff, S. Renner, M. Rivain, and P. K. Vadnala. "Conversion of Security Proofs from One Leakage Model to Another: A New Issue". In: *Constructive Side-Channel Analysis and Secure Design - Third International Workshop, COSADE 2012, Darmstadt, Germany, May 3-4, 2012. Proceedings*. 2012, pp. 69–81.

[67] J. Coron and L. Goubin. "On Boolean and Arithmetic Masking against Differential Power Analysis". In: *Cryptographic Hardware and Embedded Systems - CHES 2000, Second International Workshop, Worcester, MA, USA, August 17-18, 2000, Proceedings*. 2000, pp. 231–237.

[68] J. Coron, A. Greuet, and R. Zeitoun. "Side-Channel Masking with Pseudo-Random Generator". In: *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part III*. 2020, pp. 342–375.

[69] J. Coron, E. Prouff, M. Rivain, and T. Roche. "Higher-Order Side Channel Security and Mask Refreshing". In: *Fast Software Encryption - 20th International Workshop, FSE 2013, Singapore, March 11-13, 2013. Revised Selected Papers*. 2013, pp. 410–424.

[70] J. Coron, E. Prouff, and T. Roche. "On the Use of Shamir's Secret Sharing against Side-Channel Analysis". In: *Smart Card Research and Advanced Applications - 11th International Conference, CARDIS 2012, Graz, Austria, November 28-30, 2012, Revised Selected Papers*. 2012, pp. 77–90.

[71] D. Dachman-Soled and Y. T. Kalai. "Securing Circuits and Protocols against 1/poly(k) Tampering Rate". In: *Theory of Cryptography - 11th Theory of Cryptography Conference, TCC 2014, San Diego, CA, USA, February 24-26, 2014. Proceedings*. 2014, pp. 540–565.

[72] J. Daemen and V. Rijmen. "Resistance against implementation attacks: A comparative study of the AES proposals". In: *Proceedings of the Second Advanced Encryption Standard (AES) Candidate Conference*. 1999.

[73] A. Dehbaoui, J. Dutertre, B. Robisson, and A. Tria. "Electromagnetic Transient Faults Injection on a Hardware and a Software Implementations of AES". In: *2012 Workshop on Fault Diagnosis and Tolerance in Cryptography, Leuven, Belgium, September 9, 2012*. 2012, pp. 7–15.

[74] S. Dhooghe and S. Nikova. "My Gadget Just Cares For Me - How NINA Can Prove Security Against Combined Attacks". In: *IACR Cryptol. ePrint Arch.* (2019), p. 615.

[75] S. Dhooghe and S. Nikova. "My Gadget Just Cares for Me - How NINA Can Prove Security Against Combined Attacks". In: *Topics in Cryptology - CT-RSA 2020 - The Cryptographers' Track at the RSA Conference 2020, San Francisco, CA, USA, February 24-28, 2020, Proceedings*. 2020, pp. 35–55.

[76] W. Diffie and M. E. Hellman. "New directions in cryptography". In: *IEEE Trans. Inf. Theory* 6 (1976), pp. 644–654.

[77] C. Dobraunig, M. Eichlseder, H. Groß, S. Mangard, F. Mendel, and R. Primas. "Statistical Ineffective Fault Attacks on Masked AES with Fault Countermeasures". In: *Advances in Cryptology - ASIACRYPT 2018 - 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part II*. 2018, pp. 315–342.

[78] A. Duc, S. Dziembowski, and S. Faust. "Unifying Leakage Models: From Probing Attacks to Noisy Leakage". In: *J. Cryptol.* 1 (2019), pp. 151–177.

[79] M. Dumont, M. Lisart, and P. Maurine. "Electromagnetic Fault Injection : How Faults Occur". In: *2019 Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC 2019, Atlanta, GA, USA, August 24, 2019*. 2019, pp. 9–16.

[80] O. Dunkelman, N. Keller, and J. Kim. "Related-Key Rectangle Attack on the Full SHACAL-1". In: *Selected Areas in Cryptography, 13th International Workshop, SAC 2006, Montreal, Canada, August 17-18, 2006 Revised Selected Papers*. 2006, pp. 28–44.

[81] P. Dupont, J. Hesse, D. Pointcheval, L. Reyzin, and S. Yakoubov. "Fuzzy Password-Authenticated Key Exchange". In: *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part III*. 2018, pp. 393–424.

[82] S. Dziembowski and S. Faust. "Leakage-Resilient Circuits without Computational Assumptions". In: *Theory of Cryptography - 9th Theory of Cryptography Conference, TCC 2012, Taormina, Sicily, Italy, March 19-21, 2012. Proceedings*. 2012, pp. 230–247.

[83] S. Dziembowski, S. Faust, and M. Skorski. "Noisy Leakage Revisited". In: *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II*. 2015, pp. 159–188.

[84] S. Dziembowski, S. Faust, and F. Standaert. "Private Circuits III: Hardware Trojan-Resilience via Testing Amplification". In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*. 2016, pp. 142–153.

[85] S. Dziembowski, S. Faust, and K. Zebrowski. "Simple Refreshing in the Noisy Leakage Model". In: *Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part III*. 2019, pp. 315–344.

[86] S. Dziembowski and K. Pietrzak. "Leakage-Resilient Cryptography in the Standard Model". In: *IACR Cryptol. ePrint Arch.* (2008), p. 240.

[87] S. Dziembowski, K. Pietrzak, and D. Wichs. "Non-Malleable Codes". In: *Innovations in Computer Science - ICS 2010, Tsinghua University, Beijing, China, January 5-7, 2010. Proceedings*. 2010, pp. 434–452.

[88] J. Ernst and A. Mitrokotsa. "A Framework for UC Secure Privacy Preserving Biometric Authentication Using Efficient Functional Encryption". In: *Applied Cryptography and Network Security - 21st International Conference, ACNS 2023, Kyoto, Japan, June 19-22, 2023, Proceedings, Part II*. 2023, pp. 167–196.

[89] A. Erwig, J. Hesse, M. Orlt, and S. Riahi. "Fuzzy Asymmetric Password-Authenticated Key Exchange". In: *IACR Cryptol. ePrint Arch.* (2020), p. 987.

[90] A. Erwig, J. Hesse, M. Orlt, and S. Riahi. "Fuzzy Asymmetric Password-Authenticated Key Exchange". In: *Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part II*. 2020, pp. 761–784.

[91] S. Faust, V. Grosso, S. M. D. Pozo, C. Paglialonga, and F. Standaert. "Composable Masking Schemes in the Presence of Physical Defaults & the Robust Probing Model". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 3 (2018), pp. 89–120.

[92] S. Faust, K. Hostáková, P. Mukherjee, and D. Venturi. "Non-Malleable Codes for Space-Bounded Tampering". In: *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part II*. 2017, pp. 95–126.

[93] S. Faust, J. Krämer, M. Orlt, and P. Struck. "On the Related-Key Attack Security of Authenticated Encryption Schemes". In: *IACR Cryptol. ePrint Arch.* (2022), p. 140.

[94] S. Faust, J. Krämer, M. Orlt, and P. Struck. "On the Related-Key Attack Security of Authenticated Encryption Schemes". In: *Security and Cryptography for Networks - 13th International Conference, SCN 2022, Amalfi, Italy, September 12-14, 2022, Proceedings*. 2022, pp. 362–386.

[95] S. Faust, L. Masure, E. Micheli, M. Orlt, and F. Standaert. "Connecting Leakage-Resilient Secret Sharing to Practice: Scaling Trends and Physical Dependencies of Prime Field Masking". In: *Advances in Cryptology - EUROCRYPT 2024 - 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26-30, 2024, Proceedings, Part IV*. 2024, pp. 316–344.

[96] S. Faust, P. Mukherjee, J. B. Nielsen, and D. Venturi. "Continuous Non-malleable Codes". In: *Theory of Cryptography - 11th Theory of Cryptography Conference, TCC 2014, San Diego, CA, USA, February 24-26, 2014. Proceedings*. 2014, pp. 465–488.

[97] S. Faust, P. Mukherjee, D. Venturi, and D. Wichs. "Efficient Non-malleable Codes and Key-Derivation for Poly-size Tampering Circuits". In: *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*. 2014, pp. 111–128.

[98] S. Faust, K. Pietrzak, and D. Venturi. "Tamper-Proof Circuits: How to Trade Leakage for Tamper-Resilience". In: *Automata, Languages and Programming - 38th International Colloquium, ICALP 2011, Zurich, Switzerland, July 4-8, 2011, Proceedings, Part I*. 2011, pp. 391–402.

[99] J. Feldtkeller, T. Güneysu, T. Moos, J. Richter-Brockmann, S. Saha, P. Sasdrich, and F. Standaert. "Combined Private Circuits - Combined Security Refurbished". In: *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, CCS 2023, Copenhagen, Denmark, November 26-30, 2023*. 2023, pp. 990–1004.

[100] J. Feldtkeller, J. Richter-Brockmann, P. Sasdrich, and T. Güneysu. "CINI MINIS: Domain Isolation for Fault and Combined Security". In: *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022*. 2022, pp. 1023–1036.

[101] I. Florescu. *Probability and Stochastic Processes*. 2014.

[102] E. Galois. *Sur la théorie des nombres*. Bulletin des Sciences Mathématiques XIII: 428. 1830.

[103] B. M. Gammel and S. Mangard. “On the Duality of Probing and Fault Attacks”. In: *J. Electron. Test.* 4 (2010), pp. 483–493.

[104] K. Gandolfi, C. Mourtel, and F. Olivier. “Electromagnetic Analysis: Concrete Results”. In: *Cryptographic Hardware and Embedded Systems - CHES 2001, Third International Workshop, Paris, France, May 14-16, 2001, Proceedings*. 2001, pp. 251–261.

[105] S. Gao and T. D. Mateer. “Additive Fast Fourier Transforms Over Finite Fields”. In: *IEEE Trans. Inf. Theory* 12 (2010), pp. 6265–6272.

[106] D. Genkin, Y. Ishai, M. Prabhakaran, A. Sahai, and E. Tromer. “Circuits resilient to additive attacks with applications to secure computation”. In: *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*. 2014, pp. 495–504.

[107] D. Genkin, A. Shamir, and E. Tromer. “RSA Key Extraction via Low-Bandwidth Acoustic Cryptanalysis”. In: *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*. 2014, pp. 444–461.

[108] R. Gennaro and Y. Lindell. “A Framework for Password-Based Authenticated Key Exchange”. In: *Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003, Proceedings*. 2003, pp. 524–543.

[109] C. Gentry, P. D. MacKenzie, and Z. Ramzan. “A Method for Making Password-Based Key Exchange Resilient to Server Compromise”. In: *Advances in Cryptology - CRYPTO 2006, 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006, Proceedings*. 2006, pp. 142–159.

[110] O. Goldreich. *Foundations of Cryptography: Basic Tools*. USA, 2000.

[111] L. Goubin and A. Martinelli. “Protecting AES with Shamir’s Secret Sharing Scheme”. In: *Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings*. 2011, pp. 79–94.

[112] L. Goubin and J. Patarin. “DES and differential power analysis the “Duplication” method”. In: *Cryptographic Hardware and Embedded Systems: First InternationalWorkshop, CHES’99 Worcester, MA, USA, August 12–13, 1999 Proceedings 1*. Springer. 1999, pp. 158–172.

[113] D. Goudarzi, A. Joux, and M. Rivain. “How to Securely Compute with Noisy Leakage in Quasilinear Complexity”. In: *Advances in Cryptology - ASIACRYPT 2018 - 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part II*. 2018, pp. 547–574.

[114] D. Goudarzi, T. Prest, M. Rivain, and D. Vergnaud. "Probing Security through Input-Output Separation and Revisited Quasilinear Masking". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 3 (2021), pp. 599–640.

[115] V. Goyal, Y. Ishai, and Y. Song. "Private Circuits with Quasilinear Randomness". In: *Advances in Cryptology - EUROCRYPT 2022 - 41st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Trondheim, Norway, May 30 - June 3, 2022, Proceedings, Part III*. 2022, pp. 192–221.

[116] L. Grassi, D. Khovratovich, C. Rechberger, A. Roy, and M. Schofnegger. "Poseidon: A New Hash Function for Zero-Knowledge Proof Systems". In: *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021*. 2021, pp. 519–535.

[117] L. Grassi, R. Lüftenegger, C. Rechberger, D. Rotaru, and M. Schofnegger. "On a Generalization of Substitution-Permutation Networks: The HADES Design Strategy". In: *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part II*. 2020, pp. 674–704.

[118] J. Ha, S. Kim, B. Lee, J. Lee, and M. Son. "Rubato: Noisy Ciphers for Approximate Homomorphic Encryption". In: *Advances in Cryptology - EUROCRYPT 2022 - 41st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Trondheim, Norway, May 30 - June 3, 2022, Proceedings, Part I*. 2022, pp. 581–610.

[119] S. Han, S. Liu, and L. Lyu. "Efficient KDM-CCA Secure Public-Key Encryption for Polynomial Functions". In: *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part II*. 2016, pp. 307–338.

[120] D. G. Harris. "Critique of the related-key attack concept". In: *Des. Codes Cryptogr.* 1-3 (2011), pp. 159–168.

[121] J. Hesse. "Separating Symmetric and Asymmetric Password-Authenticated Key Exchange". In: *Security and Cryptography for Networks - 12th International Conference, SCN 2020, Amalfi, Italy, September 14-16, 2020, Proceedings*. 2020, pp. 579–599.

[122] R. Horn and C. Johnson. *Topics in Matrix Analysis*. 1994.

[123] K. Hostáková. "Foundations of Generalized State Channel Networks". PhD thesis. Technical University of Darmstadt, Germany, 2021.

[124] J. B. J. Hwang. *Introduction to Probability*. 2014.

[125] Y. Ishai, M. Prabhakaran, A. Sahai, and D. A. Wagner. "Private Circuits II: Keeping Secrets in Tamperable Circuits". In: *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings*. 2006, pp. 308–327.

[126] Y. Ishai, A. Sahai, and D. A. Wagner. "Private Circuits: Securing Hardware against Probing Attacks". In: *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*. 2003, pp. 463–481.

[127] T. Isobe. "A Single-Key Attack on the Full GOST Block Cipher". In: *J. Cryptol.* 1 (2013), pp. 172–189.

[128] S. Jarecki, H. Krawczyk, and J. Xu. "OPAQUE: An Asymmetric PAKE Protocol Secure Against Pre-computation Attacks". In: *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part III*. 2018, pp. 456–486.

[129] Y. T. Kalai, B. Kanukurthi, and A. Sahai. "Cryptography with Tamperable and Leaky Memory". In: *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*. 2011, pp. 373–390.

[130] D. Kalman. "The Generalized Vandermonde Matrix". In: *Mathematics Magazine* 1 (1984), pp. 15–21. (Visited on 11/23/2023).

[131] J. Katz and Y. Lindell. *Introduction to Modern Cryptography, Second Edition*. 2nd. 2014.

[132] J. Katz, R. Ostrovsky, and M. Yung. "Efficient Password-Authenticated Key Exchange Using Human-Memorable Passwords". In: *Advances in Cryptology - EUROCRYPT 2001, International Conference on the Theory and Application of Cryptographic Techniques, Innsbruck, Austria, May 6-10, 2001, Proceeding*. 2001, pp. 475–494.

[133] J. Katz and V. Vaikuntanathan. "Round-Optimal Password-Based Authenticated Key Exchange". In: *J. Cryptol.* 4 (2013), pp. 714–743.

[134] A. Kerckhoffs. "La cryptographie militaire". In: *Journal des sciences militaires, vol. IX, Paris*. 1883, pp. 5–38.

[135] A. Kiayias and Y. Tselekounis. "Tamper Resilient Circuits: The Adversary at the Gates". In: *Advances in Cryptology - ASIACRYPT 2013 - 19th International Conference on the Theory and Application of Cryptology and Information Security, Bengaluru, India, December 1-5, 2013, Proceedings, Part II*. 2013, pp. 161–180.

[136] S. Kiyoshima. "Round-Efficient Black-Box Construction of Composable Multi-Party Computation". In: *J. Cryptol.* 1 (2019), pp. 178–238.

[137] D. Knichel, P. Sasdrich, and A. Moradi. "SILVER - Statistical Independence and Leakage Verification". In: *Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part I*. 2020, pp. 787–816.

[138] P. C. Kocher. "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems". In: *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*. 1996, pp. 104–113.

[139] P. C. Kocher, J. Jaffe, and B. Jun. "Differential Power Analysis". In: *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*. 1999, pp. 388–397.

[140] B. Koo, D. Hong, and D. Kwon. "Related-Key Attack on the Full HIGHT". In: *Information Security and Cryptology - ICISC 2010 - 13th International Conference, Seoul, Korea, December 1-3, 2010, Revised Selected Papers*. 2010, pp. 49–67.

[141] F. Liu and A. Lysyanskaya. "Tamper and Leakage Resilience in the Split-State Model". In: *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*. 2012, pp. 517–532.

[142] R. S. (London). *Philosophical Transactions of the Royal Society of London: Giving Some Accounts of the Present Undertakings, Studies, and Labours, of the Ingenious, in Many Considerable Parts of the World*. Bd. 148, pp. 17–37.

[143] X. Lu, B. Li, and D. Jia. "KDM-CCA Security from RKA Secure Authenticated Encryption". In: *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*. 2015, pp. 559–583.

[144] S. Mangard, T. Popp, and B. M. Gammel. "Side-Channel Leakage of Masked CMOS Gates". In: *Topics in Cryptology - CT-RSA 2005, The Cryptographers' Track at the RSA Conference 2005, San Francisco, CA, USA, February 14-18, 2005, Proceedings*. 2005, pp. 351–365.

[145] S. Mangard, N. Pramstaller, and E. Oswald. "Successfully Attacking Masked AES Hardware Implementations". In: *Cryptographic Hardware and Embedded Systems - CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings*. 2005, pp. 157–171.

[146] L. Masure, P. Méaux, T. Moos, and F. Standaert. "Effective and Efficient Masking with Low Noise Using Small-Mersenne-Prime Ciphers". In: *Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part IV*. 2023, pp. 596–627.

[147] T. S. Messerges. "Securing the AES Finalists Against Power Analysis Attacks". In: *Fast Software Encryption, 7th International Workshop, FSE 2000, New York, NY, USA, April 10-12, 2000, Proceedings*. 2000, pp. 150–164.

[148] A. Muffet. "Facebook: Password hashing & authentication". In: *Real World Crypto*. 2015.

[149] C. Namprempre, P. Rogaway, and T. Shrimpton. "Reconsidering Generic Composition". In: *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*. 2014, pp. 257–274.

[150] *National Institute of Standards and Technology. Lightweight cryptography standardization process*. 2015.

[151] N. Pippenger. "On Networks of Noisy Gates". In: *26th Annual Symposium on Foundations of Computer Science, Portland, Oregon, USA, 21-23 October 1985*. 1985, pp. 30–38.

[152] T. Prest, D. Goudarzi, A. Martinelli, and A. Passelègue. "Unifying Leakage Models on a Rényi Day". In: *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part I*. 2019, pp. 683–712.

[153] E. Prouff and M. Rivain. "Masking against Side-Channel Attacks: A Formal Security Proof". In: *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*. 2013, pp. 142–159.

[154] E. Prouff and T. Roche. "Higher-Order Glitches Free Implementation of the AES Using Secure Multi-party Computation Protocols". In: *Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings*. 2011, pp. 63–78.

[155] S. Rasoolzadeh, A. R. Shahmirzadi, and A. Moradi. "Impeccable Circuits III". In: *IEEE International Test Conference, ITC 2021, Anaheim, CA, USA, October 10-15, 2021*. 2021, pp. 163–169.

[156] I. S. Reed and G. Solomon. "Polynomial Codes Over Certain Finite Fields". In: *Journal of the Society for Industrial and Applied Mathematics* 2 (1960), pp. 300–304. eprint: `https://doi.org/10.1137/0108018`.

[157] E. Rescorla. “The Transport Layer Security (TLS) Protocol Version 1.3”. In: *RFC* (2018), pp. 1–160.

[158] J. Richter-Brockmann, J. Feldtkeller, P. Sasdrich, and T. Güneysu. “VERICA - Verification of Combined Attacks Automated formal verification of security against simultaneous information leakage and tampering”. In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 4 (2022), pp. 255–284.

[159] M. Rivain and E. Prouff. “Provably Secure Higher-Order Masking of AES”. In: *Cryptographic Hardware and Embedded Systems, CHES 2010, 12th International Workshop, Santa Barbara, CA, USA, August 17-20, 2010. Proceedings*. 2010, pp. 413–427.

[160] T. Roche and E. Prouff. “Higher-order glitch free implementation of the AES using Secure Multi-Party Computation protocols - Extended version”. In: *J. Cryptogr. Eng.* 2 (2012), pp. 111–127.

[161] P. Rogaway. “Nonce-Based Symmetric Encryption”. In: *Fast Software Encryption, 11th International Workshop, FSE 2004, Delhi, India, February 5-7, 2004, Revised Papers*. 2004, pp. 348–359.

[162] T. Schneider, A. Moradi, and T. Güneysu. “ParTI - Towards Combined Hardware Countermeasures Against Side-Channel and Fault-Injection Attacks”. In: *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II*. 2016, pp. 302–332.

[163] O. Seker, A. Fernandez-Rubio, T. Eisenbarth, and R. Steinwandt. “Extending Glitch-Free Multiparty Protocols to Resist Fault Injection Attacks”. In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 3 (2018), pp. 394–430.

[164] A. R. Shahmirzadi, S. Rasoolzadeh, and A. Moradi. “Impeccable Circuits II”. In: *57th ACM/IEEE Design Automation Conference, DAC 2020, San Francisco, CA, USA, July 20-24, 2020*. 2020, pp. 1–6.

[165] A. Shamir. “Cryptography: State of the Science”. In: *ACM A.M. Turing Award Lecture*. 2002.

[166] A. Shamir. “How to Share a Secret”. In: *Commun. ACM* 11 (1979), pp. 612–613.

[167] S. P. Skorobogatov and R. J. Anderson. “Optical Fault Induction Attacks”. In: *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*. 2002, pp. 2–12.

[168] S. Vaudenay. “Clever Arbiters Versus Malicious Adversaries - On the Gap Between Known-Input Security and Chosen-Input Security”. In: *The New Codebreakers - Essays Dedicated to David Kahn on the Occasion of His 85th Birthday*. 2016, pp. 497–517.

[169] C. D. Walter. “Sliding Windows Succumbs to Big Mac Attack”. In: *Cryptographic Hardware and Embedded Systems - CHES 2001, Third International Workshop, Paris, France, May 14-16, 2001, Proceedings*. 2001, pp. 286–299.

[170] M. Wang, K. He, J. Chen, Z. Li, W. Zhao, and R. Du. “Biometrics-Authenticated Key Exchange for Secure Messaging”. In: *CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021*. 2021, pp. 2618–2631.

[171] E. Waring. “Problems concerning interpolations”. In: *Philosophical Transactions of the Royal Society*, pp. 59–67.

[172] K. Xagawa. “Message Authentication Codes Secure against Additively Related-Key Attacks”. In: *IACR Cryptol. ePrint Arch.* (2013), p. 111.

[173] A. L. Young and M. Yung. “The Dark Side of "Black-Box" Cryptography, or: Should We Trust Capstone?” In: *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*. 1996, pp. 89–103.

[174] J. G. Zehfuss. “Ueber eine gewisse Determinante”. In: *Zeitschrift für Mathematik und Physik*. 1858, pp. 298–301.

[175] J. Zeitschner, N. Müller, and A. Moradi. “PROLEAD_SW Probing-Based Software Leakage Detection for ARM Binaries”. In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 3 (2023), pp. 391–421.

[176] L. Zussa, J. Dutertre, J. Clédière, and A. Tria. “Power supply glitch induced faults on FPGA: An in-depth analysis of the injection mechanism”. In: *2013 IEEE 19th International On-Line Testing Symposium (IOLTS), Chania, Crete, Greece, July 8-10, 2013*. 2013, pp. 110–115.

List of Figures

List of Tables

List of Abbreviations

PDT	Probe Distribution Table
DG	Dependency Graph
faPAKE	Fuzzy Asymmetric Password-Authenticated Key Exchange
aPAKE	Asymmetric Password-Authenticated Key Exchange
fPAKE	Fuzzy Password-Authenticated Key Exchange
PAKE	Password-Authenticated Key Exchange
UC	Universal Composability
E&M	Encrypt-and-MAC
EtM	Encrypt-then-MAC
MtE	MAC-then-Encrypt
GGM	Generic Group Model
STRAPS	Sampled Testing of the RAndom Probing Security
RKA	Related-Key Attack
OT	Oblivious Transfer
AEAD	Authenticated Encryption with Associated Data

A. Towards Tight Random Probing Security

This chapter corresponds to our published article at CRYPTO 2021 [57], with minor edits. Our full version can be found in [56].

Towards Tight Random Probing Security

Gaëtan Cassiers[1], Sebastian Faust[2],
Maximilian Orlt[2], François-Xavier Standaert[1]

[1] Crypto Group, ICTEAM Institute, UCLouvain, Louvain-la-Neuve, Belgium
[2] TU Darmstadt, Darmstadt, Germany

Abstract. Proving the security of masked implementations in theoretical models that are relevant to practice and match the best known attacks of the side-channel literature is a notoriously hard problem. The random probing model is a promising candidate to contribute to this challenge, due to its ability to capture the continuous nature of physical leakage (contrary to the threshold probing model), while also being convenient to manipulate in proofs and to automate with verification tools. Yet, despite recent progress in the design of masked circuits with good asymptotic security guarantees in this model, existing results still fall short when it comes to analyze the security of concretely useful circuits under realistic noise levels and with low number of shares. In this paper, we contribute to this issue by introducing a new composability notion, the *Probe Distribution Table (PDT)*, and a new tool (called STRAPS, for the Sampled Testing of the RAndom Probing Security). Their combination allows us to significantly improve the tightness of existing analyses in the most practical (low noise, low number of shares) region of the design space. We illustrate these improvements by quantifying the random probing security of an AES S-box circuit, masked with the popular multiplication gadget of Ishai, Sahai and Wagner from Crypto 2003, with up to six shares.

1 Introduction

Context. Modern cryptography primarily analyzes the security of algorithms or protocols in a black-box model where the adversary has only access to their inputs and outputs. Since the late nineties, it is known that real-world implementations suffer from so-called side-channel leakage, which gives adversaries some information about intermediate computation states that are supposedly hidden. In this work, we focus on an important class of side-channel attacks against embedded devices, which exploits physical leakage such as their power consumption [26] or electro-magnetic radiation [22]. We are in particular concerned with the masking countermeasure [14], which is one of the most investigated solutions to mitigate side-channel attacks. In this context, the main scientific challenge we tackle is to find out security arguments that are at the same time practically relevant and theoretically sound.

Two separated worlds. In view of the difficulty to model side-channel attacks, their practical and theoretical investigations have first followed quite independent paths. On the practical side, the analysis of masked implementations as currently performed by evaluation laboratories is mostly based on statistical testing. Approaches for this purpose range from detection-based testing, which aims at identifying leakage independently of whether it can be exploited [32], to attack-based testing under various adversarial assumptions, which aims at approximating (if possible bounding) the concrete security level of the implementation with actual (profiled or non-profiled) attacks such as [15,11] and their numerous follow ups. On the theoretical side, the first model introduced to capture the security of masked implementations is the t-threshold probing model introduced by Ishai, Sahai and Wagner (ISW) [24]. In this model, leaky computation is captured as the evaluation of an arithmetic circuit, and the adversary may choose t wires of the circuit for which she receives the value they carry. The adversary succeeds if she recovers a secret input variable of the circuit.

The pros and cons of both approaches are easy to spot. On the one hand, statistical testing provides quantitative evaluations against concrete adversaries, but the guarantees it offers are inherently heuristic and limited to the specific setting used for the evaluations. On the other hand, theoretical models enable more general conclusions while also having a good potential for automation [5], but they may imperfectly abstract physical leakage. For some imperfections, tweaking the model appeared to be feasible. For example,

ISW's threshold probing model initially failed to capture physical defaults such as glitches that can make masking ineffective [27,28]. Such glitches were then integrated in the model [21] and automated [10,4,6]. Yet, it remained that the threshold probing model is inherently unable to capture the continuous nature of physical leakage, and therefore the guarantees it provides can only be qualitative, as reflected by the notion of probing security order (i.e., the number of shares that the adversary can observe without learning any sensitive information). This also implies that so-called horizontal attacks taking advantage of multiple leakage points to reduce the noise of the implementations cannot be captured by this model [7].

An untight unifying approach. As a result of this limitation, the noisy leakage model was introduced by Prouff and Rivain [30]. In this model, each wire in the circuit leaks independently a noisy (i.e., partially randomized) value to the adversary. In an important piece of work, Duc et al. then proved that security in the threshold probing model implies security in the noisy leakage model, for some values of the model parameters [17]. This result created new bridges between the practical and theoretical analyzes of masked implementations. In particular, it made explicit that the security of this countermeasure depends both on a security order (which, under an independence assumption, depends on the number of shares) and on the noise level of the shares' leakage. So conceptually, it implies that it is sound to first evaluate the probing security order of an implementation, next to verify that this security order is maintained in concrete leakages (e.g., using detection-based statistical testing) and finally to assess the noise level. Yet, and as discussed in [18], such an analysis is still not tight: choosing security parameters based on this combination of models and the reductions connecting them would lead to overly expensive implementations compared to a choice based on the best known (profiled) side-channel attacks.

A tighter middle-ground. Incidentally, the reduction of Duc et al. also considered an intermediate level of abstraction denoted as the random probing model. In this model, each wire in the circuit independently leaks its value with probability p (and leaks no information with probability $1-p$). Technically, it turns out that the aforementioned tightness issue is mostly due to the reduction from the threshold probing model to the random probing model, while there is a closer relationship between the random probing model and the noisy leakage model [19,29]. Since the random probing model remains relatively easy to manipulate (and automate) in circuit-level proofs, it therefore appears as an interesting candidate to analyze masking schemes with tight security guarantees.

Like the noisy leakage model, the random probing model captures the concept of "noise rate", which specifies how the noise level of an implementation must evolve with the number of shares in order to remain secure against horizontal attacks. As a result, different papers focused on the design and analysis of gadgets with good (ideally constant) noise rate [1,3,2,23,20]. While these papers provide important steps in the direction of asymptotically efficient masking schemes, the actual number of shares they need to guarantee a given security level and/or the noise level they require to be secure remain far from practical. To the best of our knowledge, the most concrete contribution in this direction is the one of Belaïd et al. [8,9], which introduced a compiler that can generate random probing secure circuits from small gadgets satisfying a notion of "random probing expandability", together with a tool (called VRAPS) that quantifies the random probing security of a circuit from its leakage probability. With this tool, they reduce the level of noise required for security to practically acceptable values, but the number of shares required in order to reach a given security level for their (specialized) constructions is still significantly higher than expected from practical security evaluations – we give an example below.

Our contributions. In this paper, we improve the tightness of masking security proofs in the most practical (low noise, low number of shares) region of the design space, focusing on practical ISW-like multiplication gadgets, integrated in an AES S-box design for illustration purposes. More precisely:

We first introduce STRAPS, a tool for the Sampled Testing of the RAndom Probing Security of small circuits, which uses the Monte-Carlo technique for probability bounding and is released under an open source license.[1]

Since this tool is limited to the analysis of small circuits and/or small security orders due to computational reasons, we next combine it with a new compositional strategy that exploits a new security property for

[1] https://github.com/cassiersg/STRAPS

masked gadgets, the Probe Distribution Table (PDT), which gives tighter security bounds for composed circuits and is integrated in the STRAPS tool. This combination of tool and compositional strategy allows us analyzing significantly larger circuits and security orders than an exhaustive approach, while also being able to analyze any circuit (i.e., it does not rely on an expansion strategy [2]).

We finally confirm the practical relevance of our findings by applying them to a masked AES S-box using ISW gadgets. We show how to use them in order to discuss the trade-off between the security order and the noise level (i.e., leakage probability) of concrete masked implementations on formal bases. As an illustration, we use our tools to compare the impact of different refreshing strategies for the AES S-box (e.g., no refresh, simple refreshes or SNI refreshes) in function of the noise level. We can also claim provable security levels for useful circuits that are close to the worst-case attacks discussed in [18] which is in contrast to previous works. Precisely, we are able to prove the same statistical security order (i.e., the highest statistical moment of the leakage distribution that is independent of any sensitive information) as in this reference, for realistic leakage probabilities in the range $[10^{-1}; 10^{-4}]$. For example, our AES S-box with 6 shares and leakage probability of $\approx 10^{-3}$ ensures security against an adversary with up to one billion measurements. Belaïd et al. would need 27 shares to reach the same security.

Open problems and related works. While providing tight results for a masked AES S-box implementation with up to 6 shares, therefore opening the way towards tight random probing security in general, we note that our composition results are not completely tight in certain contexts which (we discuss in the paper and) could pop up in other circuits than the AES S-box. Hence, generalizing our results to be tight for any circuit is an interesting open problem and the same holds for optimizing the complexity of our verification techniques in order to scale with even larger circuits and number of shares.

Besides, we illustrated our results with the popular ISW multiplications in order to show their applicability to non-specialized gadgets, which are concretely relevant for the number of shares and noise levels we consider. Yet, since one of the motivations to use the random probing model is to capture horizontal attacks, it would also be interesting to analyze multiplication algorithms that provide improved guarantees against such attacks thanks to a logarithmic or even constant noise rate and could not be proven so far (e.g., [7,13]).

2 Background

Notations. In this work, we consider Boolean or arithmetic circuits over finite fields $\mathbb{F}_{2^m}$ and refer to the underlying additive and multiplicative operations as $\oplus$ and $\odot$, respectively. For the sake of simplicity we also use these operations for a share-wise composition of vectors $(v_i)_{i\in[n]}$ and $(w_i)_{i\in[n]}$ with $[n] = \{0, 1, \ldots, n-1\}$ such that $(v_i)_{i\in[n]} \odot (w_i)_{i\in[n]} := (v_i \odot w_i)_{i\in[n]}$ and $(v_i)_{i\in[n]} \oplus (w_i)_{i\in[n]} := (v_i \oplus w_i)_{i\in[n]}$. Furthermore, we use the Kronecker product to compose two real matrices $A = (a_{i,j})_{i\in[m],j\in[n]}$, $B = (b_{i,j})_{i\in[k],j\in[l]}$ such that $A \otimes B = (a_{i,j}B)_{i\in[m],j\in[n]}$. We also denote $x \xleftarrow{\$} \mathcal{X}$ as choosing x uniformly at random from the set $\mathcal{X}$, and $\mathcal{X}^{(k)}$ as the set of subsets of $\mathcal{X}$ of size k.

Masking. Masking is a well known countermeasure against side-channel attacks. With an encoding scheme $(\mathsf{Enc}(\cdot), \mathsf{Dec}(\cdot))$, sensitive data x is split into n shares (represented as a vector) $(x_i)_{i\in[n]} \leftarrow \mathsf{Enc}(x)$, and the decoding function takes as input the n shares and recovers the unshared value x, i.e., $x \leftarrow \mathsf{Dec}((x_i)_{i\in[n]})$. For security we require that any subset of $n-1$ shares does not reveal any information about the sensitive data x. In this work, we focus on additive sharing $\mathsf{Dec}((x_i)_{i\in[n]}) = \bigoplus_{i=0}^{n-1} x_i$, which is the most studied scheme.

Circuit model. As common in masking scheme literature, we model computation as arithmetic circuits operating over a finite field $\mathbb{F}_{2^m}$. The circuit is represented by a directed acyclic graph, where each node is a gate that has a fixed number of input and output wires (incoming and outgoing edges) that carry arithmetic values. We consider the following types of gates in our circuits: addition  and multiplication gates have two input wires and one output wire, and perform the corresponding arithmetic operation. The copy gate has one input and two outputs, and is used to duplicate a value. Finally, the random gate has no input and one output, which carries a uniformly distributed value. The constant gate outputs a constant value a.

In a masked circuit the gates are represented by subcircuits called gadgets G. These gadgets operate on encoded inputs and produce encoded outputs. The gadgets contain: (1) A set of gates; (2) The set of wires that connect the inputs and outputs of those gates named internal wires ($\mathcal{W}$); (3) The set of wires only connected with those gates' input named input wires ($\mathcal{I}$); (4) The set of output gates $\hat{\mathcal{O}}$ (which is the subset of its gates that output wires that are not connected to another gate of the gadget). The gadgets, however, contain no output wires, such that each wire in a circuit composed of multiple gadgets belongs to only one of its composing gadgets. For convenience, we also write $\mathcal{O}$ for the set of output wires of the gates in $\hat{\mathcal{O}}$, although these wires are not part of the gadget but are the next gadgets input wires. We denote $\mathcal{A} = \mathcal{W} \cup \mathcal{I}$ the set of all wires in the gadget. The inputs and outputs of a gadget are partitioned in (ordered) sets of n elements named sharings (and each element is a share). A gadget G_f that implements the function $f : \mathbb{F}^l \mapsto \mathbb{F}^k$ with n shares has l input sharings and k output sharings. Let $(y_i^0)_{i\in[n]}$, ..., $(y_i^{k-1})_{i\in[n]}$ be the values of the output sharings when the input sharings have the values $(x_i^0)_{i\in[n]}$, ..., $(x_i^{l-1})_{i\in[n]}$. It must hold that

$$f(\mathsf{Dec}((x_i^0)_{i\in[n]}), \ldots, \mathsf{Dec}((x_i^{l-1})_{i\in[n]})) = (\mathsf{Dec}((y_i^0)_{i\in[n]}), \ldots, \mathsf{Dec}((y_i^{k-1})_{i\in[n]})).$$

In this work, we use various gadgets. First, gadgets that implement linear operations (addition $\mathsf{G}_{\oplus}$, copy $\mathsf{G}_{\copyright}$, squaring $\mathsf{G}_{.^2}$), which we implement share-wise. Next, we use the ISW multiplication gadget [24]. Finally, we use refresh gadgets $\mathsf{G}_{\circledR}$ which re-randomize a sharing $(x_i)_{i\in[n]}$ to $(y_i)_{i\in[n]}$ such that $\mathsf{Dec}((x_i)_{i\in[n]}) = \mathsf{Dec}((y_i)_{i\in[n]})$. We consider two refresh gadget implementations: the simple refresh and the SNI, randomness-optimized refresh gadgets from [12]. Their algorithmic description is given in the extended version of the paper .

Leakage model. In this work we consider the p-random probing model as originally introduced by Ishai, Sahai and Wagner [24]. This model defines the following random probing experiment. Let $\mathcal{W}$ be a set of wires in a circuit, $\mathcal{L}_p(\mathcal{W})$ is a random variable with $\mathcal{L}_p(\mathcal{W}) \subseteq \mathcal{W}$, such that each wire $w \in \mathcal{W}$ is in $\mathcal{L}_p(\mathcal{W})$ with probability p (independently for each wire). Following this notation, for a gadget G, we denote by $\mathcal{L}_p(\mathsf{G}) := \mathcal{L}_p(\mathcal{W}, \mathcal{I}) := (\mathcal{L}_p(\mathcal{W}), \mathcal{L}_p(\mathcal{I}))$, where $\mathcal{W}$ and $\mathcal{I}$ are the set of internal and input wires of G, respectively.

For a gadget G, a set of probes is a successful attack for an input sharing $(x_i)_{i\in[n]}$ if the joint distribution of the values carried by the probes depends on $\mathsf{Dec}((x_i)_{i\in[n]})$ (assuming that the other input sharings are public). The security level of G in the p-random probing model (or p-random probing security) with respect to an input sharing $(x_i)_{i\in[n]}$ is the probability (over the randomness in $\mathcal{L}_p$) that a set of probes $\mathcal{L}_p(\mathsf{G})$ is a successful attack. As a result, the security of a gadget in bits is worth $-\log_2$(security level). We omit to mention the attacked input sharing when the gadget has only one input sharing.

3 Random probing security of small circuits

In this section, we show how to efficiently compute an upper bound on the random probing security level of relatively small gadgets, and we illustrate the results on well-known masked gadgets. We also describe the high-level ideas that will lead to the STRAPS tool that we describe in Section 5.3.

3.1 Derivation of a random probing security bound

We first derive a way to compute the security level of a gadget for various values of p, using some computationally heavy pre-processing. Next, we explain a way to use statistical confidence intervals to reduce the cost of the pre-processing. Finally, we detail how these techniques are implemented in a practical algorithm.

A simple bound. We can obtain the security level of a small circuit by computing first the statistical distribution of $\mathcal{L}_p(\mathsf{G})$ (i.e., $\Pr[\mathcal{L}_p(\mathcal{A}) = \mathcal{A}']$ for each subset $\mathcal{A}' \subset \mathcal{A}$). Then, for each possible set of probes $\mathcal{A}'$, we do a dependency test in order to determine if the set is a successful attack, denoted as $\delta_{\mathcal{A}'} = 1$, while $\delta_{\mathcal{A}'} = 0$ otherwise [8]. There exist various tools that can be used to carry out such a dependency test, such

as maskVerif [4] or SILVER [25] (while such tools are designed to prove threshold probing security, they perform dependency tests as a sub-routine). A first naive algorithm to compute the security level ϵ is thus given by the equation

$$\epsilon = \sum_{\substack{\mathcal{A}' \subset \mathcal{A} \\ \text{s.t. } \delta_{\mathcal{A}'}=1}} \Pr[\mathcal{L}_p(\mathcal{A}) = \mathcal{A}']. \tag{1}$$

The computational cost of iterating over all possible probe sets grows exponentially with $|\mathcal{A}|$: for a circuit with $|\mathcal{A}|$ internal wires, one has to do $2^{|\mathcal{A}|}$ dependency tests, for each value of p (e.g., we have $|\mathcal{A}| = 57$ for the ISW multiplication with three shares). To efficiently cover multiple values of p, we introduce a first improvement to the naive algorithm given by Equation (1). For each $i \in \{0, \ldots, |\mathcal{A}|\}$, we compute the number c_i of sets of probes of size i that are successful attacks $c_i = \left|\left\{\mathcal{A}' \in \mathcal{A}^{(i)} \text{ s.t. } \delta_{\mathcal{A}'} = 1\right\}\right|$. Then, we can compute

$$\epsilon = \sum_{i=0}^{|\mathcal{A}|} p^i (1-p)^{|\mathcal{A}|-i} c_i, \tag{2}$$

which gives us a more efficient algorithm to compute random probing security, since it re-uses the costly computation of c_i for multiple values of p.

The VRAPS tool [8] computes c_i for small values of i by computing $\delta_{\mathcal{A}'}$ for all $\mathcal{A}' \in \mathcal{A}^{(i)}$. This is however computationally intractable for larger i values, hence they use the bound $c_i \leq \binom{|\mathcal{A}|}{i}$ in such cases.

A statistical bound. Let us now show how to improve the bound $c_i \leq \binom{|\mathcal{A}|}{i}$ while keeping a practical computational cost. At a high level, we achieve this by using a Monte-Carlo method whose idea is as follows: instead of computing directly ϵ, we run a randomized computation that gives us information about ϵ (but not its exact value). More precisely, the result of our Monte-Carlo method is a random variable ϵ^U that satisfies $\epsilon^U \geq \epsilon$ with probability at least $1-\alpha$ (the confidence level), where α is a parameter of the computation. That is, $\Pr_{\text{MC}}\left[\epsilon^U \geq \epsilon\right] \geq 1-\alpha$, where $\Pr_{\text{MC}}$ means the probability over the randomness used in the Monte-Carlo method.[2] In the rest of this work, we use $\alpha = 10^{-6}$ since we consider that it corresponds to a sufficient confidence level.[3]

Let us now detail the method. First, let $r_i = c_i / \left|\mathcal{A}^{(i)}\right|$. We remark that r_i can be interpreted as a probability: $r_i = \Pr_{\mathcal{A}' \xleftarrow{\$} \mathcal{A}^{(i)}}[\delta_{\mathcal{A}'} = 1]$. The Monte-Carlo method actually computes r_i^U such that $r_i^U \geq r_i$ with probability at least $1 - \alpha/(|\mathcal{A}|+1)$. Once the r_i^U are computed, the result is

$$\epsilon^U = \sum_{i=0}^{|\mathcal{A}|} p^i (1-p)^{|\mathcal{A}|-i} \binom{|\mathcal{A}|}{i} r_i^U, \tag{3}$$

which ensures that $\epsilon^U \geq \epsilon$ for any p with confidence level $1-\alpha$, thanks to the union bound. Next, r_i^U is computed by running the following experiment: take t_i samples $\mathcal{A}' \xleftarrow{\$} \mathcal{A}^{(i)}$ uniformly at random (this sampling is the random part of the Monte-Carlo method) and compute the number s_i of samples for which $\delta_{\mathcal{A}'} = 1$. By definition, s_i is a random variable that follows a binomial distribution $B(t_i, r_i)$: the total number of samples is t_i and the "success" probability is r_i. We can thus use the bound derived in [33]. If r_i^U satisfies $\text{CDF}_{\text{binom}}(s_i; t_i, r_i^U) = \alpha/(|\mathcal{A}|+1)$, then $\Pr[r_i^U \geq r_i] = 1 - \alpha/(|\mathcal{A}|+1)$, which gives

$$r_i^U = \begin{cases} 1 & \text{if } s_i = t_i, \\ x \quad \text{s.t.} \quad I_x(s_i+1, t_i - s_i) = 1 - \alpha/(|\mathcal{A}|+1) & \text{otherwise,} \end{cases} \tag{4}$$

[2] In other words, $[0, \epsilon^U]$ is a conservative confidence interval for ϵ with nominal coverage probability of $1-\alpha$.

[3] This parameter is not critical: we can obtain a similar value for ϵ^U with higher confidence level by increasing the amount of computation: requiring $\alpha = 10^{-12}$ would roughly double the computational cost of the Monte-Carlo method.

where $I_x(a,b)$ is the regularized incomplete beta function. We can similarly compute a lower bound ϵ^L such that $\epsilon^L \leq \epsilon$ with confidence coefficient $1-\alpha$, which we compute by replacing r_i^U with r_i^L in Equation (3), where:

$$r_i^L = \begin{cases} 0 & \text{if } s_i = 0, \\ x \quad \text{s.t.} \quad I_x(s_i, t_i - s_i + 1) = \alpha / \left(|\mathcal{A}| + 1\right) & \text{otherwise.} \end{cases} \tag{5}$$

A hybrid algorithm. Our Monte-Carlo method has a main limitation: when $r_i = 0$ the bound r_i^U will not be null (it will be proportional to $1/t_i$). This means that we cannot prove tightly the security of interesting gadgets when p is small. For instance, let us take a fourth-order secure gadget (that is, $r_0 = r_1 = r_2 = r_3 = r_4 = 0$). If $r_1^U \neq 1$, then ϵ^U scales like $r_1^U p$ as p becomes small (other, higher degree, terms become negligible). A solution to this problem would be to set t_i to a large number, such that, in our example, r_1^U would be small enough to guarantee that $r_1^U p \ll r_5 p^5$ for all considered values of p. If we care about $p = 10^{-3}$, this means $r_1^U \ll 10^{-12} \cdot r_5 \leq 10^{-12}$. This is however practically infeasible since the number of samples t_1 is of the order of magnitude $1/r_1^U > 10^{12}$.

There exist another solution, which we call the hybrid algorithm: perform a full exploration of $\mathcal{A}^{(i)}$ (i.e., use the algorithm based on Equation (2)) when it is not computationally too expensive (i.e., when $\left|\mathcal{A}^{(i)}\right|$ is below some limit N_{max}), and otherwise use the Monte-Carlo method. The goal of this hybrid algorithm is to perform a full exploration when $r_i = 0$ (in order to avoid the limitation discussed above), which can be achieved for gadgets with a small number n of shares. Indeed, r_i can be null only for $i < n$ (otherwise there can be probes on all the shares of the considered input sharing), and the number of cases for the full exploration is therefore $\left|\mathcal{A}^{(i)}\right| = \binom{|\mathcal{A}|}{i} \leq \binom{|\mathcal{A}|}{n-1}$, which is smaller than N_{max} if n and $|\mathcal{A}|$ are sufficiently small. The latter inequality holds if $|\mathcal{A}| \geq 2(n-1)$, which holds for all non-trivial gadgets.

Algorithm 1 Random probing security algorithm: compute r_i^U, r_i^L for a given $\mathcal{A}$ and i. The parameters are N_{max} and N_t.

Require $N_t \leq N_{max}$
$N_{sets} = \binom{|\mathcal{A}|}{i}$
$t_i \leftarrow 1$, $s_i \leftarrow 0$ ▷ t_i: total number of samples, s_i: successful attacks
while $t_i \leq N_{max} \wedge s_i < N_t$ **do** ▷ First Monte-Carlo sampling loop
  $\mathcal{A}' \xleftarrow{\$} \mathcal{A}^{(i)}$
  if $\delta_{\mathcal{A}'} = 1$ **then**
    $s_i \leftarrow s_i + 1$.
  $t_i \leftarrow t_i + 1$
if $N_{sets} \leq t_i$ **then** ▷ Enumerate $\mathcal{A}^{(i)}$ if it is cheaper than Monte-Carlo.
  $s_i \leftarrow 0$
  for all $\mathcal{A}' \in \mathcal{A}^{(i)}$ **do**
    if $\delta_{\mathcal{A}'} = 1$ **then**
      $s_i \leftarrow s_i + 1$
  $r_i^U \leftarrow s_i / N_{sets}$, $r_i^L \leftarrow s_i / N_{sets}$
else ▷ Re-run Monte-Carlo to avoid bias due to N_t early stopping.
  $s_i \leftarrow 0$
  Repeat t_i **times**
    $\mathcal{A}' \xleftarrow{\$} \mathcal{A}^{(i)}$
    if $\delta_{\mathcal{A}'} = 1$ **then**
      $s_i \leftarrow s_i + 1$
  Compute r_i^U and r_i^L using Equations (4) and (5).

Algorithm 1 describes how we choose between full enumeration and Monte-Carlo sampling, which is the basis of our STRAPS tool (see Section 5.3 for more details). The algorithm adds a refinement on top of the

above explanation: if we can cheaply show that r_i is far from zero, we do not perform full exploration even if it would not be too expensive. It accelerates the tool, while keeping a good bound. This optimization is implemented by always starting with a Monte-Carlo sampling loop that takes at most N_{max} samples, with an early stop if s_i goes above the value of a parameter N_t (we typically use parameters such that $N_{max} \gg N_t$). The parameter N_t determines the relative accuracy of the bound we achieve when we do the early stop: in the final sampling, we will have $s_i \approx N_t$, which means that the uncertainty on r_i decreases as N_t increases. The parameter N_{max} has an impact when r_i is small and we do not reach N_t successful attacks: it limits both the maximum size of $\mathcal{A}^{(i)}$ for which full exploration is performed, and the number of samples used for the Monte-Carlo method.

Remark. The Monte-Carlo method is limited to the random probing model and cannot be used to prove security in the threshold probing model since proving security in this model means proving that $r_i = 0$, which it cannot do. Our hybrid algorithm, however, can prove threshold probing security for the numbers of probes i where it does full enumeration of $\mathcal{A}^{(j)}$ for all $j \in \{0, \ldots, i\}$.

Dependency test. We use the dependency test algorithm from maskVerif [4], as it offers two important characteristics: (i) it gives the set of input shares on which the probes depend, not only if there is a dependency to the unshared variable (the reason for this appears in Section 5.1), and (ii) it is quite efficient. One drawback of the maskVerif dependency test is that in some cases, it wrongly reports that the adversary succeeds, which implies that the statistical lower bound is not anymore a lower bound for the security level, and the statistical upper bound is not completely tight (but it is still an upper bound for the true security level). In this case, we refer to the statistical lower bound as the *stat-only* lower bound. While the stat-only lower bound is not indicative of the security level, it remains useful to quantify the statistical uncertainty and therefore to assess whether one could improve the tightness of the upper bound by increasing the number of samples in the Monte Carlo method.

3.2 Security of some simple gadgets

We now present the results of random probing security evaluations using the previously described tools. First, we discuss the sharewise XOR gadget and the ISW multiplication gadget with n shares. Next, we discuss the impact of the two parameters of our algorithm (N_{max} and N_t) on the tightness of the results and on the computational complexity (i.e., the execution time) of the tool.

In Figure 1 (left), we show the security level (with respect to one of the inputs) of the addition gadget for $n = 1, \ldots, 6$ shares. We can see that the security level of the gadget is proportional to p^n, which is expected. Indeed, the graph of this share-wise gadget is made of n connected components (so-called "circuit shares" [12]) such that each share of a given input sharing belongs to a distinct component, and the adversary needs at least one probe in each of them to succeed. This trend can also be linked with the security order in the threshold probing model. Since the gadget is $n-1$-threshold probing secure, a successful attack contains at least n probes, hence has probability proportional to p^n.

We can observe a similar trend for the ISW multiplication gadget (Figure 1, right). Since the gadget is $n-1$-threshold probing secure, the security level scales proportionally to p^n for small values of p. For larger values of p, the security level of this gadget is worse than p^n, which is due to the larger number of wires, and the increased connectivity compared to the addition gadgets. It implies that there are many sets of probes of sizes $n+1$, $n+2$, ... that are successful attacks (which is usually referred to as horizontal attacks in the practical side-channel literature [7]). These sets make up for a large part of the success probability when $p > 0.05$ due to their large number, even though they individually have a lower probability of occurring than a set of size n (for $p < 0.5$).

Next, we discuss the impact of parameters N_{max} and N_t in Algorithm 1 on the tightness of the bounds we can compute. We first focus on the impact of N_t, which is shown on Figure 2. For $N_t = 10$, we have a significant distance between the statistical upper and lower bounds, while the gap becomes small for $N_t = 100$ and $N_t = 1000$. This gap appears as a bounded factor between the upper and lower bounds which, as discussed previously, is related to the accuracy of the estimate of a proportion when we have about N_t positive samples.

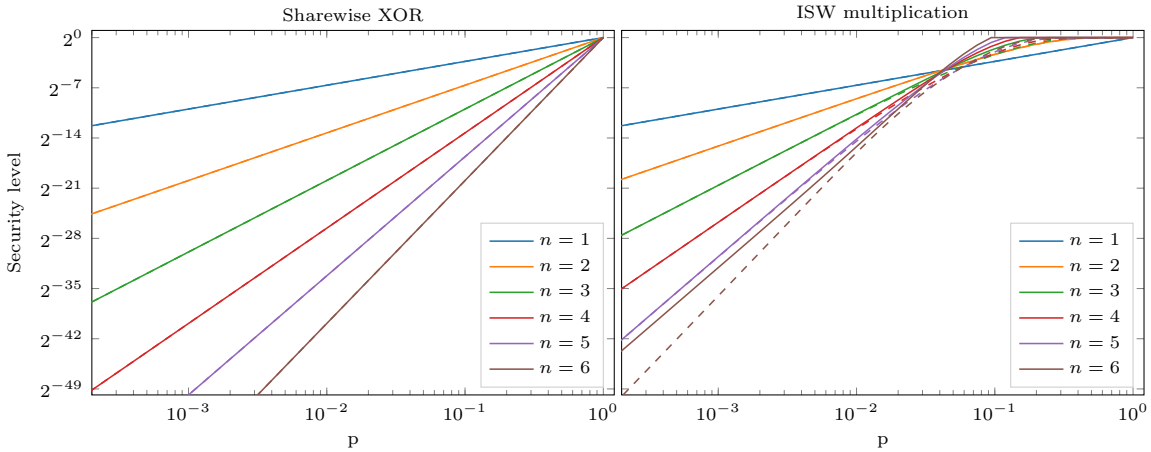


Fig. 1: Security of masked gadgets (with respect to the input sharing x, assuming the input sharing y is public). The continuous line is an upper bound, while the dashed line is the stat-only lower bound. $N_{max} = 10^7$, $N_t = 1000$.

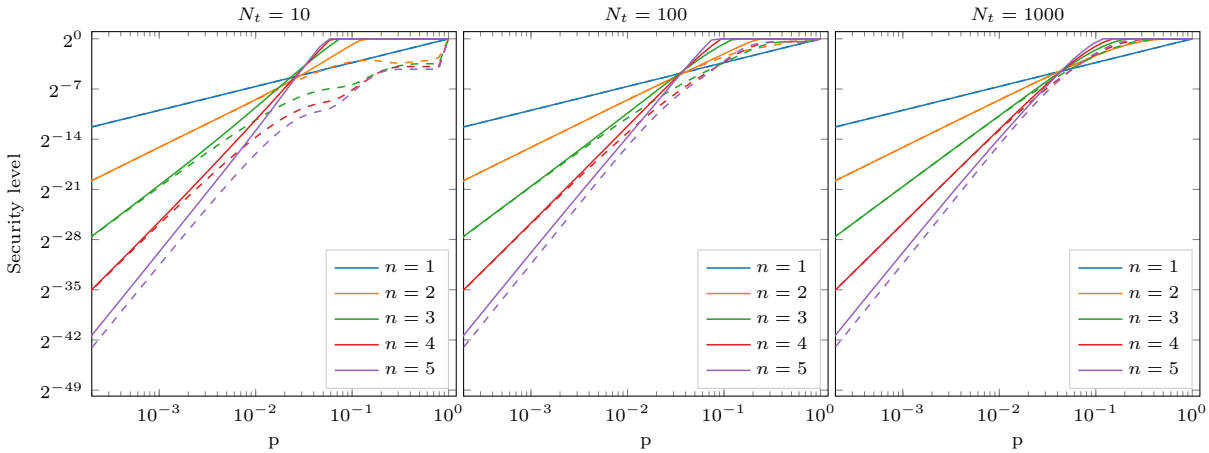


Fig. 2: Impact of the parameter N_t of Algorithm 1 on the security bounds of masked ISW multiplication gadgets (w.r.t. the input sharing x). $N_{max} = 10^7$.

We also look at the impact of N_{max} on Figure 3. We observe a gap between the bounds for too low N_{max} values, which gets worse as the number of shares increases. Indeed, when N_{max} is too small, we cannot do an enumeration of all the sets of $n-1$ probes, hence we cannot prove that the security order of the gadget is at least $n-1$, which means that the upper bound is asymptotically proportional to $p^{n'}$, with $n' < n-1$.

We finally observed that the computational cost is primarily dependent on N_{max} and the circuit size, while N_t has a lower impact (for the values considered). For instance, the execution time of the tool for the ISW multiplication with $n = 6$, $N_{max} = 10^8$ and $N_t = 100$ is about 33 h on a 24-core computer.

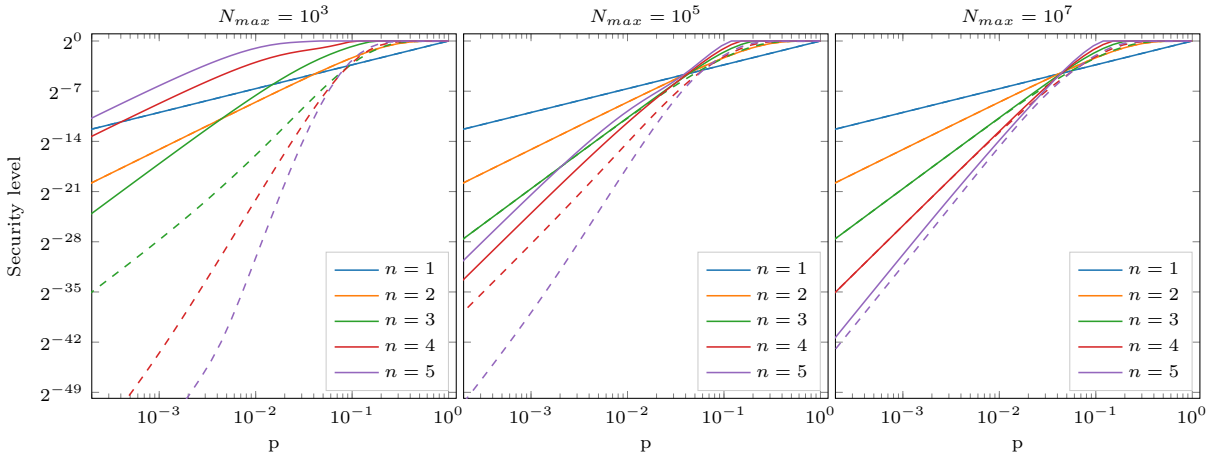


Fig. 3: Impact of the parameter N_{max} of Algorithm 1 on the security bounds of masked ISW multiplication gadgets (w.r.t. the input sharing x). $N_t = 1000$.

4 New composition results

In the previous section, it became clear that the tool is limited if it directly computes the security of complex circuits. This leads to the need to investigate composition properties. The existing definitions of random probing composability and random probing expandability in [8] are based on counting probes at the inputs and outputs of gadgets which are needed to simulate the leakage. We have recognized that ignoring the concrete random distribution over the needed input/output wires, and only counting the wires leads to a significant loss of tightness. Therefore we introduce our new security notion, the **PDT**. Before we define the **PDT** in Section 4.3 and present the composition results in Section 4.4, we recall the idea of simulatability in the leakage setting. Refining the dependency test of Section 3, we analyze the information a simulator needs to simulate a gadget's leakage in Section 4.2. In contrast to the previous section, we take into account the output gates, which is needed for composition. Further, we recall the definitions of parallel and sequential composition in Section 4.1, and present formal definitions adapted for our **PDT**s.

4.1 Definitions

Given two gadgets G_0 and G_1 with n shares, we define in this section the gadgets formed by their sequential composition written $\mathsf{G} = \mathsf{G}_1 \circ \mathsf{G}_0$ or their parallel composition written $\mathsf{G} = \mathsf{G}_1 || \mathsf{G}_0$.

We first introduce notations that allows us to keep track of input wires, output gates and internal wires in gadget compositions. We work with ordered finite sets. That is, given a finite set A (e.g., one of the sets $\mathcal{W}$, $\mathcal{I}$ or $\hat{\mathcal{O}}$ of a gadget G), we assign to each element of A a unique index in $[|A|] = \{0, 1, \ldots, |A|\}$. Then, given disjoint finite sets A and B, we denote by $C = A||_{(k)}B$ the union of A and B ordered such that a wire with index i in A has index i in C, and a wire with index i in B has index $k+i$ in B. The $||_{(\cdot)}$ operator is right-associative, which means that $A_2||_{(k_1)}A_1||_{(k_0)}A_0 = A_2||_{(k_1)}\left(A_1||_{(k_0)}A_0\right)$.

The sequential composition of gadgets allows implementing compositions of functions and is formally defined next.

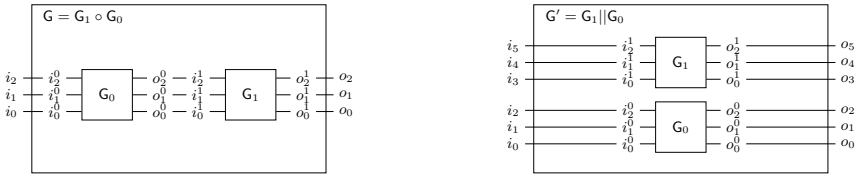
Definition 1 (Sequential composition). *Let* G_0 *and* G_1 *two gadgets with n shares, input wires* $\mathcal{I}_i$*, output gates* $\hat{\mathcal{O}}_i$*, and internal wires* $\mathcal{W}_i$*, respectively, such that* $|\mathcal{I}_1| = |\hat{\mathcal{O}}_0|$*. The sequential composition of* G_0 *and* G_1 *is the gadget* G *denoted as* $\mathsf{G}_1 \circ \mathsf{G}_0$ *whose set of input wires is* $\mathcal{I} = \mathcal{I}_0$ *and set of output gates is* $\hat{\mathcal{O}} = \hat{\mathcal{O}}_1$*. The set of internal wires of* G *is* $\mathcal{W} = \mathcal{W}_1||_{(k_1)}\mathcal{I}_1||_{(k_0)}\mathcal{W}_0$ *with* $k_1 = |\mathcal{W}_0| + |\mathcal{I}_1|$ *and* $k_0 = |\mathcal{W}_0|$*. The input wires of* G_1 *are connected to the output gates of* G_0 *such that for all i the input wire with index i is the output wire of the* i^{th} *output gate. If* G_0 *(resp.* G_1*) implements* f_0 *(resp.* f_1*), then* G *implements* $f_1 \circ f_0$*.*

The parallel composition of gadgets allows implementing a gadget for the function $f(x, y) = (f_0(x), f_1(y))$, using gadgets implementing f_0 and f_1.

Definition 2 (Parallel composition). *Let* G_0 *and* G_1 *two gadgets with n shares, input wires* $\mathcal{I}_i$*, output gates* $\hat{\mathcal{O}}_i$*, and internal wires* $\mathcal{W}_i$*, respectively. The parallel composition of* G_0 *and* G_1 *is the gadget* G *denoted as* $\mathsf{G}_1||\mathsf{G}_0$ *whose set of input wires is* $\mathcal{I} = \mathcal{I}_1||_{(|\mathcal{I}_0|)}\mathcal{I}_0$*, set of output gates is* $\hat{\mathcal{O}} = \hat{\mathcal{O}}_1||_{(|\hat{\mathcal{O}}_0|)}\hat{\mathcal{O}}_0$*, and set of internal wires is* $\mathcal{W} = \mathcal{W}_1||_{(|\mathcal{W}_0|)}\mathcal{W}_0$*.*

Figure 4 illustrates how to renumber the input wires and output gates in the case of gadgets with three inputs wires and three output gates. Figure 4a describes the sequential composition defined in Definition 1 and Figure 4b describes the parallel composition defined in Definition 2. For example, the input wire set of G' is $\mathcal{I} = \{i_5, i_4, \ldots, i_0\}$ which is the wire union $\mathcal{I} = \mathcal{I}_1||_{(|\mathcal{I}_0|)}\mathcal{I}_0$ of the input wires $\mathcal{I}_0 = \{i_2^0, i_1^0, i_0^0\}$ and $\mathcal{I}_1 = \{i_2^1, i_1^1, i_0^1\}$ of the gadgets G_0 and G_1.

We emphasize that both compositions are a basis for dividing a circuit into an arbitrary set of subcircuits. Therefore, if we have a masked gadget implementation of each gate type that appears in a circuit, we can build a masking compiler for that circuit: first decompose the circuit in sequential and parallel compositions down to subcircuits containing a single gate, then replace each gate with the corresponding masked gadget, and finally compose those gadgets according to the initial decomposition. As a case study, we depict a masked AES S-box implementation in Figure 6. The gadgets G_0-G_{10} are a parallel composition of the basis gadgets and $\mathsf{G}_{\text{S-box}}$ is a sequential composition of the gadgets G_0-G_{10}. The formal description of the S-box composition is given in Table 1.



(a) Sequential Composition (b) Parallel Composition

Fig. 4: Examples of sequential composition (4a) and parallel composition (4b).

4.2 Simulatability

So far, we described how to measure the amount of information leaked by a circuit by analyzing it directly. As observed in previous works, the complexity of such an approach rapidly turns out to be unrealistic. We now formalize simulatability-based definitions following the ideas outlined in [5], which are useful to analyze large circuits thanks to compositional reasoning.

Definition 3 (Simulatability). *A set of wires* $\mathcal{W}$ *in a gadget* G *is simulatable by a subset* $\mathcal{I}' \subset \mathcal{I}$ *of its inputs if there exists a probabilistic simulator function taking as input the values of the inputs* $\mathcal{I}'$*, and outputs a distribution of values on wires. Conditioned on the values of the wires in* $\mathcal{I}$ *the distribution output by the simulator is identical to the leakage from wires in* $\mathcal{W}$ *when the gadget is evaluated (conditioned on* $\mathcal{I}$*).*

The simulatability of small circuits, and particularly gadgets, is well studied and can be proven with tools such as maskVerif [4] and SILVER [25]. In this work we use the distribution of the smallest set of input wires such that there exists a simulator whose output has the same distribution as the leakage. More precisely, let $\mathcal{W}'$ be a subset of input and internal wires of a gadget G and $\mathcal{O}'$ an arbitrary subset of output wires, then we write $\mathcal{I}' = \mathcal{S}^{\mathsf{G}}(\mathcal{W}', \mathcal{O}')$ to define the smallest subset $\mathcal{I}'$ of input wires of G by which $(\mathcal{W}', \mathcal{O}')$ is perfectly simulatable.

Definition 4 (Simulatability set). *Let G be a gadget with input wire, internal wire and output gate sets $\mathcal{I}$, $\mathcal{W}$, and $\hat{\mathcal{O}}$. Further, let $\mathcal{O}$ be the set of output wires of $\hat{\mathcal{O}}$. The* simulatability set *of a subset $\mathcal{W}' \subseteq (\mathcal{W}, \mathcal{I})$ and $\mathcal{O}' \subseteq \mathcal{O}$, denoted $\mathcal{S}^{\mathsf{G}}(\mathcal{W}', \mathcal{O}')$, is the smallest subset of $\mathcal{I}$ by which $\mathcal{W}'$ and $\mathcal{O}'$ can be simulated.*

In the random probing model, $\mathcal{W}' = \mathcal{L}_p(\mathsf{G})$ is a random variable, hence the simulatability set $\mathcal{S}^{\mathsf{G}}(\mathcal{L}_p(\mathsf{G}), \mathcal{O}')$ is itself a random variable.

We now introduce rules for simulatability of parallel and sequential gadget compositions. Indeed, it is not enough to give a simulator for each gadget, but we also have to ensure that each individual simulator is consistent with the distribution generated by the other simulators, and that each simulator is provided with correct values for the input shares.

Claim 1 *For any parallel gadget composition $\mathsf{G} = \mathsf{G}_1 || \mathsf{G}_0$ with output gates $\hat{\mathcal{O}} = \hat{\mathcal{O}}_1 ||_{(|\hat{\mathcal{O}}_1|)} \hat{\mathcal{O}}_0$ an its output wires $\mathcal{O}$. It holds that*

$$\mathcal{S}^{\mathsf{G}}(\mathcal{L}_p(\mathsf{G}), \mathcal{O}') = \mathcal{S}^{\mathsf{G}_1}(\mathcal{L}_p(\mathsf{G}_1), \mathcal{O}'_1) \; ||_{(|\mathcal{I}_0|)} \; \mathcal{S}^{\mathsf{G}_0}(\mathcal{L}_p(\mathsf{G}_0), \mathcal{O}'_0)$$

for any subset of output wires $\mathcal{O}' = \mathcal{O}'_1 ||_{(|\mathcal{O}_0|)} \mathcal{O}'_0 \subseteq \mathcal{O}$.

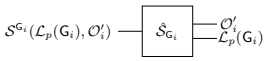
The proof is given in the extended version of the paper.

Claim 2 *For any sequential gadget composition $\mathsf{G} = \mathsf{G}_1 \circ \mathsf{G}_0$ with output gates $\hat{\mathcal{O}}$ and its output wires $\mathcal{O}$, it holds that*

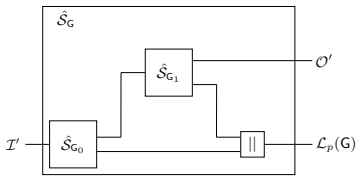
$$\mathcal{S}^{\mathsf{G}}(\mathcal{L}_p(\mathsf{G}), \mathcal{O}') \subseteq \mathcal{S}^{\mathsf{G}_0}\left(\mathcal{L}_p(\mathsf{G}_0), \mathcal{S}^{\mathsf{G}_1}(\mathcal{L}_p(\mathsf{G}_1), \mathcal{O}')\right)$$

for any subset of output wires $\mathcal{O}' \subseteq \mathcal{O}$.

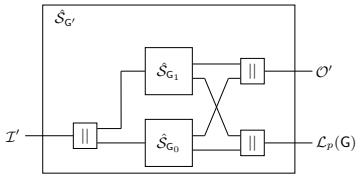
The proof is given in the extended version of the paper.



(a) Tight Simulator for Gadget G_i used in the proof of Claim 1 and 2



(b) Simulator for a serial gadget compositions.



(c) Simulator for a parallel gadget compositions.

Fig. 5: Simulators for the gadgets depicted in Figure 4 to prove Claims 1 and 2.

4.3 Probe distributions

In this section, we introduce our new security properties, the **PD** (Probe Distribution) and the **PDT** (Probe Distribution Table). Intuitively, given a set of wires $\mathcal{W}$ and a leakage process $\mathcal{L}$ (hence $\mathcal{L}(\mathcal{W}) \subseteq \mathcal{W}$), the **PD** of $\mathcal{L}(\mathcal{W})$ is a vector of size $2^{|\mathcal{W}|}$ that represents the statistical distribution of $\mathcal{L}(\mathcal{W})$. In more detail, for each subset $\mathcal{W}' \subseteq \mathcal{W}$, there is a corresponding element of the **PD** with value $\Pr[\mathcal{L}(\mathcal{W}) = \mathcal{W}']$. The **PDT** notion extends the idea in a way that makes it useful for analyzing gadget compositions: it links the set of output probes on the gadget to the distribution of the simulatability set of the gadget (i.e., to the inputs needed to simulate the leakage). More precisely, for a gadget G, the **PDT** is a matrix in $[0,1]^{|\mathcal{I}|\times|\mathcal{O}|}$, such that each column is associated to a subset of the outputs $\mathcal{O}' \subseteq \mathcal{O}$. Each column is a **PD** that represents the distribution of $\mathcal{S}^{\mathsf{G}}(\mathcal{L}(\mathsf{G}), \mathcal{O}')$ (viewed as a subset of the set of inputs $\mathcal{I}$). The two main results (Theorems 1 and 2) of the next section relate the **PDT** of a sequential (resp., parallel) gadget composition to the matrix (resp., tensor) product of the **PDT**s of the composing gadgets. We first formalize the mapping between subsets of wires and indices in vectors/matrices.

Definition 5 (Index representation of subsets of wires). *For any set of wires $\mathcal{W}$ of which each element has a unique index in $[|\mathcal{W}|]$, we associate to each subset $\mathcal{W}'$ of $\mathcal{W}$ the index*

$$\tilde{\mathcal{W}'} = \sum_{i\in[|\mathcal{W}|]} b_i 2^i \quad \textit{with} \quad \begin{cases} b_i = 1 & \textit{if element } i \textit{ of } \mathcal{W} \textit{ belongs to } \mathcal{W}', \\ b_i = 0 & \textit{otherwise.} \end{cases}$$

For example, the wire set $\mathcal{W} = \{\omega_0, \omega_1\}$ has 4 subsets $\mathcal{W}'$, that we represent with their index below:

$\mathcal{W}'$	$\emptyset$	$\{\omega_0\}$	$\{\omega_1\}$	$\{\omega_0, \omega_1\}$
$\tilde{\mathcal{W}'}$	0	1	2	3

Let use now give the formal definition of the **PD**.

Definition 6 (Probe Distribution PD). *Let $\mathcal{L}$ be a probabilistic process that outputs subsets of a set of wires $\mathcal{W}$. The probe distribution (**PD**) of $\mathcal{L}$ with respect to $\mathcal{W}$ is $\mathbf{p} \in [0,1]^{2^{|\mathcal{W}|}}$ such that for all $\mathcal{W}' \subset \mathcal{W}$, $\mathbf{p}_{\tilde{\mathcal{W}'}} = Pr[\mathcal{L}(\mathcal{W}) = \mathcal{W}']$.*

The **PD** of $\mathcal{L}_p(\mathcal{W})$ in the previous example is $\mathbf{p} = \big((1-p)^2, p(1-p), p(1-p), p^2\big)$.

We next give the definition of the **PDT**, which can be seen as the **PD**s of $\mathcal{S}^{\mathsf{G}}(\mathcal{L}_p(\mathsf{G}), \mathcal{O}')$ conditioned on the set of output probes $\mathcal{O}'$.

Definition 7 (Probe Distribution Table (PDT)). *Let G be a gadget with input wires $\mathcal{I}$ and output wires $\mathcal{O}$. For any $\mathcal{O}' \subseteq \mathcal{O}$, let $\mathbf{p}_{\tilde{\mathcal{O}'}}$ be the **PD** of $\mathcal{S}^{\mathsf{G}}(\mathcal{L}_p(\mathsf{G}), \mathcal{O}')$. The **PDT** of G ($\boldsymbol{PDT}_{\mathsf{G}}$) is a $[0,1]^{2^{|\mathcal{I}|}\times 2^{|\mathcal{O}|}}$ matrix with all the $\mathbf{p}_{\tilde{\mathcal{O}'}}$ as columns, that is*

$$\boldsymbol{PDT}_{\mathsf{G}} = \left(\mathbf{p}_j\right)_{j\in[2^{|\mathcal{O}|}]},$$

with $j = \tilde{\mathcal{O}'}$ for all subsets $\mathcal{O}' \subseteq \mathcal{O}$. The notation $\boldsymbol{PDT}_{\mathsf{G}}(\tilde{\mathcal{I}'}, \tilde{\mathcal{O}'})$ refers to the element of $\mathbf{p}_{\tilde{\mathcal{O}'}}$ associated to $\mathcal{I}'$.

$\mathbf{PDT}_{\mathsf{G}}(\tilde{\mathcal{I}'}, \tilde{\mathcal{O}'}) = \Pr\left[\mathcal{S}^{\mathsf{G}}(\mathcal{L}_p(\mathsf{G}), \mathcal{O}') = \mathcal{I}'\right]$. Furthermore, the **PDT** of a gadget is independent of its environment (i.e., of the **PD** of its output wires).

A first example of **PDT** is the one of the $\oplus$- and $\cdot$- gates (when viewed as gadgets with one share). In the first column, no output has to be simulated, and thus the only leakage comes from the two input wires. For the second column, knowledge of both inputs is needed to simulate the output. This gives:

$\mathbf{PDT}_{\oplus} = \mathbf{PDT}_{\cdot} =$

PDT	$\mathcal{O}' = \emptyset$	$\mathcal{O}' = \{0\}$
$\mathcal{I}' = \emptyset$	$(1-p^2)$	0
$\mathcal{I}' = \{0\}$	$p(1-p)$	0
$\mathcal{I}' = \{1\}$	$p(1-p)$	0
$\mathcal{I}' = \{0,1\}$	p^2	1

The second example is the simple refresh gadget G_r with two shares where a random value is added to two different wires. The random value leaks three times with probability p (one time in the  and two times in the). Thus the leakage probability of the random value is $q = 1 - (1-p)^3$, and we get:

$$\mathbf{PDT}_{\mathsf{G}_r} = \begin{array}{c|cccc} \mathbf{PDT} & \mathcal{O}' = \emptyset & \mathcal{O}' = \{0\} & \mathcal{O}' = \{1\} & \mathcal{O}' = \{1,0\} \\ \hline \mathcal{I}' = \emptyset & (1-p)^2 & (1-q)(1-p)^2 & (1-q)(1-p)^2 & 0 \\ \mathcal{I}' = \{0\} & p(1-p) & (q+qp)(1-p) & (1-q)p(1-p) & 0 \\ \mathcal{I}' = \{1\} & p(1-p) & (1-q)p(1-p) & (q+(1-q)p)(1-p) & 0 \\ \mathcal{I}' = \{0,1\} & p^2 & qp+(1-q)p^2 & qp+(1-q)p^2 & 1 \end{array}$$

The **PDT** is related to the security level in the random probing model.

Claim 3 (Security level from PDT) *Let G be a gadget and $\boldsymbol{PDT}_{\mathsf{G}}$ its Probe Distribution Table. Let s be the the security level of G with respect to an input sharing. If the set of shares of the considered input sharing is $\mathcal{I}'$, then*

$$\mathbf{e}^T \cdot \boldsymbol{PDT}_{\mathsf{G}} \cdot \mathbf{p}_\emptyset = \sum_{\mathcal{I}'' \supseteq \mathcal{I}'} \boldsymbol{PDT}_{\mathsf{G}}(\tilde{\mathcal{I}''}, 0) \geq s,$$

*where $\mathbf{p}_\emptyset = (1, 0, \ldots, 0)$ is the **PD** corresponding to no output leakage and $e_i = 1$ for all $i = \tilde{\mathcal{I}''}$ with $\mathcal{I}'' \supseteq \mathcal{I}'$, while $e_i = 0$ otherwise.*

Proof. Let $\mathcal{A}'$ be a set of wires that is an attack, that is, that depends on the considered unshared value which we denote Simulating $\mathcal{A}'$ therefore requires at least all the shares in $\mathcal{I}'$, hence

$$s \leq \Pr_{\mathcal{A}' \leftarrow \mathcal{L}_p(\mathsf{G})} \left[\mathcal{S}^{\mathsf{G}}(\mathcal{A}', \emptyset) \subseteq \mathcal{I}' \right].$$

Then, by definition of $\mathcal{L}_p(\mathsf{G})$ and of the **PDT**,

$$s \leq \Pr\left[\mathcal{S}^{\mathsf{G}}(\mathcal{L}_p(\mathsf{G}), \emptyset) \subseteq \mathcal{I}' \right] = \sum_{\mathcal{I}'' \supseteq \mathcal{I}'} \Pr\left[\mathcal{S}^{\mathsf{G}}(\mathcal{L}_p(\mathsf{G}), \emptyset) = \mathcal{I}'' \right] = \sum_{\mathcal{I}'' \supseteq \mathcal{I}'} \mathbf{PDT}_{\mathsf{G}}(\tilde{\mathcal{I}''}, 0).$$

This proves the inequality. The equality claim holds by construction of $\mathbf{e}$. □

We now give a few results that constitute the basis for the composition theorems of the next section. A first result links the **PD** of the input wires needed to simulate the leakage of the gadget and some of its outputs to the **PDT** of the gadget and the **PD** of its outputs. This claim is the foundation for the analysis of sequential gadget composition.

Claim 4 (PDT and PD) *Let G be a gadget with output wire set $\mathcal{O}$ and input wire set $\mathcal{I}$. If a probabilistic process $\mathcal{L}'(\mathcal{O})$ has a **PD** $\mathbf{p}$ with respect to $\mathcal{O}$, then $\boldsymbol{PDT}_{\mathsf{G}} \cdot \mathbf{p}$ is the **PD** of $\mathcal{S}^{\mathsf{G}}(\mathcal{L}_p(\mathsf{G}), \mathcal{L}'(\mathcal{O}))$ with respect to input wires $\mathcal{I}$.*

Proof. The solution can be directly derived from the definitions: Let $(v_i)_{i \in 2^{|\mathcal{I}|}} = \mathbf{PDT}_{\mathsf{G}} \cdot \mathbf{p}$. For any $\mathcal{I}' \subseteq \mathcal{I}$, it holds that

$$\begin{aligned} v_{\tilde{\mathcal{I}'}} &= \sum_{\mathcal{O}' \subseteq \mathcal{O}} \mathbf{PDT}_{\mathsf{G}}(\tilde{\mathcal{I}'}, \tilde{\mathcal{O}'}) \cdot p_{\tilde{\mathcal{O}'}} \\ &= \sum_{\mathcal{O}' \subseteq \mathcal{O}} \Pr\left[\mathcal{S}^{\mathsf{G}}(\mathcal{L}_p(\mathsf{G}), \mathcal{O}') = \mathcal{I}' \right] \cdot \Pr\left[\mathcal{L}'(\mathcal{O}) = \mathcal{O}' \right] \\ &= \sum_{\mathcal{O}' \subseteq \mathcal{O}} \Pr\left[\mathcal{S}^{\mathsf{G}}(\mathcal{L}_p(\mathsf{G}), \mathcal{O}') = \mathcal{I}', \mathcal{L}'(\mathcal{O}) = \mathcal{O}' \right] \\ &= \Pr\left[\mathcal{S}^{\mathsf{G}}(\mathcal{L}_p(\mathsf{G}), \mathcal{L}'(\mathcal{O})) = \mathcal{I}' \right]. \end{aligned}$$

The final equation gives the claim since it is exactly the i^{th} entry of the **PD** of $\mathcal{S}^{\mathsf{G}}(\mathcal{L}_p(\mathsf{G}), \mathcal{L}'(\mathcal{O}))$ with $i = \tilde{\mathcal{I}'}$. □

We next want to compare two probe distributions $\mathbf{p}$, $\mathbf{p}'$ to describe a partial order for distributions "$\dot{\leq}$". The high-level idea is that $\mathbf{p}$ is "larger" than $\mathbf{p}'$ (denoted $\mathbf{p}\dot{\geq}\mathbf{p}'$) if $\mathcal{L}$ gives more information than $\mathcal{L}'$. In other words, $\mathbf{p}$ is "larger" than $\mathbf{p}'$ if we can simulate $\mathcal{L}'(\mathcal{W})$ with $\mathcal{L}(\mathcal{W})$, where $\mathcal{L}$ (resp., $\mathcal{L}'$) is the probabilistic process associated to $\mathbf{p}$ (resp., $\mathbf{p}'$).

Definition 8 (Partial order for distributions). *For a set of wires $\mathcal{W}$, let $\mathcal{L}$ and $\mathcal{L}'$ be probabilistic processes with **PDs** $\mathbf{p}$ and $\mathbf{p}'$. We say that $\mathbf{p}$ is larger than $\mathbf{p}'$ and write $\mathbf{p}\dot{\geq}\mathbf{p}'$ iff the $\mathcal{L}'$ is simulatable by $\mathcal{L}$, that is, if there exists a probabilistic algorithm S that satisfies $S(\mathcal{X}) \subset \mathcal{X}$ such that the distribution of $\mathcal{L}'(\mathcal{W})$ and $S(\mathcal{L}(\mathcal{W}))$ are equal.*

On the one hand, it is clear that the definition is reflexive, antisymmetric, and transitive. Let $\mathbf{p}$, $\mathbf{p}'$, $\mathbf{p}''$ three **PD**s, it holds:

- $\mathbf{p}\dot{\geq}\mathbf{p}$, since we can always use the identity as simulator.
- If we know $\mathbf{p}\dot{\geq}\mathbf{p}'$ and $\mathbf{p}\dot{\leq}\mathbf{p}'$, both **PD**s describe processes with the same distribution, and we know $\mathbf{p} = \mathbf{p}'$.
- If it holds that $\mathbf{p}\dot{\geq}\mathbf{p}'$ and $\mathbf{p}'\dot{\geq}\mathbf{p}''$, it exists a simulator S' that simulates the process defined by $\mathbf{p}'$ with the process defined by $\mathbf{p}$, and a simulator S'' that does the same for $\mathbf{p}''$ and $\mathbf{p}'$. Hence, $S := S'(S''(\cdot))$ simulates the process defined by $\mathbf{p}''$ with the process of $\mathbf{p}$ and it follows $\mathbf{p}\dot{\geq}\mathbf{p}''$.

On the other hand, the order is only partial since it can happen that we have two probabilistic processes such that for both processes there exist no simulator to simulate the other.

The partial order for **PD**s is respected by linear combinations:

Claim 5 *Let $(\mathbf{p}_i)_{i\in[k]}$, $(\mathbf{p}'_i)_{i\in[k]}$ be **PDs** such that $\mathbf{p}_i\dot{\geq}\mathbf{p}'_i$ for all i. let $(\alpha_i)_{i\in[k]}$ be such that $0 \leq \alpha_i \leq 1$ for all i and $\sum_{i\in[k]} \alpha_i = 1$. If we denote $\mathbf{p} = \sum_{i\in[k]} \alpha_i\mathbf{p}_i$ and $\mathbf{p}' = \sum_{i\in[k]} \alpha_i\mathbf{p}'_i$, then $\mathbf{p}$ and $\mathbf{p}'$ are **PDs** and furthermore, $\mathbf{p}\dot{\geq}\mathbf{p}'$.*

Proof. Let $\mathcal{W}$ be a set of wires such that the random processes $(\mathcal{L}_i)_{i\in[k]}$ (resp. $(\mathcal{L}'_i)_{i\in[k]}$) have $(\mathbf{p}_i)_{i\in[k]}$ (resp. $(\mathbf{p}'_i)_{i\in[k]}$) as **PD**s. Further, let S^i be such that $S^i(\mathcal{L}_i(\mathcal{W}))$ has the same distribution as $\mathcal{L}'_i$. Let $\mathcal{L}$ be such that

$$\Pr\left[\mathcal{L}(\mathcal{W}) = \mathcal{W}'\right] = \sum_{i\in[k]} \alpha_i \Pr\left[\mathcal{L}_i(\mathcal{W}) = \mathcal{W}'\right],$$

and similarly for $\mathcal{L}'$. Firstly, $\mathcal{L}$ and $\mathcal{L}'$ are well-defined: the probabilities given above are non-negative and sum to 1. Next, the **PD** of $\mathcal{L}$ (resp. $\mathcal{L}'$) is $\mathbf{p}$ (resp. $\mathbf{p}'$). Finally, we build the simulator S. Let $\mathcal{L}''$ be a random process that, on input $\mathcal{W}$, selects randomly $i \in [k]$ (such that the probability of taking the value i is α_i), and outputs $S^i(\mathcal{L}_i(\mathcal{W}))$. Then, let S be a random process such that $\Pr[S(\mathcal{W}'') = \mathcal{W}'] = \Pr[\mathcal{L}'' = \mathcal{W}'|\mathcal{L} = \mathcal{W}'']$ for all $\mathcal{W}', \mathcal{W}'' \subseteq \mathcal{W}$. We observe that for all $\mathcal{W}' \subseteq \mathcal{W}$,

$$\begin{aligned}
\Pr[S(\mathcal{L}) = \mathcal{W}'] &= \sum_{\mathcal{W}''\subseteq\mathcal{W}} \Pr[S(\mathcal{W}'') = \mathcal{W}'] * \Pr[\mathcal{L} = \mathcal{W}''] \\
&= \sum_{\mathcal{W}''\subseteq\mathcal{W}} \Pr[\mathcal{L}'' = \mathcal{W}'|\mathcal{L} = \mathcal{W}''] * \Pr[\mathcal{L} = \mathcal{W}''] \\
&= \Pr[\mathcal{L}'' = \mathcal{W}'].
\end{aligned}$$

Since $\mathcal{L}''$ has the same distribution as $\mathcal{L}'$, this means that $\Pr[S(\mathcal{L}) = \mathcal{W}'] = \Pr[\mathcal{L}' = \mathcal{W}']$. □

The **PDT** has a partial structure. As described above each column i of the **PDT** is the **PD** of $\mathcal{S}^{\mathsf{G}}(\mathcal{L}_p(\mathsf{G}), \mathcal{O}')$ with $\tilde{\mathcal{O}}' = i$. Since we know that the input set required by a leakage simulator can only grow (or stay constant) if it has to simulate additional (output) leakage, we get:

Claim 6 *For any gadget with output wires $\mathcal{O}$, the columns $\mathbf{p}$. of the **PDT** have the following property: $\mathbf{p}_{\tilde{\mathcal{O}}'}\dot{\geq}\mathbf{p}_{\tilde{\mathcal{O}}''}$ for all $\mathcal{O}'' \subseteq \mathcal{O}' \subseteq \mathcal{O}$.*

Proof. It follows directly from claim 4. It holds that $\mathcal{S}^{\mathsf{G}}(\mathcal{L}_p(\mathsf{G}), \mathcal{O}'') \subseteq \mathcal{S}^{\mathsf{G}}(\mathcal{L}_p(\mathsf{G}), \mathcal{O}')$ and thus $\Pr\left[\mathcal{S}^{\mathsf{G}}(\mathcal{L}_p(\mathsf{G}), \mathcal{O}'') \subseteq \mathcal{S}^{\mathsf{G}}(\mathcal{L}_p(\mathsf{G}),\right.$ 1. The last equation is the claim $p_{\tilde{\mathcal{O}}'} \dot{\geq} p_{\tilde{\mathcal{O}}''}$. □

Finally, we want to extend the partial order of **PD**s to the whole **PDT**, with the same meaning: if $\mathbf{PDT}_{\mathsf{G}_0} \dot{\leq} \mathbf{PDT}_{\mathsf{G}_1}$, the amount of information leaked in G_0 is less than the information leaked in G_1:

Definition 9 (Partial order for PDT's). *Let* $\mathbf{A}, \mathbf{B} \in [0,1]^{2^{|\mathcal{I}|} \times 2^{|\mathcal{O}|}}$ *be two* ***PDT****s, we write*

$$\mathbf{A} \dot{\leq} \mathbf{B}$$

if for any ***PD*** $\mathbf{p} \in [0,1]^{2^{|\mathcal{O}|}}$ *it holds* $\mathbf{A} \cdot \mathbf{p} \dot{\leq} \mathbf{B} \cdot \mathbf{p}$.

As shown in Claim 4, $\mathbf{A} \cdot \mathbf{p}$ and $\mathbf{B} \cdot \mathbf{p}$ are **PD**s, therefore the partial order of **PDT**s is well defined.

Corollary 1 (PDT order is column-wise). *Let* ***PDT*** *and* ***PDT′*** *be* ***PDT****s, with columns* $(\mathbf{p}_i)_{i \in [|\mathcal{O}|]}$ *and* $(\mathbf{p}'_i)_{i \in [|\mathcal{O}|]}$ *respectively. Then,* $\mathbf{PDT} \dot{\geq} \mathbf{PDT}'$ *iff* $\mathbf{p}_i \dot{\geq} \mathbf{p}'_i$ *for all* $i \in [|\mathcal{O}|$.

Proof. If $\mathbf{PDT} \dot{\geq} \mathbf{PDT}'$, then for any $i \in [|\mathcal{O}|$, let $\mathbf{e}$ be such that $e_j = 1$ if $i = j$ and $e_j = 0$ otherwise. Since $\mathbf{e}$ is a **PD**, we have $\mathbf{p}_i = \mathbf{PDT} \cdot \mathbf{e} \dot{\geq} \mathbf{PDT}' \cdot \mathbf{e} = \mathbf{p}'_i$.

In the other way, let use assume that $\mathbf{p}_i \dot{\geq} \mathbf{p}'_i$, for all i. Then for any **PD** α (whose elements are denoted α_i), $\mathbf{PDT} \cdot \alpha$ is a linear combination of $\mathbf{p}_i$ with coefficients α_i, for which Claim 5 applies. Therefore $\mathbf{PDT} \cdot \alpha \dot{\geq} \mathbf{PDT}' \cdot \alpha$. □

Another useful property is that we can merge the order of **PD**s and **PDT**s:

Claim 7 *Let* $\mathbf{A}, \mathbf{B} \in [0,1]^{2^{|\mathcal{I}|} \times 2^{|\mathcal{O}|}}$ *be two* ***PDT****s, and* $\mathbf{p}, \mathbf{p}' \in [0,1]^{2^{|\mathcal{O}|}}$ *be two* ***PD****s. If* $\mathbf{A} \dot{\leq} \mathbf{B}$ *and* $\mathbf{p} \dot{\leq} \mathbf{p}'$*, then* $\mathbf{A} \cdot \mathbf{p} \dot{\leq} \mathbf{B} \cdot \mathbf{p}'$.

Proof. We prove the claim $\mathbf{A} \cdot \mathbf{p} \dot{\leq} \mathbf{B} \cdot \mathbf{p}'$ in two steps. First we show (i) $\mathbf{A} \cdot \mathbf{p} \dot{\leq} \mathbf{A} \cdot \mathbf{p}'$, and then we show (ii) $\mathbf{A} \cdot \mathbf{p}' \dot{\leq} \mathbf{B} \cdot \mathbf{p}'$.

(i) By Definition 8, there exists $\mathcal{W}$, $\mathcal{L}$ and $\mathcal{L}'$ associated to $\mathbf{p}$, $\mathbf{p}'$, respectively, with $\Pr[\mathcal{L}(\mathcal{W}) \subset \mathcal{L}'(\mathcal{W})] = 1$. Further, it holds $\Pr[\mathbf{A}_{\mathcal{L}(\mathcal{W})} \dot{\leq} \mathbf{A}_{\mathcal{L}'(\mathcal{W})}] = 1$ with Claim 6. Hence, $\mathbf{A} \cdot \mathbf{p} \dot{\leq} \mathbf{A} \cdot \mathbf{p}'$.

(ii) $\mathbf{A} \cdot \mathbf{p}' \dot{\leq} \mathbf{B} \cdot \mathbf{p}'$ follows from Definition 9 and $\mathbf{A} \dot{\leq} \mathbf{B}$. □

This leads to the preservation of **PDT** ordering through matrix product.

Corollary 2. *Let* $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, $\mathbf{D}$ *be* ***PDT****s. If* $\mathbf{A} \dot{\leq} \mathbf{B}$ *and* $\mathbf{C} \dot{\leq} \mathbf{D}$*, then* $\mathbf{A} \cdot \mathbf{C} \dot{\leq} \mathbf{B} \cdot \mathbf{D}$.

Proof. Let us denote by $\mathbf{X}_{*,i}$ the $(i+1)$-th column of a matrix $\mathbf{X}$. Then, for all $i \in [|\mathcal{O}|]$, $(\mathbf{A} \cdot \mathbf{C})_{*,i} = \mathbf{A} \cdot \mathbf{C}_{*,i}$ and $(\mathbf{B} \cdot \mathbf{D})_{*,i} = \mathbf{B} \cdot \mathbf{D}_{*,i}$. Hence, by Corollary 1, $\mathbf{A} \cdot \mathbf{C} \dot{\leq} \mathbf{B} \cdot \mathbf{D}$ iff $\mathbf{C}_{*,i} \dot{\leq} \mathbf{D}_{*,i}$ for all i. Using the same Corollary, we have $\mathbf{C}_{*,i} \dot{\leq} \mathbf{D}_{*,i}$. Finally, using Claim 7, we get $\mathbf{A} \cdot \mathbf{C}_{*,i} \dot{\leq} \mathbf{B} \cdot \mathbf{D}_{*,i}$ for all i. □

Finally, we relate the partial order for **PD**s and **PDT**s to the security level.

Claim 8 (Security level bound from PDT bound) *Let* s *be the security level of a gadget* G *with respect to a set of input shares* $\mathcal{I}'$*. Let* ***PDT*** *be the* ***PDT*** *of* G *and let* ***PDT′*** *be a* ***PDT****. If* $\mathbf{PDT}' \dot{\geq} \mathbf{PDT}$*, then* $\mathbf{e}^T \cdot \mathbf{PDT}' \cdot \mathbf{p}_\emptyset \geq s$*, where* $\mathbf{e}$ *is defined as in Claim 3.*

Proof. Using Claim 3, we know that $\mathbf{e}^T \cdot \mathbf{PDT} \cdot \mathbf{p}_\emptyset \geq s$. With Claim 7, we know that $\mathbf{PDT}' \cdot \mathbf{p}_\emptyset \dot{\geq} \mathbf{PDT} \cdot \mathbf{p}_\emptyset$. Let $\mathcal{L}$ (resp. $\mathcal{L}'$) be the random process associated to $\mathbf{PDT}' \cdot \mathbf{p}_\emptyset$ (resp. $\mathbf{PDT} \cdot \mathbf{p}_\emptyset$), and let S be the simulator that simulates $\mathcal{L}$ from $\mathcal{L}'$. We have $S(\mathcal{L}'(\mathcal{I})) \subseteq \mathcal{L}'(\mathcal{I})$, hence $\Pr[\mathcal{I}' \subseteq S(\mathcal{L}'(\mathcal{I}))] \leq \Pr[\mathcal{I}' \subseteq \mathcal{L}'(\mathcal{I})]$. Since S simulates $\mathcal{L}(\mathcal{I})$, $\Pr[\mathcal{I}' \subseteq S(\mathcal{L}'(\mathcal{I}))] = \Pr[\mathcal{I}' \subseteq \mathcal{L}(\mathcal{I})]$, which leads to $\mathbf{e}^T \cdot \mathbf{PDT} \cdot \mathbf{p}_\emptyset = \Pr[\mathcal{I}' \subseteq \mathcal{L}(\mathcal{I})] \leq \Pr[\mathcal{I}' \subseteq \mathcal{L}'(\mathcal{I})] = \mathbf{e}^T \cdot \mathbf{PDT}' \cdot \mathbf{p}_\emptyset$. □

4.4 Composition rules

In this section, we give the two main composition theorems for the **PDT** of parallel and sequential gadget compositions. Next, we show how the compositions theorems can be used to compute **PDT**s for larger composite gadgets and illustrate our results on the AES S-box example.

Theorem 1 (parallel composition). *Let* G_1 *and* G_2 *be two gadgets with* $\boldsymbol{PDT}_{\mathsf{G}_0}$ *and* $\boldsymbol{PDT}_{\mathsf{G}_1}$*. Further let* $\mathsf{G} = \mathsf{G}_1||\mathsf{G}_0$ *with* $\boldsymbol{PDT}_{\mathsf{G}}$*. It holds that*

$$\boldsymbol{PDT}_{\mathsf{G}} = \boldsymbol{PDT}_{\mathsf{G}_1} \otimes \boldsymbol{PDT}_{\mathsf{G}_0}.$$

Proof. Let $\mathcal{I}_0$, $\mathcal{I}_1$, $\mathcal{O}_0$, and $\mathcal{O}_1$ the input and output wires of G_0 and G_1, respectively. Hence, $\mathcal{I} = \mathcal{I}_1||_{(n)}\mathcal{I}_0$, $\mathcal{O} = \mathcal{O}_1||_{(m)}\mathcal{O}_0$ are the input and output wires of G with $n = |\mathcal{I}_0|$ and $m = |\mathcal{O}_0|$. From Definition 2 follows for any $\mathcal{I}' = \mathcal{I}'_1||_{(n)}\mathcal{I}'_0 \subseteq \mathcal{I}$ and $\mathcal{O}' = \mathcal{O}'_1||_{(m)}\mathcal{O}'_0 \subseteq \mathcal{O}$ that $\Pr\left[\mathcal{S}(\mathcal{L}_p(\mathsf{G}) \cup \mathcal{O}') = \mathcal{I}'\right]$ is the matrix entry $(\tilde{\mathcal{I}}', \tilde{\mathcal{O}}')$ of $\mathbf{PDT}_{\mathsf{G}}$. Considering Claim 1, we get

$$\begin{aligned}
\mathbf{PDT}_{\mathsf{G}}(\tilde{\mathcal{I}}', \tilde{\mathcal{O}}') &= \Pr\left[\mathcal{S}^{\mathsf{G}}(\mathcal{L}_p(\mathsf{G}), \mathcal{O}') = \mathcal{I}'\right] \\
&= \Pr\left[\mathcal{S}^{\mathsf{G}_1}\left(\mathcal{L}_p(\mathsf{G}_1) \cup \mathcal{O}'_1\right)||_{(n)}\mathcal{S}^{\mathsf{G}_0}\left(\mathcal{L}_p(\mathsf{G}_0), \mathcal{O}'_0\right) = \mathcal{I}'_1||_{(n)}\mathcal{I}'_0\right] \\
&= \Pr\left[\mathcal{S}^{\mathsf{G}_1}\left(\mathcal{L}_p(\mathsf{G}_0), \mathcal{O}'_0\right) = \mathcal{I}'_0, \mathcal{S}^{\mathsf{G}_0}\left(\mathcal{L}_p(\mathsf{G}_1), \mathcal{O}'_1\right) = \mathcal{I}'_1\right] \\
&= \Pr\left[\mathcal{S}^{\mathsf{G}_1}(\mathcal{L}_p(\mathsf{G}_0), \mathcal{O}'_0) = \mathcal{I}'_0\right] \cdot \Pr\left[\mathcal{S}^{\mathsf{G}_0}(\mathcal{L}_p(\mathsf{G}_1), \mathcal{O}'_1) = \mathcal{I}'_1\right] \\
&= \mathbf{PDT}_{\mathsf{G}_0}(\tilde{\mathcal{I}}'_0, \tilde{\mathcal{O}}'_0) \cdot \mathbf{PDT}_{\mathsf{G}_1}(\tilde{\mathcal{I}}'_1, \tilde{\mathcal{O}}'_1).
\end{aligned}$$

The last transformation of the formula uses the fact that the set of probes of both gadgets are independent, and the resulting term is exactly the matrix entry $(\tilde{\mathcal{I}}', \tilde{\mathcal{O}}')$ of $\mathbf{PDT}_{\mathsf{G}_1} \otimes \mathbf{PDT}_{\mathsf{G}_0}$. □

Remark. Theorem 1 can be generalized to any parallel composition of sub-circuits, even if those sub-circuits are not gadgets. For instance, a share-wise gadget with n shares is the parallel composition of n identical sub-circuits (a single addition gate for the addition gadget). The **PDT** of the addition gate $\mathbf{PDT}_{\oplus}$ is given in Section 4.3, therefore $\mathbf{PDT}_{\mathsf{G}_{\oplus,n}}$ can be computed as

$$\mathbf{PDT}_{\mathsf{G}_{\oplus,n}} = P\left(\bigotimes_{i=0}^{n-1} \mathbf{PDT}_{\oplus}\right),$$

where P reorders the index of the input wires from $(x_0^0, x_0^1, x_1^0, x_1^1, \ldots x_{n-1}^0, x_{n-1}^1)$ to $(x_0^0, \ldots, x_{n-1}^0, x_0^1, \ldots, x_{n-1}^1)$ where x_i^0 and x_i^1 are the first and second input wires of the i^{th} addition gate, respectively.

Theorem 2 (sequential composition). *Let* G_0 *and* G_1 *be two gadgets with* $\boldsymbol{PDT}_{\mathsf{G}_0}$*,* $\boldsymbol{PDT}_{\mathsf{G}_1}$*, and with* n_i *input wires and* m_i *output wires, respectively such that* $m_0 = n_1$*. Further let* $\mathsf{G} = \mathsf{G}_1 \circ \mathsf{G}_0$ *with* $\boldsymbol{PDT}_{\mathsf{G}}$*. It holds that*

$$\boldsymbol{PDT}_{\mathsf{G}} \dot{\leq} \boldsymbol{PDT}_{\mathsf{G}_0} \cdot \boldsymbol{PDT}_{\mathsf{G}_1}.$$

Proof. Let $\overline{\mathbf{PDT}} = \mathbf{PDT}_{\mathsf{G}_0} \cdot \mathbf{PDT}_{\mathsf{G}_1}$ and $\mathcal{I}_0$, $\mathcal{I}_1$, $\mathcal{O}_0$, $\mathcal{O}_1$ the input and output wire sets of G_0 and G_1, respectively. It also means that $\mathcal{I}_0$ and $\mathcal{O}_1$ are the input and output wire sets of G. Considering the fact that $\overline{\mathbf{PDT}}$ is the result of a matrix multiplication of $\mathbf{PDT}_{\mathsf{G}_0}$ and $\mathbf{PDT}_{\mathsf{G}_1}$, we get for any $\mathcal{I}' \subseteq \mathcal{I}_0$ and $\mathcal{O}' \subseteq \mathcal{O}_1$

$$\begin{aligned}
\overline{\mathbf{PDT}}(\tilde{\mathcal{I}}'', \tilde{\mathcal{O}}') &= \sum_{\mathcal{O}'' \subseteq \mathcal{O}_0} \Pr\left[\mathcal{S}^{\mathsf{G}_0}(\mathcal{L}_p(\mathsf{G}_0), \mathcal{O}'') = \mathcal{I}'\right] \cdot \Pr\left[\mathcal{S}^{\mathsf{G}_1}(\mathcal{L}_p(\mathsf{G}_1), \mathcal{O}') = \mathcal{O}''\right] \\
&= \sum_{\mathcal{O}'' \subseteq \mathcal{O}_0} \Pr\left[\mathcal{S}^{\mathsf{G}_0}(\mathcal{L}_p(\mathsf{G}_0), \mathcal{O}'') = \mathcal{I}', \mathcal{S}^{\mathsf{G}_1}(\mathcal{L}_p(\mathsf{G}_1), \mathcal{O}') = \mathcal{O}''\right] \\
&= \Pr\left[\mathcal{S}^{\mathsf{G}_0}\left(\mathcal{L}_p(\mathsf{G}_0), \mathcal{S}^{\mathsf{G}_1}(\mathcal{L}_p(\mathsf{G}_1), \mathcal{O}')\right) = \mathcal{I}'\right].
\end{aligned}$$

Further, $\mathbf{PDT}_{\mathsf{G}}(\tilde{\mathcal{I}}', \tilde{\mathcal{O}}') = \Pr\left[\mathcal{S}^{\mathsf{G}}(\mathcal{L}_p(\mathsf{G}), \mathcal{O}') = \mathcal{I}'\right]$, and thus for any $\mathcal{O}' \subseteq \mathcal{O}_1$ the columns $\mathbf{PDT}_{\mathsf{G}}(\tilde{\mathcal{O}}')$ and $\overline{\mathbf{PDT}}(\tilde{\mathcal{O}}')$ are the **PD**s of $\mathcal{S}^{\mathsf{G}}(\mathcal{L}_p(\mathsf{G}), \mathcal{O}')$ and of $\mathcal{S}^{\mathsf{G}_0}\left(\mathcal{L}_p(\mathsf{G}_0), \mathcal{S}^{\mathsf{G}_1}(\mathcal{L}_p(\mathsf{G}_1), \mathcal{O}')\right)$, respectively. Because of Claim 2, it holds that

$$\Pr\left[\mathcal{S}^{\mathsf{G}}(\mathcal{L}_p(\mathsf{G}), \mathcal{O}') \subseteq \mathcal{S}^{\mathsf{G}_0}\left(\mathcal{L}_p(\mathsf{G}_0), \mathcal{S}^{\mathsf{G}_1}(\mathcal{L}_p(\mathsf{G}_1), \mathcal{O}')\right)\right] = 1.$$

The last equation proves that it exists a simulator that simulates the simulatability set $\mathcal{S}^{\mathsf{G}}(\mathcal{L}_p(\mathsf{G}), \mathcal{O}')$ with $\mathcal{S}^{\mathsf{G}_0}\left(\mathcal{L}_p(\mathsf{G}_0), \mathcal{S}^{\mathsf{G}_1}(\mathcal{L}_p(\mathsf{G}_1), \mathcal{O}')\right)$. Hence, it holds that $\mathbf{PDT}_{\mathsf{G}}(\tilde{\mathcal{O}}') \dot{\leq} \overline{\mathbf{PDT}}(\tilde{\mathcal{O}}')$ for any column with $\mathcal{O}' \subseteq \mathcal{O}_1$. Since the inequality holds for any column, the inequality is independent from the distribution of the output wires $\mathcal{O}_1$. It follows that $\mathbf{PDT}_{\mathsf{G}}\mathbf{p} \dot{\leq} \mathbf{PDT}_{\mathsf{G}_0} \cdot \mathbf{PDT}_{\mathsf{G}_1}\mathbf{p}$ for all **PD**s $\mathbf{p}$. This results in the claim of the theorem $\mathbf{PDT}_{\mathsf{G}} \dot{\leq} \mathbf{PDT}_{\mathsf{G}_0} \cdot \mathbf{PDT}_{\mathsf{G}_1}$. □

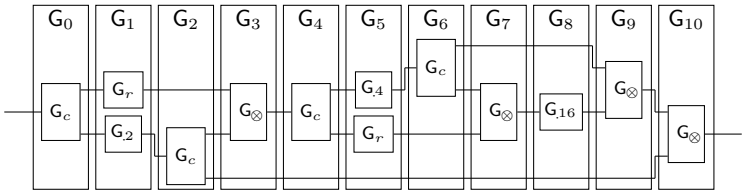


Fig. 6: AES S-box circuit (using the implementation from [31]) as a serial composition of gadgets. The symbols G_c, G_r, $\mathsf{G}_\otimes$ and $\mathsf{G}_{.x}$ are respectively copy, refresh and exponentiation to the power of x gadgets.

Corollary 3. *Let* $(\mathsf{G}_i)_{i\in[k]}$ *be gadgets that can be sequentially composed to form* $\mathsf{G} = \mathsf{G}_{k-1} \circ \cdots \circ \mathsf{G}_0$*. It holds that*

$$\boldsymbol{PDT}_{\mathsf{G}} \dot{\leq} \boldsymbol{PDT}_{\mathsf{G}_0} \cdot \ldots \cdot \boldsymbol{PDT}_{\mathsf{G}_{k-1}}.$$

Proof. This is a direct consequence of Theorem 2 and Corollary 2. □

The **PDT** of the AES S-box depicted in Figure 6 is bounded by $\mathbf{PDT}_{\text{S-box}}$ defined in Table 1. We compute the S-box with the gadgets $\mathsf{G}_{.2}$, $\mathsf{G}_\otimes$, G_r, and G_c. In addition, we also use a identity gadget G_{id}^l as a placeholder for composition results (this gadget does not leak and has as many inputs as outputs), whose **PDT** is the identity matrix. As described in Table 1, the gadgets G_0-G_{10} are a parallel composition of the gadgets $\mathsf{G}_{.2}$, $\mathsf{G}_{.4}$, $\mathsf{G}_{.16}$, $\mathsf{G}_\otimes$, G_r, G_c, and G_{id}^l (we can compute their **PDT**s using Theorem 1). Thus, $\mathsf{G}_{\text{S-box}}$ is a sequential composition of G_0-G_{10}. We can compute its **PDT** using Corollary 3, as shown in Table 1.

G_0	G_c	$\mathbf{PDT}_{\mathsf{G}_0} = \mathbf{PDT}_{\mathsf{G}_c}$
G_1	$\mathsf{G}_r \Vert \mathsf{G}_{.2}$	$\mathbf{PDT}_{\mathsf{G}_1} = \mathbf{PDT}_{\mathsf{G}_r} \otimes \mathbf{PDT}_{\mathsf{G}_{.2}}$
G_2	$\mathsf{G}_{id} \Vert \mathsf{G}_c$	$\mathbf{PDT}_{\mathsf{G}_2} = \mathbf{PDT}_{\mathsf{G}_{id}} \otimes \mathbf{PDT}_{\mathsf{G}_c}$
G_3	$\mathsf{G}_\otimes \Vert \mathsf{G}_{id}$	$\mathbf{PDT}_{\mathsf{G}_3} = \mathbf{PDT}_{\mathsf{G}_\otimes} \otimes \mathbf{PDT}_{\mathsf{G}_{id}}$
G_4	$\mathsf{G}_c \Vert \mathsf{G}_{id}$	$\mathbf{PDT}_{\mathsf{G}_4} = \mathbf{PDT}_{\mathsf{G}_c} \otimes \mathbf{PDT}_{\mathsf{G}_{id}}$
G_5	$\mathsf{G}_{.4} \Vert \mathsf{G}_r \Vert \mathsf{G}_{id}$	$\mathbf{PDT}_{\mathsf{G}_5} = \mathbf{PDT}_{\mathsf{G}_{.4}} \otimes \mathbf{PDT}_{\mathsf{G}_r} \otimes \mathbf{PDT}_{\mathsf{G}_{id}}$
G_6	$\mathsf{G}_c \Vert \mathsf{G}_{id} \Vert \mathsf{G}_{id}$	$\mathbf{PDT}_{\mathsf{G}_6} = \mathbf{PDT}_{\mathsf{G}_c} \otimes \mathbf{PDT}_{\mathsf{G}_{id}} \otimes \mathbf{PDT}_{\mathsf{G}_{id}}$
G_7	$\mathsf{G}_{id} \Vert \mathsf{G}_\otimes \Vert \mathsf{G}_{id}$	$\mathbf{PDT}_{\mathsf{G}_7} = \mathbf{PDT}_{\mathsf{G}_{id}} \otimes \mathbf{PDT}_{\mathsf{G}_\otimes} \otimes \mathbf{PDT}_{\mathsf{G}_{id}}$
G_8	$\mathsf{G}_{id} \Vert \mathsf{G}_{.16} \Vert \mathsf{G}_{id}$	$\mathbf{PDT}_{\mathsf{G}_8} = \mathbf{PDT}_{\mathsf{G}_{id}} \otimes \mathbf{PDT}_{\mathsf{G}_{.16}} \otimes \mathbf{PDT}_{\mathsf{G}_{id}}$
G_9	$\mathsf{G}_\otimes \Vert \mathsf{G}_{id}$	$\mathbf{PDT}_{\mathsf{G}_9} = \mathbf{PDT}_{\mathsf{G}_\otimes} \otimes \mathbf{PDT}_{\mathsf{G}_{id}}$
G_{10}	$\mathsf{G}_\otimes$	$\mathbf{PDT}_{\mathsf{G}_{10}} = \mathbf{PDT}_{\mathsf{G}_\otimes}$
$\mathsf{G}_{\text{S-box}}$	$\mathsf{G}_{10} \circ \mathsf{G}_9 \circ \ldots \circ \mathsf{G}_0$	$\mathbf{PDT}_{\text{S-box}} \dot{\leq} \mathbf{PDT}_{\mathsf{G}_0} \cdot \mathbf{PDT}_{\mathsf{G}_1} \cdot \ldots \cdot \mathbf{PDT}_{\mathsf{G}_{10}}$

Table 1: Composition of the AES S-box and its approximated **PDT**.

We conclude by noting that some well-known matrix product and tensor product distributive and associative properties mirror the properties of the gadget compositions (when the operations are well-defined):

$$(\mathbf{A} \cdot \mathbf{B}) \cdot \mathbf{C} = \mathbf{A} \cdot (\mathbf{B} \cdot \mathbf{C}) \qquad (\mathsf{G}_0 \circ \mathsf{G}_1) \circ \mathsf{G}_2 = \mathsf{G}_0 \circ (\mathsf{G}_1 \circ \mathsf{G}_2)$$
$$(\mathbf{A} \otimes \mathbf{B}) \otimes \mathbf{C} = \mathbf{A} \otimes (\mathbf{B} \otimes \mathbf{C}) \qquad (\mathsf{G}_0 || \mathsf{G}_1) \,||\mathsf{G}_2 = \mathsf{G}_0 || \,(\mathsf{G}_1 || \mathsf{G}_2)$$
$$(\mathbf{A} \cdot \mathbf{B}) \otimes (\mathbf{C} \cdot \mathbf{D}) = (\mathbf{A} \otimes \mathbf{C}) \cdot (\mathbf{B} \otimes \mathbf{D}) \qquad (\mathsf{G}_0 \circ \mathsf{G}_1) \,||\, (\mathsf{G}_2 \circ \mathsf{G}_3) = (\mathsf{G}_0 || \mathsf{G}_2) \circ (\mathsf{G}_1 || \mathsf{G}_3)$$

This means that our composition theorems give the same result independently of the way we decompose a composite gadget. This gives us freedom to choose, e.g., the most efficient way when we deal with relatively large computations.

5 Practical security of composite circuits

In this section, we adapt the method of Section 3 to compute bounds for **PDT**s. We then show how to turn those bounds into gadget security levels using the **PDT** properties and composition theorems. We finally describe the tool that implements our methodology and discuss its result for well-known gadgets.

5.1 Bounding PDTs

We first describe how to adapt the method of Section 3 to bound **PDT**s. That is, given a gadget G, we want to generate an upper bound $\mathbf{PDT}^U$ such that $\mathbf{PDT}^U \geq \mathbf{PDT}$ with probability at least $1 - \alpha$ (e.g., $1 - 10^{-6}$), and the $\geq$ operator defined for matrices and vectors as element-wise. We note that $\mathbf{PDT}^U$ is not a **PDT**: the sum of the elements in one of its columns may be ≥ 1.

There are two main differences with the bound of Section 3: (1) we have to handle all possible cases for the probes on the output shares of the gadgets (i.e., all the columns of the **PDT**), and (2) we care about the full distribution of the input probes, not only the probability of successful attack.

The upper bound $\mathbf{PDT}^U$ can be computed by grouping probe sets by size (similarly to Equation (3)):

$$\mathbf{PDT}^U(\tilde{\mathcal{I}}', \tilde{\mathcal{O}}') = \sum_{i=0}^{|\mathcal{W}|} p^i (1-p)^{|\mathcal{W}|-i} \cdot \left|\mathcal{W}^{(i)}\right| \cdot \mathbf{R}_i^U(\tilde{\mathcal{I}}', \tilde{\mathcal{O}}')$$

satisfies $\mathbf{PDT}^U(\tilde{\mathcal{I}}', \tilde{\mathcal{O}}') \geq \mathbf{PDT}(\tilde{\mathcal{I}}', \tilde{\mathcal{O}}')$ if

$$\mathbf{R}_i^U(\tilde{\mathcal{I}}', \tilde{\mathcal{O}}') \geq \frac{\left|\left\{\mathcal{W}' \subseteq \mathcal{W}^{(i)} \text{ s.t. } \mathcal{S}^{\mathsf{G}}(\mathcal{L}_p(\mathsf{G}), \mathcal{O}') = \mathcal{I}'\right\}\right|}{\left|\mathcal{W}^{(i)}\right|} \tag{6}$$

for all $i \in \{0, \ldots, |\mathcal{W}|\}$. Therefore, if Equation (6) is satisfied for each $(\mathcal{I}', \mathcal{O}', i)$ tuple with probability at least $1 - \alpha / \left((|\mathcal{W}| + 1)\, 2^{|\mathcal{I}| \cdot |\mathcal{O}|}\right)$, then $\mathbf{PDT}^U \geq \mathbf{PDT}$ with probability at least $1 - \alpha$ (by the union bound).

The computation of all the elements $P_i^U(\tilde{\mathcal{I}}', \tilde{\mathcal{O}}')$ can be performed identically to the computation of r_i^U in Section 3.1, except for changing the criterion for a Monte-Carlo sample $\mathcal{W}'$ to be counted as positive (i.e., be counted in s_i): $\mathcal{S}(\mathcal{W}', \mathcal{O}') = \mathcal{I}'$ (instead of $\delta_{\mathcal{W}'} = 1$). Furthermore, the algorithm can be optimized by running only one sampling for each $(i, \mathcal{O}')$ pair: we take $t_{i,\mathcal{O}'}$ samples, and we classify each sample $\mathcal{W}'$ according to $\mathcal{S}(\mathcal{W}', \mathcal{O}')$. This gives sample counts $s_{i,\mathcal{O}',\mathcal{I}'}$ for all $\mathcal{I}' \subseteq \mathcal{I}$, and from there we can use Equation (4).[4]

Finally, we use the hybrid strategy of Algorithm 1, with the aforementioned modifications.[5] The computation of a statistical-only lower bound $\mathbf{PDT}^L$ is done in the same way, except that Equation (5) is used instead of Equation (4).

[4] The random variables $s_{i,\mathcal{O}',\mathcal{I}'}$ for all $\mathcal{I}' \subseteq \mathcal{I}$ are not mutually independent, hence the derived bounds are not independent from each other, but this is not an issue since the union bound does not require independent variables.

[5] And additionally the change of the condition $s_i < N_t$ by $s_{i,\mathcal{O}'\mathcal{I}} < N_t$. The rationale for this condition is that, intuitively, if we have many "worst-case" samples, then we should have a sufficient knowledge of the distribution $\left(P_i(\tilde{\mathcal{I}}', \tilde{\mathcal{O}}')\right)_{\mathcal{I}' \subseteq \mathcal{I}}$.

5.2 From PDT bound to security level bound.

Let us take positive matrices $A^U \geq A$ and $B^U \geq B$. It always holds that $A^U \otimes B^U \geq A \otimes B$ and $A^U \cdot B^U \geq A \cdot B$. Therefore, if we use **PDT** bounds in composition Theorem 1 (resp., Corollary 3), we get as a result – denoted $\overline{\mathbf{PDT}}^U$ and computed as $A^U \cdot B^U$ (resp., $A^U \otimes B^U$) – a corresponding bound for the composite **PDT** – denoted $\overline{\mathbf{PDT}}$ and computed as $A \cdot B$ (resp., $A \otimes B$): $\overline{\mathbf{PDT}}^U \geq \overline{\mathbf{PDT}} \dot{\geq} \mathbf{PDT}$. Then, if we use $\overline{\mathbf{PDT}}^U$ in the formula for the computation of the security level (Claim 8) instead of $\overline{\mathbf{PDT}}$, we get

$$s^U = \mathbf{e}^T \cdot \overline{\mathbf{PDT}}^U \cdot \mathbf{p}_\emptyset \geq \mathbf{e}^T \cdot \overline{\mathbf{PDT}} \cdot \mathbf{p}_\emptyset \geq s.$$

We compute the statistical-only lower bound s^L in a similar manner. One should however keep in mind that $s^L \leq s$ does not hold in general, since Claim 8 and the sequential composition theorem only guarantee an upper bound (in addition to the non-tightness coming from the maskVerif algorithm). Again, the statistical-only lower bound is however useful for estimating the uncertainty on the security level that comes from the Monte-Carlo method: if there is a large gap between s^L and s^U, increasing the number of samples in the Monte-Carlo sampling can result in a better s^U (on the other hand, s^L gives a limit on how much we can hope to reduce s^U by increasing the number of samples).

5.3 Tool

We implemented the computation of the above bounds in the open-source tool STRAPS (Sampled Testing of the RAndom Probing Security). This tool contains a few additional algorithmic optimizations that do not change the results but significantly reduce the execution time (e.g., we exploit the fact that, in some circuits, many wires carry the same value, and we avoid to explicitly compute **PDT**s of large composite gadgets to reduce memory usage). Regarding performance, for the computation of the security of the AES S-box (see Figure 10), almost all of the execution time goes into computing the **PDT** of the ISW multiplication gadgets. Computing the **PDT**s of the other gadgets is much faster as they are smaller, and computing the composition takes a negligible amount of time (less than 1 %). The total running time for the AES S-box is less than 5 s for 1, 2 and 3 shares, 30 s for 4 shares, 3 min for 5 shares, and 33 h for 6 shares on a 24-core computer (dual 2.3 GHz Intel(R) Xeon(R) CPU E5-2670 v3).

STRAPS presents a few similarities with VRAPS [8]. While STRAPS mainly computes **PDT** bounds and VRAPS computes random probing expandability bounds, both metrics relate to the random probing security of a gadget, and both tools are based on the maskVerif dependency test algorithm. The main differences between these tools are twofold. First, STRAPS uses a mix of Monte-Carlo sampling and full exploration of the sets of probes, whereas VRAPS does only full exploration. Second, STRAPS computes and uses the simulatability set for a given set of internal and output probes, while VRAPS only stores whether the size of the simulatability set exceeds a given threshold. Thanks to this weaker requirement, VRAPS is able to exploit the set exploration algorithm of maskVerif, which accelerates the full exploration of the sets of probes by avoiding an exhaustive enumeration of all subsets [4].

5.4 Experiments & SOTA comparison

In this final section, we illustrate how to use our **PDT** bounding tool and the **PDT** composition theorems in order to bound the security of larger circuits, and to extract useful intuitions about the trade-off between the number of shares and level of noise required to reach a given security level. We also compare our results with previous works by Dziembowski et al. [20] and Belaïd et al. [8,9].

We begin by evaluating the impact of using composition theorems instead of a direct security evaluation. In Section 3.2, we concluded that directly analyzing the security of even a single multiplication gadget in the random probing model tightly is computationally intensive. On Figure 7, we show the security of a slightly more complex $\mathsf{ISW}(\mathsf{x}, \mathsf{SNI\text{-}Ref}(\mathsf{x}^2))$ gadget evaluated as either the composition of four gadgets (a split gadget, a squaring, an SNI refresh and an ISW multiplication), or as a single gadget (we call it integrated evaluation). We can see that when the gadget becomes large ($n = 5$) and for a similar computational complexity, the

results for the **PDT** composition are statistically tighter thanks to the lower size of its sub-gadgets. We also observe that, when upper and lower bounds converge, the security level computed from **PDT** composition is close to the one computed by the integrated evaluation, although the latter one is slightly better. We conclude that the **PDT** composition technique can provide useful results in practically relevant contexts where we build gadget compositions for which the integrated evaluation is not satisfying.

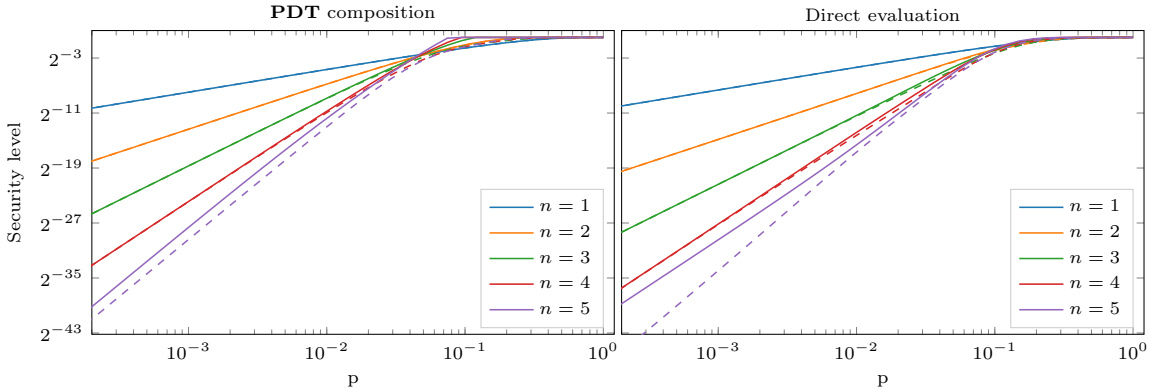


Fig. 7: Security of a cubing gadget $\mathsf{ISW}(\mathsf{x}, \mathsf{SNI\text{-}Ref}(\mathsf{x}^2))$. The left plot comes from **PDT** composition while the right plot is a direct security evaluation of the full circuit as a single gadget. The continuous line is an upper bound, while the dashed line is the stat-only lower bound. $N_{max} = 2 \times 10^6$, $N_t = 1000$.

Next, we investigate different refreshing strategies when computing the x^3 operation with an ISW multiplication gadget. Namely, we compare the situation with no refreshing which is known to be insecure in the threshold probing model [16], the simple refreshing with linear randomness complexity which does not offer strong composability guarantees, and an SNI refresh gadget from [12]. The results are illustrated in Figure 8. In the first case (with no refreshing), we observe the well-known division by two of the statistical security order (reflected by the slope of the security curves in the asymptotic region where the noise is sufficient and curves become linear): the security level is asymptotically proportional to $p^{\lceil (n-1)/2 \rceil}$. On the other side of the spectrum, the composition with an SNI refresh guarantees a statistical security order of $n - 1$. Finally, the most interesting case is the one of the simple refresh gadget, for which we observe a statistical security order reduction for $n \geq 3$, of which the impact may remain small for low noise levels. For instance, we can see that for $p \geq 2 \times 10^{-3}$, the curves for the simple and the SNI refresh gadgets are almost the same, with the security order reduction becoming more and more apparent only for lower values of p. So this analysis provides us with a formal quantitative understanding of a gadget's security level which, for example, suggests that depending on the noise levels, using SNI gadgets may not always be needed.

We extend this analysis of a simple gadget to the case of a complete AES S-box in Figure 9. All the previous observations remain valid in this case as well. Furthermore, this figure confirms that our results get close to the ones reported for concrete worst-case attacks in [18]. Namely, already for the (low) number of shares and (practical) levels of noise we consider, we observe a statistical security order of $n - 1$ for a practically relevant (AES S-box) circuit.[6]

Eventually, we compare our bounds with state-of-the-art results for the non-linear part of the AES S-box in Figure 10, in order to highlight that such tight results were not available with existing solutions. Precisely, we compare our results with the works that provide the best bounds in the low-noise region that we consider: the Simple Refreshing (SR) strategy of Dziembowski et al. [20], and the first (RPE1) [8] and

[6] To make the results more easily comparable, one can just assume connect the leakage probability with the mutual information of [18] by just assuming that the mutual information per bit (i.e., when the unit is the field element) equals p.

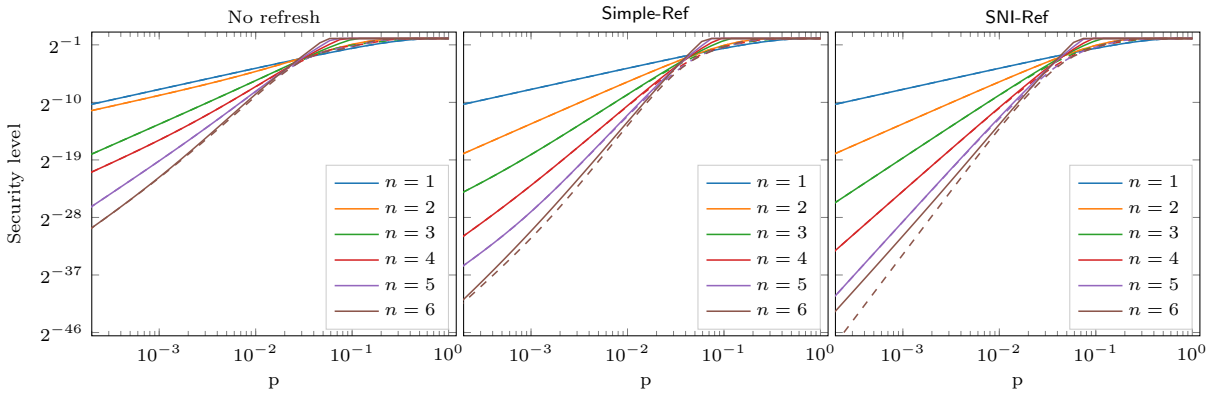


Fig. 8: Security of the cubing $\mathsf{ISW}(\mathsf{x}, \mathsf{Ref}(\mathsf{x}^2))$, where Ref is identity (no refreshing), Simple-Ref, or SNI-Ref gadget. The continuous line is an upper bound, while the dashed line is the stat-only lower bound. $N_{max} = 10^8$, $N_t = 100$.

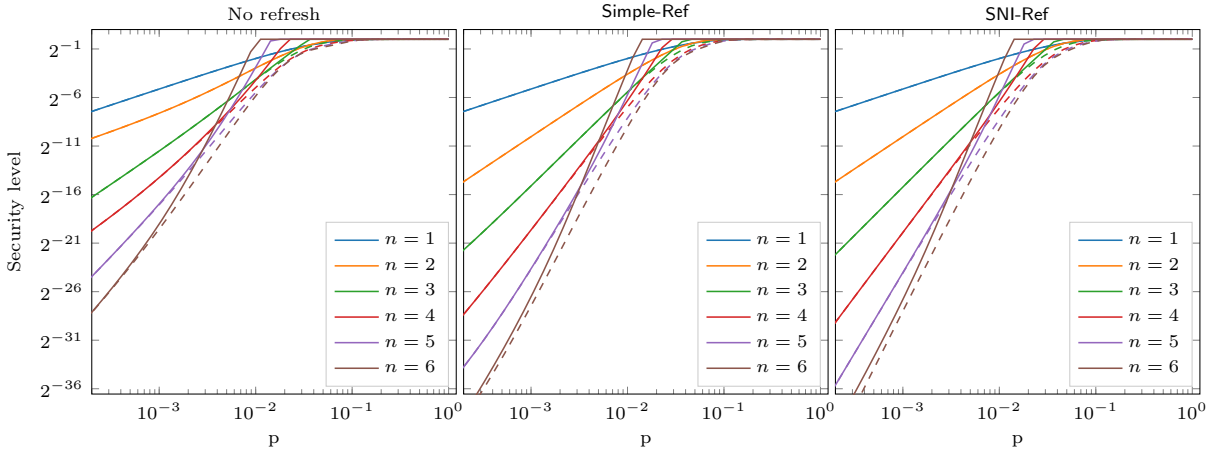


Fig. 9: Security of the non-linear part of an AES S-box in $\mathbb{F}_{256}$, where Ref is either an identity (no refreshing), the Simple-Ref gadget, or the SNI-Ref gadget. The continuous line is an upper bound, while the dashed line is the stat-only lower bound. $N_{max} = 10^8$, $N_t = 100$.

second (RPE2) [9] sets of gadgets from the Random Probing Expansion strategy of Belaïd et al. We see that amongst the previous works we consider here, RPE2 with 27 shares achieves the best maximum tolerated leakage probability and statistical security order. Our **PDT**-based analysis of the SNI-refreshed AES S-box with the ISW multiplication achieves a similar security level with only 6 shares. In this last experiment, the number of shares n is an indicator for the circuit size since all schemes have a circuit size in $\mathcal{O}(n^2)$. So we conclude that our results enable a significant improvement of the provable security claims of practical masked circuits in the random probing model.

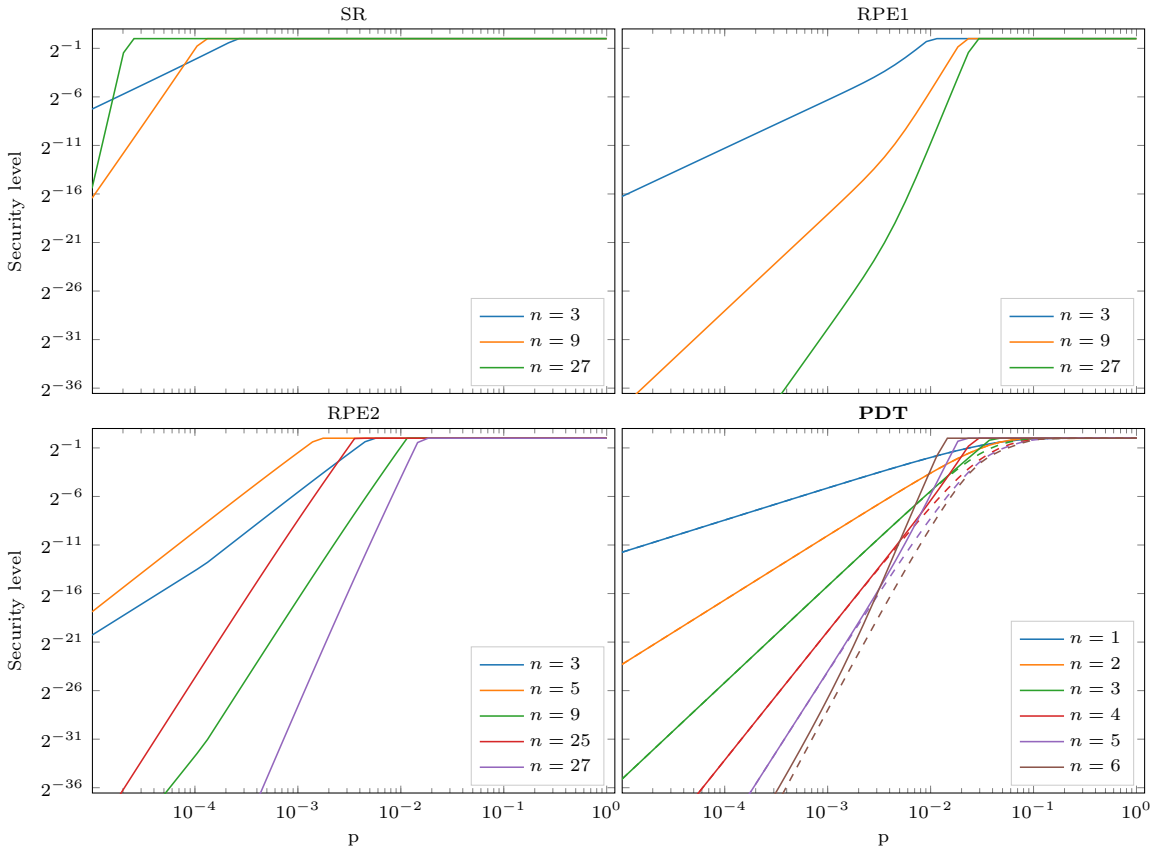


Fig. 10: Security of the non-linear part of an AES S-box in $\mathbb{F}_{256}$, based on the best result of each paper. For the **PDT**, we take use a SNI refresh gadget. All the circuits have a size $\mathcal{O}(n^2)$.

Acknowledgments. Gaëtan Cassiers and François-Xavier Standaert are resp. Research Fellow and and Senior Associate Researcher of the Belgian Fund for Scientific Research (FNRS-F.R.S.). Sebastian Faust and Maximilan Orlt are founded by the Emmy Noether Program FA 1320/1-1 of the German Research Foundation (DFG). This work has been funded in part by the ERC project 724725 and by Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - SFB 1119 - 236615297 (Project S7).

References

1. M. Ajtai. Secure computation with information leaking to an adversary. In *STOC*, pages 715–724. ACM, 2011.

2. P. Ananth, Y. Ishai, and A. Sahai. Private circuits: A modular approach. In *CRYPTO (3)*, volume 10993 of *Lecture Notes in Computer Science*, pages 427–455. Springer, 2018.
3. M. Andrychowicz, S. Dziembowski, and S. Faust. Circuit compilers with o(1/\log (n)) leakage rate. In *EUROCRYPT (2)*, volume 9666 of *Lecture Notes in Computer Science*, pages 586–615. Springer, 2016.
4. G. Barthe, S. Belaïd, G. Cassiers, P. Fouque, B. Grégoire, and F. Standaert. maskverif: Automated verification of higher-order masking in presence of physical defaults. In *ESORICS (1)*, volume 11735 of *Lecture Notes in Computer Science*, pages 300–318. Springer, 2019.
5. G. Barthe, S. Belaïd, F. Dupressoir, P. Fouque, B. Grégoire, P. Strub, and R. Zucchini. Strong non-interference and type-directed higher-order masking. In *CCS*, pages 116–129. ACM, 2016.
6. G. Barthe, M. Gourjon, B. Grégoire, M. Orlt, C. Paglialonga, and L. Porth. Masking in fine-grained leakage models: Construction, implementation and verification. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(2):189–228, 2021.
7. A. Battistello, J. Coron, E. Prouff, and R. Zeitoun. Horizontal side-channel attacks and countermeasures on the ISW masking scheme. In *CHES*, volume 9813 of *Lecture Notes in Computer Science*, pages 23–39. Springer, 2016.
8. S. Belaïd, J. Coron, E. Prouff, M. Rivain, and A. R. Taleb. Random probing security: Verification, composition, expansion and new constructions. In *CRYPTO (1)*, volume 12170 of *Lecture Notes in Computer Science*, pages 339–368. Springer, 2020.
9. S. Belaïd, M. Rivain, and A. R. Taleb. On the power of expansion: More efficient constructions in the random probing model. *IACR Cryptol. ePrint Arch.*, 2021:434, 2021.
10. R. Bloem, H. Groß, R. Iusupov, B. Könighofer, S. Mangard, and J. Winter. Formal verification of masked hardware implementations in the presence of glitches. In *EUROCRYPT (2)*, volume 10821 of *Lecture Notes in Computer Science*, pages 321–353. Springer, 2018.
11. E. Brier, C. Clavier, and F. Olivier. Correlation power analysis with a leakage model. In *CHES*, volume 3156 of *Lecture Notes in Computer Science*, pages 16–29. Springer, 2004.
12. G. Cassiers, B. Gregoire, I. Levi, and F. X. Standaert. Hardware private circuits: From trivial composition to full verification. *IEEE Transactions on Computers*, pages 1–1, 2020.
13. G. Cassiers and F. Standaert. Towards globally optimized masking: From low randomness to low noise rate or probe isolating multiplications with reduced randomness and security against horizontal attacks. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2019(2):162–198, 2019.
14. S. Chari, C. S. Jutla, J. R. Rao, and P. Rohatgi. Towards sound approaches to counteract power-analysis attacks. In *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 398–412. Springer, 1999.
15. S. Chari, J. R. Rao, and P. Rohatgi. Template attacks. In *CHES*, volume 2523 of *Lecture Notes in Computer Science*, pages 13–28. Springer, 2002.
16. J. Coron, E. Prouff, M. Rivain, and T. Roche. Higher-order side channel security and mask refreshing. In *FSE*, volume 8424 of *Lecture Notes in Computer Science*, pages 410–424. Springer, 2013.
17. A. Duc, S. Dziembowski, and S. Faust. Unifying leakage models: From probing attacks to noisy leakage. *J. Cryptol.*, 32(1):151–177, 2019.
18. A. Duc, S. Faust, and F. Standaert. Making masking security proofs concrete (or how to evaluate the security of any leaking device), extended version. *J. Cryptol.*, 32(4):1263–1297, 2019.
19. S. Dziembowski, S. Faust, and M. Skorski. Noisy leakage revisited. In *EUROCRYPT (2)*, volume 9057 of *Lecture Notes in Computer Science*, pages 159–188. Springer, 2015.
20. S. Dziembowski, S. Faust, and K. Zebrowski. Simple refreshing in the noisy leakage model. In *ASIACRYPT (3)*, volume 11923 of *Lecture Notes in Computer Science*, pages 315–344. Springer, 2019.
21. S. Faust, V. Grosso, S. M. D. Pozo, C. Paglialonga, and F. Standaert. Composable masking schemes in the presence of physical defaults & the robust probing model. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(3):89–120, 2018.
22. K. Gandolfi, C. Mourtel, and F. Olivier. Electromagnetic analysis: Concrete results. In *CHES*, volume 2162 of *Lecture Notes in Computer Science*, pages 251–261. Springer, 2001.
23. D. Goudarzi, A. Joux, and M. Rivain. How to securely compute with noisy leakage in quasilinear complexity. In *ASIACRYPT (2)*, volume 11273 of *Lecture Notes in Computer Science*, pages 547–574. Springer, 2018.
24. Y. Ishai, A. Sahai, and D. A. Wagner. Private circuits: Securing hardware against probing attacks. In *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481. Springer, 2003.
25. D. Knichel, P. Sasdrich, and A. Moradi. SILVER - statistical independence and leakage verification. In *ASIACRYPT (1)*, volume 12491 of *Lecture Notes in Computer Science*, pages 787–816. Springer, 2020.
26. P. C. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.

27. S. Mangard, T. Popp, and B. M. Gammel. Side-channel leakage of masked CMOS gates. In *CT-RSA*, volume 3376 of *Lecture Notes in Computer Science*, pages 351–365. Springer, 2005.
28. S. Nikova, V. Rijmen, and M. Schläffer. Secure hardware implementation of nonlinear functions in the presence of glitches. *J. Cryptol.*, 24(2):292–321, 2011.
29. T. Prest, D. Goudarzi, A. Martinelli, and A. Passelègue. Unifying leakage models on a rényi day. In *CRYPTO (1)*, volume 11692 of *Lecture Notes in Computer Science*, pages 683–712. Springer, 2019.
30. E. Prouff and M. Rivain. Masking against side-channel attacks: A formal security proof. In *EUROCRYPT*, volume 7881 of *Lecture Notes in Computer Science*, pages 142–159. Springer, 2013.
31. M. Rivain and E. Prouff. Provably secure higher-order masking of AES. In *CHES*, volume 6225 of *Lecture Notes in Computer Science*, pages 413–427. Springer, 2010.
32. T. Schneider and A. Moradi. Leakage assessment methodology - extended version. *J. Cryptogr. Eng.*, 6(2):85–99, 2016.
33. F. Scholz. Confidence bounds & intervals for parameters relating to the binomial, negative binomial, poisson and hypergeometric distributions with applications to rare events, 2008.

B. Provable Secure Parallel Gadgets

This chapter corresponds to our published article at TCHES 2022 [39], with major edits: At CHES 2023 [39], we argued that we achieve security for leakage probability $p = O(1/\sqrt{n})$. It turned out that this is not the case for arbitrary circuits. For this reason, we updated our paper on the IACR Cryptology ePrint Archive [38] with a more fine grained analysis that proves our result for $p = O(1/n^2)$ in general, and $p = O(1)$ for affine circuits. In the following, we highlight the changes in red.

Provable Secure Parallel Gadgets

Francesco Berti[1], Sebastian Faust[2] and Maximilian Orlt[2]

[1] Bar-Ilan University, Ramat-Gan 529002, Israel francesco.berti@biu.ac.il
[2] Department of Computer Science, TU Darmstadt, Darmstadt, Germany
{sebastian.faust,maximilian.orlt}@tu-darmstadt.de

Abstract. Side-channel attacks are a fundamental threat to the security of cryptographic implementations. One of the most prominent countermeasures against side-channel attacks is masking, where each intermediate value of the computation is secret shared, thereby concealing the computation's sensitive information. An important security model to study the security of masking schemes is the *random probing model*, in which the adversary obtains each intermediate value of the computation with some probability p. To construct secure masking schemes, an important building block is the *refreshing gadget*, which updates the randomness of the secret shared intermediate values. Recently, Dziembowski, Faust, and Zebrowski (ASIACRYPT'19) analyzed the security of a simple refreshing gadget by using a new technique called the *leakage diagram*. In this work, we follow the approach of Dziembowski et al. and significantly improve its methodology. Concretely, we refine the notion of a leakage diagram via so-called *dependency graphs*, and show how to use this technique for arbitrary complex circuits via composition results and approximation techniques. To illustrate the power of our new techniques, as a case study, we designed provably secure parallel gadgets for the random probing model, and adapted the ISW multiplication such that all gadgets can be parallelized. Finally, we evaluate concrete security levels, and show how our new methodology can further improve the concrete security level of masking schemes. This results in a compiler provable secure up to a noise level of $O(1)$ for affine circuits and $O(1/n^2)$ in general.

Keywords: Random Probing Model · Masking · Composability · Leakage Diagram

1 Introduction

Context. Proving the security of cryptographic schemes is the de-facto standard of modern cryptography. The most widely used security model is the black-box model, where the adversary has access to the inputs and outputs but has no knowledge or control over the algorithm's inner workings. It is well known, however, that real-world implementations may reveal information about the inner workings, and in particular about the secret key of a cryptographic scheme. Multiple side-channel attacks exploit physical phenomena such as power consumption [23], cache accesses [10], acoustic signals [18], or timing [22].

Masking schemes. A popular countermeasure against power analysis attacks is *masking*. At a high-level, the idea is to conceal sensitive intermediate values through secret sharing. A masking scheme relies on an *encoding function* that takes as input a value on a wire x and shares it over multiple wires that carry the shares $x_0, \ldots, x_{n-1}$. The encoding function we consider in this work samples $x_0, \ldots, x_{n-1}$ uniformly at random such that $x = \sum_i x_i$, where n is called the order of the masking scheme. If $x \in \mathbb{F}_2$, then such masking schemes are called Boolean masking. The main challenge in designing secure masking schemes is to develop operations – often called *gadgets* – that securely compute on shared values. Security here means that even if the adversary learns information on the internals of the gadget, such information does not reveal sensitive information. In addition, we need a method to compose gadgets without violating security. This is often done via the *refreshing algorithm*, which takes as input a sharing $x_0, \ldots, x_{n-1}$ that encodes x, and outputs a fresh sharing $x'_0, \ldots, x'_{n-1}$ of the same secret value. Here, for security, we have to guarantee that even given leakage from the refreshing procedure, the output $x'_0, \ldots, x'_{n-1}$ is a fresh encoding of x.

Security analysis of masking countermeasure. As the design of the masking scheme is complex, we analyze their security using security proofs. To this end, we require a *leakage model* to describe the leakage emitting from a masked device formally. The most widely used leakage model is the *t-threshold probing model* originally introduced in the seminal work of Ishai, Sahai, and Wagner (ISW) [21]. In this model, cryptographic computation

is described as a Boolean (sometimes arithmetic) circuit, where the adversary is allowed to probe up to t wires and learn the values carried on these wires during the computation.

Although a security analysis in the threshold probing model provides the first evidence of the soundness of a masking scheme, it does not accurately model the quantitative nature of leakage, thereby excluding important types of attacks [31, 12]. To address this problem, Prouff and Rivain introduced the *noisy leakage model* [27]. In this model, the adversary obtains a noisy version of each wire, where the noise is sampled from a certain distribution (e.g., the Gaussian distribution). Noisy leakages accurately model physical leakage from power consumption and, in particular, allow for quantitative statements about the noise required to conceal sensitive information – crucial information for cryptographic engineers. In detail, they define the noise as a set of probabilistic leakage functions that are restricted by an upper bound using the Euclidean Norm (or statistical distance) as a metric. An important shortcoming of the noisy leakage model, however, is that it is very hard to work with. Concretely, in comparison to the threshold probing model, security proofs are highly cumbersome, and proving the security of natural constructions often requires to rely on unrealistic assumptions (e.g., the use of leak-free gates). To resolve these problems, somewhat surprisingly, Duc et al. [14] showed that noisy leakages and the seemingly much weaker threshold probing model of ISW are related. For their proof, they considered an intermediate model – the *p-random probing model* – and showed that security in this model directly implies security against noisy leakages. The p-random probing model considers a particular noise distribution, where each wire leaks with probability p, while the adversary obtains no knowledge of the wire's value with probability $1 - p$. The security in the random probing model only implies security in the noisy model with a loss of the field size. There are two approaches to avoiding the security loss. The first approach was presented by Dziembowski et al.[15] who proposed the *average random probing model*, a modified version of the random probing model. However, security proofs in this model are still rather complex for two reasons. First, they assume a more powerful class of leakage functions because the adversary can choose leakage functions where only the average leakage probability is p. In other words, for any possible input value, the leakage probability p can be different (up to p times the field size). Second, the adversary also learns the internal randomness used by the leakage function to decide whether a value leaks or not. Hence, the adversary even learns something about the values when the leakage function does not output the value. An alternative approach is given by Goudarzi et al. [26]. They eliminate the field size by using an alternative metric for the noisy model. In other words, they do not modify the probing model where the actual proof is done, but the noisy model that should model the natural leakage. In detail, they use a worst-case metric called (Average) Related Error and show that security in the random probing model tightly implies security in the modified noisy model. Since the result of Goudarzi et al. and Duc et al., security in the random probing model has been studied intensively by the research community [2, 4, 7, 11, 16]. There are two important goals in this research area. First, we aim to design masking schemes that obtain security for values of p independent of the order n of the masking scheme and are, in particular, close to 1. This is important as it implies that the masked computation remains secure in the presence of larger amounts of leakage. Second, the masking schemes that we design need to be efficient, where efficiency is typically measured in terms of circuit and randomness complexity. In particular, since all of our gadgets have low depth, the latency of the compiled circuits is significantly improved. Our main contribution is to improve on both of these goals for certain classes of masked computation.

1.1 Contribution

Improved analysis of refreshing gadgets. As discussed above, the main ingredient of any masked computation is a secure refreshing gadget. It is typically placed throughout the masked computation to ensure composition. In addition, refreshing gadgets also have applications for key refreshing, e.g., as part of a masked AES, where the secret key has to be refreshed periodically to ensure security. There is a large body of literature on designing secure refreshing schemes. For our work, the most important is the work of Dziembowski et al. [16], who gave a security analysis of a very simple and efficient refreshing. Their scheme essentially uses only n randomness and n operations, which is optimal for a refreshing gadget of order n masking. Dziembowski et al. show that this simple refreshing gadget surprisingly is $O(\sqrt{p}^n)$-secure. Our first contribution is to improve their construction and show that it achieves asymptotically better security of $O(p^n)$.

Improved analysis of affine masked computation. As a second contribution, we extend our analysis of the refreshing gadget to protect affine computation. Affine computation (i.e., addition and multiplication by a constant) is frequently used in cryptographic schemes since it is less costly than non-linear operations. This is the reason why, for instance, many symmetric cryptographic schemes make massive use of affine computation. In our work, we give

an improved analysis for simple masked affine computation. In particular, we consider a very simple addition gadget, which computes the addition of two sharings $a_0, \ldots, a_{n-1}$ and $b_0, \ldots, b_{n-1}$ by adding the shares component-wise, i.e., $c_i = a_i + b_i$ followed by a refreshing of $c_0, \ldots, c_{n-1}$. We can show that this gadget remains $O(p^n)$-secure, where earlier works either require significantly more randomness (namely, [4] with $O(n^{2.4})$-randomness required, while ours needs $O(n)$-randomness to refresh the inputs) or require more noise (namely [16], $O(\sqrt{p}^n)$).

We also show how to extend our results to the masked computation of non-linear operations. Our multiplication gadget is essentially the widely used and analyzed ISW multiplication. While it is known that asymptotically, there are more advanced constructions that achieve security for a constant p, we improve the analysis of the ISW multiplication for small share numbers. Concretely, we can prove the security up to

$$[18p + 2(1 - (1 - p)^{8n} + 1 - (1 - \sqrt{3p})^{n-1})]^n \leq 8[1 - (1 - \sqrt{3p})^{8n}] ,$$

instead of $(32np + 4n\sqrt{3p})^n)$ as in [16]. Interestingly, this illustrates that the security for larger share number n is much better than previously assumed [16]. We believe that this is a worthwhile goal due to the following two reasons. First, it was shown [11] that the ISW-multiplication achieves better security than more advanced constructions for small values of n. Second, the ISW multiplication is widely used in many masking schemes, and hence it is important to better understand its security in the random probing model.

Parallel computation. Finally, we note that all our constructions are highly parallelizable. Parallel gadgets [13, 3] are particularly interesting for masked circuits as they are faster due to executing many operations at the same time. In addition, it is also more challenging to perform a side-channel attack against a parallel implementation than against a serial one. The basic idea is that parallel computations can increase the noise in the attacks, as shown in [24].

1.2 Related Work

Proof techniques. Analyzing leakage resilience of circuits via graphs was already proposed in [28] at Crypto 2015. They described a transformation of circuits based on graphs to generalize the ISW Multiplication and showed that it is closely related to Threshold Implementations [25] and the Trichina gate [30]. In particular, they give a generalized graph for the multiplication gadget using different layers, such as linear and non-linear layers, and compare the security of the different multiplication gadgets. In contrast to the work in [28], the work of [16] did not use the graph to analyze a gadget but a full circuit in the random probing model. They also use a graph based approach that is in particular useful to analyze the linear layers of circuits. We formalized the approach of [16], and give tighter security proves in the random probing model. Further, we propose gadgets with lower latency. Alternative approaches to analyze the security in the random probing model were proposed in [2, 4, 7, 9]. They introduce definitions for random probing composability based on counting the number of probes at the inputs and outputs of gadgets that are needed to simulate the leakage. To improve this approach, we could follow the recent work of Cassiers et al. [11] and tighten the analysis. Concretely, in [11], the authors use a definition, which they call the *Probe Distribution Table (PDT)*. The *PDT* allows a tighter analysis since it considers the concrete wires that the simulator needs. The drawback of the *PDT* approach is that the table grows exponentially with the number of shares of the gadgets, and thus a generic analysis is not possible. The work of [4, 7, 5] allows analysis for generic order, but it only provides security proofs for circuits with special structures. For this reason, the constructions are typically less efficient, as discussed above.

Compiler. As mentioned in Section 1.1, many compilers produce masked circuits with provable security in the random probing model. At Eurocrypt 2016, Andrychowicz et al. [2] presented a compiler with constant leakage probability using expander graphs. This rather is a feasibility result since expander graphs require a high number of shares. Two years later, Goudarzi et al. [19] gave a compiler for polynomial sharing requiring noise $p = O(1/\log(n))$. Here, they presented an NTT-based secure multiplication with complexity $O(n \log(n))$. The compiler was further improved in [20] to allow more general fields $\mathbb{F}$ and complexity $\Theta(n \log(n))$. They use the additive FFT algorithm proposed by Gao and Mateerin 2010 [17] to avoid the limitations of the classical NTT. With self-folding bases, a generalization of Cantor bases, they further optimized the gadget. However, the field size still restricts the number of shares $n < |\mathbb{F}|$ due to the share-wise different support points of poly sharings. Considering affine circuits, our compiler is more efficient. For example, our refresh gadget has linear complexity and does not use multiplication gates, while the one in [20] uses $n \log(n)/2$ multiplications. Further, our compiler allows a leakage rate of $O(1)$ for affine circuits instead of $O(1/\log(n))$. Regarding non-affine circuits, their construction has

better complexities with respect to efficiency and security due to their NTT-based multiplication. However, for our compiler, we slightly modified the ISW multiplication such that it is parallelizable.

To allow security for a constant leakage probability p, Ananth et al. [1] proposed a *modular approach* how to compose a secure compiler multiple times. Finally, several follow-up works further improved this approach [4, 8, 6]. However, as described in Section 1.1, this approach leads to relatively costly circuits with randomness complexity of at least $O(n^{2.4})$ for affine and non-affine circuits, while our compiler only requires $O(n)$ and $O(n^2)$, respectively. In particular, our work analyzes the widely used ISW multiplication that is still promising for reasonable share number ($2 \geq n \geq 32$) and noise parameters [11]. For this reason, we try to close the gap between practice and theory and give a tighter security analysis in the random probing model.

2 Background

Notations. Let $[n] := \{0, 1, \dots n-1\}$. Let $(\mathbb{F}, +, \cdot)$ be a finite field with its addition and multiplication (and let $-$ be its subtraction). We denote with $\mathbf{x}$ and $(x_i)_{i\in[n]}$ vectors with coefficients in the field $x_i \in \mathbb{F}$. Let X_0, X_1 be two random variables over a set $\mathcal{X}$. Their *statistical distance* is: $\Delta(X_0; X_1) := \frac{1}{2}\sum_{x\in\mathcal{X}} |\Pr[X_0 = x] - \Pr[X_1 = x]|$. If $\Delta(X_0; X_1) \leq \epsilon$, we say that X_0 *and* X_1 *are* ϵ*-close*.

Directed graphs. A *directed graph* is a pair $G = (V, E)$ with a set of vertices/nodes V and a set of edges $E \subseteq \{(x, y) | (x, y) \in V^2 \text{ and } x \neq y\}$, which are ordered tuples of vertices. Further, we write (x, y) to refer to such edges. We call x its *source node* and y its *destination node*. When we draw our graphs, we represent the edge (x, y) with an arrow from x pointing to y. We write $-(x, y) := (y, x)$ to exchange destination and source. A *sub-graph* $G' \subset G$ is a graph $G' = (V, E')$ with $E' \subset E$. This allows us to define unions of sub-graphs $G' = (V, E')$, $G'' = (V, E'')$ with $G' \cup G'' := (V, E' \cup E'')$. Note that all sub-graphs also have all nodes V. In our work, we are only interested in the edges and assume that each sub-graph still consists of all nodes. Further, if we consider graphs $G' = (V', E')$, $G'' = (V'', E'')$ with different nodes $V' \neq V''$, we also write $G = G' \cup G'' := (V' \cup V'', E' \cup E'')$. Hence, G is a graph consisting of two (sometimes unconnected) sub-graphs G' and G''. Let G be a graph. A *path* is the image of continuous functions $f : [0, 1] \subset \mathbb{R} \to G$. A *loop* is a path s.t. $f(0) = f(1)$. We consider only loops s.t. $f_{|(0,1)}$ is injective to avoid loops of type $(x, y), -(x, y)$ or containing it as a sub-loop.

Circuits. An *arithmetic circuit* over a finite field $\mathbb{F}$ is a labeled acyclic graph. Its edges are the *wires*, and its vertices are the *gates*. The edges pointing to a gate are the *input wires* of the gate, while those coming from it are the *output wires*. We use the following gates: *addition* ⊕ : with fan-in 2 and fan-out 1, outputting the addition of the 2 input variables; *subtraction* ⊖ : as the addition one, outputting the subtraction; *multiplication* ⊙ as the addition, outputting the multiplication; *constant* @$_a$: with fan-in 0 and fan-out 1, outputting the constant value a; *random* ® : with fan-in 0 and fan-out 1 outputting a uniform random variable; *copy* : with fan-in 1 and fan-out 2, outputting 2 copies of the input variable ; *input* I: with fan-in 1 and fan-out 1, outputting the input variable; *output* O: with fan-in 1 and fan-out 1, outputting the output variable. The last two gates (I and O) are added for syntactic reasons. A *complete* circuit is a circuit where there is an I gate at every input wire of the circuit and an O gate at every output wire; otherwise, the circuit is *incomplete*. The *completion* of an incomplete circuit is the addition of I and O whenever needed to make the circuit complete. An *affine* circuit is a circuit without multiplication gates. We denote with $\mathcal{W}(\mathsf{C})$ the set of wires of the circuit C.
A wire carries a *variable*. We say that two variables, x and y, are the *same variable* if the wires carrying x and y are connected only via copy gates. The *value* of a variable x is the value that is carried on the wire carrying x during an execution with fixed inputs and randomness.

Masking. One of the most common countermeasures against side-channel attacks is *masking*. The idea is to split the sensitive variables into n shares and then perform the computations on these shares and finally recover the output. We use an *encoding* scheme $(\mathtt{Enc}, \mathtt{Dec})$ to encode variables, *gadgets* to perform computations on encoded variables, and a *refreshing gadget* to securely compose multiple gadgets. We discuss these individual components below in more detail.

Encoding/decoding schemes. An *encoding* scheme Enc is a probabilistic algorithm that takes as input $x \in \mathbb{F}$ and outputs an n-tuple $(x_0, ..., x_{n-1}) = \mathtt{Enc}(x)$, where n is the *masking order*. The *decoding* scheme Dec takes as input an n-tuple $(x_0, ..., x_{n-1})$ and outputs $x = \mathtt{Dec}(x_0, ..., x_{n-1})$. For correctness, we want that for any $x \in \mathbb{F}$ it holds

that $\mathtt{Dec}(\mathtt{Enc}(x)) = x$. For security, we need that any subset of $n-1$ shares of $\mathtt{Enc}(x)$ are independent of x. We use *arithmetic encoding*. $\mathtt{Enc}(x)$ provides a randomized n-tuple $x_0, ..., x_{n-1}$ s.t. $\sum_{i=0}^{n-1} x_i = x$, and $\mathtt{Dec}(x_0, ..., x_{n-1}) = \sum_{i=0}^{n-1} x_i$. In the following, we will often denote an encoding of x by $(x_i)_{i\in[n]}$.

Gadgets. To perform computations on encoded variables, we construct *gadgets*. Gadgets are made out of simple gates such that even if the internals of the gadgets leak, the adversary will not learn any "useful" information. Suppose we have a gate implementing the function $f : \mathbb{F}^l \to \mathbb{F}^k$ (e.g.,$l = 2$ and $k = 1$, for ⊛). The corresponding gadget G_f is composed of many gates and performs the same operation where the input wires hold l encodings and the output wires carry k encodings of the outputs. We require *soundness* from the gadgets, i.e., gadgets that perform the same operation as the underlying gate, just in the encoded domain. Formally, we have $\forall \mathbf{x} = (x^0, ..., x^{l-1}) \in \mathbb{F}^l$,

$$f(x^0, ..., x^{l-1}) = (\mathtt{Dec}(y_i^0)_{i\in[n]}, \ldots, \mathtt{Dec}(y_i^{k-1})_{i\in[n]})$$

with $((y_i^0)_{i\in[n]}, \ldots, (y_i^{k-1})_{i\in[n]}) \leftarrow \mathsf{G}_f(\mathtt{Enc}(x^0), \ldots, \mathtt{Enc}(x^{l-1}))$.

Refreshing schemes. Refreshing schemes (or refreshing gadgets) are gadgets G_f where f is the identity.[1] The scheme takes as input an encoding $(x_i)_{i\in[n]}$, and outputs a re-randomized encoding $(y_i)_{i\in[n]}$, such that $\mathtt{Dec}((x_i)_{i\in[n]}) = \mathtt{Dec}((y_i)_{i\in[n]})$. In this work, we consider refreshing schemes using a linear number of random gates ® .

In Figure 1a the simple refresh `sRef` of [16] is depicted, initially introduced in [29]. The gadget adds a random value to each input $y_i \leftarrow x_i + r_i$ with $i = 0, \ldots n-2$ and subtracts each random value from the last input $y_{n-1} \leftarrow x_{n-1} - (r_0 + \cdots + r_{n-2})$.

This work considers an alternative to `sRef` that we call `pRef`; see Figure 1b. This gadget was initially introduced in [3] and has the key feature that it is highly parallelizable. `pRef` takes as an input $(x_i)_{i\in[n]}$ with n random values r_i and processes them in two parallel steps. In the first step, it computes $b_i \leftarrow x_i + r_i$, and in the second step, it subtracts r_{i-1} from b_i such that $y_i \leftarrow b_i - r_{i-1 \pmod n}$ for all $i \in [n]$, obtaining $(y_i)_{i\in[n]}$.

Circuit compilers. Given the components from above, we can transform circuit C into a masked circuit $\widehat{\mathsf{C}}$. This is done via the concept of a *circuit compiler* $\mathtt{CC}$. $\mathtt{CC}$ works as follows: First, $\mathtt{CC}$ replaces each wire carrying x with a *bundle* of n-wires carrying an encoding of x, $(x_i)_{i\in[n]}$. Next, it replaces all gates in C with the corresponding sound gadgets, input I gates with $\widehat{\mathsf{I}}$ *input encoders* (which encodes the input), and output O gates with $\widehat{\mathsf{O}}$ *output decoders*. Finally, between every two gadgets, the compiler $\mathtt{CC}$ adds a *refreshing gadget* to ensure secure composition. The masked transformation $\widehat{\mathsf{C}}$ of a complete circuit C is *sound* if $\widehat{\mathsf{C}}(\mathbf{x}) = \mathsf{C}(\mathbf{x})$ for every possible input $\mathbf{x}$ of C. For an incomplete circuit C, we say that the transformation $\widehat{\mathsf{C}}$ is *sound* if the transformation of its completion is sound. A compiler $\mathtt{CC}$ is *sound* if for all circuits C the transformation $\widehat{\mathsf{C}} = \mathtt{CC}(\mathsf{C})$ is sound.

Random probing model. As discussed in the introduction, we use the p-random probing model, originally introduced in [21] to model side-channel leakage of the transformed circuit $\widehat{\mathsf{C}}$. In the p-random probing model, each wire leaks the value that it carries with probability p. Following [21], we assume that the wires of the input encoders $\widehat{\mathsf{I}}$ and output decoders $\widehat{\mathsf{O}}$ *do not leak*. Notice that, as in [21], this is without loss of generality when we move from stateless to stateful circuits. To make it explicit what wires leak, we will denote in the following with $\mathcal{W}'(\widehat{\mathsf{C}}) \subset \mathcal{W}(\widehat{\mathsf{C}})$ the set of wires of the circuit $\widehat{\mathsf{C}}$ that can leak. The definition below formalizes security in the p-random probing model.

The transformed circuit is private if its leakage reveals nothing about its inputs and outputs. We can define this with a security experiment.

Definition 1 (Privacy [16])**.** Let C be a circuit with fan-in k with input $\mathbf{x} = (x_1, ..., x_k)$. Further, let $\widehat{\mathsf{C}}$ be a sound transformation of C and $p \in [0, 1]$ its leakage probability. The *leakage experiment* $\mathsf{Leak}(\widehat{\mathsf{C}}, \mathbf{x}, p)$ is defined as follows:

- We fed $\mathbf{x}$ to $\widehat{\mathsf{C}}$ resulting in some assignments of the wires of $\widehat{\mathsf{C}}$. If C is incomplete, the input bundle corresponding to the input wire containing x is fed with an encoding $(x_i)_{i\in[n]}$ of x.
- Each wire w of $\mathcal{W}'(\widehat{\mathsf{C}})$ is added to $\mathcal{L}_p(\widehat{\mathsf{C}})$ with probability p.

[1] We emphasize that this does not imply that G_f is also the identity. Since the gadget can be probabilistic, the encoding of the outputs can be re-randomized.

$\texttt{sRef}((x_i)_{i\in[n]})$

for $j \in \{0 \dots n-2\}$
  $r_j \leftarrow_{\$} \mathbb{F}$
$c_0 \leftarrow 0$
for $j \in \{0 \dots n-2\}$
  $c_{j+1} \leftarrow c_j + r_j$
  $y_j \leftarrow x_j + r_j$
$y_{n-1} \leftarrow x_{n-1} - c_{n-1}$
return $(y_i)_{i\in[n]}$

(a) Simple refresh gadget [29].

$\texttt{pRef}((x_i)_{i\in[n]})$

for $j \in \{0 \dots n-1\}$
  $r_j \leftarrow_{\$} \mathbb{F}$
for $j \in \{0 \dots n-1\}$
  $b_j \leftarrow x_j + r_j$
  $y_j \leftarrow b_j - r_{j-1 \pmod n}$
return $(y_i)_{i\in[n]}$

(b) Parallel refresh gadget [3].

Figure 1: Refresh gadgets with linear random complexity.

- Output: $(\mathcal{L}_p(\widehat{\mathsf{C}}), A_{|\mathcal{L}_p(\widehat{\mathsf{C}})})$, where A is the set of the values carried by the wires of $\mathcal{W}$ during the circuit evaluation of $\widehat{\mathsf{C}}$ on input $\mathbf{x}$.

$\widehat{\mathsf{C}}$ is (p, ϵ)-*private* if there is a simulation algorithm that, not knowing $\mathbf{x}$, outputs a random variable that is ϵ-close to the actual output of $\mathsf{Leak}(\widehat{\mathsf{C}}, \mathbf{x}, p)$

In other words, the masked circuit $\widehat{\mathsf{C}}$ is (p, ϵ)-*private* if the leakage in experiment $\mathsf{Leak}(\widehat{\mathsf{C}}, \mathbf{x}, p)$ can be simulated independently from the inputs up to ϵ statistical distance. More precisely, if for any two inputs $\mathbf{x}, \mathbf{x'}$ of the circuit, the distributions $\mathsf{Leak}(\widehat{\mathsf{C}}, \mathbf{x}, p)$ and $\mathsf{Leak}(\widehat{\mathsf{C}}, \mathbf{x'}, p)$ are ϵ-close, then $\widehat{\mathsf{C}}$ is (p, ϵ)-private. This observation was used to prove security in [16]. Therefore, they defined *Extended Leakage Shiftability* to describe when the leakage is independent of the input. In particular, shiftability accurately describes the fact that we can change the input of a circuit so that the observed leakage does not contradict the new input.

Definition 2 (Leakage Shiftability [16])**.** Let $\widehat{\mathsf{C}}$ be a sound transformation of a circuit C. We say that an output L of the experiment $\mathsf{Leak}(\widehat{\mathsf{C}}, \mathbf{x}, p)$ is *shiftable* to $\mathbf{x'}$ if it can be output of the experiment $\mathsf{Leak}(\widehat{\mathsf{C}}, \mathbf{x'}, p)$.

In other words, let $L \leftarrow \mathsf{Leak}(\widehat{\mathsf{C}}, \mathbf{x}, p)$ be the leakage with $L = A_{|\mathcal{L}_p(\widehat{\mathsf{C}})}$, where A is the set of the values carried by the wires during the circuit evaluation of $\widehat{\mathsf{C}}$ on input $\mathbf{x}$. Then, L is shiftable if there is an assignment A' with the same probability during the circuit evaluation of $\widehat{\mathsf{C}}$ on input $\mathbf{x'}$ s.t. it still holds for both leakages $L = A'_{|\mathcal{L}_p(\widehat{\mathsf{C}})}$. In this case, we can shift the values of the variables from A to A' without modifying the values of the variables leaked. This technique allows more fine-grained security analyzes than simulatability since we show where we can modify the input encodings of each gadget (without ignoring where exactly, as done for simulatability). So, if the leakage is shiftable and the leakage of the shifted encoding has the same distribution as the unshifted one, the leakage is independent of the encoding. Hence, the leakage can be simulated without knowing the encoded value. This property was also used in [16] to prove their compiler security.

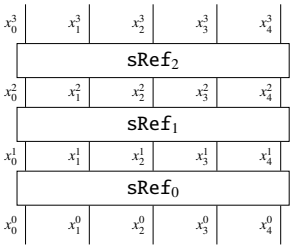
Corollary 1 ([16])**.** *Let $\widehat{\mathsf{C}}$ be the sound transformation of a circuit C via Dziembowski et al.'s compiler [16]. If*

$$\Pr[\mathsf{Leak}(\widehat{\mathsf{C}}, \boldsymbol{x}, p) \textit{ is not shiftable to } \boldsymbol{x'}, \textit{ for any } \boldsymbol{x'}] \leq \epsilon,$$

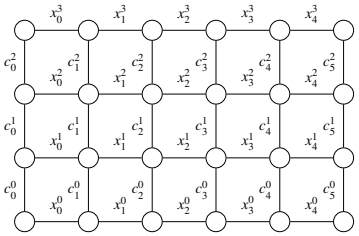
for any input $\boldsymbol{x}, \boldsymbol{x'}$ then $\widehat{\mathsf{C}}$ is (p, ϵ)-secure.

To compute the shift probability, the authors give a new technique to transform this problem into a graph path problem. Next, we present the class of graphs they consider.

Original leakage diagram. Dziembowski et al. [16] introduced the concept of *leakage diagrams*. They represent all the variables on a graph and those variables whose values are leaked in a sub-graph called the *leakage diagram*. For example, they represent multiple consecutive executions of `sRef` (depicted in Figure 2a) with the diagram depicted in Figure 2b. The edges of the diagram represent the intermediate values that are computed during the execution of this circuit. The intermediate values of each `sRef` execution are represented by two consecutive rows and by the vertical edges between these two rows. On the lower row, there are n edges that represent the input



(a) Simple refresh gadget [16].



(b) The graph of the simple refresh used for the leakage diagram [16].

Figure 2: Refresh gadgets with linear random complexity.

shares (one edge per share), while on the upper row, there are n edges representing the output shares of a refreshing gadget. The vertical edges between two rows represent the partial sum c^i_j of the random values used during that execution. Since $c^i_{j+1} = c^i_j + r^i_j$, the variable r^i_j is represented by both the edges c^i_j and c^i_{j+1}. They also add the edges corresponding to the variable c^i_n. These c^i_n are defined similarly to the others as follows $c^i_n = c^i_{n-1} + x^i_{n-1} - x^{i-1}_{n-1}$. Thus, it always holds $c^i_n = 0$ because $c^i_n = c^i_{n-1} + x^i_{n-1} - x^{i-1}_{n-1} = c^i_{n-1} + (x^{i-1}_{n-1} - c^i_{n-1}) - x^{i-1}_{n-1} = 0$.

During k executions of `sRef` the adversary receives a leakage $\mathcal{L}_p$ (see Def. 1). The *leakage diagram* corresponding to $\mathcal{L}_p$ is the subgraph of Fig. 2b composed by all the edges corresponding to the variables belonging to $\mathcal{L}_p$ (and all c^i_0 and c^i_n since they are always equal to 0. Thus their values are always known by the adversary). Further, Dziembowski et al. [16] proved that the leakage can be simulated independently from the input $x = \mathtt{Dec}((x^0_i)_{i\in[n]})$ if there is no path from the left to the right of the leakage diagram. The technique was extended to analyze more complex masked circuits where each gadget's output is refreshed with the `sRef` gadget. Further, they show that the security can be bounded with the probability that there is such a path.

3 Parallel Compiler

The circuit compiler we present in this paper has the key feature that its operations are highly parallelizable. It uses the standard gadgets for addition (cf. Fig. 3a), copy (cf. Fig. 3b), random (cf. Fig. 3d), and constant (cf. Fig. 3c) operations. When considering their circuit representation, the gadgets have a low depth; hence, they can be executed highly parallelly. The more interesting gadgets are the ones for the multiplication of two encoded inputs (cf. Fig. 3e) and for refreshing an encoding (cf. Fig. 1b). The multiplication gadgets with input encodings $(a_i)_{i\in[n]}$ and $(b_i)_{i\in[n]}$ computes the tensor product $a_i b_j$ of all shares. As in the ISW multiplication, this results in n^2 products $a_i b_j$ that we need to compress to a random encoding of the output by appropriately adding up these values and blinding the intermediate results by injecting fresh randomness. In contrast to the ISW multiplication that has depth n, we change the way in which this final addition is done to reduce the depth to $log(n)$. This significantly reduces the latency of the gadget from n to $\log(n)$. The refresh gadget re-randomizes an encoding such that it still decodes to the same value. The refresh gadget of our compiler has the key feature that it only has depth 2 in contrast to the simple refreshing from [16], which has an asymptotic depth of n.

Compiler CC^p. The parallel compiler CC^p takes as input an arbitrary circuit C using the gates  ,  ,  ,  ,  and  and replaces each gate with the corresponding gadget from Figure 3 in $\widehat{\mathsf{C}}$. Note that the gadget for works by slightly modifying the `Add` gadget such that the second input is first share-wise transformed to its additive inverse, i.e., by setting $-b_i \leftarrow b_i$. At the high level, the topology of C and $\widehat{\mathsf{C}}$ is the same, i.e., if two gates are connected by wires in C, then the corresponding gadgets are connected in the same way through wire bundles in $\widehat{\mathsf{C}}$. These wire bundles carry the encodings of the variables corresponding to the wires in C. Finally, to guarantee composability, the compiler CC^p inserts refresh gadgets between each computational gadget to inject further randomness. By applying the compiler to a circuit C, we get a parallelizable masked circuit that we denote with $\widehat{\mathsf{C}} \leftarrow \mathsf{CC}^p(\mathsf{C})$. We start by showing the soundness of the compiler CC^p in the following corollary.

$\texttt{Add}((a_i)_{i\in[n]}, (b_i)_{i\in[n]})$

for $i \in \{0 \dots n-1\}$

$c_i = a_i + b_i$

return $(c_i)_{i\in[n]}$

(a) The addition gadget Add for $\oplus$ and $\ominus$ with $b_i \leftarrow -b_i$.

$\texttt{Copy}((a_i)_{i\in[n]})$

for $i \in \{0 \dots n-1\}$

$b_i = a_i$

$c_i = a_i$

return $(b_i)_{i\in[n]}, (c_i)_{i\in[n]}$

(b) The copy gadget Copy for $\copy$.

$\texttt{Const}_a$

return $(a_i)_{i\in[n]}$

(c) The constant gadget $\texttt{Const}_a$ for $@$ with $(a_i)_{i\in[n]} \leftarrow \texttt{Enc}(a)$.

$\texttt{Rand}()$

for $i \in \{0 \dots n-1\}$

$r_j \leftarrow_\$ \mathbb{F}$

return $(r_i)_{i\in[n]}$

(d) The random gadget Rand for $\circledR$.

$\texttt{Mult}((a_i)_{i\in[n]}, (b_i)_{i\in[n]})$

for $i \in \{1, ..., n-1\}$

  for $j \in \{0, ..., i-1\}$

    $z_{0,i,j} \leftarrow_\$ \mathbb{F}$

for $i \in \{0 \dots n-1\}$

  for $j \in \{i+1, ..., n-1\}$

    $w_{i,j} = a_i \cdot b_j - z_{0,j,i}$

    $z_{0,i,j} = w_{i,j} + a_j \cdot b_i$

for $i \in \{0, ..., n-1\}$

  $z_{0,i,i} = a_i \cdot b_i$

$L = log_2(n+1)$

for $i \in \{0, n-1\}$

  for $l \in \{1, ..., L\}$

    for $j \in \{0, ..., 2^{L-l} - 1\}$

      $z_{l,i,j} = z_{l-1,i,2j} + z_{l-1,i,2j+1}$

for $i \in \{0, ..., n-1\}$

  $c_i = z_{L,i,0}$

return $(c_i)_{i\in[n]}$

(e) The multiplication gadget Mult for $\odot$.

Figure 3: Parallel gadgets.

Corollary 2. *Let* $\mathsf{C} : \mathbb{F}^s \to \mathbb{F}^t$ *be an arbitrary circuit and* $\widehat{\mathsf{C}} \leftarrow \mathsf{CC}^p(\mathsf{C})$. *For any* $x^0, \dots, x^{s-1} \in \mathbb{F}$, *we have:*

$$((y_i^0)_{i\in[n]}, \dots (y_i^{t-1})_{i\in[n]}) \leftarrow \widehat{\mathsf{C}}(\texttt{Enc}(x^0), \dots, \texttt{Enc}(x^{s-1}))$$

with $\mathsf{C}(x^0, \dots, x^{s-1}) = (\texttt{Dec}((y_i^0)_{i\in[n]}), \dots, \texttt{Dec}((y_i^{t-1})_{i\in[n]}))$.

Proof. The proof of the corollary immediately follows from the soundness of the gadgets depicted in Figure 3. □

Our compiler has similar features to the compiler of [16]. More precisely, we can show that Corollary 1 also holds for our compiler.

Corollary 3. *Let* $\widehat{\mathsf{C}}$ *be the sound transformation of a circuit* C *via our compiler,* CC^p. *If*

$$\Pr[\mathsf{Leak}(\widehat{\mathsf{C}}, \boldsymbol{x}, p) \textit{ is not shiftable to } \boldsymbol{x'}, \textit{ for any } \boldsymbol{x'}] \leq \epsilon,$$

for any input $\boldsymbol{x}, \boldsymbol{x'}$ *then* $\widehat{\mathsf{C}}$ *is* (p, ϵ)*-secure.*

Proof. This proof is similar to the one in [16]. According to Definition 1, we need to simulate $\mathsf{Leak}(\widehat{\mathsf{C}}, \mathbf{x}, p)$ independently from the input $\mathbf{x}$ up to ϵ statistical distance. Therefore, we prove that the distribution of $\mathsf{Leak}(\widehat{\mathsf{C}}, \mathbf{x'}, p)$-experiment is ϵ-close to $\mathsf{Leak}(\widehat{\mathsf{C}}, \mathbf{x}, p)$-experiment for any $\mathbf{x}, \mathbf{x'}$ if it is shiftable. This immediately gives the required simulator[2]. So, we are left with the proof that shiftability for any $\mathbf{x}, \mathbf{x'}$ implies that the outputs of the experiments $\mathsf{Leak}(\widehat{\mathsf{C}}, \mathbf{x}, p)$ and $\mathsf{Leak}(\widehat{\mathsf{C}}, \mathbf{x'}, p)$ are ϵ-close. Let n be the number of shares used by $\widehat{\mathsf{C}}$. We will follow a sequence

[2]The simulator takes a random input $\mathbf{x'}$ and outputs $\mathsf{Leak}(\widehat{\mathsf{C}}, \mathbf{x'}, p)$.

Input: $(x_0, ..., x_n), (y_0, ..., x_n^{i+1})$
$r_{n-1} \leftarrow_{\$} \mathbb{F}$
for $j = 1, ..., n-1$
$r_j = x_j - y_j - r_{j-1}$
endfor
Return $(r_0, ..., r_n)$

Figure 4: The Randomness Reconstructor *Rand*(pRef) for $(y_i)_{i\in[n]} \leftarrow \texttt{pRef}((x_i)_{i\in[n]})$.

of Games where the first game represents the circuit with input **x** and the last one with input **x'**:

Game 0: The leakage experiment $\mathsf{Leak}(\widehat{\mathsf{C}}, \mathbf{x}, p)$.

Game 1: The modified Game 0, where we have modified how pRef gadgets pick the randomness. Instead of picking the randomness uniformly at random, pRef picks uniformly at random a new encoding of x, $(x_0^*, ..., x_n^*)$, then, via the randomness reconstructor *RandR*(pRef) (described in Fig. 4), it computes the internal randomness. Hence, the output shares $(x_0^i, ..., x_n^i)$ are the random encoding $(x_0^*, ..., x_n^*)$.
Transition between Game 0 and Game 1: We show that the randomness reconstructor *RandR*(pRef) outputs randomness indistinguishable from that used by pRef of Game 0. In both cases, r_0 is picked uniformly at random, and hence, r_0 is picked in the same way. Since it holds $r_j = x_j - y_j - r_{j-1}$ and all y_j are picked uniformly at random, all r_j have the same distribution as if they are picked uniformly at random. Since Game 0 and Game 1 only differ in how the randomness is used, and the randomness in Game 0 and Game 1 is indistinguishable. Hence, both games are indistinguishable.

Game 2: It is the modified Game 1, where we have replaced the non-leaked intermediate values of the variables in such a way that $\widehat{\mathsf{C}}$ has the input **x'**. Note that this is possible due to the shiftablity assumption, and hence, we can apply shiftabilty without giving further details.
Transition from Game 1 to Game 2: Since an outcome of the $\mathsf{Leak}(\widehat{\mathsf{C}}, \mathbf{x}, p)$-experiment can be shifted to an outcome of the $\mathsf{Leak}(\widehat{\mathsf{C}}, \mathbf{x'}, p)$-experiment except with probability ϵ, we can do this shift with probability $1 - \epsilon$. Due to the ϵ, this is the only game hop in the proof with a loss.

Game 3: Game 3 is the modified Game 2, where we have replaced the input and output encodings of all gadgets with random encodings of the same value as we did in Game 1. In other words, every input/output encoding is again replaced with a random encoding that still decodes to the same value (the one that we got in Game 2).
Transition between Game 2 and Game 3: Since the inputs and outputs of both games are still random encodings, we cannot distinguish Games 2 and 3. Note that the encodings in Game 2 are still random because the shiftability depends only on the leaked variables and not on the values that this variable assumes.

Game 4: The leakage experiment $\mathsf{Leak}(\widehat{\mathsf{C}}, \mathbf{x'}, p)$.
Transition between Game 3 and Game 4: It is the inverse of the transition between Game 0 and Game 1. Thus, using *RandR*(pRef), we can prove that these two games are indistinguishable.

As mentioned in the transition from Game 1 to Game 2, this is the only step with a security loss of ϵ. This proves the claim since it immediately flows that Game 0 and Game 4 are ϵ-close. □

3.1 Dependency Graph for our Gadgets

The values carried by the wires of a masked circuit can be considered as a set of random variables randomized by the random input encodings and the internal random gates. When we analyze such random variables X and Y representing intermediate wires, they can carry values $x, y \in \mathbb{F}$ with $\Pr[X = x] \geq 0$ and $\Pr[Y = y] \geq 0$. When we analyze the leakage resilience of circuits, we can distinguish two cases (i) intermediate values are independent $\Pr[X = x, Y = y] = \Pr[X = x] \cdot \Pr[Y = y]$ or (ii) they are dependent $\Pr[X = x, Y = y] \neq \Pr[X = x] \cdot \Pr[Y = y]$. To describe the dependencies of such a set of random variables T occurring as intermediate values during the

computation of a masked circuit, we use a *dependency graph* (DG), represented as a *directed labeled* graph where all edges have a source and destination node. Further, each edge has a label containing at least one variable $x \in T$. For any subset of random variables $S \subset T$, we get a subgraph consisting of all edges whose labels contain at least a variable in S. Further, the edges of the dependency graph are linked so that any subset S of variables describes a subgraph with no loops (a *loop* is a path with the same starting and ending point) if the set S consists of random variables that are independent of the decoded inputs or outputs of the circuit.

Definition 3. Let C be a masked circuit C with intermediate values T, and $\mathcal{G}$ a labeled graph with k edges $e_0, e_1, \ldots, e_{k-1}$ each labeled with T_i such that $\bigcup_{i\in[k]} T_i = T$. $\mathcal{G}$ is a dependency graph if for each sub graph $\mathcal{G}' \subset \mathcal{G}$ with edges e_i $i \in I \subset [k]$, it holds

$$\mathcal{G}' \text{ has no loop} \Rightarrow \bigcup_{i\in I} T_i \text{ is independent of the unmasked inputs.}$$

Dependency graphs are helpful for three reasons: First, when we consider the leakage as a subset S of the random variables T, we can represent the leakage as a subgraph, the so-called leakage diagram, $\mathcal{LD}$ (see Definition 5). Second, using the graph property that all subsets with elements dependent on the decoding of the inputs are structured as loops; we can classify leakage diagrams as "good" or "bad" (see Section 4.1). All "good" leakage diagrams correspond to leakages that can be simulated without knowing sensitive values. Third, with the classification, we can upper bound the probability that the leakage corresponds to a "bad" leakage diagram if all wires leak their values with probability p. Hence, we can analyze the leakage resilience of a circuit (Sec. 5). In the following, we describe the dependency graphs of a simple encoding and for each gadget used by our compiler. Then, we show how to compose the dependency graphs of our gadgets to get the dependency graph for any output of our compiler.

Dependency graphs of masked values. Let us consider an encoding of a secret x, with $(x_i)_{i\in[n]} \leftarrow \texttt{Enc}(x)$. The corresponding set of random variables is

$$T = \{x_0, x_1, \ldots, x_{n-1}\}\ .$$

We represent this with the dependency graph depicted in Figure 5a with $T_i = \{x_i\}$. There are n edges, each labeled with one of the variables of T. For simplicity, we call the edge labeled with $\{x_i\}$ the x_i-edge. All n edges form a loop, which we can see as a "circle" [3]. It is easy to see that this "circle" is the only loop in the graph, and any strict subset of the n edges does not form a loop. This describes the abovementioned property that dependent random variables form a loop. The variables $x_0, x_1, \ldots, x_{n-1}$ describe a loop because they depend on the secret with $\texttt{Dec}((x_i)_{i\in[n]}) = x$. However, any strict subset $S \subset T$ is a set of independent random variables due to the security property of the secret sharing, and that is why they do not form a loop in the dependency graph.

Further, the dependency graph is a directed graph, and the x_i-edge connects the destination node of the x_{i-1}-edge with the source node of the x_{i+1}-edges.[4] The direction of an edge represents the sign of the edges labels. Thus, we can also think that there is an edge labeled $-x_i$ which connects the source node of the x_{i+1}-edge with the destination node of the x_{i-1}-edge. In detail, the path ove rall x_i's can be considered as the sum over all x_i's. If the path also consists of edges with opposite directions, we subtract the variables represented as an edge with the opposite direction instead of adding them. Since the random variables of the `Rand` and `Const` gadget only consist of output variables, their dependency graphs are simple dependency graphs of maskings. Next, we give the dependency graph of `pRef`.

Dependency graph for `pRef`. Let `pRef` refresh an input $(x_i)_{i\in[n]}$ with random values $(r_i)_{i\in[n]}$. As intermediate variables it computes $(b_i)_{i\in[n]}$ with $b_i = x_i + r_i$ and outputs $(y_i)_{i\in[n]}$ with $y_i = b_i - r_{i-1}$. This leads to the set of random variables

$$T_{\texttt{pRef}} = \{x_0, x_1, \ldots, x_{n-1}, r_0, r_1, \ldots, r_{n-1}, b_0, b_1, \ldots, b_{n-1}, y_0, y_1, \ldots, y_{n-1}\}.$$

The dependency graph of `pRef` is depicted in Figure 5b. Each edge is labeled by a single value of $T_{\texttt{pRef}}$. Thus, we can use the same convention as before, and the edge labeled with $\{x\}$ is the x-edge for all $x \in T_{\texttt{pRef}}$. The dependency graph (Fig. 5b) forms a skeleton of a cylinder. The $x_0, ..., x_{n-1}$-edges form a loop, which is the bottom circle of the cylinder, while the $y_0, ..., y_{n-1}$-edges form a loop, which is the top circle of the cylinder. These two loops are

[3] In the following, when we use *circles* and *rectangles* for the elemental geometrical shapes, while *loops* for the graph loops defined before. Clearly, "circles" and "rectangles" are loops if they are defined only with the edges of a graph.

[4] To simplify the notion, we omit the (mod n) in all the operations with the index of variables, as for $i - 1$.

identical to the dependency graph of masked values described in the previous paragraph. The remaining edges form the lateral surface of the cylinder. More precisely, this lateral surface consists of n rectangles defined by the x_i, r_i, y_i, r_{i-1}-edges. The loops defined by those rectangles describe the subset $\{x_i, r_i, y_i, r_{i-1}\}$ of dependent random variables because $x_i + r_i - r_{i-1} = y_i$. Here it becomes clear why the dependency graph is a directed graph because the r_{i-i} has the opposite direction when we consider the alternative path r_{i-1}, x_i r_i that connects the same nodes as y_i. Hence we only add x_i and r_i but subtract r_{i-1} to compute y_i. Further, the n rectangles each have a diagonal edge: the b_i-edges that describe the remaining intermediate values b_i. We add them to the graph such that they fulfill the same additive properties as the n rectangles. In detail, an alternative path for the edge b_i is r_{i-1}, y_i or x_i, r_i because it holds $b_i = r_{i-1} + y_i$ and $b_i = x_i + r_i$. This construction fulfills the property again that all subsets $S \subset T_{\texttt{pRef}}$ that depend on the decoding of the input (or output) form a loop in the graph. More precisely, they form a loop that orbits the lateral surface of the cylinder structure of the graph. In Proposition 6 (Sec. 4.2), we give the formal proof.

Dependency graph for `Copy`. The copy gadget `Copy` (Fig. 3b) takes as input an encoding $(a_i)_{i\in[n]}$ and outputs two encodings $(b_i)_{i\in[n]}$ and $(c_i)_{i\in[n]}$ with $a_i = b_i = c_i$ for all i. Note that the dependency graph only considers the random variables carried by the wires, and a_i, b_i, and c_i represent the same variable. Thus,

$$T_{\texttt{Copy}} = \{a_0, ..., a_{n-1}, b_0, ..., b_{n-1}, c_0, ..., c_{n-1}\} = \{a_0, \dots, a_{n-1}\}$$

results in the same dependency graph as the usual masking described above. The dependency graph is the graph depicted in Figure 5a, with $T_i = \{a_i, b_i, c_i\} = \{a_i\}$. Again, it is clear that the graph is a dependency graph because any set of possible leaked values is a set of independent values if the set does not describe a subgraph with a loop.

Dependency graph for `Add`. The addition gadget `Add` (Fig. 3a) is a share-wise addition. It takes as input two encodings $(a_i)_{i\in[n]}$ and $(b_i)_{i\in[n]}$ and outputs an encoding $(c_i)_{i\in[n]}$ with $c_i = a_i + b_i$ for all $i \in [n]$. This leads to

$$T_{\texttt{Add}} = \{a_0, ..., a_{n-1}, b_0, ..., b_{n-1}, c_0, ..., c_{n-1}\}.$$

A possible dependency graph is depicted in Figure 5a with $T_i = \{a_i, b_i, c_i\}$. Compared with the graphs presented before, the difference is that each edge represents multiple different variables. More precisely, we map all values of each share-wise computation to a single edge. However, it holds again that any strict sub-graph has no loop and describes a subset that is independent of the decoded input or output (see Sec. 4.2 Prop. 5). Note that this fact immediately follows with the same argument as the one for dependency graphs of usual masking when we consider the approximation that an adversary learns all variables a_i, b_i, c_i if at least one is leaked.

Dependency graph for `Mult`. The `Mult` gadget takes as input two encodings $(a_i)_{i\in[n]}$, and $(b_i)_{i\in[n]}$, and outputs an encoding $(c_i)_{i\in[n]}$ with

$$\texttt{Dec}((c_i)_{i\in[n]}) = \texttt{Dec}((a_i)_{i\in[n]}) \cdot \texttt{Dec}((b_i)_{i\in[n]}) \ .$$

Therefore, the gadget computes the intermediate values $z_{l,i,j}$, and $w_{i,j}$, as defined in Figure 3e. This leads to the set of random variables generated by the circuit

$$T_{\texttt{Mult}} = \{a_0, ..., a_{n-1}, b_0, ..., b_{n-1}, a_i \cdot b_j, z_{l,i,j}, w_{i,j} \text{ and } i, j \in [n], l \in [L+1]\},$$

with $L = \log_2(n+1)$. Considering Figure 3e, it turns out that $w_{i,j}$ and $z_{l,i,j}$ are not defined for all j and l. For the sake of simplicity, we omit the precise treatment and assume that all the variables $w_{\widetilde{i,j}}$ not defined by the algorithm are zero. We see them as not elements in $T_{\texttt{Mult}}$. A possible dependency graph is depicted in Figure 5a with

$$T_i = \{a_i, b_i, c_i, a_i \cdot b_i, a_i \cdot b_j, a_j \cdot b_i, w_{i,j}, w_{j,i}, z_{l,i,j} \text{ and } j \in [n], l \in [L+1]\} \ .$$

It is very similar to the graph used for `Add`. The idea is to label the T_i-edge with the i^{th} share of the inputs and outputs a_i, b_i, c_i. The difference to `Add` is that `Mult` also has intermediate values that we still have to add to the graph. Therefore, we add to T_i all the monomials $a_i b_j$ and $a_j b_i$, $j \in [n]$. Note the monomial $a_i b_j$ (and $a_j b_i$) belongs to *both* labels T_i and T_j. Finally, we add all intermediate addends $w_{i,j}$s and the $z_{l,i,j}$ of the i^{th} output share to T_i. This is inspired by [21, 16]. In Section 4.2 (Proposition 5), we prove that the variables $S \subset T_{\texttt{Mult}}$ that do not describe a sub-graph with a loop are independent of the decoding of $(a_i)_{i\in[n]}$, $(b_i)_{i\in[n]}$, and $(c_i)_{i\in[n]}$. Next, we give the composition results to construct the dependency graph of any output of our compiler.

3.2 Composition of Dependency Graphs

In the previous section, we introduced the dependency graphs for our gadgets. Since our compiler always outputs a composition of those gadgets, we are interested in how to get the dependency graphs for the composition of those gadgets. Therefore, we give composition results to obtain the dependency graphs of composed gadgets G_1 and G_2 with dependency graphs DG_1 and DG_2, respectively. In [11], they distinguish two different compositions, the sequential composition written $G = \mathsf{G}_1 \circ \mathsf{G}_2$ where G_1 gets as input the output of G_2, or the parallel composition written $G = \mathsf{G}_1 \| \mathsf{G}_2$ where both gadgets compute parallel and independently of each other. When we consider parallel compositions of two gadgets where both gadgets run independently (with no shared inputs), it is easy to see that the dependency graphs of both gadgets do not affect each other. Hence, the dependency graph of $\mathsf{DG}_{\mathsf{G}_1 \| \mathsf{G}_2}$ can be seen as a union of sets $\mathsf{DG}_1 \cup \mathsf{DG}_2$ where both graphs are considered as one graph but there is no edge connecting DG_1 and DG_2 because there is no further dependency generated by the parallel composition. To compute the dependency graph of the sequential composed gadgets $\mathsf{G}_1 \circ \mathsf{G}_2$ out of DG_1 and DG_2 we use a modified union of both dependency graphs. When we consider sequential compositions, an output wire becomes an input wire of another gadget. Hence, two wires merge to only one wire, and a modified union is required where the two edges of such connected wires become the same. For this reason, we define a function f (so-called attaching function) that maps the edges of the G_2's output wires to the according edges of the G_1's input wires. The result is a union of both graphs where f defines which edges of DG_1 and DG_2 are the same, and we can write

$$\mathsf{DG}_{\mathsf{G}_1 \circ \mathsf{G}_2} = \mathsf{DG}_{\mathsf{G}_1} \cup_f \mathsf{DG}_{\mathsf{G}_2}$$

For example, let $\mathsf{G}_1 = \texttt{pRef}$ and $\mathsf{G}_2 = \texttt{Add}$, then the output $(c_i)_{i\in[n]}$ is the input of $\texttt{pRef}$. This can be described with the attaching function f that maps the edge of c_i in $\mathsf{DG}_{\texttt{pRef}}$ to the edge of T_i in $\mathsf{DG}_{\texttt{Add}}$, and the resulting dependency graph $\mathsf{DG}_{\texttt{pRef}\circ\texttt{Add}} = \mathsf{DG}_{\texttt{pRef}} \cup_f \mathsf{DG}_{\texttt{Add}}$ is depicted in Figure 6a where T_i is labeled with the inputs and outputs of the addition gadget $\{a_i, b_i, c_i\}$ as the dependency graph of $\texttt{Add}$ (Fig. 5a). Further, due to the composition, the function f merges the edges related to the output of the addition with the edges related to the input of the refresh. For this reason, the edge labeled with T_i is also the edge that represents the input edge of the dependency graph of $\texttt{pRef}$ (Fig. 5b).

Additionally, we can also refresh the inputs of the addition. Let G be an addition or multiplication gadget with $(c_i)_{i\in[n]} \leftarrow \mathsf{G}((a_i)_{i\in[n]}, (b_i)_{i\in[n]})$ where $(a_i)_{i\in[n]}$ and $(b_i)_{i\in[n]}$ are refreshed outputs $(a_i)_{i\in[n]} \leftarrow \texttt{pRef}((x_i)_{i\in[n]})$, $(b_i)_{i\in[n]} \leftarrow \texttt{pRef}((y_i)_{i\in[n]})$, respectively, and $(c_i)_{i\in[n]}$ is refreshed afterwards $(z_i)_{i\in[n]} \leftarrow \texttt{pRef}((c_i)_{i\in[n]})$. This composition can be written as $\texttt{pRef} \circ \mathsf{G} \circ (\texttt{pRef} \| \texttt{pRef})$ because the refresh of $(a_i)_{i\in[n]}$ and $(b_i)_{i\in[n]}$ is a parallel composition, and the remaining ones are sequential. This results in a dependency graph

$$\mathsf{DG}_{\texttt{pRef}(\mathsf{G}(\texttt{pRef}(\cdot),\texttt{pRef}(\cdot)))} = \mathsf{DG}_{\texttt{pRef}} \cup_{f_1} \mathsf{DG}_{\mathsf{G}} \cup_{f_2} (\mathsf{DG}_{\texttt{pRef}} \cup \mathsf{DG}_{\texttt{pRef}})$$

depicted in Figure 6b with f_1 mapping the input edges $\mathsf{DG}_{\texttt{pRef}((c_i)_{i\in[n]})}$ to the output edges of DG_{G}, and f_2 input edges DG_{G} to the output edges $\mathsf{DG}_{\texttt{pRef}((x_i)_{i\in[n]})}$ and $\mathsf{DG}_{\texttt{pRef}((y_i)_{i\in[n]})}$. Note that T_i is determined by the choice of G, and in case of $\texttt{Add}$ it is $\{a_i, b_i, c_i\}$. Formally, the operation defined by $\mathsf{DG}_1 \cup_f \mathsf{DG}_2$ is a topological definition called *adjunction space* and can be formalized as follows.

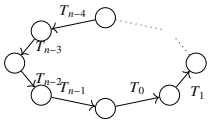
Definition 4 (Attaching)**.** Let DG_1 and DG_2 two disjoint dependency graphs, and f a function as described above mapping some edges of DG_1 to edges in DG_2. The composed graph is

$$\mathsf{DG}_1 \cup_f \mathsf{DG}_2 = (\mathsf{DG}_1 \cup \mathsf{DG}_2)/\sim,$$

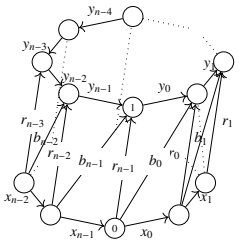
where $\sim$ is the smallest equivalence relation with $x \sim f(x)$.

The adjunction space of two dependency graphs preserves the properties of the underlying dependency graphs and merges them in such a way that the resulting graph describes the dependencies from both dependency graphs simultaneously. For the sake of simplicity, we can consider the composition as described above, where we merge the edges of two dependency graphs if the according wires become one due to the circuit composition. More formally, we will prove in Propositions 5, 6, 8, and Theorem 10 that the variables contained *only* in the labels of the edges of a sub-graph that does not contain a loop are independent of the inputs or outputs.

The key observation is that the dependency graphs for all our gadgets are either the loop depicted in Figure 5a or the skeleton of a cylinder, as shown in Figure 5. Further, the compiler places a refresh gadget between every gadget that is not a refresh gadget. This means that the resulting dependency graph can be seen as a composition of cylinders (defined by the refresh gadgets), where the bottom and the top of the cylinder are labeled with the T_i's defined of the gadgets between the refresh gadgets.

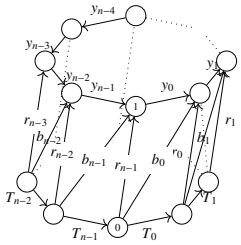


(a) Dependency graph of `Enc`, `Dec`, `Copy`, `Add`, or `Mult` with labels T_i defined in Section 3.1.

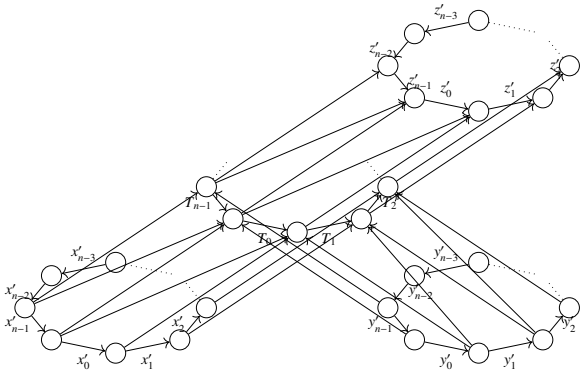


(b) The dependency graph for $(y_i)_{i\in[n]} \leftarrow \texttt{pRef}((x_i)_{i\in[n]})$.

Figure 5: Dependency graphs of the gadgets.



(a) Dependency graph of $\texttt{pRef} \circ \mathsf{G}$, with $(y_i)_{i\in[n]} \leftarrow \texttt{pRef}((x_i)_{i\in[n]})$ taking as input the output of G. The DG_{G} is described in Figure 5a while $\mathsf{DG}_{\texttt{pRef}}$ is described in Figure 5b. Note that the label T_i contains x_i.



(b) The dependency graph of the composition of $(x_i)_{i\in[n]} = \texttt{pRef}((x'_i)_{i\in[n]})$ with $(y_i)_{i\in[n]} = \texttt{pRef}((y'_i)_{i\in[n]})$, $(z_i)_{i\in[n]} = \mathsf{G}((x_i)_{i\in[n]}, (y_i)_{i\in[n]})$ (thus, G is either the `Add` or the `Mult` gadget) and $(z'_i)_{i\in[n]} = \texttt{pRef}((z_i)_{i\in[n]})$. (For simplicity, we have omitted the edges of b_is and r_is.)

Figure 6: Dependency graphs of composed gadgets.

4 Security Analyzes of the Gadgets

Before we analyze the privacy of our compiler's output in Section 5, we first give the privacy of our gadgets in this section. First, in Section 4.1, we formally show how to describe the leakage with sub-graphs of the gadgets' dependency graphs presented in the previous chapter (Sec. 3.1). Then, in Section 4.2, we give all leakages that are not shiftable using the sub-graphs of our dependency graph, and finally, we compute the probabilities of such sub-graphs under the condition that each wire leaks its value with probability p.

4.1 Leakage Diagram

Using Corollary 3, it is enough to show shiftability for the privacy proof. To characterize which outputs $\mathsf{Leak}(\widehat{\mathsf{C}}, \mathbf{x}, p)$ of the experiment in Definition 1 are shiftable, we start representing the leakage as a subgraph of the dependency graph. As already discussed in the random probing model, the adversary receives via leakage the values carried by some of the wires, $\mathcal{L}_p(\widehat{\mathsf{C}}) \subseteq \mathcal{W}'(\widehat{\mathsf{C}})$ (Def. 1). We can represent these variables as a subgraph of the dependency graph.

Definition 5 (Leakage diagram)**.** Let DG be the dependency graph of the circuit $\widehat{\mathsf{C}}$ and $\mathcal{L}_p(\widehat{\mathsf{C}})$ be the set of wires that leak in the experiment $\mathsf{Leak}(\widehat{\mathsf{C}}, \mathbf{x}, p)$.The *leakage diagram*, $\mathcal{LD}(\mathcal{L}_p(\widehat{\mathsf{C}}), \mathsf{C})$, corresponding to the leakage $\mathcal{L}_p(\widehat{\mathsf{C}})$ is the subgraph of DG composed by all edges whose label contains at least one of the variables carried by the wires in $\mathcal{L}_p(\widehat{\mathsf{C}})$.

Since $\mathcal{L}_p(\widehat{\mathsf{C}})$ is randomized by the leakage probability p, it is a random variable over all possible subsets of the wires $\mathcal{W}'(\widehat{\mathsf{C}})$ that may leak during the computation of $\widehat{\mathsf{C}}$. Hence, also $\mathcal{LD}(\mathcal{L}_p(\widehat{\mathsf{C}}), \widehat{\mathsf{C}})$ is a random variable over all possible sub-graphs of the dependency graph. Next, we give some examples of leakage diagrams LD_i with $\Pr[\mathcal{LD}(\mathcal{L}_p(\widehat{\mathsf{C}}), \widehat{\mathsf{C}}) = LD_i] > 0$. They are also represented in the full version. For this reason, we consider the $\mathtt{pRef}$ gadget refreshing an encoding $(x_i)_{i\in[n]} \leftarrow \mathtt{Enc}(x)$ of a secret $x \in \mathbb{F}$

$$(y_i)_{i\in[n]} \leftarrow \mathtt{pRef}((x_i)_{i\in[n]})$$

with random values r_i, and intermediate values $y_i = b_i - r_{i+1}$ and $b_i = x_i + r_i$ defined in Figure 1b.

1) $LD_1 = (x_0, ..., , x_{n-1})$ reveals the secret because $\mathtt{Dec}((x_i)_{i\in[n]}) = x$

2) $LD_2 = (y_0, ..., , y_{n-1})$ reveals the secret because $\mathtt{Dec}((y_i)_{i\in[n]}) = \mathtt{Dec}((x_i)_{i\in[n]}) = x$

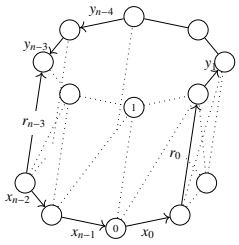
3) Let $LD_3 = (x_0, ..., x_{i-1}, r_{i-1}, y_i, r_i, x_{i+1}, ..., x_{n-1})$. Since $y_i = x_i + r_i - r_{i+1}$, we have that these values reveal x. In fact,
$$\left(\sum_{j\in[n], j\neq i} x_j\right) + r_i + y_i + r_{i+1} = \left(\sum_{j\in[n], j\neq i} x_j\right) + x_i = \mathtt{Dec}((x_i)_{i\in[n]}) = x.$$

4) Let $LD_4 = (x_0, ..., x_i, r_i, y_{i+1}, ..., y_{n-1})$. We observe that if instead of r_{n-1}, we have $r'_{n-1} = r_{n-1} + x' - x$, these values come from a refreshing of x'. In fact
$$\left(\sum_{j=0}^{i-1} x_j\right) + r_i + \left(\sum_{j=i}^{n-1} y_j\right) = \left(\sum_{j=0}^{n-1} x_j\right) - r_{n-1} = x - r_{n-1}$$
since $r_{j-1} + y_j = x_j + r_j$. Thus, LD_4 does not reveal x because the leakage is shiftable to x'.

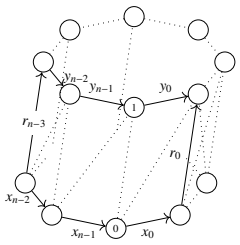
   Note that, $LD'_4 = LD_4 \cup \{r_{n-1}\}$ would reveal x similarly to B3).

5) Now, consider $LD_5 = \{x_0, ..., x_{i-1}, x_{i+1}, ..., x_{n-1}, y_0, ..., y_{i-1}, y_{i+1}, ..., y_{n-1}, r_0, ..., r_{n-1}\}$. LD_5 does not reveal x since $LD_5 \cup \{x_i\}$ comes from $\mathcal{L}(\mathtt{pRef}((x_i)_{i\in[n]}, \mathbf{r}), p)$, while $LD_5 \cup \{x_i + x' - x\}$ comes from $\mathcal{L}(\mathtt{pRef}((x'_i)_{i\in[n]}, \mathbf{r}), p)$, where $(x'_i)_{i\in[n]}$ is defined as $x'_j = x_j$ if $j \neq i$ and $x'_i = x + i + x' - x$, $\mathbf{r} = (r + 0, ..., r_{n-1})$. We prove this in Section 4.2, Proposition 6.

6) Finally, consider $LD_6 = (x_1, b_1, r_1)$, which clearly does not reveal the secret, as we will prove in the next section.



(a) Leakage diagram of $(y_i)_{i\in[n]} \leftarrow \texttt{pRef}((x_i)_{i\in[n]})$ with an orbiting loop.



(b) Leakage diagram of $(y_i)_{i\in[n]} \leftarrow \texttt{pRef}((x_i)_{i\in[n]})$ with a not orbiting loop.

Figure 7: Dependency graphs of the gadgets.

Considering the examples above, we observe that if the leakage diagram reveals information about the secret, the leakage diagram also consists of at least one loop. This is not surprising because this is exactly the property that we presented in Section 3, and immediately follows from the dependency graph property described in Section 3.1. However, the presence of a loop does not always prevent shiftability, e.g., LD_4, LD_5, ad LD_6 have a loop and are still shiftable. For tightness reasons, we want to distinguish loops that reveal the secret from loops that do not reveal the secret. For this reason, we remember that the dependency graphs of the gadgets (Fig. 5) are either a circle or the skeleton of a cylinder. Hence, its compositions are composed *hollow* cylinders, as depicted in Figure 6. We observe all loops revealing the secret *orbit* around this hollow cylinder structure. Further, the loops that do not reveal the secret do not orbit the hollow structure. Figure 7 illustrates the differences between such loops. The first loop is an example of a loop that does orbit the skeleton, while the second one gives an example of a loop that does not orbit the skeleton. In the following, we say that a leakage diagram *orbits* if it contains such an orbiting loop. Similarly to the adjunction space used to compose the dependency graphs, also the property of *hollow* and *orbit* are well-known in Topology and can be described with the topological definitions *simply connected* and *homotopically equivalent*: *Hollow* means that all the loops carrying the shares of a value are not *simply connected*, and *orbiting* means that the loop is *homotopically equivalent* to a loop containing all x_i-edges of a sharing $(x_i)_{i\in[n]}$. Roughly speaking, two paths are *homotopically equivalent* (Defined in the full version) if we can put an infinitely elastic rope over the first path, and we can slide it to cover the second path. For this, we assume that the DG of $\texttt{pRef}$ contains the lateral surface of the cylinder, and the rope, as in LD_3, can slide over it. With this, it is easy to see that LD_3 is homotopically equivalent to LD_1 and LD_2. Thus, the Topology allows us to distinguish between "bad" leakage diagrams from good, and all not orbiting leakage diagrams represent shiftable leakages. This intuition will be proved in the following subsection (Sec. 4.2). In the following, we will also call "good" diagrams *shiftable diagrams*.

4.2 Security Analysis for our Gadgets

To avoid that, we have to prove the security of any possible leakage diagram. So we start with a useful observation.

Proposition 1. *If a leakage diagram LD is shiftable, all sub graphs LD′ with LD′ ⊂ LD are also shiftable.*

Hence, when we prove shiftability for a given leakage diagram, it immediately follows shiftability for any subgraph.

Proof. A leakage diagram LD is shiftable if all the outcomes L of the $\mathsf{Leak}(\widehat{\mathsf{C}}, \mathbf{x}, p)$ which are represented by the leakage diagram LD are shiftable to $\mathbf{x'}$ for any $\mathbf{x'} \neq \mathbf{x}$. That is, (Def. 2) L can be also an outcome of the $\mathsf{Leak}(\widehat{\mathsf{C}}, \mathbf{x'}, p)$-experiment.

Now, consider the leakage diagram LD' with $LD' \subset LD$. Let L' be an outcome of the $\mathsf{Leak}(\widehat{\mathsf{C}}, \mathbf{x}, p)$, which can be represented by the leakage diagram LD'. Since $LD' \subset LD$, there exists an outcome L of the Leak experiment represented by LD s.t. $L_{|\text{var} \in LD'} = L'$, where with $var \in \mathcal{LD}'$, we mean the variables carried only by the labels of the edges in LD'. Since LD is shiftable, then L can be an outcome of the $\mathsf{Leak}(\widehat{\mathsf{C}}, \mathbf{x'}, p)$-experiment. Thus, $L' = L_{|\text{var} \in LD'}$ can be an outcome of the $\mathsf{Leak}(\widehat{\mathsf{C}}, \mathbf{x'}, p)$-experiment. Thus, all outcomes of the Leak-experiment represented by LD' are shiftable to $\mathbf{x'}$. Thus, the leakage diagram LD' is shiftable. □

For simplicity, we use this result and introduce the concept of *maximal diagram* $\mathcal{MAX}(LD)$ for any leakage diagram LD of our gadgets. $\mathcal{MAX}(LD)$ is a subgraph of the dependency graph, which contains LD

$$LD \subseteq \mathcal{MAX}(LD) \subset \mathsf{DG}$$

and is maximal in the sense that if we add any further edge, it orbits as depicted in Figure 7a. Due to Proposition 1 we only need to consider all possible maximal diagrams for the security proofs of our gadgets when we want to show that leakage diagrams are shiftable if they do not orbit. The existence of maximal diagrams is proved in the following Proposition.

Proposition 2. *Let LD be a leakage diagram that does not orbit. Then, there exists a maximal diagram LD′ containing it.*

Proof. Let LD be a subgraph of DG, which does not orbit. Suppose that $\forall$ edge $e \in \mathsf{DG} \setminus LD$, $LD \cup \{e\}$ orbits. Then, LD is a maximal simply connected subgraph containing LD.

Otherwise, there is an $e \in \mathsf{DG} \setminus LD$ s.t $LD^1 = LD \cup \{e\}$ that does not orbit. Then, we iterate with LD^1 until there is no such an e left. Since there is only a finite number of edges that can be added, this sequence of subgraphs must end after at most I step (with $I \leq |\{e \in \mathsf{DG} \setminus \mathcal{LD}\}|$). Hence, the final LD^j is a maximal diagram. □

In general, there are many different maximal diagrams, as we will see in the next section.

Maximal diagrams. Since we use the maximal diagrams in most of the following proofs, we characterize them for all of our gadgets. The dependency graphs of `Enc`, `Dec`, `Copy`, `Add`, `Mult` (Fig. 5a) are a circle with n edges, all subgraphs containing all edges except one are maximal, that is, we have n different maximal diagrams M_i defined as follows

$$\mathcal{M}_i = \{T_0, ...T_{i-1}, T_{i+1}, ..., T_{n-1}\} = \mathsf{DG} \setminus \{T_i\}$$

where with $T_j \in \mathcal{M}_i$, we mean that $\mathcal{M}_i$ contains the edge labeled with T_j as depicted in Figure 8b and 8b. We formally prove this in the following Proposition.

Proposition 3. *Let* DG *be the dependency graph of one of the* `Enc`, `Dec`, `Copy`, `Add`, *and* `Mult` *gadget. Then, there are n different maximals (Sec. 3.1)*

$$\mathcal{M}_i = \{T_0, ..., T_{n-1}\} = \mathsf{DG} \setminus \{T_i\}$$

for $i = 0, ..., n-1$.

Proof. $\mathcal{M}_i$ is a line starting from $node_{i+1}$, the starting node of the T_{i+1}-edge, to $node_i$, the arriving node of the T_{i-1}-edge (and the starting node of the T_i-edge). Thus, it is simply connected. Moreover, $\mathcal{M}_i \cup \{T_i\} = \mathsf{DG}$, where T_i is the only edge in DG and not in $\mathcal{M}_i$. Since DG orbits, $\mathcal{M}_i$ is maximal. In fact, any other subset LD that does not orbit and is composed of less than $n-1$ edges is a subset of one of these $\mathcal{M}_i$. Hence, there is no other maximal apart from the $\mathcal{M}_i$'s. □

Further, note that $\mathcal{MAX}(LD)$ is not always unique. As an example, suppose that $LD = \emptyset$, then, there are n different maximal simply connected subgraph $\mathcal{MAX}^i$ containing LD, with

$$\mathcal{MAX}^i = \{e_0, ..., e_{i-1}, e_{i+1}, ..., e_{n-1}\},$$

that is, $\mathcal{MAX}^i$ contains all edges except e_i. All the $\mathcal{MAX}^i$s contain LD, do not orbit, and, adding the only remaining edge (e_i), they orbit.

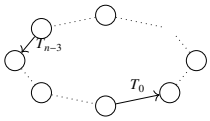
The dependency graph of `pRef` describes the skeleton of a cylinder (Fig. 5b). There are two families of maximal diagrams (both depicted in Figure 9): The first family is called $\mathcal{M}^{\text{right}}$ since there is a gap of missing edges turning right, which we call *RGap* (Fig. 9a).

$$\mathcal{M}^{\text{right}}_{i,j} := \{x_0, ..., x_{i-1}, x_{i+1}, ..., x_{n-1}, y_0, ..., y_{j-1}, y_{j+1}, ..., y_{n-1}, r_j, ..., r_{i-1}, b_j, ..., b_i\},$$

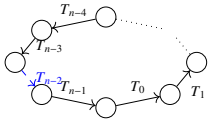
$$RGap_{i,j} = \{x_i, y_j, r_i, ..., r_{j-1}, b_{i+1}, ..., b_{j-1}\}$$

for $i, j \in [n]$ with $\mathcal{M}^{\text{right}}_{i,j} = \mathsf{DG} \setminus RGap_{i,j}$. Note that $r_j, ..., r_{i-1}$ is a short way to write: if $i < j$ $r_0, ...r_{i-1}, r_j, ..., r_{n-1}$; if $i = j$ $\emptyset$; if $i > j$ $r_{j+1}, ..., r_{i-1}$. The same holds for the $b_{i'}$s. The second family is called $\mathcal{M}^{\text{left}}$ since there is a gap, *LGap* (Fig. 9b), which turns left (or stays straight for $LGap_{i,i}$)

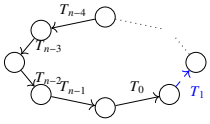
$$\mathcal{M}^{\text{left}}_{i,j} := \{x_0, ..., x_{i-1}, x_{i+1}, ..., x_{n-1}, y_0, ..., y_{j-1}, y_{j+1}, ..., y_{n-1}, r_i, ..., r_{j-1}, b_{i+1}, ..., b_{j-1}\},$$



(a) A possible leakage diagram LD only consisting of two edges (solid).



(b) Maximal leakage diagram $\mathcal{MAX}(LD) = \mathcal{M}_{n-2}$ (solid, black edges).



(c) Maximal leakage diagram $\mathcal{MAX}(LD) = \mathcal{M}_1$ (solid, black edges).

Figure 8: Two possible maximal leakage diagrams (Fig. 8b and 8b) of a possible leakage diagram LD (Fig. 8a). The blue dashed edge is the missing one so that the diagram does not orbit.

$$LGap_{i,j} = \{x_i, y_j, r_j, ..., r_{i-1}, b_j, ..., b_i\}$$

for $i, j \in [n]$ with $\mathcal{M}^{\text{left}}_{i,j} = \mathsf{DG} \setminus LGap_{i,j}$. The proof that these are maximal diagrams and that the classification is complete can be found in the following Proposition:

Proposition 4. *The maximal diagrams of* $\mathsf{DG}_{\mathtt{pRef}}$ *are either of the type* $\mathcal{M}^{right}_{i,j}$ *or* $\mathcal{M}^{left}_{i,j}$ *with* $i, j \in [n]$

Proof. In other words, if LD is a maximal diagram of $\mathsf{DG}_{\mathtt{pRef}}$, there are $i, j \in [n]$ s.t. either $LD = \mathcal{M}^{\text{right}}_{i,j}$ or $LD = \mathcal{M}^{\text{left}}_{i,j}$.

First, we start observing that if LD is a maximal diagram, then it must be connected. Otherwise, since it does not orbit, thus all connected components are simply connected (since the fundamental group of $\mathsf{DG}_{\mathtt{pRef}}$ is $\mathbb{Z}$). Thus, we can deform them homotopically to be points. Now adding a single edge to a set of points, it cannot make it not simply connected (and, thus, orbiting).

Second, in any maximal diagram, there exists one and only one $i \in [n]$ s.t. $x_i \notin LD$, and one and only one $j \in [n]$ s.t. $y_j \notin LD$. In fact, let us suppose that there exist two edges x_i and $x_{i'}$ not in LD. Let $node_i$ and the $node_{i'}$ be the source nodes of the edge x_i, and $x_{i'}$, respectively, $node_{i+1}$ and $node_{i'+1}$ be their respective destination nodes. Now, look at LD. Suppose that $i \neq i'+1, i'-1$. Since LD is maximal, there is a path $path$ from $node_{i+1}$ to $node_i$ s.t. $path \cup \{x_i$ orbits. Similarly, there is a path $path'$ from $node_{i'+1}$ to $node_{i'}$ s.t. $path' \cup \{x_{i'}$ orbits. For the structure of $\mathsf{DG}_{\mathtt{pRef}}$ $path$ and $path'$ meet in two points. Thus, taking a part of $path$ and a part $path'$, we have a path that orbits. Therefore LD is not maximal since it orbits. Now, we observe that if $x_i \notin LD$ then $\{b_i, r_i\} \not\subseteq LD$, otherwise, the path

$$x_0, ..., x_{i-1}, b_i, r_i, x_{i+1}, ..., x_{n-1}$$

is a loop in LD orbiting around the hollow graph, thus, LD would orbit. Thus, either $r_i \in LD$ or $b_i \in LD$. Similarly, if $y_j \notin LD$ then $\{b_j, r_j\} \not\subseteq LD$, otherwise, the path

$$y_0, ..., y_j, r_j, b_j, y_j, ..., y_{n-1}$$

is a loop in LD orbiting around the hollow graph. Here, we consider the case $i = j$. It depends if either b_i or r_i belongs to LD

- *Case* $r_i \in LD$. *(*$\mathcal{M}^{left}_{i,i}$*)* Since we cannot have both b_i and r_i in a maximal simply connected subgraph containing $\{x_i, ..., x_{i-1}, x_{i+1}, ..., x_{n-1}, y_0, ..., y_{i-1}, y_{i+1}, ..., y_{n-1}\}$, the natural maximal diagram is

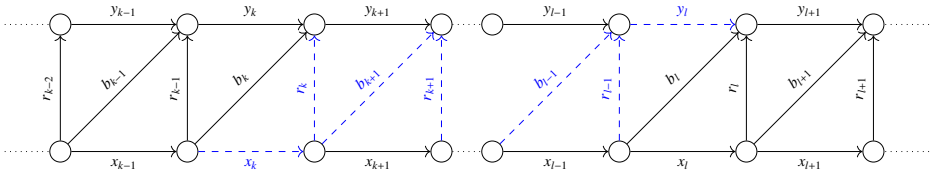
$$LD' = \{x_0, ..., x_{i-1}, x_{i+1}, ..., x_{n-1}, y_0, ..., y_{i-1}, y_{i+1}, ..., y_{n-1}, r_0, ..., r_{n-1}, b_0, ..., b_{i-1}, b_{i+1}, ..., b_{n-1}\}.$$

  We observe that it does not orbit since there is no loop turning around the hollow graph since the square defined by x_i, r_i, y_i and r_{i-1} is never crossed. Moreover, if we add only the edge in $\mathsf{DG} \setminus LD$, that is, b_i, we have a not simply connected subgraph since it is DG. Thus, LD' is a maximal simply connected subgraph.
- *Case* $b_i \in LD$. *(Case* $\mathcal{M}^{right}_{i,i}$*)* We observe that

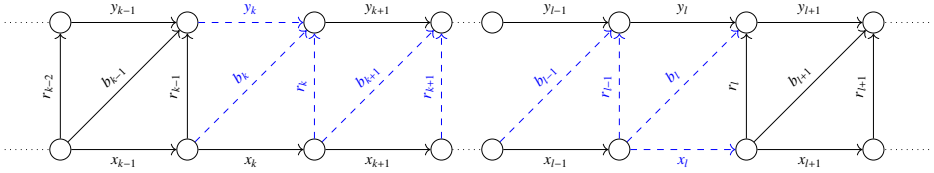
$$LD' := \{x_0, ..., x_{i-1}, x_{i+1}, ..., x_{n-1}, y_0, ..., y_{i-1}, y_{i+1}, ..., y_{n-1}, b_i\}$$

  does not orbit since there is no non-trivial loop. In fact,

$$x_{i+1}, ..., x_{n-1}, x_0, ..., x_{i-1}, b_i, y_{i+1}, ..., y_{n-1}, y_0, ..., y_{i-1}$$



(a) Edges in $RGap_{k,l}$ (dashed, blue) of the dependency graph of $(y_i)_{i\in[n]} \leftarrow \texttt{pRef}((x_i)_{i\in[n]})$ experiment.



(b) Edges in $LGap_{k,l}$ (dashed, blue) of the dependency graph of $(y_i)_{i\in[n]} \leftarrow \texttt{pRef}((x_i)_{i\in[n]})$ experiment.

Figure 9: Example of maximal leakage diagrams of `pRef`. The blue dashed edges represent $RGap_{k,l}$ and $LGap_{k,l}$, respectively. The remaining solid edges are the maximal leakage diagrams $\mathcal{M}^{\text{right}}_{k,l} = \mathsf{DG} \setminus RGap_{k,l}$ and $\mathcal{M}^{\text{left}}_{k,l} = \mathsf{DG} \setminus LGap_{k,l}$.

is a line which is not a loop.
Moreover, if we add r_l to LD', we have that

$$y_0, ..., y_l, r_l, x_{l+1}, ..., x_{i-1}, b_i, y_{i+1}, ..., y_{n-1} \quad \text{for } l < i, \quad \text{or}$$

$$x_0, ..., x_{i-1}, b_i, y_{i+1}, ..., y_l, r_l, x_{l+1}, ..., x_{n+1}, ..., x_{n-1} \quad \text{for } l > i, \quad \text{or}$$

is a loop in $LD' \cup \{r_l\}$ which orbits [5]. Similarly, if we add b_l to LD, we have that

$$y_0, ..., y_l, b_l, x_l, ..., x_{i-1}, b_i, y_{i+1}, ..., y_{n-1} \quad \text{for } l < i, \quad \text{or}$$

$$x_0, ..., x_{i-1}, b_i, y_{i+1}, ..., y_l, b_l, x_l, ..., x_{n+1}, ..., x_{n-1} \quad \text{for } l > i$$

is a loop in $LD' \cup \{b_l\}$ which orbits. Thus LD' is a maximal simply connected subgraph.
The other cases (which are done in a similar way) can be found in the full version. □

Using the maximal diagrams, we can prove the security of our gadgets with the help of Proposition 1.

Security for `Enc`, `Dec`, `Copy`, `Add`, `Mult`. Now, for all gadgets except `pRef`, we prove the condition mentioned in Section 4.1. In other words, we prove any leakage diagram that does not orbit implies that the leakage is shiftable to another input. Formally,

Proposition 5. *Let* G *be the gadget* `Add`, `Mult`, *or* `Copy` *defined in Figure 3. An outcome L of the* $\mathsf{Leak}(\mathsf{G}, \boldsymbol{x}, p)$ *experiment is shiftable to any* $\boldsymbol{x}$**'**, *if the leakage diagram corresponding to L, does not orbit the dependency graph.*

We first give a high-level proof idea.

Proof sketch. The proof is substantially the same for all gadgets: we consider the maximal $\mathcal{M}_i$ and the values of the variables it contains, and we show that we can modify the values carried by the T_i label. Hence we can shift the i^{th} share of each input and output encoding such, and prove that this modification does not change the distribution of the values in M_i. Hence, the distribution of the values in M_i is the same for $\mathsf{Leak}(\mathsf{G}, \mathbf{x}, p)$ experiment and the shifted one $\mathsf{Leak}(\mathsf{G}, \mathbf{x}', p)$. □

Next, we give the formal proof.

Proof. We prove the claim for all maximal graphs. Thus, using Proposition 1, we have that the claim holds for all not orbiting leakage diagrams. The high-level idea is that we can shift change the inputs and outputs of the edge that are not part of the maximal graph because they are uniformly distributed and not revealed due to the leakage. We start with the simplest case, (i) the masking of `Enc`, `Dec` and dependency graphs of `Copy`, then we prove the claim for (ii) `Add`, and (iii) `Mult`.

[5] The inequalities regarding l, i as all the inequalities in the remaining in the proof are done considering $l, i \in \mathbb{Z}$ (and not in $\mathbb{Z}_n$)].

(i) The masking of Enc, Dec and dependency graphs of Copy since the edges of the dependency graph are labeled with a single variable. Thus, the input is x, and the variables leaked are those in the labels of the edges of $\mathcal{M}_i$ for an $i \in [n]$. $\mathcal{M}_i$ contains all x_j with $j \neq i$. If we modify the not leaked value x_i to $x'_i = x_i + x' - x$, we have shifted the output $\mathsf{Leak}(\mathsf{G}, x, p)$ to the output of the experiment $\mathsf{Leak}(\mathsf{G}, x', p)$, since $x_0, ..., x_{i-1}, x'_i, x_{i+1}, ..., x_{n-1}$ is an encoding of x'. Hence, $\mathcal{M}_i$ has the same distribution for x and x', and therefore $\mathcal{M}_i$ is shiftable for any $i \in [n]$.

(ii) The dependency graph's edges of Add are labeled with multiple variables. Let us suppose that Add takes as input $(a_i)_{i\in[n]}$ and $(b_i)_{i\in[n]}$, and outputs $(c_i)_{i\in[n]}$, then each edge is labeled with $T_i = \{a_i, b_i, c_i\}$. Thus, the leaked variable of the maximal graph $\mathcal{M}_i$ are all a_j, b_j, and c_j with $j \neq i$. If we modify the values a_i, b_i, and c_i which are not leaked in $a'_i = a_i + a' - a$, $b'_i = b_i + b' - b$, and $c'_i = c_i + a' + b' - a - b$, we have shifted the output $\mathsf{Leak}(\mathtt{Add}, (a, b), p)$ to the output $\mathsf{Leak}(\mathtt{Add}, (a', b'), p)$. This hold because $d_0, ..., d_{i-1}, d'_i, d_{i+1}, ..., d_{n-1}$ is an encoding of d for $d = a, b, c$, and $a_j + b_j = c_j \ \forall j \neq i$, and $a'_i + b'_i = c'_i$. Note that the input sharings are uniformly distributed, and therefore the modified shares are also uniformly distributed.

(iii) Mult: let us suppose that Mult takes as input $(a_i)_{i\in[n]}$ and $(b_i)_{i\in[n]}$, and outputs $(c_i)_{i\in[n]}$. Thus, the variables leaked are those *only* in the labels of the edges of $\mathcal{M}_i$ (Def. 5), that is, $a_{i'}, b_{i'}, c_{i'}, a_{i'}b_{i'}, a_{i'}b_j, a_jb_{i'}, w_{i',j}, w_{j,i'}, z_{l,i',j}$ for $i' \neq i$, and $j \in [n]$ and $l \in [0, ..., L]$ with $L = \log_2(n+1)$. Let $\delta_1 = a' - a$ and $\delta_2 = b' - b$. We replace the values of the variable in M_i as follows: $a'_i = a' + \delta_1$, $b'_i = b_i + \delta_2$, $(a_ib_i)' := a_ib_i + \delta_1 b_i + \delta_2 a_i + \delta_1\delta_2$, $(a_ib_j)' := a_ib_j + \delta_1 b_j$, $(a_jb_i)' := a_jb_i + a_j\delta_2$, $(w_{i,j})' := w_{i,j} + \delta_1 b_j$, and $(w_{j,i})' := w_{j,i} + \delta_2 a_j$. For c'_i and $z_{l,i,j}$, since they are the sum of many $w_{i,j}$s and $a_{j'}b_i$, the modification is done according to the previous modifications. This can be seen as an output of the $\mathsf{Leak}(\mathtt{Mult}, (a', b'), p)$ as we now prove, modifying the proof of [16]:

In [16] is the proof for all values except for the $z_{l,i,j}$s. The only difference from our Mult from that in [16], is that the final addition is done sequentially thew (Figure 3e, while we do it parallel). But, we can observe that $\forall l = 1, ..., \log(n)$ and $\forall i, j$

$$z_{l,i,j} = a_ib_i + \sum_{I=2^l j}^{2^l(j+1)-1} z_{0,i,I} = a_ib_i + \sum_{I=0}^{2^l(j+1)-1} z_{0,i,I} - \sum_{I=0}^{2^l j-1} z_{0,i,I}$$

Hence, all $\sum_{I=0}^{2^l(j+1)-1} z_{0,i,I}$, $\sum^{2^l(j+1)-1} z_{0,i,j} - \sum^{2^l(j+1)-1} z_{0,i,j}$ are partial sums of the serial sum of the multiplication of Dziembowski et al. [16], and all these values are mapped on the ith edge. Thus, since their modifications cannot be detected, ours cannot be detected too.

In this way, we have shifted the output of the $\mathsf{Leak}(\mathtt{Mult}, (a, b), p)$ to an output of the experiment $\mathsf{Leak}(\mathtt{Mult}, (a', b'), p)$.

The results of (i),(ii), and (iii) conclude the proof. □

The proposition gives us the following privacy result for our gadgets.

Theorem 1. *Let* Add, Mult *and* Copy *be the gadgets defined in Figure 3. Then,* Add *and* Copy *are* $(p, (3p)^n)$*-private, for* $p \leq 1/3$*, and* Mult *is* $(p, \tilde{p})$*-private with*

$$\tilde{p} = 1 - (1-p)^{8n} + 1 - (1 - \sqrt{3p})^{n-1} \leq 2(1 - (1 - \sqrt{3p})^{8n}).$$

Proof. Using Proposition 5 and the classification of maximal diagrams, the only problems happen if all edges are leaked. Thus, these gadgets are secure if the leakage diagram does not contain all edges. In other words, they are secure with probability $1 - \Pr[LD = \{T_0, ..., T_n\}]$ and thus, they are

$$(p, \Pr[LD = \{T_0, ..., T_n\}])\text{-private.}$$

We first bound $\Pr[LD = \{T_0, ..., T_n\}]$ for (i) Add and Copy, and then for (ii) Mult.

(i) For Add and Copy, all edges are added independently with probability $1 - (1-p)^3 \leq 3p$ since there are three variables mapped to each edge: For Add there are 3 variables on the label $T_i = \{a_i, b_i, c_i\}$ (Section 3.1); For Copy there are 3 variables on the label $T_i = \{a_i, b_i, c_i\}$ (Section 3.1) with $a_i = b_i = c_i$. In both cases, each variable is carried by a single edge. Thus, the probability that all edges belong to the leakage diagram is $1 - (1-3p)^n \leq (3p)^n$ which concludes the proof.

(ii) For `Mult`, as in [16], we can prove that if we add each edge independently with probability $p' = 2(1 - (1 - \sqrt{3p})^{8n})$, the leakage diagrams obtained in this way contain the leakage diagrams obtained in the $\mathsf{Leak}(\texttt{Mult}, (x, y), p)$-experiment.

We prove this fact by doing a proof very similar to the one in [16] (which is inspired by the original proof in [21]). The two differences are the fact that our `Mult` is slightly different from theirs and, more substantially, a different analytical treatment of the bound. We start observing that there are at most $8n$ wires which carry at least one variable present in the label T_i of the T_i-edge. a_i is used in n multiplication. Thus, there are n wires carrying it. Similarly, b_i is carried by n wires. There are $2n$ different $z_{l,i,j}$ for $l > 1$. There n different $a_i b_j$ and $a_j b_i$ (for $j \in [n]$). Finally, there are at most n between $z_{0,i,j}$ and $w_{i,j}$ wires and at n different $w_{j,i}$ wires.

Thus, we add the edge T_i with probability at most $1 - (1 - p)^{8n}$ and the edges T_i and T_j with probability at most $1 - 3p$.

It is easier to work with independent variables, but in the previous situation, the edges T_is are not independently added to LD. Thus, we try to add all the T_i edges to LD independently. To do this, we add T_i with the probability

$$1 - (1 - p)^{8n} + 1 - (1 - \sqrt{3p})^{n-1}.$$

The proof is the same as in [16], where we have not used the approximation $1 - (1 - p)^{8n} \leq 8np$. A detailed discussion is given in the full version.

Thus, the probability that all edges belong to LD is bounded by $[2(1 - (1 - \sqrt{3p})^{8n})]^n$.

This proves the claim of the theorem. □

Note that we do not use the approximation $2(1 - (1 - \sqrt{3p})^{8n}) \leq 16n\sqrt{3p}$ because it is not tight and makes the security much worse.

Security for `pRef`. With the same technique, we can analyze `pRef`. The only difference is the more complex dependency graph since it forms the skeleton of a cylinder and not a simple loop. Formally:

Proposition 6. *An outcome L of the $\mathsf{Leak}(\texttt{pRef}, x, p)$ experiment, is shiftable to x', if the corresponding leakage diagram, does not orbit.*

Proof. Again due to Proposition 1, we only prove the claim for maximal diagrams. We have two types of maximal diagrams $\mathcal{M}^{\text{right}}_{i,j}$ and $\mathcal{M}^{\text{left}}_{i,j}$. For the proof, we show how to shift/modify the variables in (i) $RGap_{i,j} = \mathsf{DG} \setminus \mathcal{M}^{\text{right}}_{i,j}$ and (ii) $LGap_{i,j} = \mathsf{DG} \setminus \mathcal{M}^{\text{left}}_{i,j}$. It consists of x_i, y_j, and some b_ls and r_ls.

(i) $\mathcal{M}^{\text{right}}_{i,j}$: we shift an output of $\mathsf{Leak}(\texttt{pRef}, x, p)$ to an output of $\mathsf{Leak}(\texttt{pRef}, x', p)$ as follows: x_i is modified in $x'_i = x_i + x' - x$, y_j is modified in $y'_j = x_j + x' - x$, r_l is modified in $r'_l = r_l - (x' - x)$, for the r_ls in $RGap_{i,j}$, and b_l is modified in $b'_l = b_l - (x' - x)$, for the b_ls in $RGap_{i,j}$. Note that the modified shares and intermediate values are all uniformly random, and the modification is done in such a way that they are still uniformly random and the intermediate dependencies are consistent.

(ii) With the same technique, we can modify $\mathcal{M}^{\text{left}}_{i,j}$: we shift an output of $\mathsf{Leak}(\texttt{pRef}, x, p)$ to an output of $\mathsf{Leak}(\texttt{pRef}, x', p)$ as follows: x_i is modified in $x'_i = x_i + x' - x$, y_j is modified in $y'_j = x_j + x' - x$, r_l is modified in $r'_l = r_l + (x' - x)$, for the r_ls in $LGap_{i,j}$, and b_l is modified in $b'_l = b_l + (x' - x)$, for the b_ls in $LGap_{i,j}$.

We have to prove that every shift results in an outcome of the $\mathsf{Leak}(\texttt{pRef}, x', p)$ experiment. We prove one case and refer to the full version for the other cases.

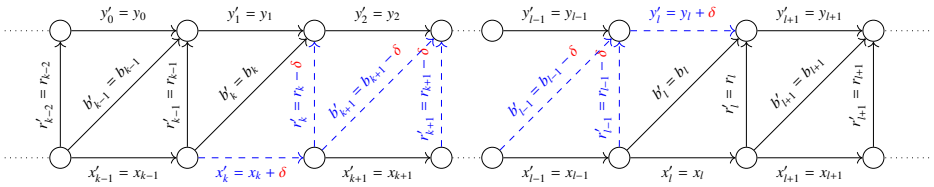
Let $\delta = x' - x$. Let A be the values carried by the variables of `pRef` during the $\mathsf{Leak}(\texttt{pRef}, x, \mathbf{r}, p)$-experiment, where $\mathbf{r} = r_0, ..., r_{n-1}$ is the randomness used.

The maximal is of type $\mathcal{M}^{right}_{i,i}$. The values carried in A' are

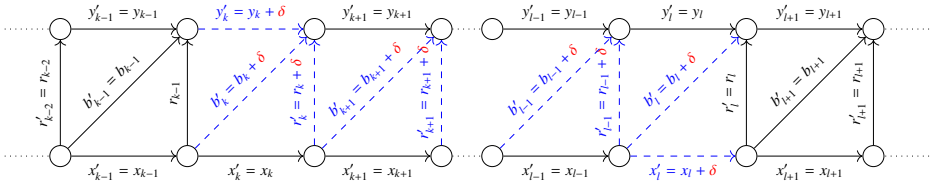
$$\{x_0, ..., x_{i-1}, x_i + \delta, x_{i+1}, ..., x_{n-1}, y_0, ..., y_{j-1}, y_i + \delta, y_{i+1}, ..., y_{n-1},$$
$$r_0 - \delta, ..., r_{n-1} - \delta, b_0 - \delta, ..., b_{i-1} - \delta, b_i, b_{i+1} - \delta, ..., b_{n-1} - \delta\}.$$

First, we observe that

$$x_0 + ... + x_{i-1} + (x_i + \delta) + x_{i+1} + ... + x_{n-1} = x_0 + ... + x_{n-1} + \delta = x + (x' - x) = x'.$$



(a) Values of the variables in $RGap_{k,l}$ in the $\mathsf{Leak}(\mathtt{pRef}, x', p)$ experiment. $\delta = x' - x$.



(b) Values of the variables in $LGap_{k,l}$ in the $\mathsf{Leak}(\mathtt{pRef}, x', p)$ experiment. $\delta = x' - x$.

Figure 10: Example of the shifting depicted in Proposition 5. The solid edges are in $\mathcal{LD}$. The maximal is of type $\mathcal{M}^{\text{right}}_{k,l}$ and $RGap_{k,l} = \mathsf{DG} \setminus \mathcal{M}^{\text{right}}_{k,l}$ (10a), and $\mathcal{M}^{\text{left}}_{k,l}$ and $LGap_{k,l} = \mathsf{DG} \setminus \mathcal{M}^{\text{left}}_{k,l}$ (10b).

The same holds for the shares of y.
Now, let us consider pRef with input $x_0, ..., x_{i-1}, x_i + \delta, x_{i+1}, ..., x_{n-1}$ and randomness $r_0 - \delta, ..., r_{n-1} - \delta$. For $i' \neq i$, $b_{i'} - \delta = x_{i'} + (r_{i'} - \delta)$ and $y_{i'} = (b_{i'} - \delta) - (r_{i'-1} - \delta)$. In stead, for i, $b_i = (x_i + \delta) + (r_i - \delta)$, and $y_i + \delta = (b_i) - (r_{i-1} - \delta)$. Thus, the values carried by A' are those of an honest evaluation of pRef with input $x_0, ..., x_{i-1}, x_i + \delta, x_{i+1}, ..., x_{n-1}$ and randomness $r_0 - \delta, ..., r_{n-1} - \delta$.

□

As already discussed before (Section 3.1), the security of the Const and Rand gadgets is given by the security of the gadgets where their output wires are used. This gives us the following privacy result for pRef.

Theorem 2. *Let* pRef *be the refreshing gadget defined in Figure 1b. Then,* pRef *is* $(p, \tilde{p})$*)-private, with* $\tilde{p} = 2 \cdot [(1 + 2(3p))3p]^n$. *If* $p \leq 1/6$ *we can approximate* $\tilde{p} \leq 2 \cdot (6p)^n$.

Proof. As in the security proof of the other gadgets, with Corollary 3 and Proposition 6, we only need to compute Pr[LD orbits] because it holds

$$\mathtt{pRef} \text{ is } (p, \Pr[LD \text{ orbits}])\text{-private.}$$

First, we observe that every edge of $\mathsf{DG}(\mathtt{pRef})$ (Figure 5b) has a label containing a single variable. We refer to Figure 5b for the notations. All variables are carried by a single wire except the r_i's, which are carried by three wires (Figure 1b). Thus, we can assume that each edge is added to LD independently with probability at most $1 - (1 - p)^3 \leq 3p$. With the result of Proposition 6, we know that the circuit is private if there is no loop that orbit. Moreover, any orbit must contain at least one of the source nodes of the edges x_0 or y_0. Next, we approximate the probability that in LD there is an orbit containing the source node of x_0 called $node^0_0$. (event E_0). Since such an orbit is at least n edges-long, we can upper-bound Pr[E_0] with the probability that the leakage diagram contains a path starting from $node_0$ with n edges. Since each node is connected with at most 4 other nodes, we have 3^n different n long paths[6] and the probability for each path is $(3p)^n$. This results in

$$\Pr[E_0] \leq (9p)^n.$$

But, we can refine the bound of E_0 as follows. We observe that we are trying to bound the probability of an orbiting path starting from $node^0_0$. Before, we observed that every path between these two nodes is at least n edges long. However, it is easy to see that there is only one n-edges long path that describes a path from $node^0_0$ and arriving to it, $2(n-1)$ paths that are $n+1$-edges long, and so on. We observe that given a node y, the paths from y to $node^0_0$ which have the minimal length[7] keeping the direction of the orbit [8], takes as the first edge from y at most two different edges.

[6] We omit paths where an edge is crossed more than once. In fact, this possibility does not give anything to the adversary.
[7] minimal means with the least number of edges.
[8] That is, making the loop $node^0_0, ..., y, node^0_0$ orbit.

To prove this, it is enough to observe that if the node y is on the x_i^0-edges orbits, thus, it is node $node_j^0$ for a $j \in [n]$, the source node of the x_j^0-edge. Thus, if the path to $node_0^0$ keeping the direction of the orbit, has to turn left, the shortest path consists of the path $x_{j-1}^0, ..., x_0^0$; otherwise, if it turns right, the shortest path is taking the path $x_j^0, ..., x_{n-1}^0$. They are the shortest path because every time a path leaves the x_i^0 orbits, it needs an additional edge to rejoin it. Moreover, it is not possible to have paths that skip the ith edge (that is, do not contain any edge indexed with i) for $i = 0, ..., j-1$ if it turns left, or with $i = j, ..., n-1$ if it turns right.

Instead, if y is on the x_i^1-edges orbits, thus, it is node $node_j^1$ for a $j \in [n]$, the source node of the x_j^1-edge. Thus, if the path to $node_0^0$ keeping the direction of the orbit has to turn right, it must take either the edge x_j^1 or r_{j-1}. In fact, the first path needs $1 + n - j$ edges (see above), while the shortest paths using the-x_j^1 can take as well at most $1 + n - j$ edges. The proof that the shortest path needs $1 + n - j$ edges can be easily done by induction. We iterate the previous argument until we arrive at $node_0^1$ for which the shortest path to $node_0^0$ is clearly r_{n-1}.

Finally, if the path to $node_0^0$ keeping the direction of the orbit has to turn left, it must take either the edge b_{j-1} or x_{j-1}^1 (for $j = 1$, it must take b_0). In fact, the first path needs $j - 1$ edges (see above, the case for y as node $node_j^0$), while the shortest paths using the-x_{j-1}^1 can take as well at most $j - 1$ edges. The proof that the shortest path needs $j - 1$ edges can be easily done by induction. We iterate the previous argument until we arrive at $node_1^1$ for which the shortest path to $node_0^0$ is clearly b_0. For $j = 0$, since we have to turn left (the case where you have to go down via r_{n-1} has already been treated in turning right, we must take x_{n-1}^1 or b_{n-1}.

Finally, we have to explain why

$$\Pr[E_0] \leq [(1 + 2(3p))3p]^n.$$

This happens because if we consider a loop starting from $node_0^0$, which orbits, and we do not take the shortest path, which is $x_0^0, ..., x_{n-1}^0$, and we take the e-th edge to deviate. In this case, we need to take one of the other two edges connected to the node we have arrived after we have taken the e-edge. But, this happens with probability at most $3p$. From there, we need at most $n - i$ edges to end our loop, where is the number of edges taken on the $x_0^0, ..., x_{n-1}^0$ before deviating via the e-edge.

Thus, if a path from y to $node_0^0$ does not pass through one of these two edges, it is at least $l + 1$ edges long. Thus, the probability that it belongs to LD is at most $(3p)^{l+1}$. Thus, we can upper-bound

$$\Pr[E_0] \leq [(1 + 2(3p))3p]^n.$$

Further, if $6p < 1$ it holds

$$[(1 + 2(3p))3p]^n \leq (6p)^n.$$

Clearly, the same holds for E_1, which is the event that there is an orbiting loop starting from the source node of the edge y_0. Thus, $\Pr[LD \text{ orbits}] \leq \Pr[E_0] + \Pr[E_1] = 2[(1 + 2(3p))3p]^n$ for $p \leq 1/3$ and $2(6p)^n$ for $p \leq 1/6$. □

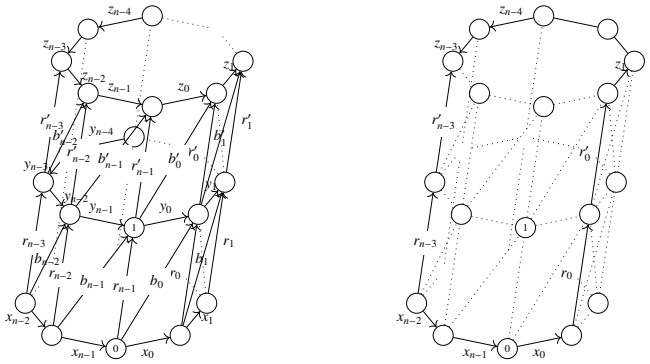
With Theorem 1 and 2, we have proven the security of the gadgets used by our compiler. However, it is well known that it is insufficient to prove their compositions' security. Next, we examine the security of compositions to argue about the security of the compiler used in the paper.

5 Security Analyzes for Circuits

In Section 4, we have investigated the privacy of the gadgets used by our compiler. In this section, we analyze the security of their compositions. Hence, we compute the privacy of our compiler's output. We start with the composition of `pRef` gadgets (Sec. 5.1), and then, we give a more general composition result for all gadgets used by our compiler (Sec.5.2)

5.1 Security Analysis for the Composition of `pRef`

In this section, we consider k sequential compositions of `pRef` gadgets (k`pRef`). Let $(x_i^0)_{i\in[n]}$ be the input of the first refresh gadget, then the j^{th} `pRef`, denoted with $\texttt{pRef}^j$ computes $(x_i^j)_{i\in[n]} \leftarrow \texttt{pRef}^j((x_i^{j-1})_{i\in[n]})$. Further, the internal variables used by $\texttt{pRef}^j$ are defined as r_i^j and b_i^j. Thus, the final output is $(x_i^k)_{i\in[n]} \leftarrow k\texttt{pRef}((x_i^0)_{i\in[n]})$. In Figure 11a, we depicted the dependency graph of two composed refresh gadgets. Compared to the more general compositions with multiple input and output sharings, as depicted in Figure 6, the dependency graph of the composed refresh gadgets is relatively simple. More precisely, it is still the skeleton of a cylinder. This cylinder is given by putting the



(a) The dependency graph for $(z_i)_{i\in[n]} \leftarrow \texttt{pRef}(\texttt{pRef}((x_i)_{i\in[n]}))$.

(b) The dependency graph for $(z_i)_{i\in[n]} \leftarrow \texttt{pRef}(\texttt{pRef}((x_i)_{i\in[n]}))$.

Figure 11: Dependency graphs of two composed refresh gadgets.

cylinder of the dependency graph of $\texttt{pRef}^j$ (denoted with DG^j) over the cylinder of $\texttt{pRef}^{j-1}$ (denoted with DG^{j-1}). In other words, the only difference with the security proof of the single refresh gadget is the length of the cylinder. The dependency graph of the composition of k refresh gadgets is a cylinder constructed out of the k cylinders (DG^j) of the dependency graph of pRef. As mentioned in the previous section, it is not sufficient to analyze the security of each gadget. This is easy to see when we remember that we have shown in Section 4.2 that pRef is secure if the leakage diagram does not orbit the cylinder. Here, it might be the case that the leakage diagram of its composition orbits the cylinder, but if we consider each leakage diagram of the composed refresh gadgets separately, it does not orbit each subcylinder. For example, Figure 11b illustrates such a case. Next, we use the same technique as in Section 4.2 to prove the security of the composition.

First, we classify the maximal diagrams of kpRef. One way to describe the gap in the cylinder is to use the multiple subgraphs *RGap* and *LGap* as defined in the proof of pRef and compose them in such a way that they describe such a gap from the bottom to the top of the cylinder. However, such a gap can also have some detours such that the gap goes partly in the opposite direction. To cover all such cases, we have to consider four more "gap sets" than *RGap* and *LGap*:

(i) $BLGap_{i,i'} := \{x_i^{j-1}, x_{i'}^{j-1}, r_{i'}^j, ..., r_{i-1}^j, b_{i'+1}^j, ..., b_i^j\}$ describes a gap that starts from the bottom circle of DG^j, then turns left, and, finally, ends in the bottom circle.

(ii) $BRGap_{i,i'} := \{x_i^{j-1}, x_{i'}^{j-1}, r_i^j, ..., r_{i'-1}^j, b_{i+1}^j, ..., b_{i'}^j\}$ describes a gap that starts from the bottom circle of DG^j, then turns right, and, finally, it ends in the bottom circle.

(iii) $ULGap_{i,i'} := \{x_i^j, x_{i'}^j, r_{i'}^j, ..., r_{i-1}^j, b_{i'}^j, ..., b_{i-1}^j\}$ describes a gap that starts from the top circle of DG^j, then turns left, and, finally, it ends in the top circle.

(iv) $URGap_{i,i'} := \{x_i^j, x_{i'}^j, r_i^j, ..., r_{i'-1}^j, b_i^j, ..., b_{i'-1}^j\}$ describes a gap that starts from the top circle of DG^j, then turns right, and, finally, it ends in the top circle.

We can use these gaps to classify maximal diagrams for kpRef. The next claim describes the structure of such maximal graphs if we consider each sub-graph of each pRef separately.

Proposition 7. *Let $(x_i^0)_{i\in[n]}$ be the input of k*pRef *and $(x_i^j)_{i\in[n]} \leftarrow \texttt{pRef}((x_i^{j-1})_{i\in[n]})$ the jth* pRef *of k*pRef*. Further, let DG^j be the sub-graph of $\mathsf{DG}_{\texttt{pRef}}$ containing the variables used in $\texttt{pRef}^j$ and $\mathcal{MAX}$ a maximal diagram of k*pRef*. It holds*

$$\mathcal{MAX} \cap \mathsf{DG}^j = \mathsf{DG}^j \setminus \left[\left(\bigcup_{I\in\mathcal{I}_1} RGap_{i_I,i'_I}\right) \cup \left(\bigcup_{I\in\mathcal{I}_2} LGap_{i_I,i'_I}\right) \cup \left(\bigcup_{I\in\mathcal{I}_3} BLGap_{i_I,i'_I}\right) \cup \left(\bigcup_{I\in\mathcal{I}_4} BRGap_{i_I,i'_I}\right) \cup \left(\bigcup_{I\in\mathcal{I}_5} ULGap_{i_I,i'_I}\right) \cup \left(\bigcup_{I\in\mathcal{I}_6} URGap_{i_I,i'_I}\right)\right]$$

For all $LD^j := \mathcal{MAX} \cap \mathsf{DG}^j$, we get $|\mathcal{I}_1| + |\mathcal{I}_2|$ is odd and all gaps are pairwise disjoint. Moreover, for LD^1 and LD^k it holds $|\mathcal{I}_1| + |\mathcal{I}_2| = 1$, for LD^1 we get $|\mathcal{I}_3| = |\mathcal{I}_4| = 0$, while for LD^k we get $|\mathcal{I}_5| = |\mathcal{I}_6| = 0$. Each gap RGap

and LGap of DG^j *is connected to one and only one gap of* DG^{j+1}*, and to one and only one gap of* DG^{j-1}*. Each gap BLGap and BRGap is connected to two distinct gaps of* DG^{j-1}*, and each gap ULGap and URGap is connected to two distinct gaps of* DG^{j+1}*.*

Proof sketch. Even when the claim looks relatively complex, it only formalizes how the maximal graph $\mathcal{MAX}$ of kpRef looks when we cut it into the sub-graphs LD^j describing the individual $(x_i^j)_{i\in[n]} \leftarrow \mathtt{pRef}((x_i^{j-1})_{i\in[n]})$. The proof idea is that the composition of all LD^j together still describes a gap that avoids a path around the dependency graph of kpRef. In other words, we can see it as construction where we can construct and compose such LD^j so that there is still a gap, and each further added edge would close the gap. Next, we give the formalized proof for this claim.

Proof. First, we prove that all the gaps are connected as described, that is, each gap *RGap* and *LGap* of DG^j is connected to at least one gap of DG^{j+1}, and at least one gap of DG^{j-1}; each gap *BLGap* and *BRGap* is connected to two distinct gaps of DG^{j-1}; and each gap *ULGap* and *URGap* is connected to two distinct gaps of DG^{j+1}. If this is not the case, we simply consider the perimeter of the gap, which is not connected. It is easy to see that this perimeter is homotopically equivalent to the gap on the $(x_i^j)_{i\in[n]}$-orbit or $(x_i^{j-1})_{i\in[n]}$-orbit where it starts. Thus, we can add the edge missing on that orbit without making $\mathcal{MAX}$ orbiting. Thus, $\mathcal{MAX}$ is not maximal, which is absurd by hypothesis.

We proceed by induction over k. For $k = 1$, the proof has already been done in Proposition 7.

Using the same argument as in the proof of Proposition 4, there is only one edge missing in the orbit defined by the $(x_i^0)_{i\in[n]}$-edges. Let j be this missing edge. Thus, there must be a gap either $RGap^{j,j'}$ or $LGap_{j,j'}$ in DG^1. Otherwise, taking the perimeter of the gap starting from x_j^0, we have a path in *LD* which is homotopically equivalent to x_j^0. Thus, *LD* orbits since it contains the path $x_{j+1}^0, ..., x_{j-1}^0$ and a path homotopically equivalent to x_j^0. All the remaining gaps are of type *ULGap* and *URGap* since they cannot go out. We have only to prove that all these gaps are disconnected. If one of these gaps is connected to the *LGap* gap or the *RGap* gap, then we have a problem. In fact, it means that from x_j^0, we can go out of the DG^1 either from x_i^1 or $x_{i'}^1$. But going out from these two gaps, we must arrive at the $(x_i^k)_{i\in[n]}$-orbit. (Otherwise, we can use the same argument to prove that the gaps are connected, to prove that $\mathcal{MAX}$ is not maximal). Thus, we can have two possibilities: 1) these two gaps do not reconnect. Thus there are two missing edges in $(x_i^k)_{i\in[n]}$, which is absurd due to the argument that we have explained in the proof of Proposition 4. 2)these two gaps reconnect. Thus $\mathcal{MAX}$ is disconnected, which is absurd due to an argument presented in the proof of Proposition 4. Similarly, we can prove that two gaps that start and ends on the top circle of DG^1, that is, the $(x_i^1)_{i\in[n]}$-one cannot intersect between each other.

Again we do the same analysis for $\mathsf{DG}^k \cap \mathcal{MAX}$. Using the same argument as for DG^1, we can prove that there must be a gap of type *RGap* or *LGap* and some gaps of type *BLGap* and *BRGap*.

Now we consider $\mathcal{MAX} \cap \mathsf{DG}^{2->k-1}$, where with $\mathsf{DG}^{2->k-1}$ we denote DG of the composition of $\mathtt{pRef}^2$ until $\mathtt{pRef}^{k-1}$. There are I holes in $(x_i^1)_{i\in[n]}$, that is, I missing edges, I' missing edges in $(x_i^{k-1})_{i\in[n]}$.

Each of these holes must be connected via a gap in $\mathsf{DG}^{2->k-1}$ to one and only one other hole. Otherwise, we have the argument against splitting or terminating at a dead end.

Now we can use the induction. Since each of these gaps is contained there, we can use the induction hypothesis to prove that each of these gaps has the desired shape.

Finally, we have to prove that $|\mathcal{I}_1| + |\mathcal{I}_2|$. We observe that we have proved that there is a "snake" of gaps that starts from x_j^0 the missing edge in the $(x_i^0)_{i\in[n]}$-orbit), to $x_{j'}^k$ (the missing edge in the $(x_i^k)_{i\in[n]}$-orbit). Each time we cross from the top to the bottom, the $(x_i^l)_{i\in[n]}$-orbit, we must cross from the bottom to the top in the same orbit. Moreover, one additional time we must cross from the top to the bottom. This concludes the proof. □

In other words, a maximal diagram can be described as a dependency graph with a gap from the bottom to the top consisting of the six subgraphs defined above. We can use this classification to prove the security for kpRef.

Proposition 8. *An outcome L of the experiment* $\mathsf{Leak}(k\mathtt{pRef}, x, p)$ *is shiftable if its leakage diagram does not orbit the dependency graph.*

Proof sketch. The high-level idea is similar to Proposition 6. We have shown that a single refresh is shiftable if the leakage diagram does not orbit the cylinder structure of its dependency graph. In detail, we have proven that a not orbiting graph of a single refresh is a sub-graph of $RGap_{i,j} = \mathsf{DG} \setminus \mathcal{M}_{i,j}^{\text{right}}$ or $LGap_{i,j} = \mathsf{DG} \setminus \mathcal{M}_{i,j}^{\text{left}}$. Further, this implies that the input share x_i can be set to an arbitrary value, and we can compute the according share y_j without changing the distribution of the leaked values. Hence, x_i and y_j are shiftable. It is easy to see that this also holds for composed gadgets with $(y_i)_{i\in[n]} \leftarrow \mathtt{pRef}^1((x_i)_{i\in[n]})$, and $(z_i)_{i\in[n]} \leftarrow \mathtt{pRef}^2((y_i)_{i\in[n]})$. If x_i and y_j are shiftable in

$\mathtt{pRef}^1$, and y_j and z_k are shiftable in $\mathtt{pRef}^2$, it follows that x_i and z_k are shiftable in $\mathtt{pRef}^2(\mathtt{pRef}^1(\cdot))$. In detail, we can set x_i to an arbitrary value and compute accordingly y_j. Since y_j and z_k are shiftable as well, we can also compute the according z_k for any y_j. For the formal proof, we also show the shiftability for the four additional types of gaps we have introduced above and extend the technique to an arbitrary number of compositions.

Proof. Similar to the security proofs of the gadgets, we use Proposition 1, and only prove the claim for maximal diagrams. We use the same approach as for the maximal diagram of a single refresh. Hence, using Proposition 1, we have to prove the claim for all maximal diagrams. Therefore, we want to show how to modify the values in the gaps $LGap^j_{i,i'}$,$RGap^j_{i,i'}$, $BLGap^j_{i,i'}$, $BRGap^j_{i,i'}$, $ULGap^j_{i,i'}$, and $URGap^j_{i,i'}$ shifting an output of $\mathsf{Leak}(\mathtt{pRef}, x, p)$ in another one without being detected. For $LGap^j_{i,i'}$ and $RGap^j_{i,i'}$ the proof was already done in Proposition 6. Hence it remains to prove it for the other four constructions:

(i) $BLGap^j_{i,i'}$: x_i is modified in $x'_i = x_i + \gamma$, $x_{i'}$ is modified in $x'_{i'} = x_{i'} - \gamma$, r_l is modified in $r'_l = r_l + \gamma$, for the r_ls in $BLGap_{i,j}$, and b_l is modified in $b'_l = b_l + \gamma$, for the b_ls in $BLGap_{i,j}$. for any γ. Note that $x'_i + x'_{i'} = x_i + x_{i'}$.

(ii) $BRGap_{i,i'}$: x_i is modified in $x'_i = x_i + \gamma$, $x_{i'}$ is modified in $x'_{i'} = x_{i'} - \gamma$, r_l is modified in $r'_l = r_l - \gamma$, for the r_ls in $BRGap_{i,j}$, and b_l is modified in $b'_l = b_l - \gamma$, for the b_ls in $BRGap_{i,j}$. Here, as well, $x'_i + x'_{i'} = x_i + x_{i'}$.

(iii) $ULGap_{i,i'}$: y_i is modified in $y'_i = y_i + \gamma$, $y_{i'}$ is modified in $y'_{i'} = y_{i'} - \gamma$, r_l is modified in $r'_l = r_l - \gamma$, for the r_ls in $ULGap_{i,j}$, and b_l is modified in $b'_l = b_l - \gamma$, for the b_ls in $ULGap_{i,j}$. Similarly to before $y'_i + y'_{i'} = y_i + y_{i'}$.

(iv) $URGap_{i,i'}$: y_i is modified in $y'_i = y_i + \gamma$, $y_{i'}$ is modified in $y'_{i'} = y_{i'} - \gamma$, r_l is modified in $r'_l = r_l + \gamma$, for the r_ls in $URGap_{i,j}$, and b_l is modified in $b'_l = b_l + \gamma$, for the b_ls in $URGap_{i,j}$. Here, as well, $y'_i + y'_{i'} = y_i + y_{i'}$.

Hence, we have modified an outcome $\mathsf{Leak}(\mathtt{pRef}, x, p)$ to another $\mathsf{Leak}(\mathtt{pRef}, x, p)$. Here, we do the proof for one case and refer to the full version for the other cases.

The gap is of type $BLGap_{i,i'}$ with $i < i'$ [9]. The values carried in A' are

$$\{x_0, ..., x_{i-1}, x_i + \gamma, x_{i+1}, ..., x_{i'-1}, x_{i'} - \gamma, x_{i'+1}, ..., x_{n-1}, y_0, ..., y_{n-1}, r_0 + \gamma, ..., r_{i-1} + \gamma,$$
$$r_i, ..., r_{i'-1}, r_{i'} + \gamma, ..., r_{n-1} + \gamma, b_0 + \gamma, ..., b_i + \gamma, b_{i+1}, ..., b_{i'}, b_{i'+1} + \gamma, ..., b_{n-1} + \gamma\}.$$

First, we observe that

$$x_0 + ... + x_{i-1} + (x_i + \gamma) + x_{i+1} + ... + x_{i'-1} + (x_{i'} - \gamma), x_{i'+1}, ..., x_{n-1} = x_0 + ... + x_{n-1} + \gamma - \gamma = x.$$

Since we have not touched the shares of y, they carry an encoding of y which is equal to x.
Now, let us consider $\mathtt{pRef}$ with input $x_0, ..., x_{i-1}, x_i + \gamma, x_{i+1}, ..., x_{i'-1}, x_{i'} - \gamma, x_{i'+1}, ..., x_{n-1}$ and randomness $r_0 + \gamma, ..., r_{i-1} + \gamma, r_i, ..., r_{i'-1}, r_{i'} + \gamma, ..., r_{n-1} + \gamma$. For $0 \leq j < i$, $b_j + \gamma = x_j + (r_j + \gamma)$, and $y_j = (b_j + \gamma) - (r_{j-1} + \gamma)$. For i, $b_i + \gamma = (x_i + \gamma) + r_i$, and $y_i = (b_i + \gamma) - (r_{i-1} + \gamma)$. For $i < j < i'$ $b_j = x_j + r_j$, and $y_j = b_j - r_j$. For i', $b_{i'} = (x_{i'} - \gamma) + (r_{i'} + \gamma)$, and $y_{i'} = b_{i'} - r_{i'}$. For $i' < j \leq n - 1$, $b_j + \gamma = x_j + (r_j + \gamma)$, and $y_j = (b_j + \gamma) - (r_{j-1} + \gamma)$. Thus, the values carried by A' are those of an honest evaluation of $\mathtt{pRef}$ with input $x_0, ..., x_{i-1}, x_i + \gamma, x_{i+1}, ..., x_{i'-1}, x_{i'} - \gamma, x_{i'+1}, ..., x_{n-1}$ and randomness $r_0 + \gamma, ..., r_{i-1} + \gamma, r_i, ..., r_{i'-1}, r_{i'} + \gamma, ..., r_{n-1} + \gamma$. For all the other gaps, we proceed in a similar way. For more details we refer to the full version. It remains to explain how we can use the modifications just described to do the shift in the maximal diagram of $k\mathtt{pRef}$. Therefore, we start with DG^1. Using the classification of the maximal of $k\mathtt{pRef}$ there is a single gap starting from its bottom circle. We modify its values according to the proof of Proposition 6. Then, there is another gap in DG^2 that starts from the edge just modified before. We can modify it coherently starting from this modified value. Iterating, due to our classification, there is always a gap in another $\mathsf{DG}^{j'}$ which starts from the final edge of the gap just modified before. Finally, we arrive at the single gap of DG^k which ends in the top circle.

Formally, we have proved the results for all the additional gaps introduced in Proposition 7. Every gap is connected only to two other gaps (except for the first and the last.) Now, we start observing that there are

$$\sum_{j=0}^{k} |\mathcal{I}_3^j| + \sum_{j=0}^{k} |\mathcal{I}_4^j| = \sum_{j=0}^{k} |\mathcal{I}_5^j| + \sum_{j=0}^{k} |\mathcal{I}_6^j|,$$

where with $\mathcal{I}_6^l$, we denote the set $\mathcal{I}_6^j$ in the classification of $\mathcal{MAX} \cap \mathsf{DG}^j$ (Proposition 7).

[9] For simplicity, when we use $i < i'$, we assume that $i, i' \in [n] \subset \mathbb{Z}$ and not in $\mathbb{Z}_n$.

Since every time when we follow a $RGap_{i,i'}$ or a $LGap_{i,i'}$ gap, we keep the same modifications for x_i and $y_{i'}$, while when we follow $BLGap$, or $BRGap$, $ULGap$, or $URGap$, we modify x_i and $y_{i'}$ with two opposite values, and we follow an even number of $BLGap$, $BRGap$, $ULGap$, and $URGap$, we have that x_j^k is modified with $+\delta = x' - x$, where x_j^k is the single edge missing in the $(x_i^k)_{i\in[n]}$-orbit. Thus, we have modified the values of the variable carried by $\mathsf{DG} \setminus \mathcal{MAX}$ to modify an outcome of $\mathsf{Leak}(k\texttt{pRef}, x, p)$ in one of $\mathsf{Leak}(k\texttt{pRef}, x', p)$. We give the formal description in the full version. □

An example of a maximal for kpRef is depicted in the full version. The proposition gives the following privacy result for our compositions. It only remains to compute the probability that a leakage diagram does not orbit the cylinder described by the dependency graph of kpRef.

Theorem 3. *Let kpRef be the composition of k refreshing gadget, defined in Figure 1b. Let n be the number of shares used. Then, kpRef is $(p, (k+1)\cdot[(2+3(3p))3p]^n))$-private, for $p \leq 1/3$ and $(p, 2\cdot(9p)^n))$.*

Proof. Let kpRef takes as input $(x_0^0, ..., x_{n-1}^0)$ and outputs $(x_0^k, ..., x_{n-1}^k)$, with the ith pRef gadget, denoted with $\texttt{pRef}^i$ does $(x_0^i, ..., x_{n-1}^i) \leftarrow \texttt{pRef}(x_0^{i-1}, ..., x_{n-1}^{i-1})$. The proof is completely similar to the one of Theorem 2 with the following differences: 1) We consider the events E_i, for $i = 0, ..., k$ (instead of $i = 0, 1$), which consists in the event that there is an orbit containing the source node of the edge x_0^i (denoted with $node_0^i$). 2) Each node is connected with 6 other nodes (and not 4). Thus, there are at most 5^n different paths. 3) There exists a single n-edges long loop orbiting from $node_0^i$, $4(n-1)$ $n+1$-edges long, and so on. From 1) and 2), we obtain that we can bound $\Pr[E_i] \leq (15p)^n$. Moreover, we can use the same argument to obtain, for $p \leq 1/9$ that

$$\Pr[E_i] \leq [(2+3(3p))3p]^n \leq (9p)^n.$$

To prove this, we can reuse the same arguments as in the proof of Theorem 2. We must add also the argument that no edge connects the $(x_i^j)_{i\in[n]}$-orbit and the $(x_i^{j'})_{i\in[n]}$-orbit if $j' \neq j-1, j, j+1$. Moreover, if node y belongs to the $(x_i^l)_{i\in[n]}$-orbit, the shortest path to $node_0^j$ can never cross and edge to go from the $(x_i^L)_{i\in[n]}$-orbit to a node in the $(x_i^{L'})_{i\in[n]}$-orbit if $|L'-j| > |L-j|$ (because we cannot skip orbits).

Finally, we explain why it holds

$$\Pr[E_0] \leq [(2+3(3p))3p]^n.$$

This is due to the fact that not all edges directly give the shortest path and require further edges that are also added with probability p, and this happens with probability at most $3p$. □

5.2 Security Analysis for the Composition of Gadgets

Now, we analyze the security of an arbitrary output of our compiler $\widehat{\mathsf{C}} \leftarrow \mathtt{CC}(\mathsf{C})$, where C can be any circuit described in the background. In Figure 6b, we depicted the dependency graph of Add (or Mult) gadgets composed with pRef gadgets to refresh its inputs and outputs. Since our compiler puts refresh gadgets after every output of Add, Mult, and Copy gadget, the dependency graph of our compiler's output is always a composition of dependency graphs depicted in Figure 6a. As mentioned, this leads to slightly more complex dependency graphs than the one described in the previous section (Sec. 5.1.) However, the main idea is the same. We have multiple dependency graphs, as depicted in Figure 6a, sharing the same upper or lower circle as already described in the previous section (Figure 11a.) The only difference is that gadgets can also have two input (or output) sharings. Therefore the cylinder does not only share the bottom (or top) of the cylinder with one but two further cylinders, as depicted in Figure 6b. Hence, the dependency graph of $\widehat{\mathsf{C}}$ is still a composition of multiple cylinders. Since the dependency graph describes multiple cylinders connected by shared upper and lower circles, we can distinguish orbiting loops from not orbiting loops again. For this reason, we can use the same technique as in Proposition 8 to prove the security of the composition. Again, we start classifying the maximals of $\widehat{\mathsf{C}}$.

Proposition 9. *Let $\mathcal{MAX}$ be a maximal diagram*

(i) For each circuit-input encoding and circuit-output encoding, there is only one edge of the encoding that does not belong to $\mathcal{MAX}$.

(ii) The intersection of the DG *of all* pRef *with $\mathcal{MAX}$ has an odd number of components which have the characterization described in Proposition 8.*

(iii) Each of these "gap sets" is connected to others to form gaps from each circuit input encoding to any output encoding of the circuit.

Proof. i) We exploit the same argument as presented in the proof of Proposition 7 to prove that this holds. Substantially, if there is more than one hole in the input/output orbit edge, the graph is either not maximal or it orbits. If there are more than two holes on the input $(x_i)_{i\in[n]}$-orbit, we must have a gap starting from each. These two gaps either intersect or do not intersect. If they intersect, then we can show that the maximal diagram orbit with an argument similar to the one introduced in the proof of Proposition 7. In the second case, $\mathcal{MAX}$ is not connected. If they do not connect, we cannot assume that they go to two different other input/output orbits. Because otherwise, we can homotopically reduce these two input sharings to the copy where the gaps split. Thus we have non-connection.

ii) The idea is the same as in the proof of Proposition 7. It comes from the fact that the gap cannot split or have a dead end in the $\mathsf{DG}_{\mathtt{pRef}}$ with the same idea as in Proposition 7.

iii) This happens because if these gaps are not connected to each other, we can homotopically take them away as done in the proof of Proposition 7.

□

As before, our classification allows us to prove the security of not orbiting leakage diagrams.

Proposition 10. *Let $\widehat{\mathsf{C}}$ be a masked circuit obtained from our compiler. An outcome L of the $\mathsf{Leak}(\widehat{\mathsf{C}}, \boldsymbol{x}, p)$ experiment, is shiftable to $\boldsymbol{x}'$ if its leakage diagram does not orbit the dependency graph.*

Proof sketch. The only difference to the proof in Section 5.1 is that we have gadgets with multiple input and output sharing. Hence we do not have only one bottom and top. (E.g. Figure 6b has two bottom circles). However, the high-level idea is the same.

Proof. When we have such dependency graphs, a leakage diagram does not orbit when gaps exist (as defined in the proof of Proposition 8) from every bottom and top circle. We have described the maximals above. Using this classification, we can modify the values of the variables carried by the edges in $\mathsf{DG}_{\widehat{\mathsf{C}}} \setminus \mathcal{MAX}$ as in the proof of Proposition 8. We start with an arbitrary input encoding, then we modify the values on the subgraphs *RGap*, *LGap*, *BRGap*, *BLGap*, *ULGap*, and *URGap* as in the previous example (and as described in the Proposition 5, 6, and 8). Iterating, we arrive at modifying all the values of $\mathsf{DG} \setminus \mathcal{MAX}$. With this technique, we can modify all inputs. Hence, L from $\mathsf{Leak}(\widehat{\mathsf{C}}, \mathbf{x}, p)$ is shiftable if its leakage diagram is not orbiting. The details that this modification gives the same output of $\mathsf{Leak}(\widehat{\mathsf{C}}, \mathbf{x}, p)$ is given now.[10]

We start with the input orbit sharings. From there, we can change the gaps until we arrive to a gap that arrives in a gap where two different inputs are modified. There using Proposition 5, we wait until we have arrived to modify the other input with gaps. We can go on, and we can arrive that at least we can modify one of the gaps with multiple inputs. [This happens due to the structure of the gasp in $\mathcal{MAX}$.]

This operation is correct because even here, every time that we go up, we add the shift, while every time we go down, we subtract the shift. This works as in the proof of Proposition 8.

□

Using Corollary 3 and Proposition 10, we can bound the actual security of the circuits obtained via our compiler.

Theorem 4. *Let $\widehat{\mathsf{C}}$ be a circuit obtained via our compiler, $\widehat{\mathsf{C}} \leftarrow \mathsf{CC}^p(\mathsf{C})$, and $|\mathsf{C}|$ be the number of gates of the circuit C, I the number of input gates and O the number of output gates. Then, $\widehat{\mathsf{C}}$ is $(p, (|\mathsf{C}| + I + O)\widehat{p}^n)$-private, with*

$$\widehat{p} = 8[1 - (1 - \sqrt{3p})^{8n}],$$

$$\textit{or even tighter: } \widehat{p} = 18p + 2(1 - (1 - p)^{8n} + 1 - (1 - \sqrt{3p})^{n-1})$$

If C is affine, then, $\widehat{\mathsf{C}}$ is $(p, (|\mathsf{C}| + I + O)(12p)^n)$. If circuit C is complete, then $\widehat{\mathsf{C}}$ is $(p, (|\mathsf{C}|\widehat{p}^n)$-private, or $\widehat{\mathsf{C}}$ is $(p, |\mathsf{C}|(12p)^n)$-private.

[10] In the proof of Proposition 8 this step is easier because each gap set meets at its end a single another gap set.

Proof. The proof is similar to the proof of Theorem 3 with the following differences: 1) We have at most $|\mathsf{C}| + I + O$ different sharings (orbits) containing the shares of an input, intermediate, or output encoding. If the circuit is complete, the number of such orbits is at most $|\mathsf{C}|$. Thus we have $|\mathsf{C}| + I + O$ events E_i to consider. 2) For each node of DG, there are at most 8 edges. 3) Each edge is added to LD with probability at most $p' = 2(1 - (1 - \sqrt{3p})^{8n})$ for general circuits (or $p' = 3p$) for affine circuits,see Proposition 5 and 6. Thus, we can prove that $\Pr[E_i] \leq (7p')^n$. Doing a more detailed analysis (full version), similar to the one done in Theorem 2 and 3, we obtain that $\Pr[E_i] \leq (4p')^n$. Putting everything together, we obtain the claim. □

With Theorem 4, we can discuss the security results for our compiler.

5.3 Compiler Security

In the previous section, we have proven that any complete circuit $\widehat{\mathsf{C}}$ obtained via our compiler, $\widehat{\mathsf{C}} \leftarrow \mathsf{CC}^p(\mathsf{C})$ is $(p, (|\mathsf{C}|)\widehat{p}^n)$-private, with $\widehat{p} = 8[1-(1-\sqrt{3p})^{8n}]$. Further, if C is affine, then, $\widehat{\mathsf{C}}$ is $(p, (|\mathsf{C}|+I+O)(12p)^n)$-private. This section discusses the results and demonstrates the improvements compared to the state-of-the-art. As in [21, 16] the condition $\widehat{p} < 1$ requires an upper-bound for the leakage probability p. In detail, Theorem 4 requires the following.

Proposition 11. *Let $p \in [0, \frac{1}{3}]$ be the leakage probability and $\alpha \in (0, 1]$. It holds $\widehat{p}^n = 8^n(1 - (1 - \sqrt{3p})^{8n})^n \leq \alpha$ if*

$$p \leq \frac{\left(1 - \sqrt[8n]{1 - \frac{\alpha^{\frac{1}{n}}}{8}}\right)^2}{3} < 1$$

Proof. If p is smaller than $1/3$ we get $8^n(1 - (1 - \sqrt{3p})^{8n})^n \leq \alpha$ for an alpha $\alpha \in (0, 1]$. The claim is a simple transformation.

$$8^n(1 - (1 - \sqrt{3p})^{8n})^n \leq \alpha \Leftrightarrow p \leq \frac{\left(1 - \sqrt[8n]{1 - \frac{\alpha^{\frac{1}{n}}}{8}}\right)^2}{3} < 1$$

For more details, we refer to the full version. □

Further, Proposition 11 gives us the required asymptotic behavior of p for our compiler.

Theorem 5. *The compiler is secure for any leakage probability p with $p = O(\frac{1}{n^2})$.*

Proof. We can define the upper-bound p with

$$F_\alpha(n) := \frac{\sqrt{1 - \sqrt[8n]{1 - \frac{\alpha^{\frac{1}{n}}}{8}}}}{3}.$$

To prove the claim, we will show that there exists a constant C s.t.

$$\lim_{n\to\infty} \frac{F_\alpha(n)}{\frac{1}{\sqrt{n}}} = C.$$

Note: Consequently, the claim of Proposition 11 follows with

$$\lim_{n\to\infty} \frac{\left(1 - \sqrt[8n]{1 - \frac{\alpha^{\frac{1}{n}}}{8}}\right)^2}{3n^{-2}} = \lim_{n\to\infty} \left(\frac{F_\alpha(n)}{\frac{1}{\sqrt{n}}}\right)^4 = C^4$$

as C^4 is a constant as well. The following is the original proof again.

We can transform the term to

$$\lim_{n\to\infty} \frac{F_\alpha(n)}{\frac{1}{\sqrt{n}}} = \lim_{n\to\infty} \sqrt{n}\frac{\sqrt{1 - \sqrt[8n]{1 - \frac{\alpha^{\frac{1}{n}}}{8}}}}{3} = \lim_{n\to\infty} \frac{1}{3}\sqrt{n\left(1 - \sqrt[8n]{1 - \frac{\alpha^{\frac{1}{n}}}{8}}\right)}.$$

It remains to study

$$\sqrt[8n]{1-\frac{\alpha^{\frac{1}{n}}}{8}} = e^{\frac{1}{8n}log\left(1-\frac{\alpha^{\frac{1}{n}}}{8}\right)}.$$

The previous equivalence is true since $0 < \frac{\alpha^{\frac{1}{n}}}{8} \leq \frac{7}{8}$. Now, if $\alpha \neq 0$, then $\lim_{n\to\infty} \frac{1}{8n} log\left(1-\frac{\alpha^{\frac{1}{n}}}{8}\right) = 0$, thus, using standard analysis techniques, we have that

$$e^{\frac{1}{8n}log\left(1-\frac{\alpha^{\frac{1}{n}}}{8}\right)} \sim \frac{1}{8n} log\left(1-\frac{\alpha^{\frac{1}{n}}}{8}\right) + 1.$$

(two functions are asymptotically equivalent if they have the same limit). Thus,

$$\lim_{n\to\infty} \frac{1}{3}\sqrt{n\left(1-\sqrt[8n]{1-\frac{\alpha^{\frac{1}{n}}}{8}}\right)} = \lim_{n\to\infty} \frac{1}{3}\sqrt{n\left(1-1-\frac{1}{8n}log\left(1-\frac{\alpha^{\frac{1}{n}}}{8}\right)\right)}$$
$$= \lim_{n\to\infty} \frac{1}{3}\sqrt{\frac{n}{8n}log\left(1-\frac{\alpha^{\frac{1}{n}}}{8}\right)} = \lim_{n\to\infty} \frac{1}{3}\sqrt{\frac{1}{8}log\left(1-\frac{1}{8}\right)} = C$$

where in the second to last equality, we have used the fact that if $\alpha > 0$, then $\lim_{n\to\infty} \alpha^{\frac{1}{n}} = 1$. Since C is a constant, this proves the claim of the theorem. □

6 Comparison with Dziemobwski et al. [16]

Our work is inspired by Dziembowski et al. [16]. Thus, we want to summarize the differences between their work and ours. Our compiler provides gadgets that work parallelly. In particular, our refresh gadget, `pRef`, works in 3 clock cycles, while their `sRef` works in $O(n)$ cycles. Moreover, our multiplication gadget, `Mult` works in $O(\log(n))$ cycles, while theirs, the ISW [21] works with $O(n)$ cycles. The other gadgets, which are the same in both compilers, need a constant number of cycles. Thus, our compiler is significantly faster than theirs for affine and general circuits.

Proof technique. Their proof technique is a particular case of our generalized one. In fact, it is enough to consider the graph depicted in Figure 2b. Since the values of the variables c_0^i and c_n^i are always 0, we can consider them as equal and glue the edge carrying c_0^i with c_n^i $\forall i$. In this way, we obtain a cylinder (the formal proof is in the full version.) In other words, the encodings of a value can also be considered like an orbit. Since $c_i^n = 0$ is fixed and not secret, we must assume that the adversary knows c_i^n. Thus, we consider the edges carrying c_i^n as always "leaked; hence, it always belongs to the leakage diagram. This condition means that our property of not orbiting becomes the property in which the leakage diagram's left, and right sides are not connected. Finally, their way to modify (that is, choosing the so-called *modifications vectors*) is a way to select a maximal containing the leakage diagram as we did in our proofs. Further, we show in the full version why we cannot use their technique for our gadgets. Informally, their technique does not work for our *LD* because choosing the so-called modification vectors would have been challenging (since there are no edges that must belong to the leakage diagrams).

Bound differences. Since every edge of $\mathsf{DG}_{\mathtt{pRef}}$ (Figure 5b) contains a single variable, while the c_j^i-edges contain two variables (Figure 2b), we have that our affine compiled circuits are $(O([9p]^n), p)$-private, while theirs is $(O([8\sqrt{3p}]^n), p)$-private. Thus, we have gained an order of magnitude. Second, by doing a more detailed analysis, we have proved that for the `Mult` gadgets and, thus, for the general compiled circuits, the security is $[18p + 2(1-(1-p)^{8n} + 1 - (1-\sqrt{3p})^{n-1})]^n \leq (8[1-(1-\sqrt{3p})^{8n}])^n$, instead of $(32np + 4n\sqrt{3p})^n$. This observation proves better security gains as depicted in the full version.

7 Conclusion

In this paper, we have started from the graphs introduced by Dziemboswki et al. [16]. Then, we showed how to use a broader class of graphs. Our graphs are more general than [16], and we used them to prove the security in the

random probing model of our parallel compiler. Using this, we have proved that our compiler has $O(p^n)$-security for the affine case and $O((n^2p)^n)$-security for the general case. Besides being parallel, our compiler has the advantage that it is one of the simplest possible. This graph technique is interesting and could be applied to other compilers. Moreover, we believe the same technique can be applied to other probing models, such as the t-threshold probing model or the average-random probing model, or considering leakage models where glitches are considered, or also considering security in the presence of faults. We also believe that our technique can be applied to gadgets corresponding to bigger circuits, as those used to mask public-key encryption scheme. Finding dependency graphs which can be useful is still an interesting challenge. Finally, it might be interesting to improve the security bounds of our `Mult` and `Add`-gadget using improved graphs.

Acknowledgment

This work was partly supported by the German Research Foundation (DFG) via the DFG CRC 1119 CROSSING (project S7), by the German Federal Ministry of Education and Research and the Hessen State Ministry for Higher Education, Research and the Arts within their joint support of the National Research Center for Applied Cybersecurity ATHENE, and by the European Commission(ERCEA), ERC Grant Agreement 101044770 CRYPTOLAYER. F. Berti was funded by Israel Science Foundation, ISF grant 2569/21.

We would like to thank Stefan Dziembowski and Karol Zebrowski for helpful discussions on earlier versions of this work. Further, we thank our reviewers for the many helpful comments to improve the paper.

References

[1] Ananth, P., Ishai, Y., Sahai, A.: Private circuits: A modular approach. In: Shacham, H., Boldyreva, A. (eds.) Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part III. Lecture Notes in Computer Science, vol. 10993, pp. 427–455. Springer (2018). https://doi.org/10.1007/978-3-319-96878-0_15, https://doi.org/10.1007/978-3-319-96878-0_15

[2] Andrychowicz, M., Dziembowski, S., Faust, S.: Circuit compilers with o(1/\log (n)) leakage rate. In: Fischlin, M., Coron, J. (eds.) Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II. Lecture Notes in Computer Science, vol. 9666, pp. 586–615. Springer (2016). https://doi.org/10.1007/978-3-662-49896-5_21, https://doi.org/10.1007/978-3-662-49896-5_21

[3] Barthe, G., Dupressoir, F., Faust, S., Grégoire, B., Standaert, F., Strub, P.: Parallel implementations of masking schemes and the bounded moment leakage model. In: Coron, J., Nielsen, J.B. (eds.) Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part I. Lecture Notes in Computer Science, vol. 10210, pp. 535–566 (2017). https://doi.org/10.1007/978-3-319-56620-7_19, https://doi.org/10.1007/978-3-319-56620-7_19

[4] Belaïd, S., Coron, J., Prouff, E., Rivain, M., Taleb, A.R.: Random probing security: Verification, composition, expansion and new constructions. In: Micciancio, D., Ristenpart, T. (eds.) Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part I. Lecture Notes in Computer Science, vol. 12170, pp. 339–368. Springer (2020). https://doi.org/10.1007/978-3-030-56784-2_12, https://doi.org/10.1007/978-3-030-56784-2_12

[5] Belaïd, S., Mercadier, D., Rivain, M., Taleb, A.R.: Ironmask: Versatile verification of masking security. In: 43rd IEEE Symposium on Security and Privacy, SP 2022, San Francisco, CA, USA, May 22-26, 2022. pp. 142–160. IEEE (2022). https://doi.org/10.1109/SP46214.2022.9833600, https://doi.org/10.1109/SP46214.2022.9833600

[6] Belaïd, S., Mercadier, D., Rivain, M., Taleb, A.R.: Ironmask: Versatile verification of masking security. In: 43rd IEEE Symposium on Security and Privacy, SP 2022, San Francisco, CA, USA, May 22-26, 2022. pp. 142–160. IEEE (2022). https://doi.org/10.1109/SP46214.2022.9833600, https://doi.org/10.1109/SP46214.2022.9833600

[7] Belaïd, S., Rivain, M., Taleb, A.R.: On the power of expansion: More efficient constructions in the random probing model **12697**, 313–343 (2021). https://doi.org/10.1007/978-3-030-77886-6_11, https://doi.org/10.1007/978-3-030-77886-6_11

[8] Belaïd, S., Rivain, M., Taleb, A.R.: On the power of expansion: More efficient constructions in the random probing model. In: Canteaut, A., Standaert, F. (eds.) Advances in Cryptology - EUROCRYPT 2021 - 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17-21, 2021, Proceedings, Part II. Lecture Notes in Computer Science, vol. 12697, pp. 313–343. Springer (2021). https://doi.org/10.1007/978-3-030-77886-6_11, https://doi.org/10.1007/978-3-030-77886-6_11

[9] Bogdanov, A., Ishai, Y., Srinivasan, A.: Unconditionally secure computation against low-complexity leakage. In: Boldyreva, A., Micciancio, D. (eds.) Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part II. Lecture Notes in Computer Science, vol. 11693, pp. 387–416. Springer (2019). https://doi.org/10.1007/978-3-030-26951-7_14, https://doi.org/10.1007/978-3-030-26951-7_14

[10] Bonneau, J., Mironov, I.: Cache-collision timing attacks against AES. In: Goubin, L., Matsui, M. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2006, 8th International Workshop, Yokohama, Japan, October 10-13, 2006, Proceedings. Lecture Notes in Computer Science, vol. 4249, pp. 201–215. Springer (2006). https://doi.org/10.1007/11894063_16, https://doi.org/10.1007/11894063_16

[11] Cassiers, G., Faust, S., Orlt, M., Standaert, F.: Towards tight random probing security. In: Malkin, T., Peikert, C. (eds.) Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part III. Lecture Notes in Computer Science, vol. 12827, pp. 185–214. Springer (2021). https://doi.org/10.1007/978-3-030-84252-9_7, https://doi.org/10.1007/978-3-030-84252-9_7

[12] Clavier, C., Feix, B., Gagnerot, G., Roussellet, M., Verneuil, V.: Horizontal correlation analysis on exponentiation. In: Soriano, M., Qing, S., López, J. (eds.) Information and Communications Security - 12th International Conference, ICICS 2010, Barcelona, Spain, December 15-17, 2010. Proceedings. Lecture Notes in Computer Science, vol. 6476, pp. 46–61. Springer (2010). https://doi.org/10.1007/978-3-642-17650-0_5, https://doi.org/10.1007/978-3-642-17650-0_5

[13] Coron, J., Prouff, E., Rivain, M., Roche, T.: Higher-order side channel security and mask refreshing. In: Moriai, S. (ed.) Fast Software Encryption - 20th International Workshop, FSE 2013, Singapore, March 11-13, 2013. Revised Selected Papers. Lecture Notes in Computer Science, vol. 8424, pp. 410–424. Springer (2013). https://doi.org/10.1007/978-3-662-43933-3_21, https://doi.org/10.1007/978-3-662-43933-3_21

[14] Duc, A., Dziembowski, S., Faust, S.: Unifying leakage models: From probing attacks to noisy leakage. J. Cryptol. **32**(1), 151–177 (2019). https://doi.org/10.1007/s00145-018-9284-1, https://doi.org/10.1007/s00145-018-9284-1

[15] Dziembowski, S., Faust, S., Skorski, M.: Noisy leakage revisited. In: Oswald, E., Fischlin, M. (eds.) Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II. Lecture Notes in Computer Science, vol. 9057, pp. 159–188. Springer (2015). https://doi.org/10.1007/978-3-662-46803-6_6, https://doi.org/10.1007/978-3-662-46803-6_6

[16] Dziembowski, S., Faust, S., Zebrowski, K.: Simple refreshing in the noisy leakage model. In: Galbraith, S.D., Moriai, S. (eds.) Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part III. Lecture Notes in Computer Science, vol. 11923, pp. 315–344. Springer (2019). https://doi.org/10.1007/978-3-030-34618-8_11, https://doi.org/10.1007/978-3-030-34618-8_11

[17] Gao, S., Mateer, T.D.: Additive fast fourier transforms over finite fields. IEEE Trans. Inf. Theory **56**(12), 6265–6272 (2010). https://doi.org/10.1109/TIT.2010.2079016, https://doi.org/10.1109/TIT.2010.2079016

[18] Genkin, D., Shamir, A., Tromer, E.: RSA key extraction via low-bandwidth acoustic cryptanalysis. In: Garay, J.A., Gennaro, R. (eds.) Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I. Lecture Notes in Computer Science, vol. 8616, pp. 444–461. Springer (2014). https://doi.org/10.1007/978-3-662-44371-2_25, https://doi.org/10.1007/978-3-662-44371-2_25

[19] Goudarzi, D., Joux, A., Rivain, M.: How to securely compute with noisy leakage in quasilinear complexity. In: Peyrin, T., Galbraith, S.D. (eds.) Advances in Cryptology - ASIACRYPT 2018 - 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part II. Lecture Notes in Computer Science, vol. 11273, pp. 547–574. Springer (2018). https://doi.org/10.1007/978-3-030-03329-3_19, https://doi.org/10.1007/978-3-030-03329-3_19

[20] Goudarzi, D., Prest, T., Rivain, M., Vergnaud, D.: Probing security through input-output separation and revisited quasilinear masking. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2021**(3), 599–640 (2021). https://doi.org/10.46586/tches.v2021.i3.599-640, https://doi.org/10.46586/tches.v2021.i3.599-640

[21] Ishai, Y., Sahai, A., Wagner, D.A.: Private circuits: Securing hardware against probing attacks. In: Boneh, D. (ed.) Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings. Lecture Notes in Computer Science, vol. 2729, pp. 463–481. Springer (2003). https://doi.org/10.1007/978-3-540-45146-4_27, https://doi.org/10.1007/978-3-540-45146-4_27

[22] Kocher, P.C.: Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In: Koblitz, N. (ed.) Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings. Lecture Notes in Computer Science, vol. 1109, pp. 104–113. Springer (1996). https://doi.org/10.1007/3-540-68697-5_9, https://doi.org/10.1007/3-540-68697-5_9

[23] Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M.J. (ed.) Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings. Lecture Notes in Computer Science, vol. 1666, pp. 388–397. Springer (1999). https://doi.org/10.1007/3-540-48405-1_25, https://doi.org/10.1007/3-540-48405-1_25

[24] Medwed, M., Standaert, F., Joux, A.: Towards super-exponential side-channel security with efficient leakage-resilient prfs. In: Prouff, E., Schaumont, P. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2012 - 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings. Lecture Notes in Computer Science, vol. 7428, pp. 193–212. Springer (2012). https://doi.org/10.1007/978-3-642-33027-8_12, https://doi.org/10.1007/978-3-642-33027-8_12

[25] Nikova, S., Rechberger, C., Rijmen, V.: Threshold implementations against side-channel attacks and glitches. In: Ning, P., Qing, S., Li, N. (eds.) Information and Communications Security, 8th International Conference, ICICS 2006, Raleigh, NC, USA, December 4-7, 2006, Proceedings. Lecture Notes in Computer Science, vol. 4307, pp. 529–545. Springer (2006). https://doi.org/10.1007/11935308_38, https://doi.org/10.1007/11935308_38

[26] Prest, T., Goudarzi, D., Martinelli, A., Passelègue, A.: Unifying leakage models on a rényi day. In: Boldyreva, A., Micciancio, D. (eds.) Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part I. Lecture Notes in Computer Science, vol. 11692, pp. 683–712. Springer (2019). https://doi.org/10.1007/978-3-030-26948-7_24, https://doi.org/10.1007/978-3-030-26948-7_24

[27] Prouff, E., Rivain, M.: Masking against side-channel attacks: A formal security proof. In: Johansson, T., Nguyen, P.Q. (eds.) Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings. Lecture Notes in Computer Science, vol. 7881, pp. 142–159. Springer (2013). https://doi.org/10.1007/978-3-642-38348-9_9, https://doi.org/10.1007/978-3-642-38348-9_9

[28] Reparaz, O., Bilgin, B., Nikova, S., Gierlichs, B., Verbauwhede, I.: Consolidating masking schemes. In: Gennaro, R., Robshaw, M. (eds.) Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I. Lecture Notes in Computer Science, vol. 9215, pp. 764–783. Springer (2015). https://doi.org/10.1007/978-3-662-47989-6_37, https://doi.org/10.1007/978-3-662-47989-6_37

[29] Rivain, M., Prouff, E.: Provably secure higher-order masking of AES. In: Mangard, S., Standaert, F. (eds.) Cryptographic Hardware and Embedded Systems, CHES 2010, 12th International Workshop, Santa Barbara, CA, USA, August 17-20, 2010. Proceedings. Lecture Notes in Computer Science, vol. 6225, pp. 413–427. Springer (2010). https://doi.org/10.1007/978-3-642-15031-9_28, https://doi.org/10.1007/978-3-642-15031-9_28

[30] Trichina, E.: Combinational logic design for AES subbyte transformation on masked data. IACR Cryptol. ePrint Arch. p. 236 (2003), http://eprint.iacr.org/2003/236

[31] Walter, C.D.: Sliding windows succumbs to big mac attack. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2001, Third International Workshop, Paris, France, May 14-16, 2001, Proceedings. Lecture Notes in Computer Science, vol. 2162, pp. 286–299. Springer (2001). https://doi.org/10.1007/3-540-44709-1_24, https://doi.org/10.1007/3-540-44709-1_24

C. Fuzzy Asymmetric Password-Authenticated Key Exchange

This chapter corresponds to our published article at ASIACRYPT 2020 [90], with minor edits. Our full version can be found in [89].

Fuzzy Asymmetric Password-Authenticated Key Exchange

Andreas Erwig[1], Julia Hesse[2], Maximilian Orlt[1], and Siavash Riahi[1]

[1] Technische Universität Darmstadt, Germany
{andreas.erwig, maximilian.orlt, siavash.riahi}@tu-darmstadt.de
[2] IBM Research - Zurich, Switzerland
jhs@zurich.ibm.com

Abstract. Password-Authenticated Key Exchange (PAKE) lets users with passwords exchange a cryptographic key. There have been two variants of PAKE which make it more applicable to real-world scenarios:

- *Asymmetric* PAKE (aPAKE), which aims at protecting a client's password even if the authentication server is untrusted, and
- *Fuzzy* PAKE (fPAKE), which enables key agreement even if passwords of users are noisy, but "close enough".

Supporting fuzzy password matches eases the use of higher entropy passwords and enables using biometrics and environmental readings (both of which are naturally noisy).
Until now, both variants of PAKE have been considered only in separation. In this paper, we consider both of them simultaneously. We introduce the notion of *Fuzzy Asymmetric PAKE* (fuzzy aPAKE), which protects against untrusted servers *and* supports noisy passwords. We formulate our new notion in the Universal Composability framework of Canetti (FOCS'01), which is the preferred model for password-based primitives. We then show that fuzzy aPAKE can be obtained from oblivious transfer and some variant of robust secret sharing (Cramer et al, EC'15). We achieve security against malicious parties while avoiding expensive tools such as non-interactive zero-knowledge proofs. Our construction is round-optimal, with message and password file sizes that are independent of the schemes error tolerance.

1 Introduction

In a world of watches interacting with smartphones and our water kettle negotiating with the blinds in our house, communicating devices are ubiquitous. Developments in user-centric technology are rapid, and they call for authentication methods that conveniently work with, e.g., biometric scans, human-memorable passwords or fingerprints derived from environmental readings.

Password-authenticated Key Exchange (PAKE) protocols [BM92, BPR00, BMP00, KOY01, GL03, KV11, CDVW12, BBC+13] are the cryptographic answer to this need. They solve the problem of establishing a secure communication channel between two users who share nothing but a low-entropy string, often simply called *password*. Two interesting variants of PAKE protocols that are known from the literature are *asymmetric* PAKE [BM93, GMR06, JKX18, BJX19] which aims at protecting the user's password even if his password file at some server is stolen, and *fuzzy* PAKE [DHP+18] which can tolerate some errors in the password. The former is useful in settings where authentication servers store thousands of user accounts and the server cannot be fully trusted. The latter introduces a usability aspect to PAKE protocols used by humans trying to remember passwords exactly. Furthermore, fuzzy PAKE broadens applicability of PAKE to the fuzzy setting and thereby allows using environmental readings or biometrics as passwords.

This work is the first to consider a combination of both PAKE variants. Namely, we introduce the notion of *fuzzy asymmetric PAKE* (fuzzy aPAKE). This new primitive allows a client and an untrusted server to authenticate to each other using a password, and both parties are guaranteed to derive the same cryptographic key as long as their passwords are within some predefined distance (in some predefined metric). Consider a client authenticating to a server using his fingerprint scan. In this setting, asymmetric PAKE protocols

would not work since subsequent scans do not match exactly. Fuzzy PAKE, on the other hand, would require the server to store the fingerprint (or at least some template of it that uniquely identifies the person) in the clear, which is unacceptable for sensitive and ephemeral personal data that is biometrics. Fuzzy asymmetric PAKE, as introduced in this paper, is the only known cryptographic solution that applies to this setting: it works with fuzzy authentication data *and* does not reveal this authentication data to the server.

Why is this hard? Given that there is a lot of literature about both asymmetric PAKE and fuzzy cryptography, one could ask whether existing techniques could be used to obtain fuzzy aPAKE. As explained already in [DHP+18], techniques from fuzzy cryptography such as information reconciliation [BBR88] or fuzzy extractors [DRS04] cannot be used with passwords of low entropy. Essentially, these techniques lose several bits of their inputs, which is acceptable when inputs have high entropy, but devastating in case of passwords.

Looking at techniques for asymmetric PAKE, all of them require some kind of password hardening such as hashing [GMR06, HL19, PW17], applying a PRF [JKX18] or a hash proof system [BJX19]. Unfortunately, such functions destroy all notions of closeness of their inputs by design. Further, it is unclear how to define a fuzzy version of, e.g., an oblivious PRF as used in [JKX18] that is not simply a constant function. While such definitions exist for "fuzzy" cryptographic hashing (e.g., robust property-preserving hashing [BLV19]), these functions either do not provide useful error correction or already their description leaks too much information about the password of the client. Overall, there seems to be no candidate asymmetric PAKE which can be made fuzzy.

Regarding more naive approaches, it is tempting to try to apply generic techniques for multi-party computation to obtain a fuzzy PAKE such as garbled circuits [Yao86]. The circuit would be created w.r.t some function of the password $h \leftarrow H(\mathsf{pw})$. The user's input would be pw'. Now the circuit finds all passwords close enough to pw' and outputs the shared key if one of these passwords yield h. Despite the inefficiency of this approach, it is unclear how to actually write down the circuit. As shown in [Hes19], h needs to be the output of some idealized assumption such as a programmable random oracle, and thus has no representation as a circuit.

Our contributions In this paper, we give the first formal definition of fuzzy asymmetric PAKE. Our definition is in the Universal Composability framework of Canetti [Can01], which is the preferred model for PAKE protocols (cf., e.g., [JKX18] for reasons). Essentially, we take the aPAKE functionality from [GMR06] (in a revised version due to [Hes19]) and equip it with fuzzy password matching (taken from the fuzzy PAKE functionality $\mathcal{F}_{\mathsf{fPAKE}}$ from [DHP+18]). Our resulting functionality $\mathcal{F}_{\mathsf{faPAKE}}$ is flexible in two ways: it can be optionally equipped with a mutual key confirmation (often called *explicit authentication*), and, just as $\mathcal{F}_{\mathsf{fPAKE}}$, $\mathcal{F}_{\mathsf{faPAKE}}$ can be parametrized with arbitrary metrics for distance, arbitrary thresholds and arbitrary adversarial leakage. Thus, our model is suitable to analyze protocols for a wide range of applications, from tolerating only few language-specific typos in passwords [CWP+17] to usage of noisy biometric scans of few thousand bits length.

We then give two constructions for fuzzy asymmetric PAKE. Our first construction Π_{faPAKE} uses error-correcting codes (ECC)[3] and oblivious transfer (OT) as efficient building blocks. Π_{faPAKE} works for Hamming distance and can correct $\mathcal{O}(\log(n))$ errors in n-bit passwords. Let us now give more details on Π_{faPAKE}.

The idea of our protocol is to first encode a cryptographic key and store it at the server, in a file together with random values to hide the codeword. The exact position of the codeword in the file is dictated by the password. A client holding a close enough password is thus able to retrieve almost the whole codeword correctly and can thus decode the session key given the error correction capabilities of the encoding. An attacker stealing the password file, however, cannot simply decode since the file contains too much randomness. To remove this randomness, he is bound to decode subsets of the file until he finds two subsets which decode to the same session key. Since decoding can be assumed to be as expensive as hashing, the effort of an off-line

[3] More precisely, we use a variant of *Robust Secret Sharing*, which can be instantiated with some class of error-correcting codes. However, since most readers are presumably more familiar with the latter, we describe our constructions in terms of codes.

dictionary attack on the password file follows from a purely combinatorial argument on the parameters of the scheme (i.e., password size and error correction threshold).

To bound the client to one password guess per run of the protocol (which is the common security requirement for PAKE), we employ an n-times 1-out-of-2 OT scheme. Each OT lets the client choose either the true or the random part of the codeword for each of the n password bits (here we assume that the codeword is from $\mathbb{F}^n$ for some large field $\mathbb{F}$). Further, we apply randomization techniques to keep a client from collecting parts of the password file over several runs of the protocol.

A plus of our protocol is that it elegantly circumvents usage of expensive techniques such as non-interactive zero-knowledge proofs to ensure security against a malicious server. Indeed, a malicious server could make the client reconstruct the session key regardless of her password by entering only the true codeword in the OT. Such attacks would be devastating in applications where the client uses the session key to encrypt her secrets and sends them to the bogus server. Thus, the client needs a means to check correct behavior of the server. We achieve this by letting the server send his transcript of the current protocol run (e.g., the full password file) to the client, symmetrically encrypted with the session key. The client decrypts and checks whether the server executed the protocol with a password close enough to his own. Crucially, a corrupted client can only decrypt (and thus learn the server's secrets) if he holds a close enough password, since otherwise he will not know the encryption key.

Our proof of security is in the UC model and thus our protocol features composability guarantees and security even in the presence of adversarially-chosen passwords. As shown in [Hes19], strong idealized assumptions are necessary in order to achieve security in the UC model in case of asymmetric PAKE protocols. The reason lies in the adaptive nature of a server compromise attack (an adversary stealing the password file), against which our fuzzy version of asymmetric PAKE should also provide some protection. And indeed, our proof is in the generic group model and additionally requires encryption to be modeled as an ideal cipher. Both assumptions provide our simulator with the power to monitor off-line password guesses (*observability*) of the environment as well as to adjust a password file to contain a specific password even after having revealed the file (*programmability*)[4]. As a technicality, usage of the generic group model requires the client to perform decoding *in the exponent*. We give an example of a code that is decodable in the exponent.

Our second construction Π_{transf} is a "naive" approach of building fuzzy aPAKE from aPAKE. Namely, for a given pw, a server could simply store a list of, say, k hashes $H(\mathsf{pw}')$ for all pw' close enough to pw. Then, client and server execute k times an aPAKE protocol, with the client entering the same password every time and the server entering all hashes one by one. The fully secure protocol would need to protect against malicious behavior, e.g., by having both parties prove correct behavior. Unfortunately, this approach has two drawbacks. First, it does not scale asymptotically and has huge password files and communication overhead depending not only on the fuzziness threshold but also on the size of the password. Second, we show that Π_{transf} cannot be considered a secure fuzzy aPAKE, but has slightly weaker security guarantees.

On the plus side, Π_{transf} is already practical (and sufficiently secure) for applications where only few passwords should let the client pass. Facebook's authentication protocol, for example, is reported to correct capitalization of the first letter [Ale15], resulting in only two hashes to be stored in the password file. As analyzed in [CAA+16, CWP+17], correcting few common typographical mistakes as, e.g., accidental caps lock, increases usability significantly more than it decreases security. For such applications, our protocol Π_{transf} is a good choice.

1.1 Roadmap

In Section 2 we give a definition of our main building blocks, error-correcting codes which are decodable in the exponent. In Section 3, we provide the formal definition of fuzzy aPAKE and discuss the design of our functionality. Our fuzzy aPAKE protocol can be found in Section 4. Our naive approach of building faPAKE from aPAKE can be found in Section 5. Efficiency is considered in Section 6.

[4] We mention that already the fuzzy PAKE construction for Hamming distance from [DHP+18] relies on both the ideal cipher and random oracle model. Usage of the generic group model (together with a random oracle) has been recently shown useful in constructing strongly secure aPAKEs [BJX19].

2 Preliminaries

2.1 Robust Secret Sharing in the exponent

An l-out-of-n secret sharing scheme allows to share a secret value s into n shares $(s_1, \cdots, s_n)$ in such a way that given at least l of these shares, the secret can be reconstructed. Simultaneously, any tuple of shares smaller than l is distributed independently of s. *Robust secret sharing* (RSS) [CDD+15] improves upon secret sharing schemes in the presence of malicious shares. Intuitively, an $(n, l-1, r)_q$-RSS is an l-out-of-n secret sharing scheme which allows the presence of up to $n-r$ corrupted shares. In detail the reconstruction of the secret is reliable for an n-tuple input $(\hat{s}_1, \cdots, \hat{s}_n)$ of r different secret shares s_i and $n-r$ random values a_i even if the positions of the correct shares are unknown.

We recall the definition of RSS as stated in [DHP+18]. For a vector $c \in \mathbb{F}_q^n$ and a set $A \subseteq [n]$, we denote with c_A the projection $\mathbb{F}_q^n \to \mathbb{F}_q^{|A|}$, i.e., the sub-vector $(c_i)_{i \in A}$.

Definition 1. *Let $\lambda \in \mathbb{N}$, q a λ-bit prime, $\mathbb{F}_q$ a finite field and $n, l, r \in \mathbb{N}$ with $l < r \leq n$. An $(n, l, r)_q$ robust secret sharing scheme (RSS) consists of two probabilistic algorithms $\mathsf{Share} : \mathbb{F}_q \to \mathbb{F}_q^n$ and $\mathsf{Rec} : \mathbb{F}_q^n \to \mathbb{F}_q$ with the following properties:*

- l-privacy*: for any $s, s' \in \mathbb{F}_q, A \subset [n]$ with $|A| \leq l$, the projections c_A of $c \xleftarrow{\$} \mathsf{Share}(s)$ and c'_A of $c' \xleftarrow{\$} \mathsf{Share}(s')$ are identically distributed.*
- r-robustness*: for any $s \in \mathbb{F}_q, A \subset [n]$ with $|A| \geq r$, any c output by $\mathsf{Share}(s)$, and any $\tilde{c}$ such that $c_A = \tilde{c}_A$, it holds that $\mathsf{Rec}(\tilde{c}) = s$.*

We now introduce a variant of RSS which produces shares that are hidden in the exponent of some group G, and which features a reconstruction algorithm that can handle shares in the exponent. At the same time we sacrifice absolute correctness of Rec and allow for a negligible error in the definition of robustness.

Definition 2 (Robust Secret Sharing in the Exponent). *Let $\lambda \in \mathbb{N}$, q a λ-bit prime, $\mathbb{F}_q$ a finite field and $n, l, r \in \mathbb{N}$ with $l < r \leq n$. Let $RSS = (\mathsf{Share}', \mathsf{Rec}')$ be a $(n, l, r)_q$ robust secret sharing scheme and let $G = \langle g \rangle$ be a cyclic group of prime order q. An $(n, l, r)_q$ robust secret sharing scheme in the exponent (RSSExp) with respect to G consists of two probabilistic algorithms $\mathsf{Share} : \mathbb{F}_q \to G^n$ and $\mathsf{Rec} : G^n \to G$ which are defined as follows:*

- $\mathsf{Share}(s)$ *: On input a secret value $s \leftarrow \mathbb{F}_q$, obtain secret shares $(s_1, \cdots, s_n) \leftarrow \mathsf{Share}'(s)$ and output $(g^{s_1}, \cdots, g^{s_n})$.*
- $\mathsf{Rec}(g^{\hat{s}_1}, \cdots, g^{\hat{s}_n})$ *: On input n group elements, this algorithm outputs $g^{\hat{s}}$, where $\hat{s} \leftarrow \mathsf{Rec}'(\hat{s}_1, \cdots, \hat{s}_n)$.*

Further, an (n, l, r)-RSSExp scheme fulfills the following properties:

- *l-privacy: as in Definition 1.*
- *r-robustness: for any $s \in \mathbb{F}_q, A \subset [n]$ with $|A| \geq r$, any c output by $\mathsf{Share}(s)$, and any $\tilde{c}$ such that $c_A = \tilde{c}_A$, it holds that $\mathsf{Rec}(\tilde{c}) = g^s$ with overwhelming probability in n.*

Note that any (n, l, r)-RSSExp scheme trivially fulfills the l-privacy property. In the next part of this section we show how to achieve r-robustness.

Instantiations of RSSExp In [DHP+18], it is shown how to construct an RSS scheme from any *maximum distance separable* (MDS) code. An $(n+1, k)_q$ MDS code is a linear q-ary code of length n and rank k, which can correct up to $\lfloor (n-k+1)/2 \rfloor$ errors. We refer to [Rot06] for a more in depth introduction to linear codes.

Concretely, [DHP+18] propose to use Reed-Solomon codes, which are closely related to Shamir's secret sharing scheme [MS81]. In general, we are not aware of any RSS scheme that is not also an MDS code. For this reason, we focus now on decoding algorithms of linear codes.

Which decoding alorithm works also in the exponent? In the following Lemma we show that it is possible to build an $(n, l-1, l+t, g)$-RSSExp scheme from an l-out-of-$(l+2t)$ Shamir's secret sharing scheme.

Lemma 1. *Let $n, l \in \mathbb{N}$ and $(\mathsf{Share}', \mathsf{Rec}')$ be an l-out-of-n Shamir's secret sharing scheme with $n = l + 2t$ for some t and $t \cdot l = \mathcal{O}(n \log n)$, $G = \langle g \rangle$ a cyclic group of order q. Further let Share be the algorithm that outputs $g^{\mathsf{Share}'(s)}$ on input $s \in \mathbb{F}_q$. Then there exists an algorithm Rec using $poly(n) \cdot \mathcal{O}(\log q)$ group operations such that $(\mathsf{Share}, \mathsf{Rec})$ is an $(n, l-1, l+t)$-RSSExp scheme with respect to G.*

Proof. $(l-1)$-privacy of l-out-of-n Shamir's secret sharing scheme is shown in [DHP+18], Lemma 5, and can be directly applied to the case where shares are lifted to the exponent of some group. Let Rec be the "unique decoding by randomized enumeration" algorithm defined by Canetti and Goldwasser [CG99] (essentially, the algorithm decodes random subsets of shares until it finds redundancy), but applied to shares in the exponent using, e.g., Lagrange interpolation. Peikert [Pei06] shows in his Proposition 2.1 that, if $t < (n+1-l)/2$ (i.e., the number of errors allows for unique decoding) and $t \cdot l = \mathcal{O}(n \log n)$, then Rec succeeds with overwhelming probability in n and requires $poly(n) \cdot \mathcal{O}(\log q)$ group operations. Since $n = l+2t$, it holds that $t < (n+1-l)/2$ and hence $(l+t)$-robustness is achieved.

3 Security Model

We now present our security definition for asymmetric fuzzy password authenticated key exchange (Π_{faPAKE}). Our functionality combines the fuzzy PAKE functionality $\mathcal{F}_{\mathsf{fPAKE}}$ from [DHP+18] with the asymmetric PAKE functionality $\mathcal{F}_{\mathsf{apwKE}}$ [GMR06] (with revisions due to [Hes19]). In order to capture the notion of fuzziness in our model, we say that a key exchange using passwords pw and pw' is successful if $d(pw, pw') \leq \delta$, where d is an arbitrary distance function and δ a fixed threshold. $\mathcal{F}_{\mathsf{fPAKE}}$ can be parametrized with arbitrary functions $hdist()$ such as Hamming distance or edit distance.

Roles: In this work we consider an asymmetric setting, namely a client $\mathcal{P}_\mathcal{C}$ and a server $\mathcal{P}_\mathcal{S}$. Each party executes different code. In this setting $\mathcal{P}_\mathcal{C}$ uses a password pw while $\mathcal{P}_\mathcal{S}$ has access to some value denoted by FILE, which is generated from a password pw' but does not immediately reveal pw'. The goal of $\mathcal{P}_\mathcal{C}$ is convincing $\mathcal{P}_\mathcal{S}$ that $d(pw, pw') \leq \delta$, while $\mathcal{P}_\mathcal{S}$ only has access to FILE (and does not have access to pw').

Modeling Adversarial Capabilities: The standard security requirement for PAKE is that an attacker is bound to one password guessing attempt per run of the protocol. This resistance to off-line dictionary attacks is also featured by our functionality $\mathcal{F}_{\mathsf{faPAKE}}$ via the TESTPWD interface that can be called by the adversary only once per session. Since we are in the setting of asymmetric PAKE, however, the adversary can also gain access to the password file FILE by compromising the server. Such a compromise is essentially a corruption query with the effect that a part of the internal state of the server is leaked to the adversary. However, opposed to standard corruption, the adversary is not allowed to control the party or modify its internal state. $\mathcal{F}_{\mathsf{faPAKE}}$ provides an interface for server compromise named STEALPWDFILE. As a consequence of such a query (which, as natural for corruption queries, can only be asked by the adversary upon getting instructions from the environment), a dictionary attack becomes possible. Such an attack is reflected in $\mathcal{F}_{\mathsf{faPAKE}}$ by the OFFLINETESTPWD interface, which allows an unbounded number of password guesses. Accounting for protocols that allow precomputation of, e.g., hash tables of the form $H(\mathsf{pw})$, $\mathcal{F}_{\mathsf{faPAKE}}$ accepts OFFLINETESTPWD queries already *before* STEALPWDFILE was issued. $\mathcal{F}_{\mathsf{faPAKE}}$ silently stores these guesses in the form of (OFFLINE, pw) records. Upon STEALPWDFILE, $\mathcal{F}_{\mathsf{faPAKE}}$ sends the client's pw_C to the adversary in case a record (OFFLINE, pw_C) exists. This models the fact that the adversary learns the client's password from his precomputated values only upon learning the password file, i.e., compromising the server[5]. Besides offline password guesses, the adversary can use FILE of the compromised server to run a key exchange session with the user. This is captured within the IMPERSONATE interface.

[5] Recent PAKE protocols [JKX18, BJX19] have offered resistance against so-called precomputation attacks, where an attacker should not be able to pre-compute any values that can be used in the dictionary attack. Our protocols do not offer such guarantees.

All these interfaces were already present in aPAKE functionalities in the literature. The key difference of $\mathcal{F}_{\mathsf{faPAKE}}$ is now that all these interfaces apply fuzzy matching when it comes to comparing passwords. Namely, $\mathcal{F}_{\mathsf{faPAKE}}$ is parametrized with two thresholds δ and γ. δ is the "success threshold", for which it is guaranteed that passwords within distance δ enable a successful key exchange. On the other hand, γ can be seen as the "security threshold", with $\gamma \geq \delta$. Guessing a password within range γ does not enable the adversary to successfully exchange a key, but it might provide him with more information than just "wrong guess". Following [DHP+18], we enable weakenings of $\mathcal{F}_{\mathsf{faPAKE}}$ in terms of leakage from adversarial interfaces (cf. Figure 2). Here, the adversary, in addition to learning whether or not his password guess was close enough, is provided with the output of different leakage functions L_c, L_m and L_f. Essentially, he learns $L_c(\mathsf{pw}, \mathsf{pw}')$ if his guess was within range δ of the other password, L_m if it was within range $\gamma > \delta$ and L_f if it was further away than γ. $\mathcal{F}_{\mathsf{faPAKE}}$ can be instantiated with any thresholds γ, δ and arbitrary functions L_c, L_m, L_f. Looking ahead, the additional threshold γ enables us to prove security of constructions using building blocks such as error-correcting codes, which come with a "gray zone" where reliable error correction is not possible, but also the encoded secret is not information-theoretically hidden. While guessing a password in this gray zone does not enable an attacker to reliably compute the same password as the client, security is still considered to be compromised since some information about the honest party's password (and thus her key) might be leaked. To keep the notion flexible, we allow describing the amount of leakage with $L_m(\cdot, \cdot)$ and mark the record `compromised` to model partial leakage of the key.

Naturally, one would aim for δ and γ to be close, where $\delta = \gamma$ offers optimal security guarantees in terms of no special adversarial leakage if passwords are only $\delta + 1$ apart (an equivalent formulation would be to set $L_m = L_f$). $\mathcal{F}_{\mathsf{faPAKE}}$ is strongest if $L_f = L_m = L_c = \perp$. Below we provide examples of nontrivial leakage functions, verbatim taken from [DHP+18].

Since in a fuzzy aPAKE protocol the password file stored at the server needs to allow for fuzzy matching, files are required to store the password in a structured or algebraic form. An adversary stealing the file could now attempt to alter the file to contain a different (still unknown) password. This kind of attack does not seem to constitute a real threat, since the attacker basically just destroyed the file and cannot use it anymore to impersonate the server towards the corresponding client. To allow for efficient protocols, we therefore choose to incorporate malleability of password files into our functionality $\mathcal{F}_{\mathsf{faPAKE}}$ by allowing the adversary to present a function f within an Impersonate query. The impersonation attack is then carried out with $f(pw)$ instead of pw, where pw denotes the server's password.

Figure 1 depicts $\mathcal{F}_{\mathsf{faPAKE}}$ with the set of leakage functions from the second example below, namely leaking whether the password is close enough to derive a common cryptographic key.

Examples of leakage functions.

1. *No leakage.* The strongest option is to provide no feedback at all to the adversary. We define $\mathcal{F}^N_{\mathsf{faPAKE}}$ to be the functionality described in Figure 1, except that TestPwd, Impersonate, OfflineTestPwd and StealPwdFile use the check depicted in Figure 2 with
$$L^N_c(\mathsf{pw}, \mathsf{pw}') = L^N_m(\mathsf{pw}, \mathsf{pw}') = L^N_f(\mathsf{pw}, \mathsf{pw}') = \perp .$$

2. *Correctness of guess.* The basic functionality $\mathcal{F}_{\mathsf{faPAKE}}$, described in Figure 1, leaks the correctness of the adversary's guess. That is, in the language of Figure 2,
$$\begin{aligned} L_c(\mathsf{pw}, \mathsf{pw}') &= \text{"correct guess"}, \\ \text{and} \qquad L_m(\mathsf{pw}, \mathsf{pw}') &= L_f(\mathsf{pw}, \mathsf{pw}') = \text{"wrong guess"}. \end{aligned}$$

3. *Matching positions ("mask").* Assume the two passwords are strings of length n over some finite alphabet, with the jth character of the string pw denoted by $\mathsf{pw}[j]$. We define $\mathcal{F}^M_{\mathsf{faPAKE}}$ to be the functionality described in Figure 1, except that TestPwd, Impersonate, OfflineTestPwd and StealPwdFile use the check depicted in Figure 2, with L_c and L_m that leak the indices at which the guessed password differs from the actual one when the guess is close enough (we will call this leakage the *mask* of the

The functionality $\mathcal{F}_{\mathsf{faPAKE}}$ is parameterized by a security parameter λ and tolerances $\delta \leq \gamma$. It interacts with an adversary $\mathcal{S}$ and a client and a server party $\mathcal{P} \in \{\mathcal{P}_C, \mathcal{P}_S\}$ via the following queries:

Password Registration

- On $(\textsc{StorePwdFile}, \mathsf{sid}, \mathcal{P}_C, \mathsf{pw})$ from $\mathcal{P}_S$, if this is the first StorePwdFile message, record $(\textsc{file}, \mathcal{P}_C, \mathcal{P}_S, \mathsf{pw})$ and mark it `uncompromised`.

Stealing Password Data

- On $\boxed{(\textsc{StealPwdFile}, \mathsf{sid})}$ from $\mathcal{S}$, if there is no record $(\textsc{file}, \mathcal{P}_C, \mathcal{P}_S, \mathsf{pw})$, return "no password file" to $\mathcal{S}$. Otherwise, if the record is marked `uncompromised`, mark it `compromised`; regardless, for all records $(\textsc{offline}, \mathsf{pw}')$ set $d \leftarrow d(\mathsf{pw}, \mathsf{pw}')$ and do:
  - If $d \leq \delta$, send ("correct guess", pw') to $\mathcal{S}$;

  If no such pw' is recorded, return "password file stolen" to $\mathcal{S}$.
- On $(\textsc{OfflineTestPwd}, \mathsf{sid}, \mathsf{pw}')$ from $\mathcal{S}$, do:
  - If there is a record $(\textsc{file}, \mathcal{P}_C, \mathcal{P}_S, \mathsf{pw})$ marked `compromised`, then set $d \leftarrow d(\mathsf{pw}, \mathsf{pw}')$ and do:
    - If $d \leq \delta$, mark record `compromised` and send "correct guess" to $\mathcal{S}$;
    - If $d > \delta$, mark record `interrupted` and send "wrong guess" to $\mathcal{S}$.
  - Else, record $(\textsc{offline}, \mathsf{pw}')$

Password Authentication

- On $(\textsc{UsrSession}, \mathsf{sid}, \mathsf{ssid}, \mathcal{P}_S, \mathsf{pw}')$ from $\mathcal{P}_C$, send $(\textsc{UsrSession}, \mathsf{sid}, \mathsf{ssid}, \mathcal{P}_C, \mathcal{P}_S)$ to $\mathcal{S}$. Also, if this is the first UsrSession message for ssid, record $(\mathsf{ssid}, \mathcal{P}_C, \mathcal{P}_S, \mathsf{pw}')$ and mark it `fresh`.
- On $(\textsc{SrvSession}, \mathsf{sid}, \mathsf{ssid})$ from $\mathcal{P}_S$, retrieve $(\textsc{file}, \mathcal{P}_C, \mathcal{P}_S, \mathsf{pw})$ and send $(\textsc{SrvSession}, \mathsf{sid}, \mathsf{ssid}, \mathcal{P}_C, \mathcal{P}_S)$ to $\mathcal{S}$. Also, if this is the first SrvSession message for ssid, record $(\mathsf{ssid}, \mathcal{P}_S, \mathcal{P}_C, \mathsf{pw})$ and mark it `fresh`.

Active Session Attacks

- On $(\textsc{TestPwd}, \mathsf{sid}, \mathsf{ssid}, \mathcal{P}, \mathsf{pw}')$ from $\mathcal{S}$, if there is a record $(\mathsf{ssid}, \mathcal{P}, \mathcal{P}', \mathsf{pw})$ marked `fresh`, then set $d \leftarrow d(\mathsf{pw}, \mathsf{pw}')$ and do:
  - If $d \leq \delta$, mark record `compromised` and send "correct guess" to $\mathcal{S}$;
  - If $d > \delta$, mark record `interrupted` and send "wrong guess" to $\mathcal{S}$.
- On $(\textsc{Impersonate}, \mathsf{sid}, \mathsf{ssid}, f)$ from $\mathcal{S}$, if there is a record $(\mathsf{ssid}, \mathcal{P}_C, \mathcal{P}_S, \mathsf{pw})$ marked `fresh` and a record $(\textsc{file}, \mathcal{P}_C, \mathcal{P}_S, \mathsf{pw}')$ marked `compromised`, then set $d \leftarrow d(\mathsf{pw}, f(\mathsf{pw}'))$ and do:
  - If $d \leq \delta$, mark record `compromised` and send "correct guess" to $\mathcal{S}$;
  - If $d > \delta$, mark record `interrupted` and send "wrong guess" to $\mathcal{S}$.

Key Generation and Implicit Authentication

- On $(\textsc{NewKey}, \mathsf{sid}, \mathsf{ssid}, \mathcal{P}, \mathsf{k})$ from $\mathcal{S}$ where $|\mathsf{k}| = \lambda$ or $\mathsf{k} = \perp$, if there is a record $(\mathsf{ssid}, \mathcal{P}, \mathcal{P}', \mathsf{pw})$ not marked `completed`, do:
  - If the record is marked `compromised`, or either $\mathcal{P}$ or $\mathcal{P}'$ is corrupted, send $(\mathsf{sid}, \mathsf{ssid}, \mathsf{k})$ to $\mathcal{P}$.
  - Else if the record is marked `fresh`, $(\mathsf{sid}, \mathsf{ssid}, \mathsf{k}')$ was sent to $\mathcal{P}'$, and at that time there was a record $(\mathsf{ssid}, \mathcal{P}', \mathcal{P}, \mathsf{pw})$ with $d(\mathsf{pw}, \mathsf{pw}') \leq \delta$ marked `fresh`, send $(\mathsf{sid}, \mathsf{ssid}, \mathsf{k}')$ to $\mathcal{P}$.
  - Else if $\mathsf{k} \neq \perp$, the record is marked `fresh`, $(\mathsf{sid}, \mathsf{ssid}, \mathsf{k}')$ was sent to $\mathcal{P}'$, and at that time there was a record $(\mathsf{ssid}, \mathcal{P}', \mathcal{P}, \mathsf{pw})$ with $d(\mathsf{pw}, \mathsf{pw}') \leq \delta$ marked `fresh`, send $(\mathsf{sid}, \mathsf{ssid}, \mathsf{k}')$ to $\mathcal{P}$.
  - Else, pick $\mathsf{k}'' \xleftarrow{\$} \{0,1\}^\lambda$ and send $(\mathsf{sid}, \mathsf{ssid}, \mathsf{k}'')$ to $\mathcal{P}$.

  Finally, mark $(\mathsf{ssid}, \mathcal{P}, \mathcal{P}', \mathsf{pw})$ `completed`.

Fig. 1: Ideal functionality $\mathcal{F}_{\mathsf{faPAKE}}$. Framed queries can only be asked upon getting instructions from $\mathcal{Z}$.

– If $d \leq \delta$, mark the record `compromised` and reply to $\mathcal{S}$ with $L_c(\mathsf{pw}, \mathsf{pw}')$;
– If $\delta < d \leq \gamma$, mark the record `compromised` and reply to $\mathcal{S}$ with $L_m(\mathsf{pw}, \mathsf{pw}')$;
– If $\gamma < d$, mark the record `interrupted` and reply to $\mathcal{S}$ with $L_f(\mathsf{pw}, \mathsf{pw}')$.

Fig. 2: Modified distance checks to allow for different leakage to be used in TestPwd, OfflineTestPwd, Impersonate and StealPwdFile. In StealPwdFile, record marking is skipped.

passwords). That is,

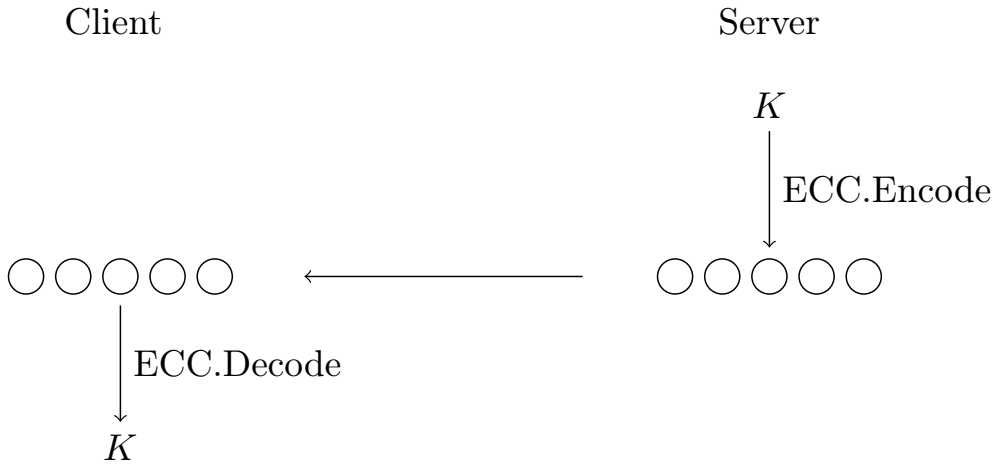
$$L_c^M(\mathsf{pw}, \mathsf{pw}') = (\{j \text{ s.t. } \mathsf{pw}[j] = \mathsf{pw}'[j]\}, \text{“correct guess”}),$$
$$L_m^M(\mathsf{pw}, \mathsf{pw}') = (\{j \text{ s.t. } \mathsf{pw}[j] = \mathsf{pw}'[j]\}, \text{“wrong guess”})$$
$$\text{and} \quad L_f^M(\mathsf{pw}, \mathsf{pw}') = \text{“wrong guess”}.$$

4. *Full password.* The weakest definition — or the strongest leakage — reveals the entire actual password to the adversary *if the password guess is close enough.* We define $\mathcal{F}_{\mathsf{faPAKE}}^P$ to be the functionality described in Figure 1, except that TestPwd, Impersonate, OfflineTestPwd and StealPwdFile use the check depicted in Figure 2, with

$$L_c^P(\mathsf{pw}, \mathsf{pw}') = L_m^P(\mathsf{pw}, \mathsf{pw}') = \mathsf{pw} \quad \text{and} \quad L_f^P(\mathsf{pw}, \mathsf{pw}') = \text{“wrong guess”}.$$

4 Fuzzy aPAKE from Secret Sharing

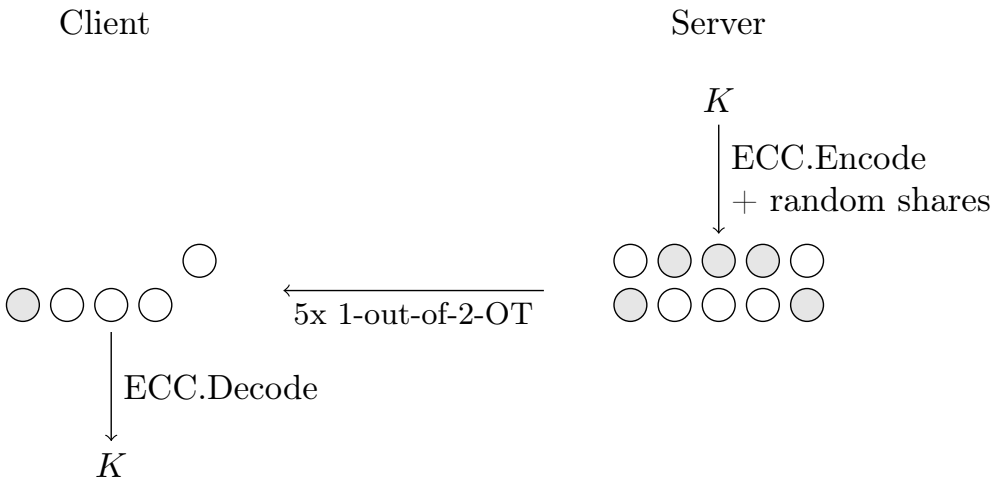
We now describe our protocol for fuzzy aPAKE with Hamming distance as metric for closeness of passwords. The very basic structure of our protocol is as follows: we let the server encode a cryptographic key K using an error-correcting code[6]. The resulting codeword (different parts of codeword are depicted as white circles in the illustration below) is then transmitted to the client, who decodes to obtain the key.



To make the retrieval of the cryptographic key password-dependent, the server stores the codeword together with randomness (depicted as grey circles below) in a password file. The position of the true codeword values in the file are dictated by the password bits. For example, in the illustration below, the server uses the password 01110. For this, we require the encoding algorithm to output codewords whose

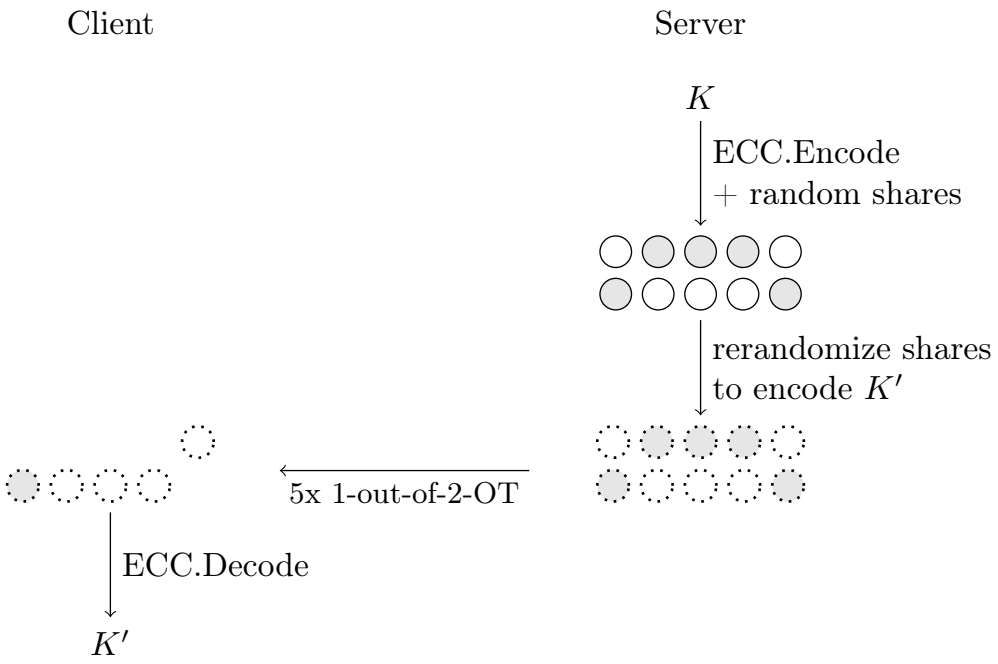
[6] Formally, we will define our scheme using the more general concept of robust secret sharing. However, for this overview it will be convenient to use the terminology of error-correcting codes.

dimension matches the number of password bits. Now instead of getting the full password file, the client can choose to see only one value per column (either a part of the codeword or a random value). Technically, this is realized by employing a n-time 1-out-of-2 oblivious transfer (OT) protocol [7], where $n = 5$ is the password size of our toy example. The oblivious part is crucial to keep the server from learning the client's password. With this approach, passwords within the error correction threshold of the password used by the server are sufficient to let the client decode the cryptographic key. In the illustration below, the client uses password 11110, letting him obtain 4/5 of the codeword correctly. Furthermore, an adversary stealing the password file is now faced with the computationally expensive task of finding the codeword within the file. Generalized to an $(n - 2t)$-out-of-n RSS, the naive approach of finding $n - 2t$ shares of the codeword by taking random subsets succeeds with probability $1/2^{n-2t}$ (as there are $\binom{n}{2t}$ "good" choices containing shares only, and $\binom{n}{2t} \cdot 2^{n-2t}$ choices overall). Here, n is the password size and t the number of errors that the fuzzy aPAKE protocol allows in passwords.



The above protocol can only be used to derive a single cryptographic key. Further, it is prone to a malicious client who could send pw and $\mathsf{pw} \oplus 1^n$ in two subsequent runs and obtain the full password file. The solution is randomization of the password file in each run of the protocol. This is straightforward for linear secret sharing.

[7] The protocol is not restricted by 1-out-of-2 OT, but can use 1-out-of-n OT for any $n \in \mathbb{N}$. In this work we consider $n = 2$, but in practice $n > 2$ might be useful to reduce the number of wrong shares (e.g. $n = 2^7$ in case of ASCII encoding).



Unfortunately, the above protocol cannot be proven UC secure. As already mentioned before, UC-secure asymmetric PAKE protocols require an idealized assumption to reveal password guesses against the file to the adversary [Hes19]. Furthermore, we need to require that a password file does not fix the password that is contained in it, in order to prove security in the presence of adaptive server compromise attacks. To remedy the situation, we let the server store the password file in the exponent of a publicly known large group and prove security of our construction in the generic group model [Sho97]. As a consequence, the client now needs to perform decoding in the exponent. We summarize in Section 2 which known decoding techniqes work also in the exponent, and detail in Section 6 how this affects the parameter choices of our scheme.

To complete our high-level protocol description, we now consider malicious behavior of client and server in the above protocol. Firstly, we observe that the client cannot cheat apart from using a different password in the OT (which does not constitute an attack) or outputting a wrong cryptographic key (which also does not constitute an attack). Things look differently when we consider a malicious server. The server could, e.g., deviate from the protocol by entering only correct codeword parts in the OT, making the key exchange succeed regardless of the password the client is using. To prevent such attacks, we let the server prove correct behavior by encrypting his view of the protocol run under the symmetric key K'. The view consists of the randomized password file as well as g^{pw}. A client being able to derive K' can now check whether the server indeed holds a password pw close enough to his own, and whether the transmitted password file parts match the password file created with pw. The formal description of our protocol can be found in Figure 3.

It is worth noting the similarity of our protocol to the fuzzy PAKE from RSS/ECC of [DHP+18]. Namely, the overall idea is the same (server choosing and encoding K, sending it to the client who can decode if and only if his password is close enough). Essentially, both protocols transmit the codeword *encrypted* with the password, using a symmetric cipher that tolerates errors in the password - let us call this a *fuzzy symmetric cipher*. [DHP+18] uses the following fuzzy symmetric cipher: XOR the codeword (the message) with cryptographic keys derived from the individual password bits. These cryptographic keys are exchanged using PAKE on individual password bits. Unfortunately, this approach does not work in the asymmetric setting, since the server would have to store the password in the clear to access its individual bits. For the asymmetric case, one has to come up with a fuzzy cipher that works with a key that is some function of the password. This function needs to have two properties: hide the password sufficiently, and still allow to evaluate distance of its input.

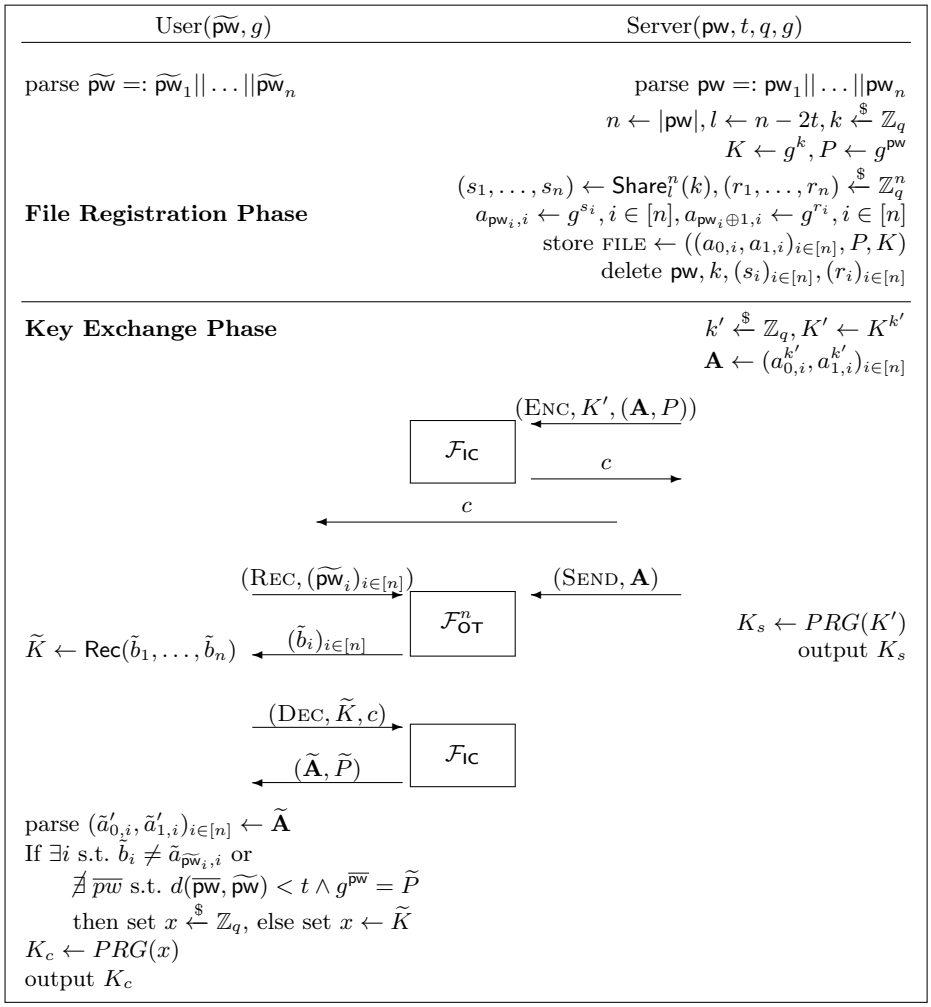


Fig. 3: Protocol Π_{faPAKE} for asymmetric fuzzy PAKE using an n times 1-out-of-2 Oblivious Transfer.

4.1 Security

Theorem 1. *Let $n, l, t \in \mathbb{N}$ with $n = l + 2t$ and $(\mathsf{Share}, \mathsf{Rec})$ be an $(n, l-1, l+t)$-RSSExp scheme with respect to a generic group G. Then the protocol depicted in Figure 3 UC-emulates $\mathcal{F}_{\mathsf{faPAKE}}^{P}$ in the $\mathcal{F}_{\mathsf{IC}}, \mathcal{F}_{\mathsf{OT}}^{n}$-hybrid model, with $\gamma = 2t$, $\delta = t$, Hamming distance $d()$ and with respect to static byzantine corruptions and adaptive server compromise.*

We now provide a proof sketch for Theorem 1. The detailed proof can be found in the full version of this paper [EHOR20].

Proof sketch: The overall proof strategy is to give a simulated transcript and output of the protocol that is indistinguishable from a real protocol execution and runs independently of the parties' passwords. The simulator is allowed to make one password guess per execution (in case of compromised server the simulator can run several offline password guesses). In the following, we describe the different cases of corruption that have to be considered.

- **Honest session:** Apart from the interaction between client and server through the UC-secure OT, the only message that needs to be simulated is one ideal cipher output which is sent from the server to the client and serves as a commitment to the servers values. Since the ideal cipher generates a uniformly random ciphertext from the ciphertext space, the simulator can replace the $\mathcal{F}_{\mathsf{IC}}$ output by a random value as long as the key is unknown. Hence, the simulator runs independently from the passwords of the parties.
- **Corrupted client:** In case of corrupted client, it is crucial to bind the client to submitting all n password bits at once such that the client is not able to adaptively change the password bits based on previous OT outputs. We achieve this by using non-adaptive n times 1-out-of-2 OT executions. Hence, $\mathcal{S}$ is able to query TestPwd on the submitted password bits before it needs to simulate the OT outputs for the client. In case TestPwd returns the server's password, $\mathcal{S}$ can simulate valid OT outputs. Otherwise, $\mathcal{S}$ chooses random outputs which is indistinguishable from the real execution due to the privacy property of the RSSExp scheme.
- **Corrupted server:** Whenever the corrupted server sends the ciphertext that contains the OT inputs and g^{pw}, $\mathcal{S}$ reconstructs pw from the inputs to the ideal cipher and the generic group operations requestes by the environment. $\mathcal{S}$ then checks whether pw is close to the client's password using the TestPwd interface. If so the simulator gets the client's password and can simulate the client. Otherwise the client's behavior is independent of its password. Hence, $\mathcal{S}$ can simulate the client with an arbitrary password that is not close to the server's.
- **Server compromise:** (1) Simulating the password file. $\mathcal{S}$ assembles a table with random group element handles as password file, and a random handle corresponding to g^{k}. As soon as $\mathcal{Z}$ starts decoding with some subset of these elements by querying the GGM, $\mathcal{S}$ learns these queries. As soon as this subset of elements corresponds to a password, the simulator submits this password to OfflineTestPwd. If the answer includes the server's password, then $\mathcal{S}$ programs the GGM such that the decoding results in the handle of g^{k}.
  (2) Impersonation attacks. The environment could use a file (e.g., the one obtained from $\mathcal{S}$ or a randomized variant of it) to impersonate the server. For this, the environment has to modify the ciphertext c to encrypt the file. Upon the environment sending an encryption query to $\mathcal{F}_{\mathsf{IC}}$ including an element P at the end of the message to be encrypted, the simulator checks if the GGM contains a tuple (pw, P). If so, $\mathcal{S}$ runs a TestPwd query on pw and learns the client's password $\widetilde{\mathsf{pw}}$ in case pw and $\widetilde{\mathsf{pw}}$ are close[8]. If there is no tuple (pw, P) in the GGM, $\mathcal{S}$ checks whether P was computed from the file (A', P') by the environment sending $f(P')$ to the GGM (and the simulator replying with P). If such a query happened, $\mathcal{S}$ issues an Impersonate query using the same function f.
- **MITM attack on honest session:** Apart from the interaction between client and server through the UC-secure OT, the only message that is sent is one ideal cipher output from the server to the client. Any attempt by $\mathcal{Z}$ to tamper with this message can be detected and hence $\mathcal{S}$ can simulate accordingly.

[8] We could alternatively let $\mathcal{S}$ issue an Impersonate query, but since the password is known issueing TestPwd works just as well.

Password Salting. In the UC modeling each protocol session has access to a fresh instantiation of the ideal functionalities. Consequently each protocol session invokes a fresh instantiation of RO or GGM, which return different values when queried on the same input in different sessions. Therefore the password files generated for two users with the same password are different. In practice however the passwords must be salted, i.e. instead of storing the g^{pw}, the server stores $g^{(\mathsf{sid}||\mathsf{pw})}$ where sid is the respective session identifier. By applying this standard technique of salting in practice, the password files for two clients who use the same password would be different.

Use Cases for Hamming Distance metric. Although hamming distance is not the most optimal way to measure the distance of two passwords, it is quite suitable for biometric applications. As an example, a server can derive the password file from a client's iris scan or fingerprint such that the client can use this biometric data for authentication. Another example would be wearable or IoT devices. Such devices can measure unique characteristics of the user or environment, such as heart beat patterns and use these measurements for authentication. Our next construction is more suitable for password matching applications where users authenticate themselves with a human memorable password, but might input some characters of the password incorrectly.

5 Fuzzy aPAKE from standard aPAKE

We now show how to construct a fuzzy aPAKE from asymmetric PAKE. Essentially, the idea is to let the server run an aPAKE protocol with the client multiple times, entering all the passwords that are close to the password he originally registered. For formally defining the protocol, it will be convenient to assume a (possibly probabilistic) function $\mathsf{close}(pw) := \{pw_i | d(\mathsf{pw}, \mathsf{pw}_i) < \delta\}$ that produces a set of all authenticating passwords. For example, for $d(), \delta$ accepting passwords where the first letter's case should be ignored, we would get $\mathsf{close}(\text{holy–moly!}) = \{\text{Holy–moly!, holy–moly!}\}$. When asking to register a password file containing pw, the server stores $\text{FILE} := \{H(\mathsf{pw}_i) | \mathsf{pw}_i \in \mathsf{close}(\mathsf{pw}) \ \forall i = 1, ..., |\mathsf{close}(\mathsf{pw})|\}$ as arbitrarily ordered list of hash values of all authenticating passwords. Let $k := |\text{FILE}|$ be the number of such passwords. Now client and server execute the aPAKE protocol k times, where the client *always* enters his password, and the server enters all values from the password file (in an order determined by a random permutation τ). Then, similar to our protocol Π_{faPAKE}, the server proves honest behavior by encrypting the (permuted) password file under all k keys generated by the aPAKE protocol. The client decrypts and looks for a password file that was generated from a password that is close to his own password. If he finds such a file, he uses the corresponding decryption key (generated from aPAKE) to perform an explicit authentication step with the server. Note that this extra round of explicit authentication cannot be skipped, since otherwise the server would not know which key to output. While the computation on the client side sounds heavy at first sight, if both parties follow the protocol, all but one decryption attempts on the client side will fail. The client can efficiently recognized a failed decryption attempt by searching the decrypted message for the hash of his own password. The protocol is depicted in Figure 4.

Π_{transf} does not scale asymptotically, neither in the size of the password nor the number of errors. As an example, for correcting only one arbitrary error in an n-bit password, the password file size is already $k = n + 1$. For correcting up to t errors, we get $k := 1 + \sum_{i=1}^{t} \binom{n}{i}$. Note that k determines not only the size of the password file but also the number of aPAKE executions. On the plus side, the construction works with arbitrary metric and distances, does not have a "security gap" between δ and γ and has reasonable computational complexity on both the client and server side.

Unfortunately Π_{transf} cannot be proven secure given the original ideal functionality $\mathcal{F}_{\mathsf{faPAKE}}$, or rather its variant with explicit authentication (see the full version of this paper [EHOR20] for more details). In a nutshell, an attacker tampering with the single aPAKE executions can issue k password guesses using arbitrary passwords from the dictionary. A fuzzy aPAKE as defined within $\mathcal{F}_{\mathsf{faPAKE}}$, however, needs to bound the attacker to use k *close* passwords. To remedy the situation we modify the TestPwd interface of our $\mathcal{F}_{\mathsf{faPAKE}}$ functionality such that it allows n single password guesses. By single guess we mean that, instead of comparing a guess to all passwords within some threshold of the password of the attacked party (as it is done

by $\mathcal{F}_{\mathsf{faPAKE}}$), it is compared to just one password. In case the client is attacked, the functionality compares with the client's password (and allows k such comparisons). In case the server is attacked, comparison is against a randomly chosen password close to the server's password[9]. Overall, the amount of information that the attacker obtains from both TestPwd interfaces is comparable: they both allow the attacker to exclude k passwords from being "close enough" to authenticate towards an honest party. Stated differently, to go through the whole dictionary D of passwords, with both TestPwd interfaces an attacker would need to tamper with $|D|/k$ key exchange sessions. We refer the reader to the full version of this paper [EHOR20] for more details regarding the modified functionalities.

We let $\mathcal{F}'_{\mathsf{faPAKE}}$ denote the ideal functionality $\mathcal{F}^{P}_{\mathsf{faPAKE}}$ with interfaces TestPwd and NewKey.

Theorem 2. *Protocol Π_{transf} UC-emulates $\mathcal{F}'_{faPAKE}$ with arbitrary distance function $d()$ and arbitrary threshold $\delta = \gamma$ in the $(\mathcal{F}_{aPAKE}, \mathcal{F}_{RO}, \mathcal{F}_{IC})$-hybrid model w.r.t static corruptions and adaptive server compromise and $H()$ denoting calls to $\mathcal{F}_{RO}$.*

We now provide a proof sketch for Theorem 2. The detailed proof can be found in the full version of this paper [EHOR20].

Proof sketch. We need to consider the following attack scenarios:

- *Passive attacks*: The environment $\mathcal{Z}$ tries to distinguish uncorrupted real and ideal execution by merely observing transcript and outputs of the protocol, while providing the inputs of both honest parties. Since the outputs of the protocol are random oracle outputs and the transcript consists of a random ciphertext vector $\overrightarrow{e}$ output by the ideal cipher, $\mathcal{Z}$ cannot distinguish real outputs from simulated random values unless it queries either the ideal cipher functionality $\mathcal{F}_{\mathsf{IC}}$ or the random oracle $\mathcal{F}_{\mathsf{RO}}$ with the corresponding inputs. This can be excluded with overwhelming probability since these inputs are uniformly random values of high entropy chosen by honest parties.
- *Active message tampering*: We consider $\mathcal{Z}$ injecting a message into a protocol execution between two honest parties. The only messages being sent in unauthenticated channels are the encryption vector $\overrightarrow{e}$ and the explicit authentication message h. Replacing the message h would simply result in two different keys as output for the parties, simulatable by sending $\perp$ via NewKey. Tampering with $\overrightarrow{e}$ is a bit more tricky. Namely, we have to consider $\mathcal{Z}$ modifying only single components of $\overrightarrow{e}$. Tampering with each element of the vector $\overrightarrow{e}$ lowers the probability for the parties to output the same key. Hence, the simulator needs to adjust the probability for the parties to output the same key by forcing the functionality to only output the same session key with this exact probability, i.e., the simulator sends $\perp$ via NewKey with the inverse probability.
- *(Static) Byzantine corruption*: We consider the case where $\mathcal{Z}$ corrupts one of the parties.
  - In case of corrupted server, given an adversarially computed $\overrightarrow{e}$, the simulator extracts all k passwords used by $\mathcal{Z}$ from the server's inputs to $\mathcal{F}_{\mathsf{IC}}$ and $\mathcal{F}_{\mathsf{RO}}$ and submits them as password guess to $\mathcal{F}'_{\mathsf{faPAKE}}$ (via TestPwd). $\mathcal{S}$ then uses the answers (either "wrong guess" or the client's true password) to continue the simulation faithfully. In case the corrupted server deviates from the protocol (e.g., $\overrightarrow{e}$ does not encrypt a set of passwords generated by $\mathsf{close}()$, or sends garbage to the $\mathcal{F}_{\mathsf{aPAKE}}$ instance in which the server uses the client's password), the simulator sends $\perp$ via the NewKey interface to simulate failure of the key exchange.
  - The case of a corrupted client is handled similarly using the freedom of k individual TestPwd queries.
- *Server compromise*: The password file is simulated without knowledge of the password by sampling random hash values. The simulator now exploits observability and programmability of the random oracle (that models the hash function) as follows: as soon as $\mathcal{Z}$ wants to compute $H(\mathsf{pw})$, $\mathcal{S}$ submits pw to its OfflineTestPwd interface. Upon learning the server's true password, $\mathcal{S}$ programs the random oracle such that the password file contains hash values of all passwords close to pw.

[9] Programming this randomized behavior into the functionality greatly simplifies proving security of Π_{transf} and does not seem to weaken the functionality compared to one using non-randomized equality checks.

– *Attacking* $\mathcal{F}_{aPAKE}$: While using $\mathcal{F}_{\mathsf{aPAKE}}$ as hybrid functionality helps the parties to exchange the key, it gives us a hard time when simulating. Essentially, the simulator has to simulate answers to all adversarial interfaces of each instance of $\mathcal{F}_{\mathsf{aPAKE}}$ since $\mathcal{Z}$ is allowed to query them. And $\mathcal{F}_{\mathsf{aPAKE}}$ has a lot of them: StealPwdFile, TestPwd, OfflineTestPwd and Impersonate. In a nutshell, OfflineTestPwd queries can be answered by querying the corresponding interface at $\mathcal{F}'_{\mathsf{faPAKE}}$. The same holds for StealPwdFile and Impersonate, only that they can be queried only once in $\mathcal{F}'_{\mathsf{faPAKE}}$. Our proof thus needs to argue that the one answer provided by $\mathcal{F}_{\mathsf{faPAKE}}$ includes already enough information to simulate answers to all k. The most annoying interface, namely TestPwd is handled by forwarding each individual TestPwd guess to $\mathcal{F}'_{\mathsf{faPAKE}}$. This explains why $\mathcal{F}'_{\mathsf{faPAKE}}$ needs to allow k individual password guesses instead of one fuzzy one (as provided by $\mathcal{F}_{\mathsf{faPAKE}}$).

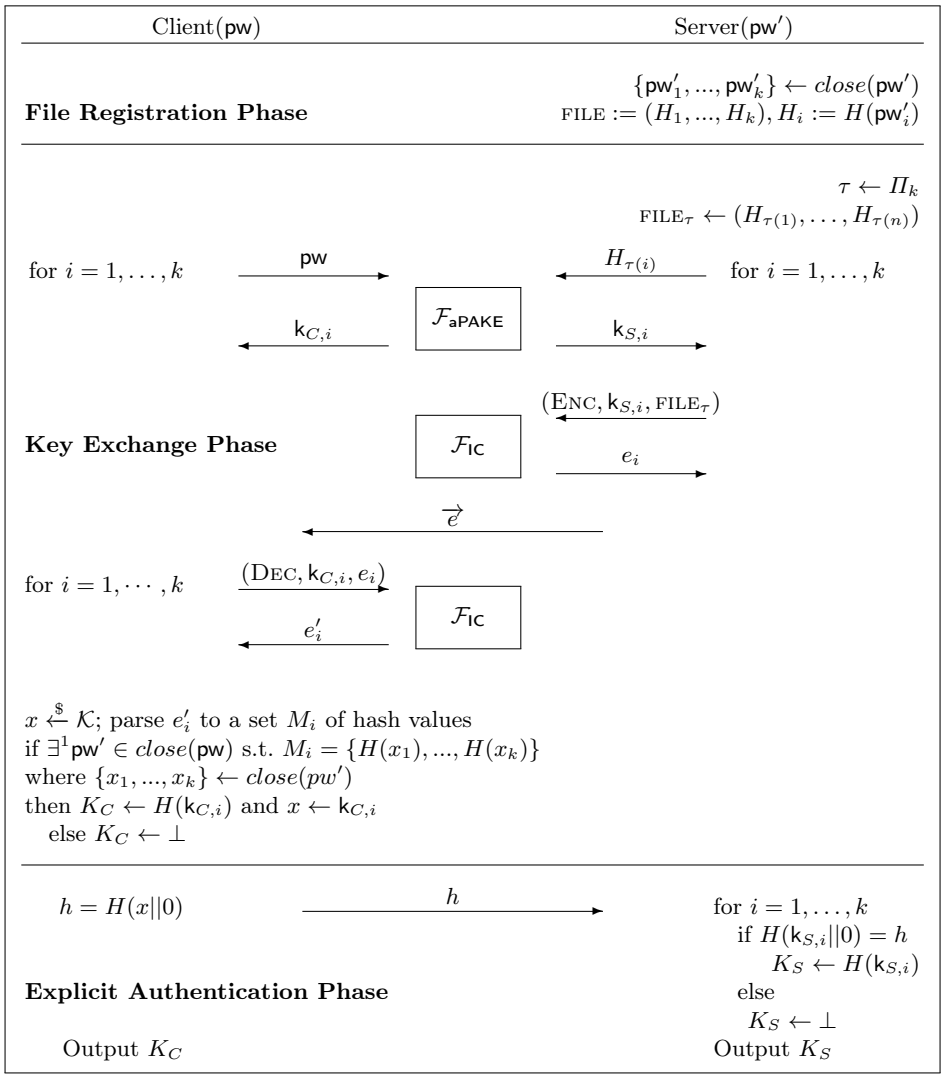


Fig. 4: Protocol Π_{transf} for fuzzy asymmetric PAKE. The parties participate in k executions of the aPAKE protocol. Afterwards they verify if at least one of the produced k keys match and agree on it. We denote $\Pi_n := perm(1, ..., k)$ the set of permutations $[k] \to [k]$. $close(\mathsf{pw})$ is a function outputting a list of all authenticating passwords (see text for a formal description).

6 Efficiency

Efficiency of Π_{faPAKE}. When instantiated with the statically secure OT from [BDD+17], Π_{faPAKE} is round-optimal and requires each party to send only one message. While 2 consecutive messages are in any case required for the OT, we can conveniently merge the ciphertext sent by the server with his message sent within the OT. In order to compute the total message size, let us first give more details on the OT instantiations that are compatible with Π_{faPAKE} and their communication complexity. Π_{faPAKE} can use any UC-secure protocol for 1-out-of-2 OT with the slight modification that the sender only continues the protocol after having received n input-dependent messages of the client (in UC-secure protocol, the client is usually committed to his input when sending his first message). E.g., one could modify the round-efficient statically secure OT protocol from [BDD+17], Figure 3, to let the sender Alice wait for receiver Bob to complete the first step of the protocol n times. The protocol requires one round of communication. In total, 3 strings, 1 public key and 2 ciphertexts are send around per 1-out-of-2 OT. For sender inputs from $\mathbb{F}_q^2$ and security parameter λ with $q = 2^\lambda$, the communication complexity of the n-fold 1-out-of-2 OT is then $8\lambda n$ bits. This results in a total message size of $8\lambda n + |c| = 8\lambda n + (2n+1)\lambda \approx 10\lambda n$ bits. For each login attempt of a client, the server needs to perform $2n + 1$ group exponentiations in order to refresh the values in the password file, as well as an encryption of $2n + 1$ group elements. Finally, the server has to perform one PRG execution. Note that the server has to do some additional computations during the initial setup phase of the protocol, however since this phase is only run once, we do not consider its complexity in this section. The client's computation is where our protocol lacks efficiency. Namely, with the naive decoding technique from [CG99], client's computation is only polynomial in $|\mathsf{pw}|$ if the error correction capability δ is not larger than $\log |\mathsf{pw}|$. And still for such δ, going beyond password sizes of, say, 40 bits does not seem feasible.

Efficiency of Π_{transf}. In order to achieve the fuzzy password matching in Π_{transf}, the server is required to store one hash value for each password that lies within distance δ of the original password. As a consequence, the password file size is highly dependent on these threshold parameters. If we consider Hamming distance as done in our first construction, for $\delta = 1$ the password file is of size $\mathcal{O}(n)$. However for $\delta = 2$ it grows to $\mathcal{O}(n^2)$ and for $\delta = 3$ to $\mathcal{O}(n^3)$. Hence, such error tolerance can only be achieved in Π_{transf} at the cost of huge password files. The same correlation to the error tolerance holds for the amount of aPAKE executions in Π_{transf}.

In order to determine the computational complexity of Π_{transf} in terms of required group operations, we chose an instantiation of an aPAKE protocol, OPAQUE [JKX18], that requires a constant number of group exponentiations. As previously discussed, Π_{transf} requires k aPAKE executions with k being the size of the password file.

Despite its shortcomings when used with Hamming distance, Π_{transf} serves as a good illustration for how to construct a general purpose faPAKE protocol that already has practical relevance. Instantiated with distance and threshold suitable to correct, e.g., capitalization of first letters or transposition of certain digits, we obtain an efficient "almost secure" fuzzy aPAKE scheme.

We present a comparison of the two schemes in Table 1. Π_{transf} is listed twice. First it is compared to Π_{faPAKE} when using Hamming distance. The last row indicates its efficiency for parameters resulting in k authenticating passwords, where k can be as small as 2.

7 Conclusion

In this paper, we initiated the study of *fuzzy asymmetric PAKE*. Our security notion in the UC framework results from a natural combination of existing functionalities. Protocols fulfilling our definition enjoy strong security guarantees common to all UC-secure PAKE protocols such as protection against off-line attacks and simulatability even when run with adversarially-chosen passwords.

We demonstrate that UC-secure fuzzy aPAKE can be build from OT and Error-Correcting Codes, where fuzziness of passwords is measured in terms of their Hamming distance. Our protocol is inspired by the ideas of [DHP+18] for building a fuzzy *symmetric* PAKE. We also show how to build a (mildly less secure)

	File size	Message size	Thresholds	Metric	Client	Server	Assumption
Π_{faPAKE}	$(2n+2)\lambda$	$10\lambda n$	$2\delta = \gamma$	Hamming	$poly(n) \cdot \mathcal{O}(\log q)$	$\mathcal{O}(n \log q)$	IC, GGM
Π_{transf}	$\mathcal{O}(n^\delta)$	$\mathcal{O}(n^\delta)$	$\delta = \gamma$	Hamming	$\mathcal{O}(n^\delta \log q)$	$\mathcal{O}(n^\delta \log q)$	IC, ROM
Π_{transf}	λk	$\mathcal{O}(k)$	$\delta = \gamma$	arbitrary	$\mathcal{O}(k)$	$\mathcal{O}(k)$	IC, ROM

Table 1: Comparison of Π_{faPAKE} and Π_{transf}. We assume n-bit passwords in case of Hamming distance. File size and communication complexity are in bits. The Client and Server column indicate the number of group operations.

fuzzy aPAKE from (non-fuzzy) aPAKE. Our construction allows for arbitrary notions of fuzziness and yields efficient, strongly secure and practical protocols for use cases such as, e.g., correction of typical orthographic errors in typed passwords.

Our two constructions nicely show the trade-offs that one can have for fuzzy aPAKE. The "naive" construction from aPAKE has large password file size when used with Hamming distance, but also works for arbitrary closeness notions possibly leading to small password files and practical efficiency. The construction using Error-Correcting Codes is restricted to Hamming distance and $\log(|\mathsf{pw}|)$ error correction threshold. I comes with a computational overhead on the client side, but has only little communication and small password file size. It is worth noting that, for this construction, all efficiency drawbacks could be remedied by finding a more efficient decoding method that works in the exponent. We leave this as well as finding more fuzzy aPAKE constructions as future work. Specifically, no fuzzy aPAKE scheme with *strong* compromise security (as defined in [JKX18]) is known.

Acknowledgments

This work was partly supported by the German Research Foundation (DFG) Emmy Noether Program *FA 1320/1-1*, by the *DFG CRC 1119 CROSSING* (project S7), by the German Federal Ministry of Education and Research (BMBF) *iBlockchain project* (grant nr. 16KIS0902), by the German Federal Ministry of Education and Research and the Hessen State Ministry for Higher Education, Research and the Arts within their joint support of the *National Research Center for Applied Cybersecurity ATHENE*, by the VeriSec project 16KIS0634 from the Federal Ministry of Education and Research (BMBF), and by the European Union's Horizon 2020 research and innovation programme under grant agreement No. 786725 – OLYMPUS.

We would like to thank Sophia Yakoubov for helpful discussions on earlier versions of this work.

References

[Ale15] Alec Muffet. Facebook: Password hashing & authentication, presentation at real world crypto, 2015.

[BBC+13] Fabrice Benhamouda, Olivier Blazy, Céline Chevalier, David Pointcheval, and Damien Vergnaud. New techniques for SPHFs and efficient one-round PAKE protocols. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 449–475. Springer, Heidelberg, August 2013.

[BBR88] Charles H. Bennett, Gilles Brassard, and Jean-Marc Robert. Privacy amplification by public discussion. *SIAM J. Comput.*, 17(2):210–229, 1988.

[BDD+17] Paulo S. L. M. Barreto, Bernardo David, Rafael Dowsley, Kirill Morozov, and Anderson C. A. Nascimento. A framework for efficient adaptively secure composable oblivious transfer in the ROM. *CoRR*, abs/1710.08256, 2017.

[BJX19] Tatiana Bradley, Stanislaw Jarecki, and Jiayu Xu. Strong asymmetric PAKE based on trapdoor CKEM. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 798–825. Springer, Heidelberg, August 2019.

[BLV19] Elette Boyle, Rio LaVigne, and Vinod Vaikuntanathan. Adversarially robust property-preserving hash functions. In Avrim Blum, editor, *ITCS 2019*, volume 124, pages 16:1–16:20. LIPIcs, January 2019.

[BM92] Steven M. Bellovin and Michael Merritt. Encrypted key exchange: Password-based protocols secure against dictionary attacks. In *1992 IEEE Symposium on Security and Privacy*, pages 72–84. IEEE Computer Society Press, May 1992.

[BM93] Steven M. Bellovin and Michael Merritt. Augmented encrypted key exchange: A password-based protocol secure against dictionary attacks and password file compromise. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM CCS 93*, pages 244–250. ACM Press, November 1993.

[BMP00] Victor Boyko, Philip D. MacKenzie, and Sarvar Patel. Provably secure password-authenticated key exchange using Diffie-Hellman. In Bart Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 156–171. Springer, Heidelberg, May 2000.

[BPR00] Mihir Bellare, David Pointcheval, and Phillip Rogaway. Authenticated key exchange secure against dictionary attacks. In Bart Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 139–155. Springer, Heidelberg, May 2000.

[CAA+16] Rahul Chatterjee, Anish Athayle, Devdatta Akhawe, Ari Juels, and Thomas Ristenpart. pASSWORD tYPOS and how to correct them securely. In *2016 IEEE Symposium on Security and Privacy*, pages 799–818. IEEE Computer Society Press, May 2016.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001.

[CDD+15] Ronald Cramer, Ivan Bjerre Damgård, Nico Döttling, Serge Fehr, and Gabriele Spini. Linear secret sharing schemes from error correcting codes and universal hash functions. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 313–336. Springer, Heidelberg, April 2015.

[CDVW12] Ran Canetti, Dana Dachman-Soled, Vinod Vaikuntanathan, and Hoeteck Wee. Efficient password authenticated key exchange via oblivious transfer. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012*, volume 7293 of *LNCS*, pages 449–466. Springer, Heidelberg, May 2012.

[CG99] Ran Canetti and Shafi Goldwasser. An efficient threshold public key cryptosystem secure against adaptive chosen ciphertext attack. In Jacques Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 90–106. Springer, Heidelberg, May 1999.

[CWP+17] Rahul Chatterjee, Joanne Woodage, Yuval Pnueli, Anusha Chowdhury, and Thomas Ristenpart. The TypTop system: Personalized typo-tolerant password checking. In Bhavani M. Thuraising-

ham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 329–346. ACM Press, October / November 2017.

[DHP+18] Pierre-Alain Dupont, Julia Hesse, David Pointcheval, Leonid Reyzin, and Sophia Yakoubov. Fuzzy password-authenticated key exchange. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part III*, volume 10822 of *LNCS*, pages 393–424. Springer, Heidelberg, April / May 2018.

[DRS04] Yevgeniy Dodis, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 523–540. Springer, Heidelberg, May 2004.

[EHOR20] Andreas Erwig, Julia Hesse, Maximilian Orlt, and Siavash Riahi. Fuzzy asymmetric password-authenticated key exchange. Cryptology ePrint Archive, Report 2020/987, 2020. `https://eprint.iacr.org/2020/987`.

[GL03] Rosario Gennaro and Yehuda Lindell. A framework for password-based authenticated key exchange. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 524–543. Springer, Heidelberg, May 2003. `http://eprint.iacr.org/2003/032.ps.gz`.

[GMR06] Craig Gentry, Philip MacKenzie, and Zulfikar Ramzan. A method for making password-based key exchange resilient to server compromise. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 142–159. Springer, Heidelberg, August 2006.

[Hes19] Julia Hesse. Separating standard and asymmetric password-authenticated key exchange. Cryptology ePrint Archive, Report 2019/1064, 2019. `https://eprint.iacr.org/2019/1064`.

[HL19] Björn Haase and Benoît Labrique. Aucpace: Efficient verifier-based PAKE protocol tailored for the iiot. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2019(2):1–48, 2019.

[JKX18] Stanislaw Jarecki, Hugo Krawczyk, and Jiayu Xu. OPAQUE: An asymmetric PAKE protocol secure against pre-computation attacks. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part III*, volume 10822 of *LNCS*, pages 456–486. Springer, Heidelberg, April / May 2018.

[KOY01] Jonathan Katz, Rafail Ostrovsky, and Moti Yung. Efficient password-authenticated key exchange using human-memorable passwords. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 475–494. Springer, Heidelberg, May 2001.

[KV11] Jonathan Katz and Vinod Vaikuntanathan. Round-optimal password-based authenticated key exchange. In Yuval Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 293–310. Springer, Heidelberg, March 2011.

[MS81] Robert J. McEliece and Dilip V. Sarwate. On sharing secrets and Reed-Solomon codes. *Commun. ACM*, 24(9):583–584, 1981.

[Pei06] Chris Peikert. On error correction in the exponent. In Shai Halevi and Tal Rabin, editors, *TCC 2006*, volume 3876 of *LNCS*, pages 167–183. Springer, Heidelberg, March 2006.

[PW17] David Pointcheval and Guilin Wang. VTBPEKE: Verifier-based two-basis password exponential key exchange. In Ramesh Karri, Ozgur Sinanoglu, Ahmad-Reza Sadeghi, and Xun Yi, editors, *ASIACCS 17*, pages 301–312. ACM Press, April 2017.

[Rot06] Ron Roth. *Introduction to Coding Theory*. Cambridge University Press, New York, NY, USA, 2006.

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer, Heidelberg, May 1997.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167. IEEE Computer Society Press, October 1986.

D. On the Related-Key Attack Security of Authenticated Encryption Schemes

This chapter corresponds to our published article at SCN 2022 [94], with minor edits. Our full version can be found in [93].

On the Related-Key Attack Security of Authenticated Encryption Schemes

Sebastian Faust[1], Juliane Krämer[2], Maximilian Orlt[1], and Patrick Struck[2]

[1] Technische Universität Darmstadt, Germany
{sebastian.faust,maximilian.orlt}@tu-darmstadt.de
[2] Universität Regensburg, Germany
{juliane.kraemer,patrick.struck}@ur.de

Abstract. Related-key attacks (RKA) are powerful cryptanalytic attacks, where the adversary can tamper with the secret key of a cryptographic scheme. Since their invention, RKA security has been an important design goal in cryptography, and various works aim at designing cryptographic primitives that offer protection against related-key attacks. At EUROCRYPT'03, Bellare and Kohno introduced the first formal treatment of related-key attacks focusing on pseudorandom functions and permutations. This was later extended to cover other primitives such as signatures and public key encryption schemes, but until now, a comprehensive formal security analysis of authenticated encryption schemes with associated data (AEAD) in the RKA setting has been missing. The main contribution of our work is to close this gap for the relevant class of nonce-based AEAD schemes.
To this end, we revisit the common approach to construct AEAD from encryption and message authentication. We extend the traditional security notion of AEAD to the RKA setting and consider an adversary that can tamper with the key K_e and K_m of the underlying encryption and MAC, respectively. We study two security models. In our weak setting, we require that tampering will change both K_e and K_m, while in our strong setting, tampering can be arbitrary, i.e., only one key might be affected. We then study the security of the standard composition methods by analysing the nonce-based AEAD schemes N1 (Encrypt-and-MAC), N2 (Encrypt-then-MAC), and N3 (MAC-then-Encrypt) due to Namprempre, Rogaway, and Shrimpton (EUROCRYPT'14). We show that these schemes are weakly RKA secure, while they can be broken under a strong related-key attack. Finally, based on the N3 construction, we give a novel AEAD scheme that achieves our stronger notion.

1 Introduction

The security of cryptographic schemes fundamentally relies on the secrecy of its keys. In particular, the secret key used by cryptographic algorithms must neither be revealed to the adversary nor must the adversary be able to change it. Unfortunately, countless advanced cryptanalytical attacks illustrate that the assumption on the secrecy of a key ceases to hold in practice. Prominent examples include side-channel attacks such as power analysis or timing attacks that partially reveal the secret key [26, 27]; or tampering and fault attacks [17], where the adversary can change the secret key and observe the effect of this change via the outputs. The latter type of attack often is referred to as a *related-key attack (RKA)*, and has been intensively studied by the research community over the last years [1, 4–6, 8, 12, 13, 16, 19, 21, 23, 24, 28, 29, 37]. But related keys may not only appear when the adversary actively tampers with the key. Another important setting where we have to deal with related keys is key updates. In this setting related-key cryptanalysis may exploit the relation of keys caused by bad key updates [14–16, 25]. Another scenario are devices with related keys. As a simple example consider a manufacturer that has some master key K. Rather than generating a fresh key for each device, it derives the key from the master key and some device id – for instance XORing the two.

The first work that provided a formal model for related-key attacks is the seminal work of Bellare and Kohno [8]. In this model, the related-key attacker can specify a related-key-deriving (RKD) function φ (from some set Φ) together with each black-box query to the cryptographic primitive, and observe the input/output behaviour for the primitive under the related key $\varphi(K)$. For instance, consider a PRF $\mathtt{F}(K, \cdot)$,

that the adversary can query on some input X. As a result of a related-key attack the adversary receives $\mathtt{F}(\varphi(K), X)$, where φ is the RKD function. Starting with [8], several works extend the notion of RKA security to a wide range of cryptographic primitives. This includes pseudorandom functions [1, 6], pseudorandom permutations [5], encryption schemes [4], and MACs [12, 37].

Somewhat surprisingly, RKA security has not been considered for the important case of authenticated encryption schemes with associated data[3] (AEAD). AEAD is a fundamental cryptographic primitive used, e.g., to secure communication in the Internet and is therefore ubiquitously deployed, especially in TLS 1.3 [32]. Lately, AEAD has received a lot of attention, for instance through the CAESAR competition [11] and the ongoing NIST standardization process on lightweight cryptography [31]. An important type of AEAD schemes, and simultaneously the focus of the NIST standardization process [31], are so-called nonce-based schemes [33]. These schemes have the advantage that they are deterministic, and hence their security does not rely on good quality randomness during encryption. Instead, they use nonces (e.g., a simple counter) and require that these nonces are never repeated to guarantee security [33].

1.1 Our Contribution

The main contribution of our work is to extend the notion of RKA security to nonce-based AEAD schemes. We study the common generic composition paradigms to construct AEAD from encryption schemes and MACs, and explore if RKA security of the underlying primitives carries over to the AEAD scheme. More concretely, let K_e and K_m be the keys of the encryption and MAC, respectively. Assuming that the encryption scheme is secure against the class Φ_e of related-key deriving functions and the MAC is secure against Φ_m, then we ask if the AEAD scheme is secure with respect to related-key derivation functions from the Cartesian product $\Phi_e \times \Phi_m$.[4] In particular, we show that under certain restrictions of $\Phi_e \times \Phi_m$ the schemes N1, N2, and N3 by Namprempre et al. [30], falling into the composition paradigms E&M, EtM, and MtE, respectively, are secure under related-key attacks. By giving concrete attacks against all schemes in case the restrictions are dropped, we show further that these restrictions are necessary. Finally, on the positive side, we give a new construction for AEAD that is secure for the general case of functions from $\Phi_e \times \Phi_m$, i.e., without the aforementioned restrictions. We provide more details on our contribution below.

RKA Security Notions for Nonce-based AEAD Schemes. We give two RKA security notions s-RKA-AE and RKA-AE for nonce-based AEAD schemes. In our weaker notion (RKA-AE), we assume that the key is updated such that each underlying primitive never uses the same key twice.[5] This is modelled by imposing an additional restriction on the adversary, where the adversary is not allowed to make queries with RKD functions that would result in keys that have already appeared during earlier RKA queries. More precisely, let K_e^i and K_m^i the result of the i-th RKA query. We require that for all i, j, we have $K_e^i = K_e^j$ if and only if $K_m^i = K_m^j$. In our stronger notion (s-RKA-AE), the above restriction is not imposed on the adversary, i.e., it is allowed to make queries i, j such that $K_e^i = K_e^j$ and $K_m^i \neq K_m^j$. Note that any adversary can trivially make such queries by repeating the RKD function for key K_e while using two different RKD functions for K_m. One may object that our weaker security notion looks rather artificial for modelling tampering attacks. We believe however that it is interesting to study for what key relations state-of-the-art AEAD constructions that are widely deployed remain secure under related-key attacks. Moreover, we emphasize another setting where such weak key relations may occur naturally – so-called bad key updates. In this setting the RKA adversary may observe ciphertexts for different related keys, where the relation stems from the key updates described by the RKD functions. Since the users update the keys, the relation between the keys is in fact not chosen by the adversary. Hence, the weaker notion guarantees security if the users ensure that, after each update, both keys K_e and K_m are fresh. In contrast, the stronger notion guarantees security even in

[3] Associated data corresponds to header information that has to be authenticated but does not need to be confidential.

[4] A similar question using the Cartesian product of the related-key deriving functions from the underlying primitives is answered in [5] for Feistel constructions.

[5] Note that the adversary can still ask for several encryptions under each key.

the case when the users might only update one of the keys. Further details on these two notions are given in Section 3.

RKA Security of the N1, N2, and N3 Construction. We study the security of the N1, N2, and N3 constructions for nonce-based AEAD schemes [30], which follow the Encrypt-and-MAC (E&M), Encrypt-then-MAC (EtM), and MAC-then-Encrypt (MtE) paradigm [9], respectively. These constructions build a nonce-based AEAD scheme from a nonce-based encryption scheme and a MAC. We show that all schemes achieve our weaker security notion, i.e., when it is ensured that both keys are updated properly. The overall proof approach is similar to the classical setting. The challenge lies in the analysis that all queries by the reduction are permitted due to the related keys. Regarding our stronger security notion, we show that all schemes have limitations. We show that N1 and N2 are insecure, irrespective of the underlying primitive, by giving concrete attacks. For N3, we show that the security crucially depends on the underlying encryption scheme, by giving an attack against any instantiation using a stream cipher. These results appear in Section 4.

RKA-secure AEAD Scheme. Finally, we give a new construction, called N*, of an AEAD scheme which is based on the N3 construction, and follows the MAC-then-Encrypt (MtE) paradigm. The underlying encryption scheme relies on an RKA-secure block cipher and a MAC. The resulting AEAD scheme achieves our stronger security notion s-RKA-AE, in fact, even in the case of nonce misuse. For simplicity we omit details regarding the nonce here, and discuss this setting more detail in Section 3. The construction and the proof is shown in Section 5.

RKA-secure Encryption and MAC from Pseudorandom Functions. We show that RKA-secure nonce-based encryption schemes and MACs can be built from RKA-secure pseudorandom functions. Combined with the results for the N1, N2, and N3 constructions, this reduces the task of constructing RKA-secure nonce-based AEAD schemes to the task of constructing RKA-secure pseudorandom functions which is a general goal in the RKA literature. More precisely, we show that the nonce-based encryption scheme and the MAC proposed by Degabriele et al. [18] in the setting of leakage-resilient cryptography achieve RKA security if the underlying pseudorandom function is RKA-secure. This is shown in the extended version of the paper [20].

1.2 Related Work

Based on the initial work by Biham [13] and Knudsen [24], the first formalisation of RKA security has been given by Bellare and Kohno [8]. They studied pseudorandom functions as well as pseudorandom permutations and showed an inherent limitation on the set of allowed RKD functions. Bellare and Cash [6] proposed RKA-secure pseudorandom functions based on the DDH assumption, which allowed a large class of RKD functions. Abdalla et al. [1] further increased the allowed class of RKD functions. Several other works study the RKA security for various primitives, e.g., pseudorandom permutations from Feistel networks [5], encryption schemes [4], and MACs [12, 37]. Harris [22], and later Albrecht et al. [3], showed inherent limitations of the Bellare-Kohno formalism by giving a generic attack against encryption schemes if the set of related-key deriving functions can depend on the primitive in question. The practical relevance of the alternative model by Harris has been questioned by Vaudenay [36].

Closer to our setting is the work by Lu et al. [29], who also study RKA security for authenticated encryption schemes. However, instead of nonce-based authenticated encryption schemes, they analyse probabilistic authenticated encryption schemes and only for the specific case of affine functions. Moreover, Han et al. [21] found their proof to be flawed, invalidating the results. To the best of our knowledge, these are the only works that consider RKA security for authenticated encryption schemes.

The practical relevance of RKA security has been shown by a number of works [16, 19, 23, 28] which present attacks against concrete primitives.

2 Preliminaries

In Section 2.1 we recall the used notation. The syntax of the cryptographic primitives and existing RKA security notions are given in Section 2.2 and Section 2.3, respectively. Additional background on security notions in the classical setting is given in the extended version of the paper [20].

2.1 Notation

By $\{0,1\}^*$ and $\{0,1\}^x$ we denote the set of bit strings with arbitrary length and length x, respectively. We refer to probabilistic polynomial-time algorithms as adversaries if not otherwise specified, and use the code-based game-playing framework by Bellare and Rogaway [10]. For a game G and adversary $\mathcal{A}$, we write $\mathsf{G}^{\mathcal{A}} \Rightarrow y$ to indicate that the output of the game, when played by $\mathcal{A}$, is y. Likewise, $\mathcal{A}^{\mathsf{G}} \Rightarrow y$ indicates that $\mathcal{A}$ outputs y when playing game G. In case $\mathcal{A}$ has access to an oracle $\mathcal{O}$ we write $\mathcal{A}^{\mathcal{O}}$. We only use distinguishing games in which an adversary $\mathcal{A}$ tries to guess a secret bit b. The advantage of $\mathcal{A}$ in such a distinguishing game G is defined as $\mathbf{Adv}^{\mathsf{G}}(\mathcal{A}) \coloneqq |2\Pr[\mathsf{G}^{\mathcal{A}} \Rightarrow \text{true}] - 1|$. Equivalent notions using adversarial advantages are $|\Pr[\mathcal{A}^{\mathsf{G}} \Rightarrow 0 \,|\, b = 0] - \Pr[\mathcal{A}^{\mathsf{G}} \Rightarrow 0 \,|\, b = 1]|$ and $|\Pr[\mathcal{A}^{\mathsf{G}} \Rightarrow 1 \,|\, b = 1] - \Pr[\mathcal{A}^{\mathsf{G}} \Rightarrow 1 \,|\, b = 0]|$. For sets $\mathcal{X}$ and $\mathcal{Y}$, the set of all functions mapping from $\mathcal{X}$ to $\mathcal{Y}$ is denoted by $\mathsf{Func}(\mathcal{X}, \mathcal{Y})$ and the set of permutations over $\mathcal{X}$ by $\mathsf{Perm}(\mathcal{X})$. We write $\mathsf{Func}(\mathcal{K}, \mathcal{X}, \mathcal{Y})$ and $\mathsf{Perm}(\mathcal{K}, \mathcal{X})$ for keyed functions in $\mathsf{Func}(\mathcal{X}, \mathcal{Y})$ and $\mathsf{Perm}(\mathcal{X})$, respectively, where $\mathcal{K}$ denotes the key space. Tables f are initialised with $\perp$ if not mentioned differently. For sets $\mathcal{S}$ and $\mathcal{T}$, we write $\mathcal{S} \xleftarrow{\cup} \mathcal{T}$ instead of $\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{T}$. Our main focus lies in the RKA setting and we use the term *classical setting* whenever we refer to the setting which does not consider related-key attacks.

2.2 Primitives

A nonce-based authenticated encryption scheme with associated data (AEAD), is a tuple of two deterministic algorithms $(\mathtt{Enc}, \mathtt{Dec})$. The encryption algorithm $\mathtt{Enc}\colon \mathcal{K} \times \mathcal{N} \times \mathcal{A} \times \mathcal{M} \to \mathcal{C}$ maps a key K, a nonce N, associated data A, and a message M, to a ciphertext C. The decryption algorithm $\mathtt{Dec}\colon \mathcal{K} \times \mathcal{N} \times \mathcal{A} \times \mathcal{C} \to \mathcal{M} \cup \{\perp\}$ maps a key K, a nonce N, associated data A, and a ciphertext C, to either a message or $\perp$ indicating an invalid ciphertext. The sets $\mathcal{K}$, $\mathcal{N}$, $\mathcal{A}$, $\mathcal{M}$, and $\mathcal{C}$, denote the key space, nonce space, associated data space, message space, and ciphertext space, respectively. An AEAD scheme is called *correct* if for any $K \in \mathcal{K}$, any $N \in \mathcal{N}$, any associated data $A \in \mathcal{A}$, and any $M \in \mathcal{M}$, it holds that $\mathtt{Dec}(K, N, A, \mathtt{Enc}(K, N, M, A)) = M$. It is called *tidy* if for any $K \in \mathcal{K}$, any $N \in \mathcal{N}$, any associated data $A \in \mathcal{A}$, any $M \in \mathcal{M}$, and any $C \in \mathcal{C}$ with $\mathtt{Dec}(K, N, A, C) = M$, it holds that $\mathtt{Enc}(K, N, A, M) = C$.

A nonce-based symmetric key encryption is similarly defined. The difference is that neither algorithm permits associated data as an input and only rejects ciphertext, i.e., outputs $\perp$, if computed on values outside the corresponding sets. For both primitives, we let c denote the length of a ciphertext.

A message authentication code (MAC) is a tuple of two deterministic algorithms $(\mathtt{Tag}, \mathtt{Ver})$. The tagging algorithm $\mathtt{Tag}\colon \mathcal{K} \times \mathcal{X} \to \{0,1\}^t$ maps a key K and message X to a tag T. The verification algorithm $\mathtt{Ver}\colon \mathcal{K} \times \mathcal{X} \times \{0,1\}^t \to \{\top, \perp\}$ takes as input a key K, a message M, and a tag T, and outputs either $\top$, indicating a valid tag, or $\perp$, indicating an invalid tag. Correctness requires that $\mathtt{Ver}(K, X, \mathtt{Tag}(K, X)) = \top$, for any $K \in \mathcal{K}$ and $X \in \mathcal{X}$. We denote the length of tags by t.

2.3 Security Notions against Related-Key Attacks

We recall some of the existing RKA security notions. All notions follow the style introduced by Bellare and Kohno [8]. That is, the set of admissible RKD functions is fixed at the start of the game. All our results, however, also apply to the alternative definition given by Harris [22], where the adversary first picks the set of RKD functions before the concrete scheme (from a family of primitives) is chosen by the game. This prevents an inherent limitation of the Bellare-Kohno formalism as the RKD function can not depend on the primitive.[6]

[6] It is questionable whether RKD functions that depend on the actual primitive are relevant in practice.

Φ-restricted Adversaries. For RKA security notions, the adversary is typically restricted to a set of functions that it can query to its oracles. This restriction is necessary, as Bellare and Kohno [8] showed that RKA security is unachievable without such restrictions. Let $\mathcal{K}$ be the key space of some primitive, then the set of permitted RKD functions is $\Phi \subset \mathsf{Func}(\mathcal{K}, \mathcal{K})$. We call an adversary that only queries functions from the set Φ to its oracles, a *Φ-restricted adversary.*

Repeating Queries. To avoid trivial wins certain queries must be excluded from the security games. In case of a MAC, we must forbid the adversary to query its challenge verification oracle on a tag it obtained from its tagging oracle. To do this, one can either adapt the game by keeping a list of such queries and let the verification oracle check for such forbidden queries. The other option, would be to simply exclude adversaries that do such queries in the security definition. For ease of exposition, we use the latter approach.

Game rkaSUF	$\mathsf{Ver}(M, T, \varphi)$	$\mathsf{Tag}(M, \varphi)$
$b \leftarrow\!\!\$\ \{0,1\}$	**if** $b = 0$	$T \leftarrow \mathtt{Tag}(\varphi(K), M)$
$K \leftarrow\!\!\$\ \mathcal{K}$	**return** $\mathtt{Ver}(\varphi(K), M, T)$	**return** T
$b' \leftarrow \mathcal{A}^{\mathsf{Ver},\mathsf{Tag}}()$	**else**	
return $(b' = b)$	**return** $\perp$	

Fig. 1: Security game rkaSUF.

RKA Security for MACs and Pseudorandom Functions/Permutations. We give the definition of related-key attack security of MACs. Existing notions define it as an unforgeability game where the adversary finally outputs a forgery attempt [12, 37]. In this work, we define unforgeability of a MAC against RKA as a distinguishing game. Here the adversary aims to distinguish whether its challenge oracle implements the real verification algorithm or simply rejects any queried tag.

Definition 1 (RKA-SUF **Security).** *Let* $\Gamma = (\mathtt{Tag}, \mathtt{Ver})$ *be a MAC and* $\Phi \subset \mathsf{Func}(\mathcal{K}, \mathcal{K})$*. Let the game* rkaSUF *be defined as in Fig. 1. For a Φ-restricted RKA adversary* $\mathcal{A}$*, that never forwards a query from its oracle* Tag*, we define its* RKA-SUF *advantage as*

$$\mathbf{Adv}^{\mathsf{rkaSUF}}_{\Gamma}(\mathcal{A}, \Phi) = 2\ \Pr[\mathsf{rkaSUF}^{\mathcal{A}} \Rightarrow \text{true}] - 1\,.$$

Games rkaPRF, rkaPRP	$\mathsf{F}(X, \varphi)$ in rkaPRF	$\mathsf{F}(X, \varphi)$ in rkaPRP
$b \leftarrow\!\!\$\ \{0,1\}$	**if** $b = 0$	**if** $b = 0$
$K \leftarrow\!\!\$\ \mathcal{K}$	$y \leftarrow F(\varphi(K), X)$	$y \leftarrow F(\varphi(K), X)$
$\mathsf{F}' \leftarrow\!\!\$\ \mathsf{Func}(\mathcal{K}, \mathcal{X}, \mathcal{Y})$	**else**	**else**
$\mathsf{P}' \leftarrow\!\!\$\ \mathsf{Perm}(\mathcal{K}, \mathcal{X})$	$y \leftarrow\!\!\$\ \mathsf{F}'(\varphi(K), X)$	$y \leftarrow \mathsf{P}'(\varphi(K), X)$
$b' \leftarrow \mathcal{A}^{\mathsf{F}}()$	**return** y	**return** y
return $(b' = b)$		

Fig. 2: Security games rkaPRF and rkaPRP.

RKA-Security for pseudorandom functions (PRFs) and pseudorandom permutations (PRPs) have been studied in many works, e.g., [3, 6–8], and are defined as the advantage in distinguishing the real function/permutation from a random function/permutation when having access to an oracle implementing either of these.

Definition 2 (RKA-PRF **Security**). *Let* $\mathtt{F}\colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ *and* $\Phi \subset \mathsf{Func}(\mathcal{K}, \mathcal{K})$. *Let the game* rkaPRF *be defined as in Fig. 2. For a* Φ*-restricted RKA adversary* $\mathcal{A}$*, that never repeats a query, we define its* RKA-PRF *advantage as*

$$\mathbf{Adv}^{\mathsf{rkaPRF}}_{\mathtt{F}}(\mathcal{A}, \Phi) = 2\ \Pr[\mathsf{rkaPRF}^{\mathcal{A}} \Rightarrow \text{true}] - 1\,.$$

Definition 3 (RKA-PRP **Security**). *Let* $\mathtt{F}\colon \mathcal{K} \times \mathcal{X} \to \mathcal{X}$ *and* $\Phi \subset \mathsf{Func}(\mathcal{K}, \mathcal{K})$. *Let the game* rkaPRP *be defined as in Fig. 2. For a* Φ*-restricted RKA adversary* $\mathcal{A}$*, that never repeats a query, we define its* RKA-PRP *advantage as*

$$\mathbf{Adv}^{\mathsf{rkaPRP}}_{\mathtt{F}}(\mathcal{A}, \Phi) = 2\ \Pr[\mathsf{rkaPRP}^{\mathcal{A}} \Rightarrow \text{true}] - 1\,.$$

3 RKA Security Notions for Nonce-based AEAD

In this section, we define security for nonce-based encryption schemes and nonce-based AEAD schemes under related-key attacks. RKA security notions for encryption and authenticated encryption schemes have been proposed by Bellare et al. [7] and Lu et al. [29], respectively. However, neither notion considers nonce-based primitives and instead considers the case of probabilistic primitives. Furthermore, both works define indistinguishability in a left-or-right sense, while we follow the stronger IND$ (indistinguishability from random bits) approach put forth by Rogaway [34]. For this notion, the adversary has to distinguish the encryption of a message from randomly chosen bits. We discuss how the classical property of nonce-respecting adversaries is extended to the RKA setting in Section 3.1 and provide two RKA security notions for nonce-based AEAD schemes in Section 3.2. In Section 3.3, we extend the notion to the nonce misuse case and Section 3.4 provides the RKA security notion for nonce-based encryption schemes.

3.1 Nonce Selection

Security notions in the classical setting are often restricted to adversaries which are *nonce-respecting*. These are adversaries that never repeat a nonce across their encryption queries. Hence, security proven against nonce-respecting adversaries guarantees security as long as the encrypting party never repeats a nonce. Below we argue why this adversarial restriction needs to be updated in the RKA setting.

Consider the following scenario. Alice and Bob communicate using an AEAD scheme across several sessions. In each session, Alice will send several encrypted messages to Bob, each time using a fresh nonce implemented as a counter. Instead of exchanging a fresh secret key for each session, they exchange a key for the first session and between two consecutive sessions, they update the key using some update function $\mathtt{F}$. There is no guarantee that Alice does not reuse a nonce in different sessions. In fact, due to using a simple counter which might be reset between the sessions, this is likely to happen. This means that an adversary can observe encryptions using the same nonce under related keys, where the relation is given by the update function $\mathtt{F}$.

The same applies to the scenario where different devices have related keys. Every user would only ensure unique nonces for the own device while there will be colliding nonces across related devices.

If we declare an RKA adversary to be nonce-respecting if and only if it never repeats a nonce, then a proof of security does not tell us anything for the scenarios depicted above. Instead, we define an RKA adversary to be *RKA-nonce-respecting* if it never repeats *the pair* of nonce and RKD function. An interpretation of this definition is that nonce-respecting is defined with respect to individual keys. Since in the classical setting there is only ever one key, this interpretation reflects this.

3.2 RKA-Security Notions for AEAD Schemes

We extend security for AEAD schemes to the RKA setting. Instead of the approach used in [29], which defines two separate RKA security notions for confidentiality and authenticity, we follow the unified security

notion by Rogaway and Shrimpton [35]. That is, the adversary has access to two oracles Enc and Dec. The goal of the adversary is to distinguish the real world, in which the oracles implement the encryption and decryption algorithm, from the ideal world, where the first oracle returns random bits while the latter rejects any ciphertext. The adversary wins the game if it can distinguish in which world it is. To make our new RKA security notion achievable, we impose standard restrictions on the adversary. That is, first, the adversary is not allowed to forward the response of an encryption query to the decryption query and, second, the adversary must not repeat a query to its encryption oracle.[7] More precisely, we say that an adversary forwards a query from its encryption oracle, if it queries its decryption oracle on a ciphertext C that it has obtained as a response from its encryption oracle, while the other queried values N, A, φ are the same for both queries. We call the resulting notion s-RKA-AE, the "s" indicating strong. The reason for that is that we introduce a weaker notion below.

Game s-rka-AE	$\mathsf{Enc}(N, A, M, \varphi)$	$\mathsf{Dec}(N, A, C, \varphi)$
$b \leftarrow\!\!\$\ \{0,1\}$	**if** $b = 0$	**if** $b = 0$
$K \leftarrow\!\!\$\ \mathcal{K}$	$C \leftarrow \mathtt{Enc}(\varphi(K), N, A, M)$	$M \leftarrow \mathtt{Dec}(\varphi(K), N, A, C)$
$b' \leftarrow \mathcal{A}^{\mathsf{Enc},\mathsf{Dec}}()$	**else**	**else**
return $(b' = b)$	$C \leftarrow\!\!\$\ \{0,1\}^c$	$M \leftarrow \perp$
	return C	**return** M

Fig. 3: Security game s-rka-AE.

Definition 4 (s-RKA-AE Security). *Let $\Sigma = (\mathtt{Enc}, \mathtt{Dec})$ be an AEAD scheme and $\Phi \subset \mathsf{Func}(\mathcal{K}, \mathcal{K})$. Let the game* s-rka-AE *be defined as in Fig. 3. For an RKA-nonce-respecting and Φ-restricted RKA adversary $\mathcal{A}$, that never repeats/forwards a query to/from* Enc*, we define its* RKA-AE *advantage as*

$$\mathbf{Adv}^{\mathsf{s\text{-}rka\text{-}AE}}_{\Sigma}(\mathcal{A}, \Phi) = 2\ \Pr[\mathsf{s\text{-}rka\text{-}AE}^{\mathcal{A}} \Rightarrow \text{true}] - 1\,.$$

The above definition treats the AEAD scheme to have a single key K. Such schemes, however, are often constructed from smaller building blocks which have individual keys. This encompasses the N constructions [30], on which we focus in the next section, but also all other constructions combining an encryption scheme and a MAC into an AEAD schemes. In this case, the set of RKD functions of the AEAD scheme is the Cartesian product of the set of RKD functions for the individual primitives. More precisely, let E and M be the underlying primitives and Φ_e and Φ_m be the respective sets of RKD functions. Then for the combined primitive AE, the set of RKD functions is $\Phi_{ae} = \Phi_e \times \Phi_m$. Thus the s-RKA-AE security game above allows the adversary to query the encryption oracle on $(N, \varphi_e, \varphi_m) \in \mathcal{N} \times \Phi_e \times \Phi_m$ and later querying it on $(N, \varphi_e, \varphi'_m) \in \mathcal{N} \times \Phi_e \times \Phi_m$, where $\varphi_m \neq \varphi'_m$. This essentially allows the adversary to bypass the nonce-respecting property of the underlying primitive.

Recall the key-update scenario described above. Allowing the adversary to query the same nonce while the queried RKD functions agree in exactly one part, models a scenario in which the key update either does not update one of the keys or later updates a key to a previously used key. We introduce a weaker security notion, in which these queries are forbidden. Security according to this notion then reflects security as long as *the pair of keys* is updated appropriately.

Definition 5 (RKA-AE Security). *Let $\Sigma = (\mathtt{Enc}, \mathtt{Dec})$ be an AEAD scheme and $\Phi = \Phi_e \times \Phi_m \subset \mathsf{Func}(\mathcal{K}, \mathcal{K})$. Let the game* rka-AE *be defined as in Fig. 4. For an RKA-nonce-respecting and Φ-restricted RKA adversary $\mathcal{A}$, that never repeats/forwards a query to/from* Enc*, we define its* RKA-AE *advantage as*

$$\mathbf{Adv}^{\mathsf{rka\text{-}AE}}_{\Sigma}(\mathcal{A}, \Phi) = 2\ \Pr[\mathsf{rka\text{-}AE}^{\mathcal{A}} \Rightarrow \text{true}] - 1\,.$$

[7] The latter restriction can also be handled by letting the encryption oracle return the same response as it did when the query was made the first time. For ease of exposition, we simply forbid such queries to avoid additional bookkeeping in the security games.

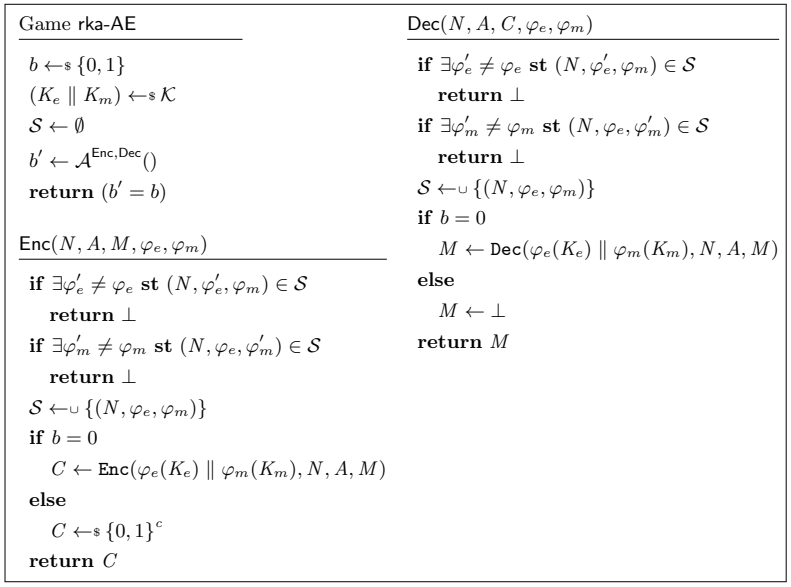


Fig. 4: Security game rka-AE. The set $\mathcal{S}$ is used to detect forbidden queries, that is, queries where the triple of nonce and the two RKD functions differ in exactly one of the functions. Both oracles reject such queries by returning $\perp$.

The weaker security notion bears similarities to split-state non-malleable codes [2]. Here, the secret is encoded in such a way that it is secure against fault attacks as long as the left and right half of the code are tampered independently. In more detail, the decoding of such tampered codes is independent from the original secret and might be invalid. However, if we consider key-related devices or bad-key updates, non-malleable codes are not helpful any more since they are used for faults and not bad randomised keys. The reason for this is that after each key update we need to take care that the resulting key is still valid. Further, we do not want to update the keys independently but simultaneously such that all keys are fresh after the key update. So the requirement to the weaker notion is the opposite of that of non malleable codes. For key updates, it is a reasonable assumption to say that all underlying keys have to be updated for a new session.

3.3 RKA-Security against Nonce Misuse

Similar to the classical setting, we extend security to nonce-misuse resistance. In this case, the adversary is allowed to repeat nonces to the encryption oracle. Below we define security in this stronger sense for s-RKA-AE security. Note that the game is the same as in Definition 4 (cf. Fig. 3), the sole difference is that the adversary is no longer restricted to be RKA-nonce-respecting.

Definition 6 (mr-s-RKA-AE **Security).** *Let* $\Sigma = (\mathtt{Enc}, \mathtt{Dec})$ *be an AEAD scheme and* $\Phi \subset \mathsf{Func}(\mathcal{K}, \mathcal{K})$. *Let the game* s-rka-AE *be defined as in Fig. 3. For an RKA-respecting and* Φ*-restricted RKA adversary* $\mathcal{A}$*, that never repeats/forwards a query to/from* Enc*, we define its* mr-s-RKA-AE *advantage as*

$$\mathbf{Adv}^{\mathsf{mr\text{-}s\text{-}rka\text{-}AE}}_{\Sigma}(\mathcal{A}, \Phi) = 2 \Pr[\mathsf{s\text{-}rka\text{-}AE}^{\mathcal{A}} \Rightarrow \text{true}] - 1\,.$$

In the same way, we can extend RKA-AE security to the nonce misuse scenario. However, we believe this notion not to be meaningful. The RKA-AE security notion already requires that keys are updated properly, i.e., they do not repeat. Since this task is way more complex than ensuring that nonces do not repeat, it seems strange to require this one while simultaneously dropping the simple requirement of unique nonces.

3.4 RKA-Security Notions for Encryption

The following definition extends the classical IND-CPA security notion for nonce-based encryption schemes to the RKA setting. The adversary has to tell apart the real encryption oracle from an idealised encryption oracle which returns random bits. The main distinction lies in the nonce selection of the adversary as it is allowed to repeat a nonce if the RKD functions are different.

Game rkaIND	$\mathsf{Enc}(N, M, \varphi)$
$b \leftarrow\!\!\$\ \{0,1\}$	**if** $b = 0$
$K \leftarrow\!\!\$\ \mathcal{K}$	$C \leftarrow \mathtt{Enc}(\varphi(K), N, M)$
$b' \leftarrow \mathcal{A}^{\mathsf{Enc}}()$	**else**
return $(b' = b)$	$C \leftarrow\!\!\$\ \{0,1\}^{c}$
	return C

Fig. 5: Security game rkaIND.

Definition 7 (RKA-IND **Security).** *Let* $\Sigma = (\mathtt{Enc}, \mathtt{Dec})$ *be an encryption scheme and* $\Phi \subset \mathsf{Func}(\mathcal{K}, \mathcal{K})$. *Let the game* rkaIND *be defined as in Fig. 5. For an RKA-nonce-respecting and* Φ*-restricted RKA adversary* $\mathcal{A}$*, that never repeats a query, we define its* RKA-IND *advantage as*

$$\mathbf{Adv}^{\mathsf{rkaIND}}_{\Sigma}(\mathcal{A}, \Phi) = 2\ \Pr[\mathsf{rkaIND}^{\mathcal{A}} \Rightarrow \text{true}] - 1\,.$$

4 RKA Security of the N1, N2, and N3 Constructions

In this section we study the security of the nonce-based AEAD schemes N1, N2, and N3 [30], which fall into the generic composition paradigms Encrypt-and-MAC (E&M), Encrypt-then-MAC (EtM), MAC-then-Encrypt (MtE) [9]. We analyse each scheme with respect to the two security notions RKA-AE and s-RKA-AE defined above. The analysis reveals that all schemes achieve RKA-AE security if the underlying primitives are RKA-secure. Regarding the stronger s-RKA-AE security, the situation is more involved. We show that both N1 and N2 are insecure irrespective of the underlying primitives. For N3, we provide a concrete attack exploiting any instantiation using a stream cipher for the underlying encryption scheme.

Section 4.1 covers the analysis of the N1 construction. The N2 construction is analysed in Section 4.2 while we analyse the N3 construction in Section 4.3.

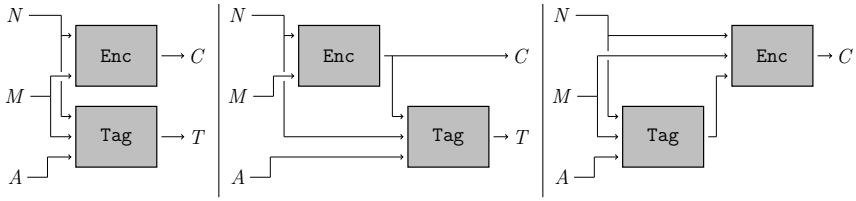


Fig. 6: The AEAD schemes N1 (left), N2 (middle), and N3 (right) [30].

4.1 N1 - Instantiation of Encrypt-and-MAC

The N1 construction composes a nonce-based encryption scheme and a MAC into an AEAD scheme. It follows the E&M paradigm. The encryption algorithm is used to encrypt the message as is the MAC to compute a tag for the message. The ciphertext of the AEAD scheme consists of the ciphertext and the tag.

The following theorem shows that the N1 construction achieves RKA-AE security if the underlying primitives are RKA-secure. The overall proof approach is similar to the classical setting but needs some extra treatment when analysing that all queries of the reductions are permitted. The full proof is given in the extended version of the paper [20].

Theorem 8. *Let $\Sigma = (\mathtt{Enc}, \mathtt{Dec})$ be an encryption scheme and $\Gamma = (\mathtt{Tag}, \mathtt{Ver})$ be a MAC with RKA function sets Φ_e and Φ_m, respectively. Further, let* N1 *be the AEAD scheme built from Σ and Γ using the N1 construction with RKA function set $\Phi_{ae} = \Phi_e \times \Phi_m$. Then for any RKA-nonce-respecting and Φ_{ae}-restricted RKA adversary $\mathcal{A}$ against* N1*, that never repeats/forwards a query to/from* Enc*, there exists an RKA-nonce-respecting and Φ_e-restricted RKA adversary $\mathcal{A}_{se}$, a Φ_m-restricted RKA adversary $\mathcal{A}_{mac}$, and a Φ_m-restricted RKA adversary $\mathcal{A}_{prf}$ such that*

$$\begin{aligned}\mathbf{Adv}^{\mathsf{rka\text{-}AE}}_{\mathrm{N1}}(\mathcal{A}, \Phi_{ae}) \leq \mathbf{Adv}^{\mathsf{rkaIND}}_{\Sigma}(\mathcal{A}_{se}, \Phi_e) + \mathbf{Adv}^{\mathsf{rkaSUF}}_{\Gamma}(\mathcal{A}_{mac}, \Phi_m) \\ + \mathbf{Adv}^{\mathsf{rkaPRF}}_{\mathsf{Tag}}(\mathcal{A}_{prf}, \Phi_m)\,.\end{aligned}$$

□

Proof (Sketch). The proof consists of multiple game hops. In the first game hop, the decryption oracle is replaced by $\perp$ which is bound by the RKA security of Γ. In the subsequent game hops, first the tag and then the ciphertext are replaced by random values which is bound by the RKA security of $\mathtt{Tag}$ and Σ, respectively.

The following theorem shows that the N1 construction does not achieve the stronger s-RKA-AE security. The reason is that a ciphertext is the concatenation of a ciphertext from the underlying encryption scheme and tag from the underlying MAC. By making two queries which solely differ in one of the RKD functions, the adversary can easily distinguishing between the real and the ideal case. The proof appears in the extended version of the paper [20].

Theorem 9. *Let $\Sigma = (\mathtt{Enc}, \mathtt{Dec})$ be an encryption scheme and $\Gamma = (\mathtt{Tag}, \mathtt{Ver})$ be a MAC with RKA function sets Φ_e and Φ_m, respectively. Further, let* N1 *be the AEAD scheme built from Σ and Γ using the N1 construction with RKA function set $\Phi_{ae} = \Phi_e \times \Phi_m$. Then* N1 *is not* s-RKA-AE*-secure. There exists an RKA-nonce-respecting and Φ_{ae}-restricted RKA adversary $\mathcal{A}$ such that*

$$\mathbf{Adv}^{\mathsf{s\text{-}rka\text{-}AE}}_{\mathrm{N1}}(\mathcal{A}) = 1\,.$$

4.2 N2 - Instantiation of Encrypt-then-MAC

The N2 construction composes a nonce-based encryption scheme and a MAC into an AEAD scheme. It follows the EtM paradigm and is displayed in Fig. 6. The scheme first encrypts the message using the encryption scheme. Subsequently, the MAC is used to compute a tag for the ciphertext. The ciphertext of the AEAD scheme consists of both the ciphertext and the tag.

The theorem below shows that the N2 construction achieves RKA-AE security if the underlying primitives are sound. The overall proof follows the classical one, except for a more complex analysis regarding the permitted queries. The full proof is given in the extended version of the paper [20].

Theorem 10. *Let $\Sigma = (\mathtt{Enc}, \mathtt{Dec})$ be an encryption scheme and $\Gamma = (\mathtt{Tag}, \mathtt{Ver})$ be a MAC with RKA function sets Φ_e and Φ_m, respectively. Further, let* N2 *be the AEAD scheme built from Σ and Γ using the N2 construction with RKA function set $\Phi_{ae} = \Phi_e \times \Phi_m$. Then for any RKA-nonce-respecting and Φ_{ae}-restricted RKA adversary $\mathcal{A}$ against* N2*, that never repeats/forwards a query to/from* Enc*, there exists an RKA-nonce-respecting and Φ_e-restricted RKA adversary $\mathcal{A}_{se}$, a Φ_m-restricted RKA adversary $\mathcal{A}_{mac}$, and a Φ_m-restricted RKA adversary $\mathcal{A}_{prf}$ such that*

$$\begin{aligned}\mathbf{Adv}^{\mathsf{rka\text{-}AE}}_{\mathrm{N2}}(\mathcal{A}, \Phi_{ae}) \leq \mathbf{Adv}^{\mathsf{rkaIND}}_{\Sigma}(\mathcal{A}_{se}, \Phi_e) + \mathbf{Adv}^{\mathsf{rkaSUF}}_{\Gamma}(\mathcal{A}_{mac}, \Phi_m) \\ + \mathbf{Adv}^{\mathsf{rkaPRF}}_{\mathsf{Tag}}(\mathcal{A}_{prf}, \Phi_m)\,.\end{aligned}$$

Proof (Sketch). In the first game hop, the decryption oracle is replaced by $\perp$ which is bound by the RKA security of Γ. In the subsequent game hops, first the tag and then the ciphertext are replaced by random values which is bound by the RKA security of $\mathtt{Tag}$ and Σ, respectively. □

Below we show that the N2 construction does not achieve s-RKA-AE security. It exhibits the same structure as the N1 construction, that is, a concatenation of a ciphertext and a tag from the underlying primitives. The difference is the tag is computed on the ciphertext rather than the message. While we give two attacks against the N1 construction, only one attack also applies against the N2 construction. The proof is given in the extended version of the paper [20].

Theorem 11. *Let* $\Sigma = (\mathtt{Enc}, \mathtt{Dec})$ *be an encryption scheme and* $\Gamma = (\mathtt{Tag}, \mathtt{Ver})$ *be a MAC with RKA function sets* Φ_e *and* Φ_m*, respectively. Further, let* N2 *be the AEAD scheme built from* Σ *and* Γ *using the N2 construction with RKA function set* $\Phi_{ae} = \Phi_e \times \Phi_m$*. Then* N2 *is not* s-RKA-AE*-secure. There exists an RKA-nonce-respecting and* Φ_{ae}*-restricted RKA adversary* $\mathcal{A}$ *such that*

$$\mathbf{Adv}^{\mathsf{s\text{-}rka\text{-}AE}}_{\mathrm{N2}}(\mathcal{A}) = 1\,.$$

4.3 N3 - Instantiation of MAC-then-Encrypt

The N3 construction composes a nonce-based encryption scheme and a MAC into an AEAD scheme. It follows the MtE paradigm and is displayed in Fig. 6. The message is first used as an input to the MAC and then both the message and the tag are encrypted. In contrast to the other compositions, the ciphertext of the AEAD scheme consists only of the ciphertext from the underlying encryption scheme.

In the theorem below, we show that the N3 construction is RKA-AE secure if both of the underlying primitives are secure. The overall proof follows the classical setting, except for the analysis that all queries by the reductions are indeed valid queries.

Theorem 12. *Let* $\Sigma = (\mathtt{Enc}, \mathtt{Dec})$ *be an encryption scheme and* $\Gamma = (\mathtt{Tag}, \mathtt{Ver})$ *be a MAC with RKA function sets* Φ_e *and* Φ_m*, respectively. Further, let* N3 *be the AEAD scheme built from* Σ *and* Γ *using the N3 construction with RKA function set* $\Phi_{ae} = \Phi_e \times \Phi_m$*. Then for any RKA-nonce-respecting and* Φ_{ae}*-restricted RKA adversary* $\mathcal{A}$ *against* N3*, that never repeats/forwards a query to/from* Enc*, there exists an RKA-nonce-respecting and* Φ_e*-restricted RKA adversary* $\mathcal{A}_{se}$*, a* Φ_m*-restricted RKA adversary* $\mathcal{A}_{mac}$*, and a* Φ_m*-restricted RKA adversary* $\mathcal{A}_{prf}$ *such that*

$$\begin{aligned}\mathbf{Adv}^{\mathsf{rka\text{-}AE}}_{\mathrm{N3}}(\mathcal{A}, \Phi_{ae}) \le\ & \mathbf{Adv}^{\mathsf{rkaIND}}_{\Sigma}(\mathcal{A}_{se}, \Phi_e) + \mathbf{Adv}^{\mathsf{rkaSUF}}_{\Gamma}(\mathcal{A}_{mac}, \Phi_m) \\ & + \mathbf{Adv}^{\mathsf{rkaPRF}}_{\mathsf{Tag}}(\mathcal{A}_{prf}, \Phi_m)\,.\end{aligned}$$

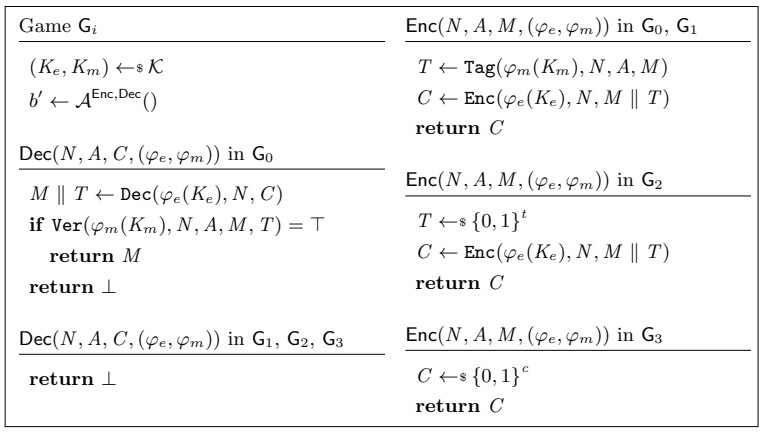


Fig. 7: Hybrid games G_i used to prove Theorem 12 (RKA-AE security of N3).

Proof. We prove the theorem using the hybrid games G_0, G_1, G_2, and G_3 displayed in Fig. 7. For sake of simplicity, the games do not contain the set $\mathcal{S}$ to detect invalid queries. Instead, we assume that the adversary

does not make such queries, which the reduction can simply answer with $\perp$. Game G_0 is rka-AE instantiated with N3 and secret bit b fixed to 0. In G_1, the decryption oracle is modified to reject any ciphertext. In G_2, encryption oracle computes a random tag which is then encrypted along with the message. Game G_3 equals rka-AE with secret bit b fixed to 1, where the encryption oracle outputs random ciphertexts and the decryption oracle rejects any ciphertext. We have

$$\begin{aligned}\mathbf{Adv}^{\mathsf{rka\text{-}AE}}_{\mathrm{N3}}(\mathcal{A}) &= \Pr[\mathcal{A}^{\mathsf{rka\text{-}AE}} \Rightarrow 0 \,|\, b = 0] - \Pr[\mathcal{A}^{\mathsf{rka\text{-}AE}} \Rightarrow 0 \,|\, b = 1] \\ &= \Pr[\mathcal{A}^{\mathsf{G}_0} \Rightarrow 0] - \Pr[\mathcal{A}^{\mathsf{G}_3} \Rightarrow 0] \\ &= \sum_{i=1}^{3} \Pr[\mathcal{A}^{\mathsf{G}_{i-1}} \Rightarrow 0] - \Pr[\mathcal{A}^{\mathsf{G}_i} \Rightarrow 0]\,.\end{aligned}$$

To bound the term $\Pr[\mathcal{A}^{\mathsf{G}_0} \Rightarrow 0] - \Pr[\mathcal{A}^{\mathsf{G}_1} \Rightarrow 0]$ we construct the following adversary $\mathcal{A}_{mac}$ against the RKA-SUF security of Γ. It chooses a random key K_e for the encryption scheme Σ and then runs $\mathcal{A}$. When $\mathcal{A}$ makes a query $(N, A, M, (\varphi_e, \varphi_m))$ to Enc, $\mathcal{A}_{mac}$ proceeds as follows. It queries its oracle Tag on (N, A, M, φ_m) to obtain a tag T. Then it locally computes $C \leftarrow \mathtt{Enc}(\varphi_e(K_e), N, M \parallel T)$ and sends C back to $\mathcal{A}$. For queries $(N, A, C, (\varphi_e, \varphi_m))$ to Dec by $\mathcal{A}$, $\mathcal{A}_{mac}$ locally computes $M \parallel T \leftarrow \mathtt{Dec}(\varphi_e(K_e), N, C)$ and queries (N, A, M, T, φ_m) to its challenge oracle Ver. If the response is $\perp$, it forwards it to $\mathcal{A}$, otherwise, it sends M to $\mathcal{A}$. When $\mathcal{A}$ outputs a bit b', $\mathcal{A}_{mac}$ outputs the same bit.

Ir remains to argue that $\mathcal{A}_{mac}$ never makes a forbidden query (forwarding from Tag to Ver) conditioned on $\mathcal{A}$ making only permitted queries. Assume, for sake of contradiction, that $\mathcal{A}$ makes a valid query $(N, A, C, \varphi_e, \varphi_m)$ to Dec for which $\mathcal{A}_{mac}$ makes a forbidden query. By construction $\mathcal{A}_{mac}$ computes $M \parallel T \leftarrow \mathtt{Dec}(\varphi_e(K_e), N, C)$ and queries Ver on (N, A, M, T, φ_m). This query is forbidden if $\mathcal{A}_{mac}$ has queried (N, A, M, φ_m) to Tag which resulted in T. This happens if $\mathcal{A}$ has made a query $(N, A, M, \varphi'_e, \varphi_m)$ to Enc. We need to distinguish between the case $\varphi'_e = \varphi_e$ and $\varphi'_e \neq \varphi_e$. The former is forbidden as this means that $\mathcal{A}$ forwards a query from Enc to Dec. The latter is forbidden since game rka-AE forbids queries that agree on the nonce and exactly one of the RKD functions while disagreeing on the other RKD function. Hence $\mathcal{A}_{mac}$ only makes permitted queries.

By construction, $\mathcal{A}_{mac}$ simulates either G_0 or G_1 for $\mathcal{A}$, depending on its secret bit b from game rkaSUF. More precisely, it simulates G_0 and G_1 if its own challenge is 0 and 1, respectively. This gives us

$$\begin{aligned}\Pr[\mathcal{A}^{\mathsf{G}_0} \Rightarrow 0] - \Pr[\mathcal{A}^{\mathsf{G}_1} \Rightarrow 0] &\leq \Pr[\mathcal{A}^{\mathsf{rkaSUF}}_{mac} \Rightarrow 0 \,|\, b = 0] - \Pr[\mathcal{A}^{\mathsf{rkaSUF}}_{mac} \Rightarrow 0 \,|\, b = 1] \\ &\leq \mathbf{Adv}^{\mathsf{rkaSUF}}_{\Gamma}(\mathcal{A}_{mac}, \Phi_m)\,.\end{aligned}$$

For the term $\Pr[\mathcal{A}^{\mathsf{G}_1} \Rightarrow 0] - \Pr[\mathcal{A}^{\mathsf{G}_2} \Rightarrow 0]$, we construct an adversary $\mathcal{A}_{prf}$ against the RKA-PRF security of the tagging algorithm Tag. First, $\mathcal{A}_{prf}$ chooses a random key K_e to simulate all encryption related functionalities. Queries to Dec by $\mathcal{A}$ are answered with $\perp$. Queries $(N, A, M, (\varphi_e, \varphi_m))$ to Enc, are processed as follows. The reduction $\mathcal{A}_{prf}$ invokes its own oracle F on (N, A, M, φ_m) to obtain T, locally computes $C \leftarrow \mathtt{Enc}(\varphi_e(K_e), N, M \parallel T)$, and sends C back to $\mathcal{A}$. When $\mathcal{A}$ terminates, $\mathcal{A}_{prf}$ also terminates and outputs whatever $\mathcal{A}$ does.

We briefly argue that $\mathcal{A}_{prf}$ never repeats a query to F. By construction, every query (N, A, M, φ_m) by $\mathcal{A}_{prf}$ stems from a query $(N, A, M, \varphi_e, \varphi_m)$ by $\mathcal{A}$. The only cases that result in a repeating query are (1) $\mathcal{A}$ repeats a query and (2) $\mathcal{A}$ makes two queries which only differ in φ_e. However, both cases are forbidden queries for $\mathcal{A}$. This yields that every output of Tag is a random value.

The adversary $\mathcal{A}_{prf}$ simulates game G_1 for $\mathcal{A}$ if its own challenge bit b equals 0, while it simulates G_2 for $\mathcal{A}$ if b equals 1. Thus it holds that

$$\begin{aligned}\Pr[\mathcal{A}^{\mathsf{G}_1} \Rightarrow 0] - \Pr[\mathcal{A}^{\mathsf{G}_2} \Rightarrow 0] &\leq \Pr[\mathcal{A}^{\mathsf{rkaPRF}}_{prf} \Rightarrow 0 \,|\, b = 0] - \Pr[\mathcal{A}^{\mathsf{rkaPRF}}_{prf} \Rightarrow 0 \,|\, b = 1] \\ &\leq \mathbf{Adv}^{\mathsf{rkaPRF}}_{\Gamma}(\mathcal{A}_{prf}, \Phi_m)\,.\end{aligned}$$

We bound the final term $\Pr[\mathcal{A}^{\mathsf{G}_2} \Rightarrow 0] - \Pr[\mathcal{A}^{\mathsf{G}_3} \Rightarrow 0]$ by constructing an adversary $\mathcal{A}_{se}$ against the RKA-IND security of the underlying encryption scheme Σ. At the start, $\mathcal{A}_{se}$ chooses a random key K_m. Any query to

Dec is answered with $\perp$. When $\mathcal{A}$ queries its oracle Enc on $(N, A, M, (\varphi_e, \varphi_m))$, $\mathcal{A}_{se}$ chooses a random tag T of length t, invokes its oracle Enc on $(N, M \parallel T, \varphi_e)$ to obtain C, and sends C to $\mathcal{A}$. At the end, $\mathcal{A}_{se}$ outputs whatever $\mathcal{A}$ outputs.

It holds that $\mathcal{A}_{se}$ is RKA-nonce-respecting as any query $(N, M \parallel T, \varphi_e)$ stems from a query $(N, A, M, \varphi_e, \varphi_m)$ by $\mathcal{A}$. This means that $\mathcal{A}_{se}$ repeats a pair of nonce N and RKD function φ_e if $\mathcal{A}$ makes two queries using $(N, \varphi_e, \varphi_m)$ and $(N, \varphi_e, \varphi'_m)$. We can distinguish between the cases (1) $\varphi_m = \varphi'_m$ and (2) $\varphi_m \neq \varphi'_m$. Case (1) does not occur, as $\mathcal{A}$ is RKA-nonce-respecting and case (2) is forbidden in game $\mathsf{rka\text{-}AE}$. The other option would be that $\mathcal{A}$ makes two queries differing only in the associated data A. This turns out not to be an issue, as the tag T that $\mathcal{A}_{se}$ queries along with the message depends on A, i.e., different A results in a different message queries by $\mathcal{A}_{se}$.

The adversary $\mathcal{A}_{se}$ perfectly simulates games G_2 or G_3 for $\mathcal{A}$ depending on its own challenge from rkaIND. Hence we have

$$\begin{aligned}\Pr[\mathcal{A}^{\mathsf{G}_2} \Rightarrow 0] - \Pr[\mathcal{A}^{\mathsf{G}_3} \Rightarrow 0] &\leq \Pr[\mathcal{A}_{se}^{\mathsf{rkaIND}} \Rightarrow 0 \,|\, b = 0] - \Pr[\mathcal{A}_{se}^{\mathsf{rkaIND}} \Rightarrow 0 \,|\, b = 1] \\ &\leq \mathbf{Adv}_{\Gamma}^{\mathsf{rkaIND}}(\mathcal{A}_{se}, \Phi_e)\,.\end{aligned}$$

Collecting the bounds above proves the claim. □

Unlike for the N1 and N2 construction, the s-RKA-AE security of the N3 construction is more subtle. The difference is that the tag is appended to the ciphertext for both the N1 and N2 construction while it is encrypted alongside the message for the N3 construction. The attacks against the N1 and N2 construction rely on the property that the ciphertext consists of two parts which can be manipulated separately. Due to the construction such attacks do not work against the N3 construction.

It turns out that the s-RKA-AE security of the N3 construction crucially depend on the used encryption scheme. Namely, if the underlying encryption scheme is a stream cipher, then the N3 construction is s-RKA-AE insecure. Below we show an attack against any instantiation using a stream cipher. For such ciphers the ciphertext is the XOR of the message and a keystream derived from the key and the nonce.

Theorem 13. *Let $\Sigma = (\mathtt{Enc}, \mathtt{Dec})$ be a stream cipher and $\Gamma = (\mathtt{Tag}, \mathtt{Ver})$ be a MAC with RKA function sets Φ_e and Φ_m, respectively. Further, let* N3 *be the AEAD scheme built from Σ and Γ using the N2 construction with RKA function set $\Phi_{ae} = \Phi_e \times \Phi_m$. Then* N3 *is not* s-RKA-AE*-secure. There exists an RKA-nonce-respecting and Φ_{ae}-restricted RKA adversary $\mathcal{A}$ such that*

$$\mathbf{Adv}_{\mathrm{N3}}^{\mathsf{s\text{-}rka\text{-}AE}}(\mathcal{A}) = 1\,.$$

Proof. Adversary $\mathcal{A}$ chooses a nonce N, associated data A, a message M, RKD functions φ_e, φ_m, and φ'_m from the respective sets such that $\varphi_m \neq \varphi'_m$. Then it queries its encryption oracle Enc on $(N, A, M, (\varphi_e, \varphi_m))$ and $(N, A, M, (\varphi_e, \varphi'_m))$ to obtain ciphertext C_1 and C_2. If the first $|M|$ bits of C_1 and C_2 are equal, $\mathcal{A}$ outputs 0, otherwise, it outputs 1.

In case $b = 0$, we have $C_1 = \mathtt{Enc}(\varphi_e(K_e), N, M \parallel \mathtt{Tag}(\varphi_m(K_m), N, A, M))$ and $C_2 = \mathtt{Enc}(\varphi_e(K_e), N, M \parallel \mathtt{Tag}(\varphi'_m(K_m), N, A, M))$. Since the encryption uses the same nonce and the same key, the same keystream for the stream cipher will be used. Together with the fact that the first $|M|$ bits are identical as the same message is encrypted, this yields that C_1 and C_2 agree on the first bits. In case $b = 1$, both C_1 and C_2 are chosen at random, hence they will not agree on the first $|M|$ bits.[8] □

In the attack above, the RKA-nonce-respecting adversary essentially bypasses the nonce-respecting property of the underlying encryption scheme by repeating the nonce N and the RKD function φ_e for the encryption scheme. Then it exploits the fact that the underlying stream cipher is secure only against nonce-respecting adversaries. We conjecture that any instantiation using an encryption scheme that can be broken in the nonce-misuse case results in an s-RKA-AE insecure instantiation of the N3 construction. The problematic part is that both the message and the tag are encrypted. While the adversary has full control over the

[8] Note that there is a negligible chance that the ciphertexts will agree on their first $|M|$ bits which we drop here for simplicity.

former, it can not choose the latter at will. This seems to thwart a simple proof showing that any nonce-misuse adversary against the underlying encryption scheme can be turned into an s-RKA-AE adversary against N3.

5 RKA Nonce-Misuse-Resistant AEAD

As described in Section 4, N1, N2, and N3 are not secure in the strong RKA setting.[9] In this section we give a new AE scheme, N*, that achieves mr-s-RKA-AE security and hence also s-RKA-AE security. The construction, following the N3 construction, is displayed in Fig. 8. The message, nonce, and associated data are first used as an input to the MAC, and then both the message and the tag are encrypted. The difference to the N3 construction is that the encryption scheme no longer takes the nonce as input. Instead, the (pseudorandom) tag ensures that the encryption is randomised.

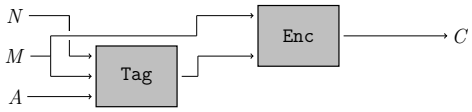


Fig. 8: The AEAD scheme N* [This work].

The theorem below shows that the new construction achieves our strong RKA security notion conditioned on the encryption scheme being an RKA-secure block cipher (pseudorandom permutation).

Theorem 14. *Let $\Sigma = (\mathtt{Enc}, \mathtt{Dec})$ be an encryption scheme and $\Gamma = (\mathtt{Tag}, \mathtt{Ver})$ be a MAC with RKA function sets Φ_e and Φ_m, respectively. Further, let* N* *be the AEAD scheme built from Σ and Γ using the* N* *construction with RKA function set $\Phi_{ae} = \Phi_e \times \Phi_m$. Then for any Φ_{ae}-restricted RKA adversary $\mathcal{A}$ against* N* *with q queries to the encryption and decryption oracle, that never repeats/forwards a query to/from* Enc*, there exists Φ_e-restricted RKA adversaries $\mathcal{A}_{prp}$, and Φ_m-restricted RKA adversaries $\mathcal{A}_{mac}$ and $\mathcal{A}_{prf}$ such that*

$$\begin{aligned}\mathbf{Adv}^{\mathsf{mr\text{-}s\text{-}rka\text{-}AE}}_{\mathrm{N}^*}(\mathcal{A}, \Phi_{ae}) \leq \mathbf{Adv}^{\mathsf{rkaSUF}}_{\Gamma}(\mathcal{A}_{mac}, \Phi_m) + \mathbf{Adv}^{\mathsf{rkaPRP}}_{\Sigma}(\mathcal{A}_{prp}, \Phi_e) \\ + \mathbf{Adv}^{\mathsf{rkaPRF}}_{\mathsf{Tag}}(\mathcal{A}_{prf}, \Phi_m) + \frac{2q^2}{2^c}\,.\end{aligned}$$

Proof. Game G_0 in Fig. 9 is the mr-s-rka-AE security game instantiated with N* and secret bit $b = 0$ and game G_5 is the mr-s-rka-AE security game with $b = 1$. To estimate the security of N*, four additional games G_1, G_2, G_3, and G_4 are needed. Starting with mr-s-rka-AE with $b = 0$ (G_0), we modify the intermediate games as follows: In game G_1 the decryption always outputs $\perp$ except if the resulting message was sent to the encryption oracle with the same N, A, and φ_m before. In G_2, the underlying encryption scheme is replaced by a random permutation. In G_3, the decryption oracle always outputs $\perp$. In G_4, the Tag algorithm is replaced by a random function. Finally, in game G_5, the encryption oracle ignores the input, and outputs a uniform random cipher C as in mr-s-rka-AE with $b = 1$. With $\mathbf{Adv}^{\mathsf{mr\text{-}s\text{-}rka\text{-}AE}}_{\mathrm{N}^*}(\mathcal{A}, \Phi_{ae}) \leq \mathbf{Adv}(\mathcal{A}^{\mathsf{G}_0}, \mathcal{A}^{\mathsf{G}_5})$ and $\mathbf{Adv}(\mathcal{A}^{\mathsf{G}_0}, \mathcal{A}^{\mathsf{G}_5}) \leq \sum_{i=0}^{4} \mathbf{Adv}(\mathcal{A}^{\mathsf{G}_i}, \mathcal{A}^{\mathsf{G}_{i+1}})$, Claim 15 - 19 conclude the proof. □

Claim 15 *For any Φ_{ae}-restricted RKA distinguisher $\mathcal{A}$ between game G_0 and G_1 defined in Fig. 9, there exists a Φ_m-restricted RKA adversary $\mathcal{A}_{mac}$ such that*

$$\mathbf{Adv}(\mathcal{A}^{\mathsf{G}_0}, \mathcal{A}^{\mathsf{G}_1}) \leq \mathbf{Adv}^{\mathsf{rkaSUF}}_{\Gamma}(\mathcal{A}_{mac}, \Phi_m)\,.$$

[9] One solution would be to use the key derivation technique proposed in [7]. However, this requires the usage of an additional PRF on top of the existing AE scheme.

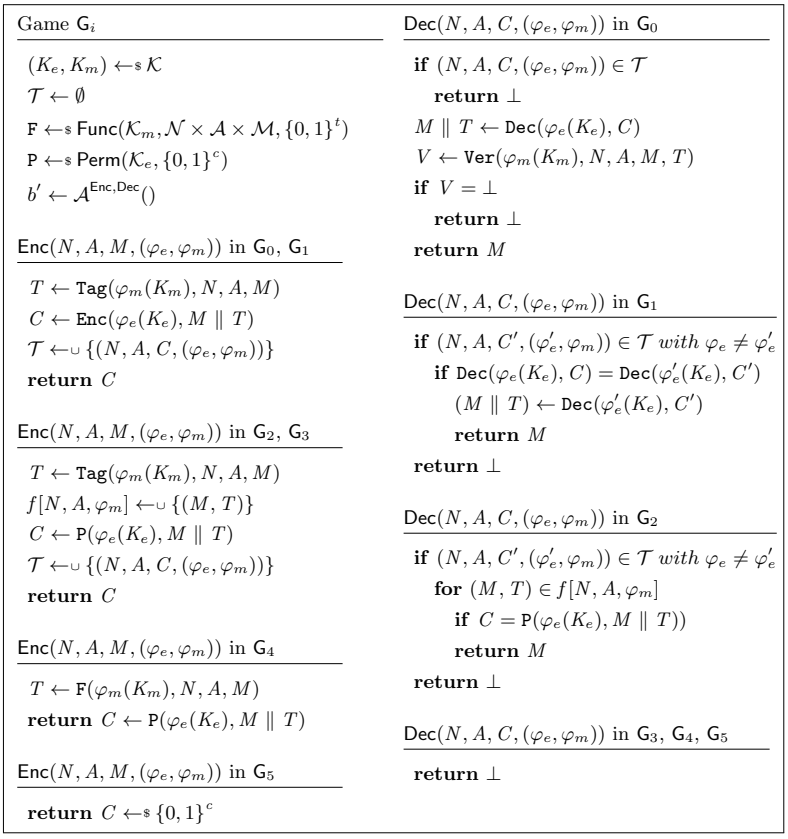


Fig. 9: Hybrid games G_i used to prove Theorem 14.

Proof. In the following, an adversary $\mathcal{A}_{mac}$ is given that wins the game with the advantage of $\mathcal{A}$. $\mathcal{A}_{mac}$ simulates the game by using the oracles of the security game rkaSUF to get the tags T for the encryption and to verify T for the decryption of the N* scheme. Further, $\mathcal{A}_{mac}$ computes the encryption scheme Σ locally with a key K_e chosen uniform at random to simulate the encryption oracle of game G_0 and G_1. $\mathcal{A}_{mac}$ simulates both games perfectly and $\mathcal{A}$ can only distinguish both games if it requests a decryption of a valid ciphertext $(N, A, C, (\varphi_e, \varphi_m))$ with $(M \parallel T) = \mathtt{Dec}(\varphi_e(K_e), N, C)$, such that (N, A, M, T, φ_m) is new and the Tag T is valid. Hence, (N, A, M, φ_m) was not forwarded to the Tag oracle of rkaSUF and $\mathcal{A}_{mac}$ can use this request to win the game rkaSUF by forwarding (N, A, M, T, φ_m) to the verification oracle Ver of rkaSUF, since it was not sent to the oracle Tag of rkaSUF before. Hence, $\mathcal{A}_{mac}$ is a Φ_m-restricted RKA adversary because $\mathcal{A}$ is Φ_{ae}-restricted, and it holds $\Pr[\mathcal{A}^{\mathsf{G}_0} \Rightarrow 0] - \Pr[\mathcal{A}^{\mathsf{G}_1} \Rightarrow 0] \leq \Pr[\mathcal{A}_{mac}^{\mathsf{rkaSUF}} \Rightarrow 0 \,|\, b = 0] - \Pr[\mathcal{A}_{mac}^{\mathsf{rkaSUF}} \Rightarrow 0 \,|\, b = 1]$, and therefore $\mathbf{Adv}(\mathcal{A}^{\mathsf{G}_0}, \mathcal{A}^{\mathsf{G}_1}) \leq \mathbf{Adv}_{\Gamma}^{\mathsf{rkaSUF}}(\mathcal{A}_{mac}, \Phi_m)$. □

Claim 16 *For any Φ_{ae}-restricted RKA distinguisher $\mathcal{A}$ between game G_1 and G_2 defined in Fig. 9, there exists an Φ_e-restricted RKA adversary $\mathcal{A}_{prp}$ such that*

$$\mathbf{Adv}(\mathcal{A}^{\mathsf{G}_1}, \mathcal{A}^{\mathsf{G}_2}) \leq \mathbf{Adv}_{\Sigma}^{\mathsf{rkaPRP}}(\mathcal{A}_{prp}, \Phi_e)\,.$$

Proof. Before we describe the simulator, we transform G_1 to G_1' to avoid that we need to query the inverse of the oracle F in rkaPRP. In G_1'[10] we replace the underling decryption function with the encryption function in such a way that the input/output behaviour of G_1 and G_1' is still the same.

$\mathsf{Dec}(N, A, C, (\varphi_e, \varphi_m))$ in G_1'	$\mathsf{Enc}(N, A, M, (\varphi_e, \varphi_m))$ in G_1'
if $(N, A, C', (\varphi_e', \varphi_m)) \in \mathcal{T}$ *with* $\varphi_e \neq \varphi_e'$	$T \leftarrow \mathtt{Tag}(\varphi_m(K_m), N, A, M)$
for $(M, T) \in f[N, A, \varphi_m]$	$C \leftarrow \mathtt{Enc}(\varphi_e(K_e), N, M \parallel T)$
if $C = \mathtt{Enc}(\varphi_e'(K_e), N, M \parallel T)$	$f[N, A, \varphi_m] \leftarrow\cup \{(M, T)\}$
return M	$\mathcal{T} \leftarrow\cup \{(N, A, C, (\varphi_e, \varphi_m))\}$
return $\perp$	**return** C

In G_1 the decryption oracle only returns the decrypted message M if M was sent to the encryption oracle before. Since the underlying $\mathtt{Enc}$ is deterministic, we can also save the queries to the encryption oracles in f and test if it encrypts to C as we do in the decryption oracle of G_1'. Hence, it holds that the games are identical and it is enough to show that $\mathbf{Adv}(\mathcal{A}^{\mathsf{G}_1'}, \mathcal{A}^{\mathsf{G}_2}) \leq \mathbf{Adv}_{\Sigma}^{\mathsf{rkaPRP}}(\mathcal{A}_{prp}, \Phi_e)$. We construct an adversary $\mathcal{A}_{prp}$ simulating G_1' and G_2 by computing the MAC locally with a uniform random key K_m and using oracle F of rkaPRP for the encryption. $\mathcal{A}_{prp}$ is Φ_e-restricted because $\mathcal{A}$ is Φ_{ae}-restricted. Hence, $\mathcal{A}_{prp}$ perfectly simulates G_1' if the challenge bit of rkaPRP is 0, and G_2 if the challenge bit is 1. It holds $\mathbf{Adv}(\mathcal{A}^{\mathsf{G}_1}, \mathcal{A}^{\mathsf{G}_2}) = \mathbf{Adv}(\mathcal{A}^{\mathsf{G}_1'}, \mathcal{A}^{\mathsf{G}_2}) \leq \mathbf{Adv}_{\Sigma}^{\mathsf{rkaPRP}}(\mathcal{A}_{prp}, \Phi_e)$. □

Claim 17 *For any Φ_{ae}-restricted RKA distinguisher $\mathcal{A}$ with q queries between game G_2 and G_3 defined in Fig. 9, it holds*

$$\mathbf{Adv}(\mathcal{A}^{\mathsf{G}_2}, \mathcal{A}^{\mathsf{G}_3}) \leq \frac{q^2}{2^c}\,.$$

Proof. Both games only differ if $\mathcal{A}$ asks for a decryption of a cipher text C which maps to a message which was already encrypted with the same (N, A, φ_m). Since the underlying encryption is a random permutation this collision happens with probability less then $\frac{q^2-q}{2^c}$. In detail $\mathcal{A}$ makes q_e queries to the encryption oracle, and q_d queries to the decryption oracle with $q_e + q_d = q$. The collision probability is less then $\frac{q_e}{2^c}$ for each query to the decryption oracle. Hence, the probability to get at least one collision is less then $\frac{q_e q_d}{2^c}$ with q_d queries to the decryption oracle. Hence, $\mathbf{Adv}(\mathcal{A}^{\mathsf{G}_1}, \mathcal{A}^{\mathsf{G}_2}) \leq \frac{q^2}{2^c}$. □

Claim 18 *For any Φ_{ae}-restricted RKA distinguisher $\mathcal{A}$ between game G_3 and G_4 defined in Fig. 9, there exists a Φ_m-restricted RKA adversary $\mathcal{A}_{prf}$ such that*

$$\mathbf{Adv}(\mathcal{A}^{\mathsf{G}_3}, \mathcal{A}^{\mathsf{G}_4}) \leq \mathbf{Adv}_{\mathsf{Tag}}^{\mathsf{rkaPRF}}(\mathcal{A}_{prf}, \Phi_m)\,.$$

[10] This transformation allows us to use a normal PRP for the simulation, and not a strong PRP.

Proof. $\mathcal{A}_{prf}$ simulates G_3 and G_4 for $\mathcal{A}$ with the oracles of the security game rkaPRF. For any request $(N, M, A, (\varphi_e, \varphi_m))$ to the oracle Enc, $\mathcal{A}_{prf}$ forwards (N, A, M, φ_m) to the rkaPRF game's oracle F to get the tag T, computes the ciphertext $C \leftarrow \mathtt{Enc}(\varphi_e(K_e), N, M \parallel T)$ locally with a random key K_e, and sends the ciphertext C to $\mathcal{A}$. Since $\mathcal{A}$ is Φ_{ae}-restricted, $\mathcal{A}_{prf}$ is Φ_m-restricted and $\mathcal{A}_{prf}$ perfectly simulates G_{b+3} where b is the challenge bit of game rkaPRF and outputs b' if $\mathcal{A}$ does. It holds that $\Pr[\mathcal{A}^{\mathsf{G}_3} \Rightarrow 0] - \Pr[\mathcal{A}^{\mathsf{G}_4} \Rightarrow 0] \leq \Pr[\mathcal{A}_{prf}^{\mathsf{rkaPRF}} \Rightarrow 0 \,|\, b = 0] - \Pr[\mathcal{A}_{prf}^{\mathsf{rkaPRF}} \Rightarrow 0 \,|\, b = 1]$. Hence, $\mathbf{Adv}(\mathcal{A}^{\mathsf{G}_3}, \mathcal{A}^{\mathsf{G}_4}) \leq \mathbf{Adv}_{\mathsf{Tag}}^{\mathsf{rkaPRF}}(\mathcal{A}_{prf}, \Phi_m)$. □

Claim 19 *For any Φ_{ae}-restricted RKA distinguisher $\mathcal{A}$ between game G_4 and G_5 with q queries to the encryption oracle defined in Fig. 9, it holds*

$$\mathbf{Adv}(\mathcal{A}^{\mathsf{G}_4}, \mathcal{A}^{\mathsf{G}_5}) \leq \frac{q^2}{2^c}\,.$$

Proof. Both games only differ from the choice of the underlying encryption. Game G_4 uses a random permutation and G_5 generates randomly chosen ciphertexts. Since the adversary is not allowed to query the same tuple $(N, A, M, (\varphi_e, \varphi_m))$ and $\mathtt{F}$ is a real random function, it follows T is fresh and uniform distributed or φ_e is fresh. In case of a fresh φ_e, it follows directly that C is chosen uniformly at random. If φ_e was already used, an adversary can only distinguish both games if it finds a collision in G_5 since G_4 uses a permutation it is not possible to get the same C twice with the same φ_e. The probability for such a collision is less then $\frac{q^2-q}{2^c}$. In detail we know that the probability to get a collision for the i^{th} query is less then $\frac{i-1}{2^c}$ and hence $\mathbf{Adv}(\mathcal{A}^{\mathsf{G}_4}, \mathcal{A}^{\mathsf{G}_5}) \leq \sum_{i=1}^{q} \frac{i-1}{2^c} \leq \frac{q^2}{2^c}$. This proves the claim. □

Similar to the N* construction, we can also use a block cipher (PRP) in the N3 construction to achieve security in the stronger security model. As discussed in the previous section this only works for N3 since the attack on N1 and N2 work independent of the underlying encryption scheme. Further, the security proof for N3 is similar to the proof of Theorem 14, we only have to adapt the block cipher that it also takes the nonce as input. We emphasize that the new construction N* is more efficient then N3, since the block cipher does not receive the nonce as an input. For the instantiation of the block cipher we refer to [5], where the authors construct an RKA-secure PRP using a three-round Feistel construction. The construction contains three RKA-secure PRFs, where the last two PRFs are initialized with the same key. Hence, with Theorem 14, we can build an mr-s-RKA-AE-secure AE scheme out of four RKA-secure PRFs, three to instantiate the block cipher and one for the MAC.

Acknowledgements

This work was funded by the German Research Foundation (DFG) – SFB 1119 – 236615297 and the Emmy Noether Program FA 1320/1-1 of the German Research Foundation (DFG).

References

1. Michel Abdalla, Fabrice Benhamouda, Alain Passelègue, and Kenneth G. Paterson. Related-key security for pseudorandom functions beyond the linear barrier. *Journal of Cryptology*, (4), 2018.
2. Divesh Aggarwal, Shashank Agrawal, Divya Gupta, Hemanta K. Maji, Omkant Pandey, and Manoj Prabhakaran. Optimal computational split-state non-malleable codes. In *TCC 2016-A, Part II*, 2016.
3. Martin R. Albrecht, Pooya Farshim, Kenneth G. Paterson, and Gaven J. Watson. On cipher-dependent related-key attacks in the ideal-cipher model. In *FSE 2011*, 2011.
4. Benny Applebaum, Danny Harnik, and Yuval Ishai. Semantic security under related-key attacks and applications. In *ICS 2011*.
5. Manuel Barbosa and Pooya Farshim. The related-key analysis of Feistel constructions. In *FSE 2014*, 2015.
6. Mihir Bellare and David Cash. Pseudorandom functions and permutations provably secure against related-key attacks. In *CRYPTO 2010*, 2010.
7. Mihir Bellare, David Cash, and Rachel Miller. Cryptography secure against related-key attacks and tampering. In *ASIACRYPT 2011*, 2011.

8. Mihir Bellare and Tadayoshi Kohno. A theoretical treatment of related-key attacks: RKA-PRPs, RKA-PRFs, and applications. In *EUROCRYPT 2003*, 2003.
9. Mihir Bellare and Chanathip Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In *ASIACRYPT 2000*, 2000.
10. Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In *EUROCRYPT 2006*, 2006.
11. Daniel J. Bernstein. CAESAR: Competition for Authenticated Encryption: Security, Applicability, and Robustness, 2014.
12. Rishiraj Bhattacharyya and Arnab Roy. Secure message authentication against related-key attack. In *FSE 2013*, 2014.
13. Eli Biham. New types of cryptanalytic attacks using related keys (extended abstract). In *EUROCRYPT'93*, 1994.
14. Eli Biham, Orr Dunkelman, and Nathan Keller. Related-key boomerang and rectangle attacks. In *EUROCRYPT 2005*, 2005.
15. Alex Biryukov and Dmitry Khovratovich. Related-key cryptanalysis of the full AES-192 and AES-256. In *ASIACRYPT 2009*, 2009.
16. Alex Biryukov, Dmitry Khovratovich, and Ivica Nikolic. Distinguisher and related-key attack on the full AES-256. In *CRYPTO 2009*, 2009.
17. Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the importance of checking cryptographic protocols for faults (extended abstract). In *EUROCRYPT'97*, 1997.
18. Jean Paul Degabriele, Christian Janson, and Patrick Struck. Sponges resist leakage: The case of authenticated encryption. In *ASIACRYPT 2019, Part II*, 2019.
19. Orr Dunkelman, Nathan Keller, and Jongsung Kim. Related-key rectangle attack on the full SHACAL-1. In *SAC 2006*, 2007.
20. Sebastian Faust, Juliane Krämer, Maximilian Orlt, and Patrick Struck. On the related-key attack security of authenticated encryption schemes. Cryptology ePrint Archive, Paper 2022/140, 2022.
21. Shuai Han, Shengli Liu, and Lin Lyu. Efficient KDM-CCA secure public-key encryption for polynomial functions. In *ASIACRYPT 2016, Part II*, 2016.
22. David G. Harris. Critique of the related-key attack concept. *Des. Codes Cryptogr.*, 2011.
23. Takanori Isobe. A single-key attack on the full GOST block cipher. In *FSE 2011*, 2011.
24. Lars R. Knudsen. Cryptanalysis of LOKI91. In *AUSCRYPT'92*, 1993.
25. Lars R. Knudsen and Tadayoshi Kohno. Analysis of RMAC. In *FSE 2003*, 2003.
26. Paul C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In *CRYPTO'96*, 1996.
27. Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *CRYPTO'99*, 1999.
28. Bonwook Koo, Deukjo Hong, and Daesung Kwon. Related-key attack on the full HIGHT. In *ICISC 10*, 2011.
29. Xianhui Lu, Bao Li, and Dingding Jia. KDM-CCA security from RKA secure authenticated encryption. In *EUROCRYPT 2015, Part I*, 2015.
30. Chanathip Namprempre, Phillip Rogaway, and Thomas Shrimpton. Reconsidering generic composition. In *EUROCRYPT 2014*, 2014.
31. National Institute of Standards and Technology. Lightweight cryptography standardization process, 2015.
32. Eric Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446, 2018.
33. Phillip Rogaway. Authenticated-encryption with associated-data. In *ACM CCS 2002*, 2002.
34. Phillip Rogaway. Nonce-based symmetric encryption. In *FSE 2004*, 2004.
35. Phillip Rogaway and Thomas Shrimpton. A provable-security treatment of the key-wrap problem. In *EUROCRYPT 2006*, 2006.
36. Serge Vaudenay. Clever arbiters versus malicious adversaries - on the gap between known-input security and chosen-input security. In *The New Codebreakers*, Lecture Notes in Computer Science. Springer, 2016.
37. Keita Xagawa. Message authentication codes secure against additively related-key attacks. Cryptology ePrint Archive, Report 2013/111, 2013.

E. Combined Fault and Leakage Resilience: Composability, Constructions and Compiler

This chapter corresponds to our published article at CRYPTO 2023 [36], with minor edits. Our full version can be found in [35].

Combined Fault and Leakage Resilience: Composability, Constructions and Compiler

Sebastian Berndt[1], Thomas Eisenbarth[2], Sebastian Faust[3], Marc Gourjon[4,5], Maximilian Orlt[3], and Okan Seker[5]

[1] Institute for Theoretical Computer Science, University of Lübeck, Germany s.berndt@uni-luebeck.de
[2] Institute for IT Security, University of Lübeck, Germany thomas.eisenbarth@uni-luebeck.de
[3] TU Darmstadt, Germany, firstname.lastname@tu-darmstadt.de
[4] Hamburg University of Technology, Germany, firstname.lastname@tuhh.de
[5] NXP Semiconductors, Germany

Abstract. Real-world cryptographic implementations nowadays are not only attacked via classical cryptanalysis but also via implementation attacks, including passive attacks (observing side-channel information about the inner computation) and active attacks (inserting faults into the computation). While countermeasures exist for each type of attack, countermeasures against combined attacks have only been considered recently. Masking is a standard technique for protecting against passive side-channel attacks, but protecting against active attacks with additive masking is challenging. Previous approaches include running multiple copies of a masked computation, requiring a large amount of randomness or being vulnerable to horizontal attacks. An alternative approach is polynomial masking, which is inherently fault-resistant.

This work presents a compiler based on polynomial masking that achieves linear computational complexity for affine functions and cubic complexity for non-linear functions. The resulting compiler is secure against attackers using region probes and adaptive faults. In addition, the notion of fault-invariance is introduced to improve security against combined attacks without the need to consider all possible fault combinations. Our approach has the best-known asymptotic efficiency among all known approaches.

1 Introduction

In classical cryptography, the security of cryptographic primitives is often analyzed in the black-box model. In this model, the adversary attacks the cryptographic algorithm via access to inputs and outputs but has no knowledge and no control over the inner workings of the algorithms. In particular, sensitive information such as the secret key is hidden from and out of the control of the adversary. Unfortunately, when running cryptographic algorithms on real-world devices countless attacks demonstrate that the black-box model is far too optimistic. Examples include *passive attacks*, where the adversary exploits physical phenomena such as the power consumption or running time of a device to extract sensitive information; or *active attacks*, where the adversary modifies temporary values via a laser or via heating up the device to introduce faulty computation.

Masking schemes. Masking schemes are a classical countermeasure to protect against passive side-channel attacks. Masking conceals sensitive information by secret sharing each value v from some finite field $\mathbb{F}$ into shares $v_0, \ldots, v_{n-1}$ such that $d+1$ shares are required to reconstruct the secret, while $\leq d$ shares reveal nothing about the sensitive value v. The most common sharing scheme is *additive* masking. Here, we choose $v_0, \ldots, v_{n-2}$ randomly from $\mathbb{F}$ and define v_{n-1} such that $v = v_0 \oplus v_2 \ldots \oplus v_{n-1}$, where $\oplus$ is the addition of the underlying field $\mathbb{F}$. The main challenge in designing masking schemes is to securely compute on the shared values. To this end, we design masked subcircuits, called *gadgets*, that securely compute on sharings and devise methods for securely composing such gadgets without violating overall security.

The security of a masking scheme is typically analyzed in the so-called probing model originally introduced by Ishai, Sahai and Wagner [ISW03]. In this model, the adversary can learn up to d values that are produced during the computation. The security proof is typically done by analyzing the d-probing security of the

gadgets and then extending it towards security of an entire masked circuit via *composition*. To argue secure composition in the probing model, an important property is *(strong) non-interference (SNI)*. Intuitively, this property guarantees that all information gained by the attacker by probing d internal values of a gadget can also be obtained by probing at most d shares of the masked input. Furthermore, probes on the output sharing can be simulated from scratch in the case of the stronger notion.

Beyond passive security. As mentioned above, passive side-channel attacks are not the only threat to cryptographic implementations. In practice, an adversary may also be able to induce faults into the computation thereby breaking the cryptographic implementation. Even worse, a physical adversary may launch combined attacks, where the adversary both passively observes side-channel leakage and introduces faults to break the cryptographic implementation. While a masking scheme can be used to protect against the passive adversary, it is easy to see that it fails to offer security against faults. For instance, if an adversary succeeds in adding an offset $c \in \mathcal{F}$ to only one of the shares, the result of the computation is faulty, which may have catastrophic consequences for security [BS97]. Hence, we need to extend probing security to also include fault attacks, where in addition to placing d probes, the adversary is allowed to induce ϵ faults. In this work, we consider arbitrary *adaptive* faults that might even depend on the information obtained via previous probes.

With adaptive faults, we cover attacks where the adversary uses information from previous leakage to insert faults. While such attackers seem unrealistically strong, they are possible in the context of side-channel attacks: Firstly, the adversary has access to the device and could stop or slow down regions of the devices similar as in cold boot attacks. Secondly, the adversary could use the leaked information not immediately but in the next cycle of a circuit that uses multiple cycles, such as AES and Present. Finally, we note that adaptive adversaries are a stronger adversarial model. Thus, from a theoretical point of view it is interesting to explore what security can be achieved in this model.

In order to protect against combined attacks, two main approaches have been considered in the literature – duplicated masking and polynomial masking. The most common one is duplicated masking to replicate the masked computation [DN20b, FRSG22].

Duplicated Approach. In this setting, the masked circuit $\hat{C}$ is executed $\epsilon + 1$ times in parallel. After each gate, the masked outputs of the computation are checked for equality to detect faults. This requires that the $\epsilon + 1$ copies use the same randomness internally. Otherwise, the output sharings would not be equal. Moreover, re-using the randomness has an additional advantage. As generating randomness is costly, by re-using randomness in all $\epsilon + 1$ copies, the overall randomness used can be reduced by a factor of $O(\epsilon)$.

The duplication approach described above has two important shortcomings. First, affine operations that traditionally can be masked at very low cost (typically at an $O(n)$ complexity overhead for $n = \epsilon + d$) get significantly more expensive as each such masked affine operation is now computed $\epsilon + 1$ times. This is especially problematic, as many modern primitives, e.g., [BDPA13, GLR+20, ARS+15, AGR+16, AAB+20, HKL+22, GKR+21] aim to reduce the number of non-linear operations by increasing the number of affine operations significantly. Even worse, in terms of security, the duplication approach is very vulnerable to so-called horizontal attacks [BCPZ16].

Horizontal Attacks. Horizontal attacks are attacks in which an adversary exploits the fact that multiple computations share the same randomness or secret key material. In the context of side-channel attacks, horizontal attacks can be particularly devastating as the attacker can amplify the leaked information and thus recover sensitive information more easily [CFG+10, ORSW12, VGS14, BS21]. To protect against horizontal attacks, it is essential to ensure that different instances of computation use independent and fresh randomness or secret key material.

Clearly, the duplication method is particularly vulnerable to horizontal attacks, as all copies share the same randomness to ensure fault detection. This drawback was already observed and explicitly stated in [FRSG22]. To illustrate this issue more clearly, the full version contains calculations describing the influence of the duplication on attacks in the random probing model. The random probing model is the standard method to analyze security of the masking countermeasure against horizontal attacks. In this model, the adversary can choose an *unbounded number of wires* and receives the value on each chosen wire with probability p.

Alternatively, such attacks can be modeled in the region probing model, also introduced in [ISW03]. Here, the threshold model is extended so that the threshold of probes applies to each gadget (or regions) in the circuit. In other words, the total number of probes increases with the number of gadgets. It has been shown by Duc et.al. [DDF14] that security in this model also implies security in the random probing model. Recently, this property has been used to construct secure compilers e.g. [ADF16, GPRV21]. In this work, we also allow up to t probes in each gadget to model horizontal attacks. One possible way to improve security against horizontal attacks is to use fresh randomness in each copy, but (a) it is not straightforward to detect faults in such randomized computation, and (b) the randomness complexity increases to $O(|C|n^3)$.

An alternative approach to executing the masked computation multiple times is to use an different sharing scheme. Recall that additive secret sharing is highly vulnerable to fault attacks as already a single fault is undetectable. Hence, using an additive sharing intuitively requires a large number of independent copies of the same computation to avoid these faults. To tackle this problem, an obvious idea is to resort to error detection codes, where one of the most promising candidates are Reed Solomon codes (often also called Shamir's secret sharing), which in addition to error detection also offer linearity. The latter is particularly useful for carrying out computation with sharings. In the literature, masking schemes based on Shamir's secret sharing are often called *polynomial masking*.

Polynomial Masking. Here, we need $|\mathbb{F}| \geq n+1$ and choose pairwise different support points $\alpha_0, \ldots, \alpha_{d-1} \neq 0$. To share a value v, we construct a polynomial $f \in \mathbb{F}[x]$ of degree d such that $f(0) = v$. The i-th share v_i is now defined as $f(\alpha_i)$. Polynomial masking [CPR12, GM11, RP12] is a well-known countermeasure against side-channel attacks, which offers advantages over additive secret sharing based schemes due to its higher algebraic complexity. Moreover, they allow for a simple protection against faults: We can add *redundant* points $\alpha_d, \alpha_{d+1}, \ldots \alpha_{n-1}$ and corresponding shares $v_{d+1}, v_{d+2}, \ldots, v_{n-1}$, but will still use a polynomial of degree d. Due to the error-correcting properties of these polynomial codes, *valid* codewords, i.e., those sharings describing a polynomial of degree d, will differ in at least $n-d$ positions. If an attacker modifies less than $n-d$ shares, the underlying polynomial (which can be interpolated from the shares v_i) will have degree at least $d+1$ due to the fundamental theorem of algebra. Hence, modification of only $n-d$ shares results in an *invalid sharing*.

The idea of using polynomial sharing was already used in the context of multiparty computations in the now classical work of Ben-Or, Goldwasser, and Wigderson [BGW88]. Using a more complicated scheme, called *verifiable secret sharing*, they show how to achieve perfect security if the number of corrupted parties is strictly less than $n/3$. Inspired by this, Seker, Fernandez-Rubio, Eisenbarth, and Steinwandt [SFRES18] adapted the BGW scheme to protect against combined attackers. Their main idea is to simplify the BGW multiplication to avoid using verifiable secret sharing in such a way that faulted inputs will lead to a faulted output with high probability. This allowed them to show that $n = 2d + \epsilon + 1$ shares are sufficient to protect against d probes and ϵ *additive* faults, i.e., faults where the attacker can add an arbitrary value to a wire (independently from the actual value on that wire). Both the randomness requirement and the computational complexity of their multiplication gadgets are asymptotical identical to those using the duplication approach, i.e., $O(n^2)$ and $O(n^3)$ respectively. But, due to their linear number of shares, they can compute affine operations in *linear* time. In this work, we show that $n = d + \epsilon + 1$ are both sufficient and necessary to protect against combined attacks. In particular, in contrast to earlier works [SFRES18, DN20b, FRSG22, RFSG22], we show security in a stronger adversarial model where the adversary can induce *adaptive* faults into the computation and security holds in the *region probing* model.

Our approach has the same asymptotical complexities for multiplication as both the duplicated approach and the approach of [SFRES18] and a linear complexity for affine operations. Furthermore, in contrast to the duplicated approach, our solution is provably resistant against horizontal attacks.

1.1 Contribution.

Our contributions are threefold. First, we present combined security notions suitable for *polynomial masking*. Second, we propose the notion of *fault-invariance*, that allows us to transform gadgets secure against probing attacks into ones that are secure against combined attacks. Third, we propose two new compilers which use

the optimal (*linear*) number of $n = e + d + 1$ shares and show security against horizontal attacks in the region probing model.

Combined Security Notions for Polynomial Sharing. In previous works [SFRES18, DN20b, FRSG22, RFSG22], an (d, ϵ)-attacker was able to choose d wires for probes and ϵ wires for faults (and corresponding fault operations from a class of possible faults). Then, the circuit was faulted according to these faults and the values of the d chosen wires were given to the attacker aiming to extract some sensitive information from these values. We strengthen the attackers significantly with regard to both probes and faults by allowing *region probes* and *adaptive faults*. Informally, region probes allow to perform d probes per *gadget*, in contrast to d probes in total. Furthermore, our attacker can choose the fault applied to a wire based on the already observed probes adaptively. A formal description about the attacker model is presented in Section 3.

A careful analysis of the differences between additive masking and polynomial masking reveals that the previously used security definitions do not transfer easily. In additive masking, we want to give an upper bound on the number of faulty outputs, while in polynomial masking we want to give a lower bound on the *degree* of the polynomial described by the sharing. We present new definitions adapted to this difference that allow to argue the composability of two secure gadgets. Here, composability means if gadgets satisfy certain security properties, these properties also hold for more complex computation that is composed of such gadgets.

Simplification of combined security analyzes. The previous approach to prove the security against combined attackers was to verify probing security of these gadget for *all* possible fault combinations [RFSG22]. This often leads to very complicated proofs with many case distinctions and many optimizations developed cannot be reused. We introduce the notion of *fault-invariance* of a gadget that allows us to *lift* probing-secure gadgets to also be secure against combined attacks *without* the need to consider all possible fault combinations. A fault-invariant gadget that is (S)NI stays (S)NI even in the presence of faults and thus allows us to reuse existing probing-secure gadgets.

A new countermeasure for combined attacks. Finally, we present two new compilers secure against combined attackers using adaptive faults in the region probing model. These are the first such compilers as the existing countermeasures using additive masking are very vulnerable to such attacks. Compared to [SFRES18], we significantly reduce the number of needed shares from $2e + d + 1$ down to $n = e + d + 1$ (which we also show as *optimal*). Along the way, we also show how to fix their approach by presenting an SNI-secure refresh. Compared to [DN19], we significantly reduce the number of needed random values down to $O(n^2)$ and the computational complexity down to $O(n^3)$. Finally, we also show that our compilers are secure against *horizontal attacks*, a feature explicitly not shared by [DN20b, FRSG22]. All of the approaches using duplicated sharing [DN20b, FRSG22] need a *quadratic* number of shares, hence their complexity will always be suboptimal for affine circuits or circuits with a very large number of affine gates, a feature of many modern blockciphers, e.g., [AGR+16, AAB+20, HKL+22, GKR+21].

For a comparison of our work to other works protecting against combined attacks, we refer to Table 1. Analysing the cryptographic primitives Keccak [BDPA13], LowMC [ARS+15] or HadesMiMc [GLR+20] yields complexity estimations shown in Table 2. These estimations show that our approach outperforms the duplication approach due to the large number of linear operations.

1.2 Related Work.

The study of private circuits was initiated by the work of Ishai, Sahai, and Wagner who presented a generic compiler to protect against probing attacks [ISW03]. In a follow-up work, Ishai, Prabhakaran, Sahai, and Wagner also considered fault attacks and presented a corresponding compiler [IPSW06]. Note that a combination of two protection mechanisms against probing attacks and fault attacks might actually *lower* the security of the protection mechanisms [REB+08, LFZD14]. Similar to the work of Ishai, Prabhakaran,

[6] This result is only present in the version `20190603:070457` of the eprint paper.

Table 1: A comparison of the complexity of the addition, multiplication and refresh gadgets with regard to $n = e + d$.

	# Shares	Multiplication		Addition	Refresh		Horizontal Att.
		Rand	Compl	Compl	Rand	Compl	Security
[DN20b, FRSG22]	$O(n^2)$	$O(n^2)$	$O(n^3)$	$O(n^2)$	$O(n^2)$	$O(n^3)$	✖
[SFRES18]	$O(n)$	$O(n^2)$	$O(n^3)$	$O(n)$	insecure		✖
[DN19][6]	$O(n)$	$O(n^3)$	$O(n^5)$	$O(n)$	$O(n^3)$	$O(n^5)$	✖
This Work	$O(n)$	$O(n^2)$	$O(n^3)$	$O(n)$	$O(n^2)$	$O(n^3)$	$O(n^{-2})$

Table 2: A rough estimation of the number of operations of our approach compared to the duplication approach of [DN20b, FRSG22] for $n = 8$ for Keccack [BDPA13], LowMC ($R = 55, m = 20$) [ARS+15], and HadesMiMc ($R_F = 10$) [GLR+20]. The numbers given here depend on estimations given in the corresponding works.

	#Add	#Mult	#Ops [DN20b, FRSG22]	#Ops This Work
Keccak	422 400	38 400	46 694 400	23 040 000
LowMC	28 894 643	3 300	1 850 946 752	232 846 744
HadesMiMc	1 820	150	193 280	91 360

Sahai, and Wagner, the use of error-detection codes together with threshold implementations was studied by Schneider, Moradi, and Güneysu [SMG16] and by De Cnudde and Nikova [CN16]. Recently, the use of explicit multi-party computation protocols as protection mechanisms was studied by Reparaz, De Meyer, Bilgin, Arribas, Nikova, Nikov, and Smart [RDB+18] and by Dhooghe and Nikova [DN20a]. Closest to this paper is the work of Seker, Fernandez-Rubio, Eisenbarth, and Steinwandt that introduced the model of statistical security against fault attacks [SFRES18][7]. They also showed that the classical multi-party protocol of Ben-Or, Goldwasser, and Wigderson [BGW88] can be adapted to this scenario and reduced the number of shares need from $n \geq 3d + 1$ down to $2d + e + 1$.

Previous security notion. The most common security notions against probing-only attacks are *non-interference (NI)* and its stronger counterpart *SNI*. The stronger security notion provides very useful composition results. Namely, it guarantees that the composition of d-SNI gadgets is d-SNI again. Now, using d-SNI gadgets prevents a probing-only attacker (where $e = 0$) from obtaining any information, but faults might still be used to obtain information. Security notions against probing-only attackers have been studied intensively and it was shown that the non-interference notions indeed prevent realistic attacks (see also Duc, Dziembowski, and Faust [DDF14]). An alternative approach, called *probe-isolating non-interference (PINI)* was introduced by Cassiers and Standaert [CS20] and also allows composability.

For fault attacks, the situation is not as easy, as different strategies might lead to different properties that are non-comparable. The simplest behavior, *fault detection*, aims to detect possible faults. Now, one can regularly check for the existence of these faults and abort the computation to prevent information leakage. In a more complex setting, *fault correction*, the computation would try to correct possible faults. While fault correction is a very useful property, it usually comes at prohibitive cost. We thus only focus on the detection of faults. Adding fault checks after every gate would detect the presence of faults as early as possible, but would increase the cost of the computation severely. Our first goal is thus to minimize the number of fault checks. The existence of multiple successive gates where no fault detection is used opens up the danger of *ineffective faults*, i.e., faults that only change some parts of the computation, but do not change the output. More informally, these faults cancel out at some point in the computation. As shown, e.g., by Clavier [Cla07] or Dobraunig, Eichlseder, Korak, Mangard, Mendel, and Primas [DEK+18] these (statistical) ineffective faults

[7] In the full version we show that the construction has a bug.

can be used by attackers in a devastating way. To protect against such faults, we design *fault-robust* gadgets: If these gadgets are given faulted inputs, their outputs will also contain faults.

As described earlier, Dhooghe and Nikova [DN20b] introduced the notion of (S)NINA, a combination of (strong) non-interference and non-accumulation, to protect against combined attackers using d probes and ϵ faults. They showed that a duplicate additive sharing is sufficient to obtain security by presenting a multiplication gadget and a refresh gadget that provided security against combined attacks. Richter-Brockmann, Feldtkeller, Sasdrich, and Güneysu [RFSG22] extended the (S)NINA notion to provide accurate definitions for the hardware context and constructed a tool, VERICA, to analyze gadgets with regard to (S)NINA. Finally, Feldtkeller, Richter-Brockmann, Sasdrich, and Güneysu [FRSG22] adapted the related notion of probe-isolating non-interference (PINI) presented by Cassiers and Standaert [CS20] to fault attacks and combined attacks. Similar to the work of Dhooghe and Nikova, they also used duplicate additive sharing and designed corresponding multiplication and refresh gadgets for these sharings that are secure against combined attacks. In contrast to our work, they only allowed static non-adaptive faults and a total number of d probes (i.e., their attacker only worked in the classical threshold probing model and not in the region probing model). Furthermore, as each copy of the computation uses the same randomness, their approach is very vulnerable to horizontal attacks, as shown in the full version. We summarize the efficiency with regard to $n = e + d$ of the constructions of the previous works in Table 1.

2 Background

In this section, we fix the notation used throughout this paper and give the needed background on polynomials and circuits.

Notation. We denote the set of the numbers between 0 and $n-1$ by $[n]$, i.e., $[n] = \{0, \dots, n-1\}$. We write $r \leftarrow_{\$} S$ to denote that r is a random, uniformly distributed element from the finite set S. To simplify notation, if we are given a vector $(v_0, \dots, v_{n-1})$, we write $(v_i)_{i\in I}$ to denote a vector only consisting of the elements v_i with $i \in I$ for $I \subseteq [n]$ and thus also $(v_i)_{i\in[n]}$ instead of $(v_0, \dots, v_{n-1})$. If D and D' are probability distribution over domain X, we write $D \equiv D'$, if $D(x) = D'(x)$ for all $x \in X$, i.e., if the distributions agree at each point. Random variables $X_0, X_1, \dots X_{n-1}$ over a set $\mathbb{F}$ are independent if it holds for any $a_0, a_1, \dots a_{n-1} \in \mathbb{F}$ that $\Pr[X_0 = a_0, X_1 = a_1, \dots X_{n-1} = a_{n-1}] = \prod_{i\in[n]} \Pr[X_i = a_i]$. We write that $X_0, X_1, \dots X_{n-1}$ are k-wise independent if for any subset $I \subset [n]$ with $|I| \leq k$, the random variables $(X_i)_{i\in I}$ are independent. For a matrix A, its rank is denoted by $\mathrm{rank}(A)$ and its kernel by $\ker(A)$. The dimension of a linear subspace H is denoted by $\dim(H)$. The weight of a vector $(a_i)_{i\in[n]}$ is the number of non-zero elements $\mathrm{weight}((a_i)_{i\in[n]}) = |\{a_i : a_i \neq 0\}|$. Further, we use polynomials in $\mathbb{F}[x]$ that are functions $f : \mathbb{F} \to \mathbb{F}$ with $f(x) = \sum_{i=0}^{k} f_i x^i$ for a natural number k and for all $f_i \in \mathbb{F}$. The degree $\deg(f)$ of f is the highest index of the non-zero f_i's. In detail, $\deg(f) = \max_i\{i\colon \text{ with } f_i \neq 0\}$.

We say that a probability distribution D is *perfectly simulatable* from a set S, if there exists a simulator Sim such that the output of $\mathsf{Sim}(S)$ has the same distribution as D. In detail, it holds for any possible x in the domain of D that $\Pr[\mathsf{Sim}(S) = x] = \Pr[D = x]$. In the following we denote this with $\mathsf{Sim}(S) \equiv D$.

Polynomial sharing. Throughout this work, we will fix a finite field $(\mathbb{F}, \oplus, \odot)$ with addition $\oplus$ and multiplication $\odot$ such that $|\mathbb{F}| \geq n + 1$, where n will be clear from the application. For the sake of simplicity, we will often also write $\cdot$ instead of $\odot$ and $+$ instead of $\oplus$. Throughout this paper, we fix n pairwise different support points $\alpha_0, \dots, \alpha_{n-1} \in \mathbb{F}$ with $\alpha_i \neq 0$. We will often represent a polynomial $f \in \mathbb{F}[x]$ via the shares $(F_i)_{i\in[n]}$ with $F_i = f(\alpha_i)$ and say that $(F_i)_{i\in[n]}$ is a degree d sharing. To see that $(F_i)_{i\in[n]}$ is indeed a valid representation of f, consider the *Vandermonde matrix* $V_{n,d} := V_{n,d}[\alpha_0, \dots, \alpha_{n-1}]$, where the i-th row has the form $(1, \alpha_i, \alpha_i^2, \dots, \alpha_i^d)$. It is now easy to see that $V_{n,d} \cdot (f_0, \dots, f_d)^T = (f(\alpha_0), f(\alpha_2), \dots, f(\alpha_{n-1}))^T = (F_0, \dots, F_{n-1})^T$, i.e., the Vandermonde matrix can be used to evaluate the polynomial on the public support points α_i. Furthermore, it is well known that $\det(V_{n,d}) = \prod_{0\leq i<j\leq n-1}(\alpha_i - \alpha_j)$. As the α_i are pairwise different and belong to a field $\mathbb{F}$ (which is free of nonzero zero divisors), this determinant is non-zero. Hence, $V_{n,d}$ is regular and the inverse matrix $V_{n,d}^{-1}$ thus exists. As $V_{n,d} \cdot (f_0, \dots, f_d)^T = (F_0, \dots, F_{n-1})^T$, we have

$(f_0, \ldots, f_d)^T = V_{n,d}^{-1} \cdot (F_0, \ldots, F_{n-1})^T$. Hence, the inverse Vandermonde matrix can interpolate the coefficients f_i from the shares F_i via a linear operator.

To share a sensitive value $s \in \mathbb{F}$ into n shares, we will construct a polynomial f with $f(x) = \sum_{i=0}^{n-1} f_i x^i$ of degree $n-1$ where the coefficients $f_1, f_2, \ldots, f_{n-1}$ are chosen randomly and f_0 is equal to the sensitive value s. Then, the value $F_i = f(\alpha_i)$ is the i^{th} share. We denote this randomized procedure that outputs $(F_i)_{i\in[n]}$ from s by $(s_i)_{i\in[n]} \leftarrow \mathsf{Enc}(s)$ with $s_i = f(\alpha_i)$. To recover the sensitive value $f_0 = s$, we only need the first row of $V_{n,d}^{-1}$. We denote the i-th element of the first row of $V_{n,d}^{-1}$ by $\lambda_i^{(0)}$. To reconstruct the shared value, we use the well-known *interpolation lemma*.

Lemma 1 (Interpolation Lemma). *Let $f \in \mathbb{F}[x]$ be a polynomial of degree $d \leq n$, let $\alpha_0, \ldots, \alpha_{n-1}$ be pairwise different support points in $\mathbb{F} \setminus \{0\}$, and let $\lambda_i^{(0)}$ be the entries of the first row of the inverse Vandermonde matrix $V_{n,d}[\alpha_0, \ldots, \alpha_{n-1}]$. Then $(\lambda_1^{(0)}, \ldots, \lambda_n^{(0)}) \cdot (f(\alpha_0), \ldots, f(\alpha_{n-1})) = f(0)$.*

To simplify notation, we also write $v \leftarrow \mathsf{Dec}((v_i)_{i\in[n]})$ with $v_i = f(\alpha_i)$ for this. Since f is of degree d, the sensitive value v can be reconstructed from $(v_i)_{i\in I}$ with any subset $I \subset [n]$ with $|I| > d$ and $(v_i)_{i\in I}$ is independent of v if $|I| \leq d$.

An important fact that we will make use of throughout this paper is the fact that a sharing $(F_i)_{i\in[n]}$ of a non-zero polynomial with many zero entries corresponds to a polynomial of high degree, as captured by the following well-known fact.

Lemma 2 (Fundamental Lemma). *Let $f \in \mathbb{F}[x]$ be a polynomial of non-zero degree. If f has k distinct roots,* $\deg(f) \geq k$.

Circuit model. We represent the computation via a circuit on the field $(\mathbb{F}, \oplus, \odot)$, i.e., we consider directed acyclic graphs G where each node is labeled as (i) input gate, (ii) output gate, (iii) random gate, (iv) addition gate, (v) multiplication gate, or (vi) constant (transformation) gate. To compute the circuit on given inputs $x_1, x_2, \ldots$ we first initialize the input gates with the according inputs. Then, at each time step, we evaluate all gates that only have parents that are already evaluated. Random gates are evaluated by sampling an element uniformly at random from $\mathbb{F}$. Constant transformation gates have two constants a and b and evaluate $a \cdot x + b$ on input x. For $a = 0$ it is the usual constant gate initialized with b. We denote the resulting output distribution $y_1, y_2, \ldots$ of circuit C with inputs $x_1, x_2, \ldots$ by $(y_1, y_2, \ldots) \leftarrow \mathsf{C}(x_1, x_2, \ldots)$. In order to simplify notation, we also write $\mathsf{C}^R(x_1, x_2, \ldots)$ for the output of C if the samples from the random gates are taken according to the *random values* R. A *gadget* is simply a subgraph of a circuit. We stress that our definition allows for an arbitrary out-degree of a gate. Hence, there is no need for copy gates or similar. Instead of outputting the result of the computation, a circuit can also *abort* the computation by returning the abort signal $\perp$.

Compiler. A *compiler* $\mathfrak{C}$ is a function transforming a circuit C into another circuit C'. We are interested in compilers that output fault- and leakage-resilient circuits C'. This can be done with polynomial sharing $(\mathsf{Enc}(\cdot), \mathsf{Dec}(\cdot))$ described above such that the circuit C' only computes on encoded values. Therefore, each gate is transformed into a sub-circuit, a so-called *gadget*, that takes as input the encoded inputs of the gate and outputs encodings such that the decoded output represents the outputs of the gate. For security reasons, additional randomness can be injected by so-called refresh gadgets that take as input an encoding and outputs a randomized encoding in such a way that the decoded value of the input and output is the same. For any circuit $\mathsf{C} : \mathbb{F}^n \to \mathbb{F}^m$ with n inputs and m outputs, the resulting compiler $\mathfrak{C}$ generates $\mathsf{C}' \leftarrow \mathfrak{C}(\mathsf{C})$ such that for any input $x^0, x^1, \ldots x^{n-1}$ and $(y_i^0)_{i\in[n]}, (y_i^1)_{i\in[n]}, \ldots (y_i^{m-1})_{i\in[n]} \leftarrow \mathsf{C}'(\mathsf{Enc}(x^0), \mathsf{Enc}(x^0), \ldots \mathsf{Enc}(x^{n-1}))$ it holds that $\mathsf{Dec}((y_i^0)_{i\in[n]}), \mathsf{Dec}((y_i^1)_{i\in[n]}), \ldots \mathsf{Dec}((y_i^{m-1})_{i\in[n]}) = \mathsf{C}(x^0, x^1, \ldots x^{n-1})$. In this case we also write that C and C' are arithmetic circuits over $\mathbb{F}$ and $\mathbb{F}^n$, respectively to highlight the fact that C' is working on the shared representation. Further, we say that C' is a *masked circuit* and has the same functionality as C. Section 4 gives a more detailed construction of the compiler and defines all the required gadgets.

Security Notions. When the adversary has access to a device to perform side-channel and fault attacks, we assume that the adversary can run the device and probe up to d intermediate values. The first and simplest security definition, *d-probing security* requires that the observation of up to d intermediate values in a masked circuit does not reveal anything about the unmasked variables.

Definition 1. *A masked circuit* C *with* k *inputs* $(x_i^j)_{i\in[n]} \leftarrow \mathsf{Enc}(x^j)$ *and* $j \in [k]$ *is* d-probing secure *if for any set* $\mathcal{L}$ *of* d *probes in*

$$\mathsf{C}((x_i^0)_{i\in[n]}, (x_i^1)_{i\in[n]}, \ldots, (x_i^{k-1})_{i\in[n]}))$$

there is a simulator Sim *only having access to* $\mathsf{C}(\cdot)$ *without the* k *secrets* x^j *such that* $\mathcal{L} \equiv \mathsf{Sim}(\mathsf{C}(\cdot))$ *for any secret* $x^0, x^1, \ldots x^{k-1}$.

Note that Sim has only access to the circuit C but not to the secrets $x^0, x^1, \ldots x^{k-1}$. In other words, the probes $\mathcal{L}$ are independent of the unmasked values. A stronger security definition is the d-region-probing model that also takes the circuit size into account. In detail it allows d-probes in each gadget of the masked circuit.

Definition 2. *A masked circuit* C *with* k *inputs* $(x_i^j)_{i\in[n]} \leftarrow \mathsf{Enc}(x^j)$ *and* $j \in [k]$ *is* d-region-probing secure *if for any set* $\mathcal{L}$ *of* d *probes in each gadget of*

$$\mathsf{C}((x_i^0)_{i\in[n]}, (x_i^1)_{i\in[n]}, \ldots, (x_i^{k-1})_{i\in[n]}))$$

there is a simulator Sim *only having access to* $\mathsf{C}(\cdot)$ *without the* k *secrets* x^j *such that* $\mathcal{L} \equiv \mathsf{Sim}(\mathsf{C}(\cdot))$ *for any secret* $x^0, x^1, \ldots x^{k-1}$.

It turned out that probing security of two circuits does not always imply probing security of its composition. Since composition results are very useful and allow the construction of compilers that work on a gate-by-gate basis, stronger definitions were subsequently developed. In the following, we give some stronger security definitions, well suited to masked circuits (or gadgets). To simplify presentation, we consider only gadgets having a single output sharing $(y_i)_{i\in[n]}$. We refer to Cassiers and Standaert for discussion of gadgets with multiple outputs [CS20]. An important notion to achieve composability in the presence of probing attacks is the notion of *d-Non-Interference* (d-NI) and *d-Strong-Non-Interference* (d-SNI) [BBD+16]. Both definitions guarantee that the leakage of up to d probes is independent of the shared secret.

Definition 3 (d-NI [BBD+15, BBD+16]). *A gadget* G *with one output sharing is* d-non-interfering (d-NI) *if and only if every set of* $d' \leq d$ *internal probes can be (perfectly) simulated with at most* d' *shares of each input sharing.*

The stronger d-SNI notion requires to distinguish between intermediate and output probes and guarantees that only the number of intermediate probes affects the number of inputs required by the simulator, easing the compilation of circuits.

Definition 4 (d-SNI [BBD+16]). *A gadget with one output sharing is* d-strong-non-interfering (d-SNI) *if and only if for every set* I *of* d_1 *internal probes and every set* O *of* d_2 *output probes such that* $d_1 + d_2 \leq d$, *the set of probes* $I \cup O$ *can be (perfectly) simulated with* d_1 *shares of each input sharing.*

Note PINI is another useful security notion for compositions in the threshold model. We omit this definition in the main body since it is vulnerable to horizontal attacks, and thus does not provide good properties for proofs in the region probing model. A detailed analysis is given in the full version.

3 Combined Security Model

As mentioned in the introduction, many countermeasures defending against fault attacks *or* probing attacks have been studied in the literature, but the task to protect against both attacks at the same time has only received more attention in the last few years. In this section we analyze the combined security of both fault resilience and probe resilience. To understand the influence of different kind of faults, we will model these faults as set of functions. An adversary with access to the class of faults $\mathcal{F}$ is able to change the value x

to $\zeta(x)$ for $\zeta \in \mathcal{F}$ during the computation. For the sake of simplicity, we always assume that the identity function id is part of every class $\mathcal{F}$. A fault attack now applies several of these faults to different wires of $\mathsf{C}^R \colon \mathbb{F}^k \to \mathbb{F}^l$. More formally, if the wires of the circuit C^R are numbered by $1, \ldots, W$, a *fault attack* T is a tuple of functions $T = (\zeta_1, \ldots, \zeta_W)$ with $\zeta_i \in \mathcal{F}$ for all $i = 1 \ldots W$ that describes how the value of each wire i is faulted. This means that such a wire i gets a value z_i from its output gate, but the following gate gets as input the faulted value $\zeta_i(z_i, \boldsymbol{u}_i)$, where ζ_i is the i^{th} function in $T = (\zeta_1, \ldots, \zeta_W)$ and $\boldsymbol{u}_i$ are the values already revealed by probes. We write $\mathsf{A}(\mathcal{F})$ to refer to the set of all possible fault attacks using the fault-functions $\mathcal{F}$. To simplify notation, we will often only use the ordering of the wires implicitly. If we tamper the circuit C^R with a fault attack $T \in \mathsf{A}(\mathcal{F})$ we write $T[\mathsf{C}^R]$. Due to physical constraints, a typical attacker cannot fault arbitrary many wires and is thus restricted (for example, [SFRES18] considers at most 3 faults and [RDB+18] at most 8 faults). For a fault attack $T \in \mathsf{A}(\mathcal{F})$, we write $|T|$ for the number of non-identity faults used, i.e., $|T| = |\{i \in \{1, \ldots, W\} \mid \zeta_i \neq \mathrm{id}\}|$ with $T = (\zeta_1, \ldots, \zeta_W)$. In the following we often consider different types of fault sets. In the most general case, we use *wire independent faults* $\mathcal{F}^{ind} := \{\text{all functions } \zeta \colon \mathbb{F} \times \mathbb{F}^* \to \mathbb{F}\}$ to show that the attacker can fault arbitrarily. We stress here that our model implies that the faults performed on different wires are somewhat *independent*, as they each only consider a single wire. An often studied restriction are *additive faults* $\mathcal{F}^+ := \{\zeta : \zeta(x, \boldsymbol{u}) = x + a \text{ for all } a \in \mathbb{F}\}$ that fault the wires value by adding an arbitrary value. We give a detailed discussion about the fault sets in the full version.

Security Experiment We are now ready to give a formal description of the underlying security experiment where an attacker is able to perform combined attacks. Therefore, we adjusted the security game of Dhooge and Nikova [DN20b] to allow adaptive faults and region probes. Let $\mathsf{C} \colon \mathbb{F}^k \to \mathbb{F}^l$ be a circuit with wires $W = \{w_i\}_i$ that is split into *regions* $R_1, \ldots, R_r$ with wires $W_1, \ldots, W_r$. An (d, e)-*attacker* A with respect to a fault class $\mathcal{F}$ takes part in the following experiment:

- The experiment chooses $b \leftarrow\!\!\$\, \{0, 1\}$ uniformly at random
- A is given input C and outputs the following:
  - a fault-attack $T \in \mathsf{A}(\mathcal{F})$ with $|T| \leq e$
  - for each region R_j, a subset $W'_j \subseteq W_j$ of wires with $|W'_j| \leq d$
  - two possible circuit inputs $x_0, x_1 \in \mathbb{F}^k$
- The experiment runs $\widetilde{y_b} \leftarrow T[\mathsf{C}](x_b)$ and the wire values corresponding to $W' = \bigcup_{j=1}^{r} W'_j$ are given to A. The attacker outputs a bit b'.

We say that C is ϵ-secure if $\Pr[b = b'] = 1/2$ and $\Pr[\widetilde{y_b} \in \{\perp, y_b\}] \geq 1 - \epsilon$ for any (d, e)-*attacker* A, where $y_b \leftarrow \mathsf{C}(x_b)$ is the output of a non-faulted run of C on x_b. In other words, the circuit is ϵ-secure if it is information-theoretic secure against leakage and detects erroneous values with probability at least $1 - \epsilon$.

In this work, we assume leakage-free encoding and decoding gadgets as defined in Definition 1. As a consequence, it is sufficient to prove that the probes can be simulated with less than d values of each masked input, if the circuit is masked with a degree d masking. The existence of such gadgets is commonly used and goes back to Ishai, Sahai, and Wagner [ISW03].

3.1 Privacy

First, we give a property that guarantees the (S)NI property of a gadget even in the presence of fault attacks. We emphasize that this property only gives probing security in the presence of faults, but ignores fault security notions such as error preservation and detection. For the general fault resilience we refer to Section 3.2. Next, we extend the probing security by requiring that a gadget is d-(S)NI even if the adversary inserts faults into the computation.

Definition 5 (fault resilient SNI). *A gadget* G *is* d fault resilient (strong-) non-interfering (d-fr(S)NI) *with respect to* $\mathcal{F}$ *if* $T[\mathsf{G}]$ *is* d*-(S)NI for any fault attack* $T \in \mathsf{A}(\mathcal{F})$.

Note that an (S)NI gadget is not always (S)NI in the presence of faults. In the full version, we give a detailed discussion and some examples. fr(S)NI is a relative strong property and in some cases it might be sufficient (or needed) to slightly weaken this notion. To do so, we will consider the situation introduced before where (a) the number of faults are bounded and, furthermore, (b) we will treat these faults additionally as probes. This is justified, e.g., in the context of constant fault functions (also called *stuck faults*) that might set a random value to a fixed value known by the adversary, as this can easily be seen to be strictly stronger than a probe on this random value. If a circuit is fault resilient under these restrictions, we say that the circuit is wfr(S)NI.

Definition 6 (weak fault resilient NI). *A gadget* G *is* d weak fault resilient non-interfering (d-wfrNI) *with respect to* $\mathcal{F}$ *if every set of* d' *probes in* $T[\mathsf{G}]$ *can be (perfectly) simulated with* $d' + |T|$ *shares of each input sharing for any fault attack* $T \in \mathsf{A}(\mathcal{F})$ *with* $|T| + d' \leq d$.

Definition 7 (weak fault resilient SNI). *A gadget* G *is* d weak fault resilient strong-non-interfering (d-wfrSNI) *with respect to* $\mathcal{F}$ *if every set of* d_1 *internal probes and* d_2 *output probes in* $T[\mathsf{G}]$ *can be (perfectly) simulated with* $d_1 + \epsilon_1$ *shares of each input sharing for any fault attack* $T \in \mathsf{A}(\mathcal{F})$ *with* ϵ_1 *internal faults and* ϵ_2 *output faults such that* $d_1 + d_2 + \epsilon_1 + \epsilon_2 \leq d$.

Note that this weaker notion does not imply that the faulted gadget $T[\mathsf{G}]$ is $(d - \epsilon) - (S)NI$ with $\epsilon = |T|$, as our d-wfr(S)NI definition gives ϵ more input values to the simulator for the simulation. This new security notions allows us to use the same composition results as the standard (S)NI gadgets even in the presence of faults. For example, the composition of two d-frSNI gadgets is easily seen to be d-frSNI again.

Theorem 1. *The composition of two d-frSNI (or d-wfrSNI) gadgets with respect to* $\mathcal{F}$ *is d-frSNI (or d-wfrSNI) with respect to* $\mathcal{F}$ *if* $\mathcal{F} \subseteq \mathcal{F}^{ind}$.

We write adaptive if the security still holds under the assumption that the adversary can choose the function with the knowledge of the probes (before). Theorem 1 implies that the composition of an arbitrary number of SNI gadgets is SNI again. This easily follows by the fact that composed SNI gadgets can be seen as SNI gadgets again, and we can compose step-by-step arbitrary many gadgets together. Theorem 1 is only an example composition for SNI. Next, we give a general proof that also implies this theorem[8] and shows that all d-(S)NI composition rules apply as well to the d-frSNI and d-wfrSNI.

Theorem 2. *The composition rules for (S)NI also apply to d-fr(S)NI and d-wfr(S)NI.*

Proof. We start with the proof for the stronger security notion. The faults $\mathcal{F} \subseteq \mathcal{F}^{ind}$ only allow independent faults on each wire, and this allows us to split the adversary in multiple adversaries that tamper each gadget independently. So let C be an arbitrary composed circuit only using d-fr(S)NI gadgets G_0, G_1, ..., G_m with respect to $\mathcal{F}$ and let T be any fault attack $T \in \mathsf{A}(\mathcal{F})$. Since we keep the proof as general as possible we just assume that C has some security properties (e.g. SNI) that follow from the (S)NI properties of G_0, G_1. Now we show that the property also holds for any T. Since we can split the circuit attack T to gadget-wise attacks T_0, T_1, ..., T_m it holds that $T[\mathsf{C}]$ can be also described as the composition of the (independently) faulted gadgets $T_0[\mathsf{G}_0]$, $T_1[\mathsf{G}_1]$, ..., $T_m[\mathsf{G}_m]$. The fr(S)NI properties still guarantees that each faulted gadget $T_j[\mathsf{G}_j]$ has the same (S)NI property as its unfaulted version G_j. Hence, the faulted gadgets $T_0[\mathsf{G}_0]$, $T_1[\mathsf{G}_1]$, ..., $T_m[\mathsf{G}_m]$ have the same composition properties as the unfaulted (S)NI gadgets G_0, G_1, ..., G_m and the faulted circuit $T[\mathsf{C}]$ has the same (S)NI properties as the original one C. Note that this holds for any fault attack $T \in \mathsf{A}(\mathcal{F})$, and it follows that the same composition rules apply for fr(S)NI as for (S)NI. The proof for the weaker notion is similar, only the fault attack is limited and the faults are counted as probes. □

Remember that d-(S)NI gadgets are also d-probing secure as defined in Definition 1. Similarly to Theorem 2 it easily follows that any d-frSNI circuit C with respect to $\mathcal{F}$ is also d-probing secure for any fault attack $T \in \mathsf{A}(\mathcal{F})$. More formally, $T[\mathsf{C}]$ is d-probing secure for any $T \in \mathsf{A}(\mathcal{F})$. The probing security of d-wfrSNI circuits is slightly weaker. Since the number of allowed probes in d-wfrSNI circuits is reduced by the number of faults, a d-wfrSNI circuit C with respect to $\mathcal{F}$ is only $(d - e)$-probing secure for any fault attack $T \in \mathsf{A}(\mathcal{F})$ with $|T| = e \leq d$.

[8] Alternatively, we give a straight forward proof of Theorem 1 in the full version

Privacy analyzes. Analysing the (S)NI property in itself is often tedious and to study the (w)f-(S)NI notion means we also need to consider all possible fault attacks. To avoid the combinatorial explosion, we present an additional property such that the classical (S)NI property directly implies the faulty one. This property is called *fault-invariance*, as the amount of information that a probe gives is independent of the faults. As the internal values z_i of a circuit only depend on the internal randomness R and the inputs $x_0, x_i, \dots, x_{l-1}$ we can also write them as functions $z_i = f_i^R(x_0, x_1, \dots x_{k-1})$.

Definition 8 (fault-invariance). *A circuit* C *is* fault-invariant *with respect to a fault set* $\mathcal{F}$ *if for any* $T \in \mathsf{A}(\mathcal{F})$*, any intermediate value* f *in* C^R *and the according value* $\tilde{f}$ *in* $T[\mathsf{C}^R]$ *there are* $\zeta, \zeta_0, \zeta_1, \dots \zeta_{k-1} \in \mathcal{F}$ *such that it holds*

$$\tilde{f}^R(x_0, x_1, \dots x_{m-1}) = \zeta(f^R(\zeta_0(x_0), \zeta_1(x_1), \dots \zeta_{m-1}(x_{m-1})))$$

for any input $(x_0, x_1, \dots x_{m-1})$ *and randomness* R*.*

In other words, Definition 8 says that all faults in $\mathcal{F}$ applied to a gadget can be pushed to the input or the output of the gadget.

Gadgets that are (S)NI and also have this property are directly (S)NI in the presence of faults.

Corollary 1. *If a gadget is d-(S)NI and fault-invariant with respect to* $\mathcal{F} \subseteq \mathcal{F}^{ind}$*, the gadget is d-fr(S)NI with respect to* $\mathcal{F}$*.*

Proof. We will prove that we can take the classical leakage simulator of the non-faulted gadget and transform it according to the faults due to the fault invariance. Fix a gadget G and some probed values $(p_0, p_1, \dots, p_{d-1})^T$. Due to the d-(S)NI property there is a simulator $S(a_0, a_1, \dots)$ that perfectly simulates the leakage with some input values $(a_0, a_1, \dots)$. This means it holds that

$$S(a_0, a_1, \dots) = (S_0(a_0, a_1, \dots), S_1(a_0, a_1, \dots), \dots S_{d-1}(a_0, a_1, \dots))^T$$

has the same distribution as $(p_0, p_1, \dots, p_{d-1})^T$, where $S_i(a_0, a_1, \dots)$ is the projection of the output of S to the wire indexed by probe p_i. Further, let any $T \in \mathsf{A}(\mathcal{F})$ be a fault-attack and $(p'_0, p'_1, \dots, p'_{d-1})^T$ be the according probes on the same wires in $T[\mathsf{G}]$. Due to the fault-invariance we know that there exist functions $\zeta_{i,j}, \zeta'_i \in \mathcal{F}$ such that the values

$$S'(a_0, a_1, \dots) = \begin{pmatrix} \zeta'_0(S_0(\zeta_{0,0}(a_0), \zeta_{0,1}(a_1), \dots)) \\ \zeta'_1(S_1(\zeta_{1,0}(a_0), \zeta_{1,1}(a_1), \dots)), \\ \vdots \\ \zeta'_{p-1}(S_{p-1}(\zeta_{p-1,0}(a_0), \zeta_{p-1,1}(a_1), \dots)) \end{pmatrix}$$

have the same distribution as $(p'_0, p'_1, \dots, p'_{d-1})^T$. Hence, the simulator S' can simulate the faulted gadget and with the same inputs as the simulator S of the unfaulted (S)NI gadget. This proves that $T[\mathsf{G}]$ is also (S)NI for any $T \in \mathsf{A}(\mathcal{F})$. □

In the following, we show that the stronger definition with regard to the additive faultset $\mathcal{F}^+$ directly implies the weaker definition for the more general independent $\mathcal{F}^{ind}$, if we consider fault-invariant gadgets.

Corollary 2. *If a gadget is d-fr(S)NI and fault-invariant with respect to* $\mathcal{F}^+$*, it is adaptively d-wfr(S)NI with respect to* $\mathcal{F}^{ind}$*.*

Proof. Let C be an d-fr(S)NI and fault-invariant circuit with respect to $\mathcal{F}^+$. We give a reduction and prove that any simulator Sim for d-wfr(S)NI with respect to $\mathcal{F}^{ind}$ can be simulated by a simulator $\widetilde{\mathsf{Sim}}$ for d-fr(S)NI with respect to $\mathcal{F}^+$ if the according gadget is fault-invariant with respect to $\mathcal{F}^+$.

So let Sim simulate a fault attack $T \in \mathsf{A}(\mathcal{F}^{ind})$ with $|T| = s$ and $d - s$ probes. Now we show that Sim can be simulated with a Simulator $\widetilde{\mathsf{Sim}}$ with fault attack $\widetilde{T} \in \mathsf{A}(\mathcal{F}^+)$ with $|\widetilde{T}| = s$ and d probes. Here, we use the fact that $\widetilde{\mathsf{Sim}}$ can simulate d values, and we can also simulate the faulted values to transform all faults into

additive faults. In detail, let v_1, v_2,... v_s the values faulted by the fault functions $\zeta_1, \zeta_2, \ldots, \zeta_s \in \mathcal{F}^{ind}$ due to T and $v_{s+1}, v_{s+2}, \ldots, v_d$ the $d-s$ values simulated by Sim in $T[\mathsf{C}]$. Now $\widetilde{\mathsf{Sim}}$ can simulate the according values $\tilde{v}_1, \tilde{v}_2, \ldots, \tilde{v}_d$ in the unfaulted C. Next, it computes for all fault functions ζ_j with $a_j = \zeta_j(v_j) - v_j$ and constructs the new additive fault function $\tilde{\zeta}_j(x) = x + a_j$. It follows that for all faults it holds $\zeta_j(v_j) = \tilde{\zeta}_j(v_j)$. Due to the invariance $\widetilde{\mathsf{Sim}}$ can move all additive faults to the inputs and outputs. In other words $\widetilde{\mathsf{Sim}}$ can compute how the additive faults affect the simulated probes $\tilde{v}_1, \tilde{v}_2, \ldots, \tilde{v}_d$, and can compute the according probes $v'_1, v'_2, \ldots, v'_d$ in the circuit faulted with the fault functions $\tilde{\zeta}_j(x)$. Since the faults are the same as $\zeta_j(x)$, the values $v'_{s+1}, v'_{s+2}, \ldots, v'_d$ and $v_{s+1}, v_{s+2}, \ldots, v_d$ have the same distribution. This implies the claim of the corollary because it shows that if we have $\widetilde{\mathsf{Sim}}$, we also have Sim. In other words if the gadget is d-fr(S)NI and fault-invariant with respect to $\mathcal{F}^+$, it is d-wfr(S)NI with respect to $\mathcal{F}^{ind}$.

Note that the simulator can choose the fault function with the knowledge of the probes. This also means that if the adversary chooses the function adaptively, it does not affect the simulator. Hence the adversary can be adaptive. □

3.2 Error Preservation and Detection

As described before, a general way to understand the countermeasures against passive and active attacks, can be viewed as *encodings*. For a concrete (randomized) encoding scheme $\mathsf{Enc}\colon \mathbb{F} \to \mathbb{F}^n$, a value $y \in \mathbb{F}^n$ is a *valid* encoding if there is an $x \in \mathbb{F}$ and some randomness such that $\mathsf{Enc}(x; r) = y$. Similar to the fr(S)NI property we are interested in a property that guarantees that errors can be detected even when we compose multiple gadgets. In our case, we want to guarantee that errors are detected by the fact that the resulting encodings are invalid. More concretely, if y and y' are valid encodings, we want to increase their Hamming distance, denoted by $\mathrm{d}(y, y')$. To argue about the behavior of a gadget in the presence of faults that can introduce computation errors, we need to guarantee that errors already present in the computation (a) stay present in the computation (to detect them) and (b) that these errors cannot accumulate over time to lead to a valid encoding of an *incorrect* value. To model this, we assume that the inputs $(x_i)_{i\in[n]}$ of our gadgets might already be faulted by a *fault vector* $(v_i)_{i\in[n]}$, i.e., the inputs will always be $(x_i)_{i\in[n]} + (v_i)_{i\in[n]}$, where $(x_i)_{i\in[n]}$ is a valid encoding. In a first approach, one might require that the input of an invalid encoding, i.e., one where $(v_i)_{i\in[n]} \neq 0$, always leads to an invalid output encoding. But this is a very strict requirement that is nearly impossible to fulfill if we consider an addition gadget: The attacker might add $(v_i)_{i\in[n]}$ to one of the inputs and then later add $-(v_i)_{i\in[n]}$ to the output. Clearly, this gives a valid encoding although the input was invalid. We thus also allow that our gadget on input $(x_i)_{i\in[n]} + (v_i)_{i\in[n]}$ with $(v_i)_{i\in[n]} \neq 0$ can produce a valid encoding of the *correct value* but this effect has to be independent of input $(x_i)_{i\in[n]}$.

Definition 9 (e-f-robust). *A gadget G with one output sharing and two input sharings is* e-fault-robust *with respect to $\mathcal{F}$, if for any valid encoding $(x_i^{(0)})_{i\in[n]}$ and $(x_i^{(1)})_{i\in[n]}$, the output $(y_i)_{i\in[n]} \leftarrow \mathsf{G}((x_i^{(0)})_{i\in[n]}, (x_i^{(1)})_{i\in[n]})$ is also valid. Further, it holds for any fault vectors $(v_i^{(0)})_{i\in[n]}$, $(v_i^{(1)})_{i\in[n]}$, and $T \in \mathsf{A}(\mathcal{F})$ with $(y_i)_{i\in[n]} + (w_i^{(1)})_{i\in[n]} + (w_i^{(2)})_{i\in[n]} \leftarrow T[\mathsf{G}]((x^{(0)} + v_i^{(0)})_{i\in[n]}, (x^{(1)} + v_i^{(1)})_{i\in[n]})$, that there are numbers t_1 and t_2 with $t_1 + t_2 \leq |T|$ such that*

(*i*) $\mathrm{weight}(w') \leq t_1$ *with* $(w'_i)_{i\in[n]} = (v_i^{(0)})_{i\in[n]} + (v_i^{(1)})_{i\in[n]} - (w_i^{(1)})_{i\in[n]}$;

(*ii*) *and $(w_i^{(2)})_{i\in[n]}$ is the zero vector or produced by the following random experiment: A polynomial $p_w \in \mathbb{F}[x]$ is chosen such that the coefficients x^{d+1}, x^{d+2}, ..., x^{n-t_2} are drawn uniformly at random from $\mathbb{F}$. Then, $w_i^{(2)} = p_w(\alpha_i)$.*

Note that any valid encoding $(y_i)_{i\in[n]}$ and any fault vector $(w_i)_{i\in[n]}$ with $\mathrm{weight}(w) \leq e$ cannot result in a valid encoding with $(y_i)_{i\in[n]} + (w_i)_{i\in[n]}$ in our setting, as we use $d + 1 + e$ shares of a polynomial of degree d. Hence, $\mathrm{weight}((y_i)_{i\in[n]} - (y'_i)_{i\in[n]}) \geq e + 1$ for all valid sharings y and y'. Next we give a useful composition result for e-f-robustness.

Theorem 3. *The composition of two e-fault-robust gadgets with respect to $\mathcal{F}$ is e-fault-robust with respect to $\mathcal{F}$ if $\mathcal{F} \subseteq \mathcal{F}^{ind}$.*

Proof. Let G and G' be two gadgets such that G is given inputs $(x_i^{(0)})_{i\in[n]}$ and $(x_i^{(1)})_{i\in[n]}$ and produces output $(y_i^{(0)})_{i\in[n]}$. Furthermore, let G' have the inputs $(y_i^{(0)})_{i\in[n]}$ (i.e., the output of G) and $(y_i^{(1)})_{i\in[n]}$ and output $(z_i)_{i\in[n]}$. Let $\tilde{\mathsf{G}}$ be the complete construction and $v_{x^{(0)}}$, $v_{x^{(1)}}$, $v_{y^{(0)}}$, $v_{y^{(1)}}$, v_z be the fault vectors.

Fix any $\tilde{T} \in \mathsf{A}(\mathcal{F})$. Due to the independence of these faults, we can split them into two parts T and T', where T corresponds to the faults introduced in G and T' corresponds to the faults introduced in G'.

As G is e-fault-robust, its output fault vector $v_{y^{(0)}}$ is of the form $v_{y^{(0)}} = v_{x^{(0)}} + v_{x^{(1)}} + w^{(1)} + w^{(2)}$ where $\text{weight}(w^{(1)}) \leq t_1$ and $w^{(2)}$ is the zero vector or drawn randomly with highest coefficient x^{n-t_2} for some numbers t_1 and t_2 with $t_1+t_2 \leq |T|$. Furthermore, as G' is e-fault-robust, its output fault vector v_z is of the form $v_z = v_{y^{(0)}} + v_{y^{(1)}} + w'^{(1)} + w'^{(2)}$ where $\text{weight}(w'^{(1)}) \leq t'_1$ and $w'^{(2)}$ is drawn randomly with highest coefficient $x^{n-t'_2}$ for some numbers t'_1 and t'_2 with $t'_1+t'_2 \leq |T'|$. Hence, $v_z = v_{x^{(0)}}+v_{x^{(1)}}+w^{(1)}+w^{(2)}+v_{y^{(1)}}+w'^{(1)}+w'^{(2)}$, where $\text{weight}(w^{(1)} + w'^{(1)}) \leq t_1 + t'_1$ and $w^{(2)} + w'^{(2)}$ corresponds to a random polynomial with highest coefficient $x^{n-\min\{t_2,t'_2\}}$. □

The definition of e-fault-robustness can be simplified for non-adaptive attackers: We can then guarantee that either $w^{(1)}$ or $w^{(2)}$ are zero. The proof of composability is similar, but uses the fact that the sum of a random polynomial and a deterministic, *independent* polynomial is again a random polynomial.

The notion of e-fault-robustness now directly allows us to give a bound on the probability that a valid encoding of an invalid value is generated by a circuit.

Theorem 4. *If a circuit is e-fault-robust, the probability that $s \leq e$ faults can produce a valid encoding of an invalid value is at most q^{s-e-1} in the case of non-adaptive attackers and $q^{s-e} \cdot (d+e+1)^{t_1}$ for all $t_1 \leq s$ in the case of adaptive attackers.*

Proof. Let $w^{(1)}$ and $w^{(2)}$ be as in the definition of e-fault-robustness and let $p^{(1)}$ and $p^{(2)}$ be the corresponding polynomials. Lemma 2 implies that the fault polynomial $p^{(1)}$ has degree at least $n - t_1$. Hence, if $p^{(2)}$ is identical to zero, the attacker can not produce a valid encoding of an invalid value. Furthermore, if $p^{(1)}$ is identical to zero, the sharing is valid if and only if all coefficients of x^{d+1}, x^{d+2}, …, x^{n-s} of $p^{(1)}$ are zero, which happens with probability q^{s-e-1}.

If $p^{(2)}$ is not identical to zero, the polynomial $p = p^{(1)} + p^{(2)}$ corresponding to $w = w^{(1)} + w^{(2)}$ needs to have degree at most d. As $p^{(2)}$ has degree at most $n - t_2$, this is only possible if $n - t_1 \leq n - t_2$, i.e., if $t_1 \geq t_2$ holds. We will now show that the number of different polynomials $p^{(1)}$ and $p^{(2)}$ such that p has degree at most d is very small. Clearly, the number of different polynomials possible for $p^{(2)}$ is q^{e-t_2} due to the fact that $p^{(2)}$ is randomly generated. Furthermore, the number of different polynomials possible for $p^{(1)}$ is $\binom{n}{t_1} \cdot q^{t_1} \leq n^{t_1} \cdot q^{t_1}$. Hence, the probability that there is a vector $w^{(1)}$ matching to the random vector $w^{(2)}$ is at most

$$\frac{q^{t_1}}{q^{e-t_2}} \cdot n^{t_1} = q^{t_1+t_2-e} \cdot n^{t_1}.$$

Hence, the probability that $s \leq e$ faults can produce a valid encoding of an invalid value is at most $q^{s-e} \cdot (d+e+1)^{t_1}$. □

3.3 Combined Security

Equipped with our new notions of d-fr(S)NI and e-fault-robustness, it is easy to see that the combination of these notions directly implies security against (d, e)-attackers.

Theorem 5. *If the circuit C is d-frSNI (or d-wfr(S)NI) and e-fault-robust, it is q^{s-e-1}-secure against any non-adaptive (d, s)-attacker (or $(d-s, s)$-attacker) with $s \leq e$. Furthermore, it is $q^{s-e} \cdot (d+e+1)^{t_1}$-secure against any adaptive (d, s)-attacker (or $(d-s, s)$-attacker) with $t_1 \leq s \leq e$.*

Proof. The perfect security with regard to the leakage (i.e., that the attacker is not able to determine the challenge bit b) directly follows from the fact that the circuit is d-SNI even in the presence of at most e faults. Furthermore, Theorem 4 directly implies that the probability that a valid encoding of an invalid value is ever output is at most q^{s-e-1} in the non-adaptive case and $q^{s-e} \cdot (d+e+1)^{t_1}$ in the adaptive case. □

4 Compiler

This section gives a compiler transforming an unprotected circuit into a fault and probe resilient circuit using a polynomial sharing with an optimal number of shares. As described in the previous section, the sensitive data v is masked with a $d+1$-out-of-n polynomial sharing. Therefore, a polynomial f with degree d such that $f(0) = v$ is generated, and the shares are given by $f(\alpha_i)$ for pairwise different non-zero inputs $\alpha_0, \ldots, \alpha_{n-1}$. The main goal of the compiler is now to take an ordinary circuit and transform it into a circuit operating on shares such that d probes and e faults do not reveal any sensitive data, i.e., into a circuit that is ϵ-secure against (d, e)-attackers. As usual, this is done by replacing gates of the original circuit with gadgets. For security reasons, refresh gadgets are also inserted that guarantee that intermediate sharings become independent. Refresh gadgets take as input an encoded secret and output a re-randomized encoding of the same secret. This procedure reduces the dependencies that might occur when one value is used as input for multiple gadgets.

In the following, let $n = d + e + 1$. Unary gates (such as addition or multiplication with a constant) taking only a single input can be handled easily, as these operations are linear and can thus be computed locally on the shares. We thus only need to focus on binary gates, i.e., addition and multiplication gates that take two inputs. Let us consider two $d+1$-out-of-n sharings $(F_i)_{i\in[n]}$ and $(G_i)_{i\in[n]}$ of secrets f_0, respectively g_0. Similar to the unary gates, Algorithm 1 computes a *share-wise* addition such that its output $(Q_i)_{i\in[n]}$ represents the addition $f_0 + g_0$. This algorithm is a gadget computing an addition since it outputs a $d+1$-out-of-n sharing again, as the addition of two degree d polynomials also has degree d.

Algorithm 1 (n, d)-**SWAdd** for $n = d + \epsilon + 1$

Input: Shares of f_0 as $(F_i)_{i\in[n]}$ and g_0 as $(G_i)_{i\in[n]}$ with degree d
Output: Shares of $f_0 + g_0$ as $(Q_i)_{i\in[n]}$ with degree d.

1: **for** $i = 0$ **to** $n - 1$ **do**
2: $\quad Q_i \leftarrow F_i + G_i$
3: **return** $(Q_i)_{i\in[n]}$

Due to our attack model, all faults added to the input of any such share-wise (linear) gadget can also be added to the output without any change in the computation. Furthermore, as $n \geq d$, these gadgets are secure against d probes, as d values of a $d+1$-out-of-n sharing do not reveal any information. We can thus state the following fact.

Theorem 6. *Share-wise affine gadgets are d-frNI with respect to $\mathcal{F}^+$ (or d-wfrNI with respect to $\mathcal{F}^{ind}$) and e-fault-robust with respect to $\mathcal{F}^{ind}$.*

Proof. The e-fault-robustness immediately follows from Definition 9(i). Next, we prove the frNI property. It is sufficient to show that a share-wise linear gadget is NI and fault invariant with respect to $\mathcal{F}^+$ because Corollary 1 shows that this implies d-frNI with respect to $\mathcal{F}^+$. Further, Corollary 2 shows that this also implies d-wfrNI with respect to $\mathcal{F}^{ind}$. Hence, it remains to prove that a share-wise linear gadget is (i) NI and (ii) fault invariant with respect to $\mathcal{F}^+$.

(i) It is well known that any share-wise gadget is NI, as all output (and intermediate) values only depend on input shares with the same index. For example, the share-wise addition with $c_i \leftarrow a_i + b_i$ only depends on the i^{th} share a_i and b_i for any $i \in [n]$. It follows that any d probes can be simulated by at most d shares of each input sharing because we never need more than one share of each input sharing for each probe. This implies the NI property.

(ii) The shift invariance w.r.t $\mathcal{F}^+$ follows from the linearity. Since an additive fault ζ only adds a constant value x on a wire, we can move the addition x to the output or input due to the linearity. For example it

holds for the share-wise addition $c_i \leftarrow a_i + b_i$ that $(a_i + b_i) + x = a_i + (b_i + x) = (a_i + x) + b_i$. In the first term, the fault is moved to the output $\zeta(c_i) = c_i + x$ and in the second two terms it is moved to the inputs $\zeta(b_i)$, and $\zeta(a_i)$, respectively. This is the property required for fault invariance. We give a more general proof in the full version. With (i) and (ii) we can conclude the proof. □

As usual, the remaining gate, the binary multiplication gate is the most complicated gate, as it does not behave linearly. Nevertheless, Algorithm 2 computes a share-wise multiplication such that its output $(Q_i)_{i\in[n]}$ represents the multiplication $f_0 \cdot g_0$. Unfortunately, the multiplication of two polynomials of degree d results in a polynomial of degree $2d$. Therefore, Algorithm 2 outputs a $2d$ degree polynomial. For $n < 2d$, this means that the shares can not represent this polynomial properly and we thus lose the information about the shared value. Furthermore, even if $n > 2d$, the next multiplication gadget in the circuit could possibly lead to a polynomial of degree $4d$ and so on. Hence, Algorithm 2 can not be used as a multiplication gadget alone and further work is required. The classical approach due to Ben-Or, Goldwasser, and Wigderson [BGW88] performs a *degree reduction* to reduce the polynomial to degree d after the share-wise multiplication to prevent the exponential blowup of the degrees. Nevertheless, the polynomial of degree $2d$ needs to be stored and their approach thus requires at least $2d$ shares.

Algorithm 2 (n, d)-**SWMult** for $n = d + \epsilon + 1$

Input: Shares of f_0 as $(F_i)_{i\in[n]}$ and g_0 as $(G_i)_{i\in[n]}$ with degree d
Output: Shares of $f_0 g_0$ as $(Q'_i)_{i\in[n]}$ with degree $2d$.

1: **for** $i = 0$ **to** $n - 1$ **do**
2: $\quad Q'_i \leftarrow F_i \cdot G_i$
3: **return** $(Q'_i)_{i\in[n]}$

To construct a multiplication gadget, the state-of-the-art gadget due to Seker, Fernandez-Rubio, Eisenbarth, and Steinwandt [SFRES18] also follows the general approach of Ben-Or, Goldwasser, and Wigderson: They first apply the share-wise multiplication (Alg. 2) to compute sharing of degree $2d$ and then reduce the degree back to d afterwards such that subsequent operations can be performed. However, the degree reduction is relatively expensive and complex. We will discuss this strategy in more depth in the following section, Section 4.1. But due to the need to store a polynomial of degree $2d$, their approach can not need less than $2d$ shares. Furthermore, they also need an additional e shares to handle faults. Our solution circumvents the need for this large number of shares. Section 4.2 presents a new compiler that needs at most $d + e + 1$ shares.

4.1 Fixed State-of-the-Art

Here, we give a more thorough explanation of the binary multiplication gadget construction due to Seker, Fernandez-Rubio, Eisenbarth, and Steinwandt [SFRES18]. We note here that this gadget can only handle *additive* faults.

Figure 1b illustrates the high-level idea of the gadget. It first performs the share-wise multiplication which outputs sharing $(Q'_i)_{i\in[n]}$ of degree $2d$ that encodes the product of the secrets of $(F_i)_{i\in[n]}$ and $(G_i)_{i\in[n]}$ and then reduces the degree of $(Q'_i)_{i\in[n]}$ such that $(Q_i)_{i\in[n]}$ carries the same secret as $(Q'_i)_{i\in[n]}$ but has degree d. This construction was proven to be d-SNI secure. As mentioned above, this multiplication leads to the intermediate $2d$ degree sharing $(Q'_i)_{i\in[n]}$ that requires $n = 2d + e + 1$ shares. To handle faults, the following idea is used: Any attacker that can add e additive faults to the shares adds a polynomial of degree at least $n - e = d + 1$ to the sharing. As a valid sharing has degree at most d, this means that the higher-order monomial $d + 1$ is set to a non-zero value iff a fault was added. These higher-order monomials are kept unchanged by share-wise additions and by careful design, are also kept unchanged with probability at least $1 - (1/q)$ in the multiplication gadget (or, more generally $1 - q^{-e+s-1}$ for $s \leq e$ faults). Unfortunately, their

refresh gadget is not (S)NI, as shown in the full version, and an alternative refresh is required. We remark that such a refresh gadget can be seen as multiplication with a fresh sharing of the value 1. It is thus sufficient to focus on addition and multiplication gadgets here. We refer to [BBD+16] for more detailed explanations. The result by Seker, Fernandez-Rubio, Eisenbarth, and Steinwandt [SFRES18] can thus be summed up via the following informal theorem:

Theorem 7 (Fixed SotA). *For any $d, e \in \mathbb{N}$ there is a compiler that is given an arithmetic circuit $\mathcal{C}$ over $\mathbb{F}$ and outputs an arithmetic circuit $\mathcal{C}'$ on $\mathbb{F}^n$ where $n = 2d + e + 1$ with*

- *$\mathcal{C}'$ has the same functionality as $\mathcal{C}$*
- *$\mathcal{C}'$ is d-probing-secure,*
- *C' is e-fault-robust with respect to $\mathcal{F}^+$.*

The proof follows from [SFRES18] with the multiplication use as SNI refresh, and the compiler presented in [BBD+16]. The idea is that the multiplication and refresh gadgets are error preserving and d-SNI and the addition gadget is error preserving and d-NI. Applying the compiler of [BBD+16] results in a d-probing secure circuit with error preserving gadgets and, consequently, the compiler outputs an error preserving circuit that is information-theoretic secure against d probes.

4.2 laOla Compiler

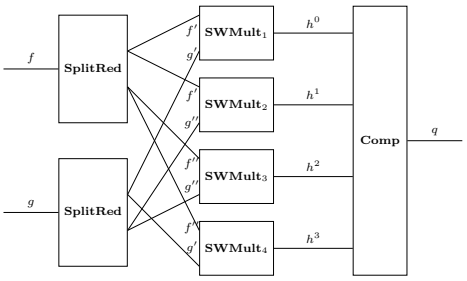
We improve our compiler in two steps. We first improve the (fr)SNI refresh, and we give a new multiplication gadget to reduce the number of shares from $2d + e + 1$ to $d + e + 1$.

Refresh. We construct a new SNI refresh gadget only using d^2 random values. Therefore, we transform this problem to a gadget that generates a zero encoding. Assuming a "secure" zero encoding $(e_i)_{i\in[n]} \leftarrow \mathsf{Enc}(0)$ it is easy to see that the refreshed output $(y_i)_{i\in[n]}$ of a sharing $(x_i)_{i\in[n]}$ with $y_i = x_i + e_i$ is SNI. Any probe y_i is uniform random, and only the (additional) internal probe e_i requires the input x_i for a perfect simulation. However, we still ignored the internal probes in $(e_i)_{i\in[n]} \leftarrow \mathsf{Enc}(0)$ to generate the zero encoding. **sZEnc** depicted in Algorithm 4 is such a gadget that is SNI secure even in the presence of internal probes and faults (Theorem 11).

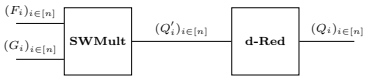
Further, we can show that the resulting refresh gadget even gives region probing security (Theorem 12). Using our new refresh gadget we can improve the compiler of [SFRES18]. The improved compiler (Imp. SotA, listed in Table 3) requires less randomness and guarantees higher security due to region probing security.

Multiplication. In order to avoid the dependency on $2d$, a trivial (insecure) approach is to *switch* the order of operations: I.e., we first reduce the degree of the input shares $(G_i)_{i\in[n]}$, $(F_i)_{i\in[n]}$ to sharings $(G'_i)_{i\in[n]}$, $(F'_i)_{i\in[n]}$ of degree $d/2$. The output of the share-wise multiplication $\mathbf{SWMult}((G'_i)_{i\in[n]}, (F'_i)_{i\in[n]})$ then has the right degree d again. However, it is easy to see that $(G'_i)_{i\in[n]}$, $(F'_i)_{i\in[n]}$ would reveal their secrets with $d/2 + 1 \leq d$ probes. Hence, this approach does not guarantee privacy against d probes.

While a naive implementation of this idea is insecure, our new construction depicted in Figure 1a still guarantees security. Instead of performing a simple degree reduction, our gadget **SplitRed** *simultaneously* constructs an *additive* sharing of the polynomials. Hence, an application of **SplitRed** on an input sharing described by the polynomial f produces *two* polynomials f' and f'' both of degree d such that the polynomial $f' + f''$ has only degree $d/2$. In other words, we produce two polynomials f' and f'' where the coefficients of the monomials $x^0, x^1, \ldots x^{d/2}$ are additive sharings of the corresponding coefficients of f and the coefficients of the monomials $x^{d/2+1}, \ldots, x^d$ are additive sharings of the all-zero vector. We compute g' and g'' similarly such that each polynomial f', f'', g', and g'' considered individually is still a polynomial of degree d. Now we can apply the share-wise multiplication four times to compute $f' \cdot g'$, $f' \cdot g''$, $f'' \cdot g'$, and $f'' \cdot g''$ and sum all four outputs with our gadget **Comp**. It follows that the output $(H_i)_{i\in[n]}$ describes a polynomial h of degree d again because $f' \cdot g' + f' \cdot g'' + f'' \cdot g' + f'' \cdot g'' = (f' + f'') \cdot (g' + g'')$ is again a polynomial of degree d. The formal description of the algorithm is given in Section 6 and the full version provides a detailed security analysis against probes and faults.



(a) Structure of the n, d-Multiplication defined in Algorithm 6



(b) Structure of the n, d-Multiplication used in [SFRES18].

Fig. 1: Our n, d-Multiplication and the multiplication in [SFRES18]

Compiler The multiplication, and refresh, together with the share-wise addition, lead to a compiler using only $n = d + e + 1$ shares against additive faults.

Theorem 8 (laOla (additive)). *For any $d, e \in \mathbb{N}$ there is a circuit compiler that is given an arithmetic circuit $\mathcal{C}$ over $\mathbb{F}$ and outputs an arithmetic circuit $\mathcal{C}'$ over $\mathbb{F}^n$ where $n = d + e + 1$ with*

- *$\mathcal{C}'$ has the same functionality as $\mathcal{C}$,*
- *$T[\mathcal{C}']$ is probing secure for any $T \in \mathsf{A}(\mathcal{F}^+)$ and*
  (i) up to d probes in $T[\mathcal{C}']$ (threshold probing security), or
  (ii) up to $d/2$ probes in every gadget of $T[\mathcal{C}']$ (region probing security).
- *C' is e-fault-robust with respect to $\mathcal{F}^+$.*

Furthermore, we also show that our approach can handle more general faults, although this comes at the cost of needing more shares.

Theorem 9 (laOla-Compiler (general)). *For any $d, e \in \mathbb{N}$ there is a circuit compiler that is given an arithmetic circuit $\mathcal{C}$ over $\mathbb{F}$ and outputs an arithmetic circuit $\mathcal{C}'$ over $\mathbb{F}^n$ where $n = d + e + 1$ with*

- *$\mathcal{C}'$ has the same functionality as $\mathcal{C}$,*
- *$T[\mathcal{C}']$ is probing secure for any $T \in \mathsf{A}(\mathcal{F}^+)$ with $|T| \leq e$ and*
  (i) up to $d - e$ probes in $T[\mathcal{C}']$ (threshold probing security), or
  (ii) with up to $d/2$ probes in every gadget of $T[\mathcal{C}']$ when the faults are counted as probes (region probing security).
- *$T[\mathcal{C}']$ is $d - e$ probing secure for any $T \in \mathsf{A}(\mathcal{F}^{ind})$ with $|T| \leq e$,*
- *C' is e-fault-robust with respect to $\mathcal{F}^{ind}$.*

In both cases, Theorem 5 implies q^{s-e-1}-security against (d, s)-attackers.

In the full version, we also prove that our compiler is optimal for affine circuits and it is inherently impossible to use a lower number of shares.

Theorem 10. *The number of shares n of any sharing procedure that protects against d probes and e faults is at least $n \geq d + e + 1$.*

5 Refresh Gadget

To construct a refresh gadget with input sharing $(x_i)_{i\in[n]}$ it is sufficient to generate a zero encoding $(e_i)_{i\in[n]}$ and output its sum $(y_i)_{i\in[n]} = (x_i)_{i\in[n]} + (e_i)_{i\in[n]}$. In this section we give two different gadgets to generate zero encodings. The first one is sufficient to inject randomness in our multiplication gadget. The latter gadget uses the weaker one and results in a d-SNI refresh gadget. Further we show that the gadget is even stronger, and we can use it to transform our d-probing secure circuit into a $d/2$-region-probing secure circuit.

Table 3: A concrete comparison with the compiler [SFRES18] fixed in this work, the constuction in [DN19], and our new compiler.

Compiler	# Shares	Randomness Cost		Comb. Sec. in the Reg. Prob. Model	Opt. for affine Circuits
		Mult. Gadget	SNI Refresh		
[DN19]	$d+e+1$	$\Theta(n^3)$	$\Theta(n^3)$	✖	✔
Fixed SotA [SFRES18]	$2d+e+1$	$2d^2+d(e+1)$	$2d^2+d(e+1)$	✖	✖
Imp. SotA[This Work]	$2d+e+1$	$2d^2+d(e+1)$	d^2	(✔)	✖
laOla [This Work]	$d+e+1$	$3d^2+2d(e+1)$	d^2	✔	✔

***ZEnc** Gadget.* The gadget **ZEnc** depicted in Algorithm 3 generates a random zero encoding. We use the polynomial representation to describe the high level idea of the gadget. Since g is a random polynomial with $g(0)=0$, it holds $g(x)=\sum_{i=1}^{d} r_j x^j$ with $r_1, r_2, \dots r_d \in \mathbb{F}$. Our gadget generates such polynomials by choosing each r_i uniform at random and outputs g. More precisely, we use the polynomial masking where each polynomial is described by n points $g(\alpha_0), g(\alpha_1), \dots g(\alpha_{n-1})$. Therefore, the algorithm does not compute $g(x)$ but each $g(\alpha_i)=\sum_{i=1}^{d} r_j \alpha_i$, separately. Hence, the final output of Algorithm 3 represent an encoding of zero with $(g_i)_{i\in[n]} := (g(\alpha_i))_{i\in[n]}$.

Algorithm 3 $\mathbf{ZEnc}_n^d$

Output: A randomized (n,d)-Encoding of zero $(g_i)_{i\in[n]}$.

1: **for** $j=1$ **to** d **do**
2: $\quad r_j \leftarrow_{\$} \mathbb{F}$
3: $\quad$ **for** $i=0$ **to** $n-1$ **do**
4: $\quad\quad g_{i+1} \leftarrow g_i \oplus r_j \alpha_i^j$
5: **return** $(g_i)_{i\in[n]}$

In the full version, we show that this encoding does not suffice for an SNI refresh. However, next we show how to use **ZEnc** to construct an SNI secure refresh.

***sZEnc** Gadget.* The gadget **sZEnc** depicted in Algorithm 4 generates a zero encoding because the sum of zero encodings is a zero encoding again.

Algorithm 4 $\mathbf{sZEnc}_n^d$

Output: A randomized (n,d)-Encoding of zero $(g_i)_{i\in[n]}$.

1: **for** $j=0$ **to** d **do**
2: $\quad (g_i)_{i\in[n]} \leftarrow \mathbf{ZEnc}_n^d$
3: $\quad (y_i)_{i\in[n]} \leftarrow (y_i)_{i\in[n]} + (g_i)_{i\in[n]}$
4: **return** $(y_i)_{i\in[n]}$

Lemma 3 (Probing security). *For any set P with $d' \le d$ probes it holds for $(e_i)_{i\in[n]} \leftarrow \boldsymbol{sZEnc}_n^d$: There is a sub set $A \subset [n]$ with $|A| = n-d'$ such that*

(i) $(e_i)_{i\in A}$ are still $(d-d')$-wise independent, independent from P and $(e_i)_{i\in[n]\setminus A}$,
(ii) P can be perfectly simulated with $(e_i)_{i\in[n]\setminus A}$

The proof is given in the full version. It is easy to see that a gadget with input sharing $(x_i)_{i\in[n]}$ and output sharing $(x_i + e_i)_{i\in[n]}$ with $(e_i)_{i\in[n]} \leftarrow \mathbf{sZEnc}_n^d$ is an SNI refresh:

Theorem 11 (Refresh). *The gadget G'_G (Alg. 5) with identity G is a d-frSNI w.r.t. $\mathcal{F}^+$ (or d-wfrSNI w.r.t. $\mathcal{F}^{ind}$) and e-fault-robust w.r.t. $\mathcal{F}^{ind}$ refresh gadget*

Proof. In the full version we give a detailed proof for fault-robustness, SNI and fault-invariance w.r.t. $\mathcal{F}^+$. With Theorem 1 we get frSNI, and with Theorem 2 follows wfrSNI. □

Algorithm 5 G'_G with $n \geq d + e + 1$

Input: The same input sharings as G. E.g., $(x_i)_{i\in[n]}$ and $(x'_i)_{i\in[n]}$, or only $(x_i)_{i\in[n]}$.
Output: A randomized output of $\mathsf{G}((x_i)_{i\in[n]}, (x'_i)_{i\in[n]})$ (or $\mathsf{G}((x_i)_{i\in[n]})$).

1: $(e_i)_{i\in[n]} \leftarrow \mathbf{sZEnc}_n^d$
2: $(y'_i)_{i\in[n]} \leftarrow \mathsf{G}((x_i)_{i\in[n]}, (x'_i)_{i\in[n]})$ (or $(y'_i)_{i\in[n]} \leftarrow \mathsf{G}((x_i)_{i\in[n]})$)
3: $(y_i)_{i\in[n]} \leftarrow (y_i)_{i\in[n]} + (e_i)_{i\in[n]}$
4: **return** $(y_i)_{i\in[n]}$

However, this refresh gadget is even more secure. Next, we show how to construct a region-probing secure compiler with the gadget depicted in Algorithm 5.

Theorem 12. *A d probing secure composition with d-NI and d-SNI secure gadgets G_i is $d/2$ region probing secure if each gadget is transformed into G'_G (Alg. 5), and outputs refreshed sharings.*

It immediately follows from Lemma 3, and the detailed proof is given in the full version. Assuming a (w)frSNI and error preserving multiplication, this theorem directly implies both Theorem 8 and Theorem 9. Using Theorem 5 then implies q^{s-e-1}-security against (d, s)-attackers. Next, we give our frSNI and error preserving multiplication gadget only using $n = d + e + 1$ shares.

6 Multiplication Gadget

In this section we introduce our new improved gadget which securely performs a masked multiplication on a polynomial sharing with just $n = d + e + 1$ shares, whereas the state-of-the-art requires $2d + e + 1$ shares.

6.1 Concept and Overview

In the following we formally introduce the new multiplication gadget depicted in Figure 1a, its formal description is given in Algorithm 6.

For a better intuition of the gadget **Mult** with inputs $(F_i)_{i\in[n]}$ and $(G_i)_{i\in[n]}$, we use the polynomial representation f and g such that $f(\alpha_i) = F_i$ and $g(\alpha_i) = G_i$. As depicted in Figure 1a, the gadget **SplitRed** first splits the inputs g and f with secrets $f(0) = f_0$ and $g(0) = g_0$ into polynomials f', f'', g', and g'' such that each polynomial is uniform random with degree d but $f' + f'$ and $g' + g''$ have degree $\frac{d}{2}$ each and, furthermore, $f'(0) + f'(0) = f_0$ and $g'(0) + g'(0) = g_0$. This allows to avoid the intermediate polynomials that require $2d$ shares. Furthermore, we can use $\mathbf{SWMult}_i$ to compute the four polynomials $h^0(x) = f'(x)g'(x)$, $h^1(x) = f'(x)g''(x)$, $h^2(x) = f''(x)g'(x)$, and $h^3(x) = f''(x)g''(x)$. The last gadget **Comp** refreshes the polynomials and sums them up into

$$f'(x)g'(x) + f'(x)g''(x) + f''(x)g'(x) + f''(x)g''(x).$$

The sum results in a polynomial $q(x)$ with $(Q_i)_{i\in[n]} = q(\alpha_i)$ which encodes the correct value $q(0) = f_0 \cdot g_0$, as

$$\begin{aligned} q(0) &= f'(0) \cdot g'(0) + f'(0) \cdot g''(0) + f''(0) \cdot g'(0) + f''(0) \cdot g''(0) \\ &= (f'(0) + f''(0)) \cdot (g'(0) + g''(0)) = g(0) \cdot f(0). \end{aligned}$$

The next sections introduce the sub-gadgets **SplitRed** (Sec. 6.2), **SWMult**, and **Comp** (Sec.6.3) needed for our multiplication gadget.

Algorithm 6 $(n, d) - \mathbf{Mult}$

Input: Shares of f_0 as $(F_i)_{0\le i<n}$ and shares of g_0 as $(G_i)_{0\le i<n}$.
Output: Shares of $f_0 g_0$ as $(\mathbf{Q}_i)_{0\le i<n}$.

1: $\left((F'_i)_{i\in[n]}, (F''_i)_{i\in[n]}\right) \leftarrow \mathbf{SplitRed}((F_i)_{i\in[n]})$
2: $\left((G'_i)_{i\in[n]}, (G''_i)_{i\in[n]}\right) \leftarrow \mathbf{SplitRed}((G_i)_{i\in[n]})$
3: $(H_i^0)_{i\in[n]} \leftarrow \mathbf{SWMult}((F'_i)_{i\in[n]}, (G'_i)_{i\in[n]})$
4: $(H_i^1)_{i\in[n]} \leftarrow \mathbf{SWMult}((F'_i)_{i\in[n]}, (G''_i)_{i\in[n]})$
5: $(H_i^2)_{i\in[n]} \leftarrow \mathbf{SWMult}((F''_i)_{i\in[n]}, (G'_i)_{i\in[n]})$
6: $(H_i^3)_{i\in[n]} \leftarrow \mathbf{SWMult}((F''_i)_{i\in[n]}, (G''_i)_{i\in[n]})$
7: $(Q_i)_{i\in[n]} \leftarrow \mathbf{Comp}((H_i^0)_{i\in[n]}, (H_i^1)_{i\in[n]}, (H_i^2)_{i\in[n]}, (H_i^3)_{i\in[n]})$
8: **return** $(Q_i)_{i\in[n]}$

6.2 SplitRedGadget

The general idea of **SplitRed**-LAOLA is best understood in the polynomial representation, on which we focus here. We are given a sharing $(F_i)_{i\in[n]}$ of a polynomial $f = \sum_i f_i x^i$, where f_0 encodes the sensitive information. We now want to split f into two polynomials f' and f''. To understand the general idea behind the algorithm, we will first focus on the case that no faults are present and later show how to adapt to faults. In this scenario, we aim for the following two properties.

(*) The sum of the sensitive information of f' and f'' is an *additive* sharing of the sensitive information of f, i.e., $(f' + f'')_0 = f_0$ (which is the *split* part of the gadget).
(**) The degrees of both f' and f'' are equal to $d = \deg(f)$, but the degree of the sum $f' + f''$ is only equal to $\frac{d}{2}$ (which is the *reduce* part of the gadget).

To obtain these properties, we proceed roughly as follows:

(i) We generate a random polynomial $g = \sum_i g_i x^i$ of degree d with $g_0 = 0$, i.e., all coefficients g_i are drawn uniformly at random for $i > 0$.

(ii) We generate another polynomial $g' = \sum_i g'_i x^i$ of degree d with $g'_0 = 0$. For $i \in [1, \frac{d}{2}]$, the coefficients g'_i are drawn uniformly at random. For $i > \frac{d}{2}$, we set $g'_i = -g_i$. This means that $\deg(g) = \deg(g') = d$, but $\deg(g + g') = \frac{d}{2}$.

(iii) Now, the second property (**) is fulfilled, but we still need to share the sensitive information of f into g and g'. Now, remember that share j holds the value $f(\alpha_j)$. We now set $g_0 = \sum_{j<n/2} \lambda_j^{(0)} f(\alpha_j)$ and $g'_0 = \sum_{j\ge n/2} \lambda_j^{(0)} f(\alpha_j)$. The interpolation lemma then implies the correctness, as $g_0 + g'_0 = \sum_j \lambda_j^{(0)} f(\alpha_j) = f(0) = f_0$.

While the correctness of this idea directly follows from the interpolation lemma, we need to be careful to secure the algorithm against both probes and faults. To obtain probing security, we simply need to generate more random polynomials and include the values $\lambda_j^{(0)} f(\alpha_j)$ more carefully over time. More concretely, for $j = 1, \ldots, n/2$, we first generate random polynomials $\hat{g}^{(j)}$ of degree d (with absolute term 0), and for $j = 1, \ldots, n$, we first generate random polynomials $\tilde{g}^{(j)}$ of degree $\frac{d}{2}$ (with absolute term 0). For $j < n/2$, we compute $g^{(j)} = \tilde{g}^{(j)} + \hat{g}^{(j)}$ and for $j \ge n/2$, we compute $g^{(j)} = \tilde{g}^{(j)} - \hat{g}^{(j-n/2)}$. Then, for $j = 1, \ldots, n$, we set $g^{(j)} = g^{(j)} + \lambda_j^{(0)} f(\alpha_j)$ and finally, obtain $f' = \sum_{j<n/2} g^{(j)}$ and $f'' = \sum_{j\ge n/2} g^{(j)}$. A careful inspection of the construction shows that the sensitive information is always sufficiently hidden against up to d probes.

To handle faults, we need to make sure that the error coefficients of f are also preserved. To do so, we do not only incorporate the terms $\lambda_j^{(0)} f(\alpha_j)$, but the term $\left(\lambda_j^{(0)} + \sum_{k>d} \lambda_j^{(k)} \alpha_i^k\right) f(\alpha_j)$, which we will denote by $\hat{\lambda}_j^{(i)} \cdot f(\alpha_j)$ in the following. Note that the interpolation lemma implies that $\sum_j \sum_i \hat{\lambda}_j^{(i)} f(\alpha_j) = f_0 + \sum_{k>d} f_k x^k$.

We show in the full version that **SplitRed** is d-NI and transfers faults from its inputs to the output.

Algorithm 7 (n, d)-**SplitRed** for $n = d + \epsilon + 1$.

Input: Shares of f_0 as $(F_i)_{i\in[n]}$.
Output: Shares of f_0' as $(F'{}_i)_{i\in[n]}$ and shares of f_0'' as $(F''{}_i)_{i\in[n]}$, such that $f_0 = f_0' + f_0''$.

for $j \in [\frac{n}{2}]$ **do**
  $(\hat{g}_i^j)_{i\in[n]} \leftarrow \mathbf{ZEnc}_n^d$
for $j \in [\frac{n}{2}]$ **do**
  $(\tilde{g}_i^j)_{i\in[n]} \leftarrow \mathbf{ZEnc}_n^{\frac{d}{2}}$
  $(g_i^j)_{i\in[n]} \leftarrow (\tilde{g}_i^j)_{i\in[n]} + (\hat{g}_i^j)_{i\in[n]}$
for $j \in [\frac{n}{2}]$ **do**
  for $i \in [n]$ **do**
    $\mathcal{F}_i'^j \leftarrow \hat{\lambda}_j^i \cdot F_j$
for $j \in [\frac{n}{2}]$ **do**
  $(\mathcal{F}_i^j)_{i\in[n]} \leftarrow (\mathcal{F}_i'^j)_{i\in[n]} + (g_i^j)_{i\in[n]}$
for $j \in [\frac{n}{2}]$ **do**
  $(F'{}_i)_{i\in[n]} \leftarrow (F'{}_i)_{i\in[n]} + (\mathcal{F}_i^j)_{i\in[n]}$
for $j \in [\frac{n}{2}]$ **do**
  $(\tilde{g}_i^{j+\frac{n}{2}})_{i\in[n]} \leftarrow \mathbf{ZEnc}_n^{\frac{d}{2}}$
  $(g_i^{j+\frac{n}{2}})_{i\in[n]} \leftarrow (\tilde{g}_i^j)_{i\in[n]} - (\hat{g}_i^j)_{i\in[n]}$
for $j \in [\frac{n}{2}]$ **do**
  for $i \in [n]$ **do**
    $\mathcal{F}_i'^{j+\frac{n}{2}} \leftarrow \hat{\lambda}_{j+\frac{n}{2}}^i \cdot F_{j+\frac{n}{2}}$
for $j \in [\frac{n}{2}]$ **do**
  $(\mathcal{F}_i^{j+\frac{n}{2}})_{i\in[n]} \leftarrow (\mathcal{F}_i'^{j+\frac{n}{2}})_{i\in[n]} + (g_i^{j+\frac{n}{2}})_{i\in[n]}$
for $j \in [\frac{n}{2}]$ **do**
  $(F''{}_i)_{i\in[n]} \leftarrow (F''{}_i)_{i\in[n]} + (\mathcal{F}_i^{j+\frac{n}{2}})_{i\in[n]}$
return $(F'{}_i)_{i\in[n]}, (F''{}_i)_{i\in[n]}$

6.3 Share-wise Multiplication and Compression Gadgets

The share-wise multiplication **SWMult** (Alg. 2) works similar to the addition. Remember that **SplitRed** shares two polynomials $f(x)$ and $g(x)$ into $f'(x)$, $f''(x)$, $g'(x)$, and $g''(x)$ such that for $\tilde{f}(x) = f'(x) + f''(x)$ and $\tilde{g}(x) = g'(x) + g''(x)$ it holds $\tilde{f}(x)$ and $\tilde{g}(x)$ have degree $\frac{d}{2}$ and $f(0) = \tilde{f}(0)$, $g(0) = \tilde{g}(0)$. The share-wise multiplication now might lead to degrees larger than d when they compute $h^0(x) = f'(x) \cdot g'(x)$, $h^1(x) = f'(x) \cdot g''(x)$, $h^2(x) = f''(x) \cdot g'(x)$, $h^3(x) = f''(x) \cdot g''(x)$, but the final gadget **Comp** sums up all h^i and this results into a polynomial $\sum_{i=0}^3 h^i$ with degree d. This follows from the fact that we can alternatively write $\sum_{i=0}^3 h^i(x) = (f'(x) + f''(x)) \cdot (g'(x) + g''(x)) = \tilde{f}(x) \cdot \tilde{g}(x)$. Since $\tilde{f}(x)$ and $\tilde{g}(x)$ have degree $\frac{d}{2}$, the product $\tilde{f}(x) \cdot \tilde{g}(x)$ has degree d. Hence the sum of the h^i results in a degree d polynomial with secret $f_0 \cdot g_0$. Note that we also add an encoding of zero in **Comp** to re-randomize the values, but this does not change the

correctness of the gadget. We show in the full version that all values of **Comp** can be simulated from a few inputs and both **SWMult** and **Comp** transfer faults from their inputs to their outputs.

Algorithm 8 Comp for $n = d + \epsilon + 1$

Input: 4 Sharings $(H_i^j)_{i\in[n]}$ of h^j
Output: Sharing $(Q_i)_{i\in[n]}$ with $h^0 + h^1 + h^2 + h^3$

1: $(Q_i)_{i\in[n]} \leftarrow \mathbf{sZEnc}_n^d$
2: $(Q_i)_{i\in[n]} \leftarrow [[[(Q_i)_{i\in[n]} + (H_i^0)_{i\in[n]}] + (H_i^1)_{i\in[n]}] + (H_i^2)_{i\in[n]}] + (H_i^3)_{i\in[n]}$
3: **return** $(Q_i)_{i\in[n]}$

6.4 Security Analysis of the Multiplication Gadget

In this section we show that the multiplication **Mult** is frSNI and e-fault-robust. Corollary 1 shows that SNI and fault-invariance implies frSNI. Due to space constraints, all security proofs can be found in the full version.

Theorem 13. *The multiplication gadget **Mult** depicted in Algorithm 6 is d-fr(S)NI with respect to $\mathcal{F}^+$ or (d-wfr(S)NI with respect to $\mathcal{F}^{ind}$) and e-fault-robust with respect to $\mathcal{F}^{ind}$.*

Proof (sketch). In the full version we prove e-fault-robustness with respect to $\mathcal{F}^{ind}$. It remains to prove the frSNI property. Due to Corollary 1, it is sufficient to show that the multiplication gadget is SNI and fault invariant with respect to $\mathcal{F}^+$ because this implies d-frSNI with respect to $\mathcal{F}^+$. Further, Corollary 2 shows that this also implies d-wfrSNI with respect to $\mathcal{F}^{ind}$. Hence, it remains to prove that the gadget **Mult** is (i) SNI and (ii) fault invariant with respect to $\mathcal{F}^+$.

(i) In the full version we first analyze the different subroutines of **Mult** separately. Combining these results shows that the complete gadget is t-SNI.
(ii) The proof is similar to the fault-invariance proof of linear gadgets. The only difference is that the gadget has (only) one non-linear layer – the share-wise multiplication. The idea is that all faults before the non-linear layer can be moved to the inputs, and the faults after the non-liner layer can be moved to the outputs.

Acknowledgment

This work was partly supported by the German Research Foundation (DFG) via the DFG CRC 1119 CROSSING (project S7), by the German Federal Ministry of Education and Research and the Hessen State Ministry for Higher Education, Research and the Arts within their joint support of the National Research Center for Applied Cybersecurity ATHENE, and by the European Commission(ERCEA), ERC Grant Agreement 101044770 CRYPTOLAYER. This work has been partially supported by BMBF through the VE-Jupiter project grants 16ME0231K and 14ME0234.

References

[AAB+20] Abdelrahaman Aly, Tomer Ashur, Eli Ben-Sasson, Siemen Dhooghe, and Alan Szepieniec. Design of symmetric-key primitives for advanced cryptographic protocols. *IACR Trans. Symm. Cryptol.*, 2020(3):1–45, 2020.

[ADF16] Marcin Andrychowicz, Stefan Dziembowski, and Sebastian Faust. Circuit compilers with $O(1/\log(n))$ leakage rate. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 586–615. Springer, Heidelberg, May 2016.

[AGR+16] Martin R. Albrecht, Lorenzo Grassi, Christian Rechberger, Arnab Roy, and Tyge Tiessen. MiMC: Efficient encryption and cryptographic hashing with minimal multiplicative complexity. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part I*, volume 10031 of *LNCS*, pages 191–219. Springer, Heidelberg, December 2016.

[ARS+15] Martin R. Albrecht, Christian Rechberger, Thomas Schneider, Tyge Tiessen, and Michael Zohner. Ciphers for MPC and FHE. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 430–454. Springer, Heidelberg, April 2015.

[BBD+15] Gilles Barthe, Sonia Belaïd, François Dupressoir, Pierre-Alain Fouque, Benjamin Grégoire, and Pierre-Yves Strub. Verified proofs of higher-order masking. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 457–485. Springer, Heidelberg, April 2015.

[BBD+16] Gilles Barthe, Sonia Belaïd, François Dupressoir, Pierre-Alain Fouque, Benjamin Grégoire, Pierre-Yves Strub, and Rébecca Zucchini. Strong non-interference and type-directed higher-order masking. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016*, pages 116–129. ACM Press, October 2016.

[BCPZ16] Alberto Battistello, Jean-Sébastien Coron, Emmanuel Prouff, and Rina Zeitoun. Horizontal side-channel attacks and countermeasures on the ISW masking scheme. In Benedikt Gierlichs and Axel Y. Poschmann, editors, *CHES 2016*, volume 9813 of *LNCS*, pages 23–39. Springer, Heidelberg, August 2016.

[BDPA13] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Keccak. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 313–314. Springer, Heidelberg, May 2013.

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *20th ACM STOC*, pages 1–10. ACM Press, May 1988.

[BS97] Eli Biham and Adi Shamir. Differential fault analysis of secret key cryptosystems. In Burton S. Kaliski Jr., editor, *CRYPTO'97*, volume 1294 of *LNCS*, pages 513–525. Springer, Heidelberg, August 1997.

[BS21] Olivier Bronchain and François-Xavier Standaert. Breaking masked implementations with many shares on 32-bit software platforms. *IACR TCHES*, 2021(3):202–234, 2021. `https://tches.iacr.org/index.php/TCHES/article/view/8973`.

[CFG+10] Christophe Clavier, Benoit Feix, Georges Gagnerot, Mylène Roussellet, and Vincent Verneuil. Horizontal correlation analysis on exponentiation. In Miguel Soriano, Sihan Qing, and Javier López, editors, *ICICS 10*, volume 6476 of *LNCS*, pages 46–61. Springer, Heidelberg, December 2010.

[Cla07] Christophe Clavier. Secret external encodings do not prevent transient fault analysis. In Pascal Paillier and Ingrid Verbauwhede, editors, *CHES 2007*, volume 4727 of *LNCS*, pages 181–194. Springer, Heidelberg, September 2007.

[CN16] Thomas De Cnudde and Svetla Nikova. More efficient private circuits II through threshold implementations. In *FDTC 2016*, pages 114–124. IEEE Computer Society, 2016.

[CPR12] Jean-Sébastien Coron, Emmanuel Prouff, and Thomas Roche. On the use of shamir's secret sharing against side-channel analysis. In *CARDIS*, volume 7771 of *Lecture Notes in Computer Science*, pages 77–90. Springer, 2012.

[CS20] Gaëtan Cassiers and François-Xavier Standaert. Trivially and efficiently composing masked gadgets with probe isolating non-interference. *IEEE Trans. Inf. Forensics Secur.*, 15:2542–2555, 2020.

[DDF14] Alexandre Duc, Stefan Dziembowski, and Sebastian Faust. Unifying leakage models: From probing attacks to noisy leakage. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 423–440. Springer, Heidelberg, May 2014.

[DEK+18] Christoph Dobraunig, Maria Eichlseder, Thomas Korak, Stefan Mangard, Florian Mendel, and Robert Primas. SIFA: Exploiting ineffective fault inductions on symmetric cryptography. *IACR TCHES*, 2018(3):547–572, 2018. `https://tches.iacr.org/index.php/TCHES/article/view/7286`.

[DN19] Siemen Dhooghe and Svetla Nikova. My gadget just cares for me - how NINA can prove security against combined attacks. Cryptology ePrint Archive, Report 2019/615, 2019. `https://eprint.iacr.org/2019/615`.

[DN20a] Siemen Dhooghe and Svetla Nikova. Let's tessellate: Tiling for security against advanced probe and fault adversaries. In Pierre-Yvan Liardet and Nele Mentens, editors, *CARDIS 2020*, volume 12609 of *Lecture Notes in Computer Science*, pages 181–195. Springer, 2020.

[DN20b] Siemen Dhooghe and Svetla Nikova. My gadget just cares for me - how NINA can prove security against combined attacks. In Stanislaw Jarecki, editor, *CT-RSA 2020*, volume 12006 of *LNCS*, pages 35–55. Springer, Heidelberg, February 2020.

[FRSG22] Jakob Feldtkeller, Jan Richter-Brockmann, Pascal Sasdrich, and Tim Güneysu. CINI MINIS: domain isolation for fault and combined security. In *CCS*, pages 1023–1036. ACM, 2022.

[GKR+21] Lorenzo Grassi, Dmitry Khovratovich, Christian Rechberger, Arnab Roy, and Markus Schofnegger. Poseidon: A new hash function for zero-knowledge proof systems. In Michael Bailey and Rachel Greenstadt, editors, *USENIX Security 2021*, pages 519–535. USENIX Association, August 2021.

[GLR+20] Lorenzo Grassi, Reinhard Lüftenegger, Christian Rechberger, Dragos Rotaru, and Markus Schofnegger. On a generalization of substitution-permutation networks: The HADES design strategy. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 674–704. Springer, Heidelberg, May 2020.

[GM11] Louis Goubin and Ange Martinelli. Protecting AES with Shamir's secret sharing scheme. In Bart Preneel and Tsuyoshi Takagi, editors, *CHES 2011*, volume 6917 of *LNCS*, pages 79–94. Springer, Heidelberg, September / October 2011.

[GPRV21] Dahmun Goudarzi, Thomas Prest, Matthieu Rivain, and Damien Vergnaud. Probing security through input-output separation and revisited quasilinear masking. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(3):599–640, 2021.

[HKL+22] Jincheol Ha, Seongkwang Kim, ByeongHak Lee, Jooyoung Lee, and Mincheol Son. Rubato: Noisy ciphers for approximate homomorphic encryption. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part I*, volume 13275 of *LNCS*, pages 581–610. Springer, Heidelberg, May / June 2022.

[IPSW06] Yuval Ishai, Manoj Prabhakaran, Amit Sahai, and David Wagner. Private circuits II: Keeping secrets in tamperable circuits. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 308–327. Springer, Heidelberg, May / June 2006.

[ISW03] Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 463–481. Springer, Heidelberg, August 2003.

[LFZD14] Pei Luo, Yunsi Fei, Liwei Zhang, and A. Adam Ding. Side-channel power analysis of different protection schemes against fault attacks on AES. In *ReConFig 2014*, pages 1–6. IEEE, 2014.

[ORSW12] Yossef Oren, Mathieu Renauld, François-Xavier Standaert, and Avishai Wool. Algebraic side-channel attacks beyond the hamming weight leakage model. In Emmanuel Prouff and Patrick Schaumont, editors, *CHES 2012*, volume 7428 of *LNCS*, pages 140–154. Springer, Heidelberg, September 2012.

[RDB+18] Oscar Reparaz, Lauren De Meyer, Begül Bilgin, Victor Arribas, Svetla Nikova, Ventzislav Nikov, and Nigel P. Smart. CAPA: The spirit of beaver against physical attacks. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 121–151. Springer, Heidelberg, August 2018.

[REB+08] Francesco Regazzoni, Thomas Eisenbarth, Luca Breveglieri, Paolo Ienne, and Israel Koren. Can knowledge regarding the presence of countermeasures against fault attacks simplify power attacks on cryptographic devices? In Cristiana Bolchini, Yong-Bin Kim, Dimitris Gizopoulos, and Mohammad Tehranipoor, editors, *DFT 2008*, pages 202–210. IEEE Computer Society, 2008.

[RFSG22] Jan Richter-Brockmann, Jakob Feldtkeller, Pascal Sasdrich, and Tim Güneysu. VERICA - verification of combined attacks automated formal verification of security against simultaneous information leakage and tampering. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2022(4):255–284, 2022.

[RP12] Thomas Roche and Emmanuel Prouff. Higher-order glitch free implementation of the AES using secure multi-party computation protocols - extended version. *Journal of Cryptographic Engineering*, 2(2):111–127, September 2012.

[SFRES18] Okan Seker, Abraham Fernandez-Rubio, Thomas Eisenbarth, and Rainer Steinwandt. Extending glitch-free multiparty protocols to resist fault injection attacks. *IACR TCHES*, 2018(3):394–430, 2018. `https://tches.iacr.org/index.php/TCHES/article/view/7281`.

[SMG16] Tobias Schneider, Amir Moradi, and Tim Güneysu. ParTI – towards combined hardware countermeasures against side-channel and fault-injection attacks. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part II*, volume 9815 of *LNCS*, pages 302–332. Springer, Heidelberg, August 2016.

[VGS14] Nicolas Veyrat-Charvillon, Benoît Gérard, and François-Xavier Standaert. Soft analytical side-channel attacks. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part I*, volume 8873 of *LNCS*, pages 282–296. Springer, Heidelberg, December 2014.