INFORMATION LEAKAGE ATTACKS AND DEFENSES IN THE IOT DOMAIN

Vom Fachbereich Informatik (FB 20)
an der Technischen Universität Darmstadt
zur Erlangung des akademischen Grades eines Doktor-Ingenieurs
genehmigte Dissertation von:

MSc. Richard Mitev

Referenten:
Prof. Dr.-Ing. Matthias Hollick (Erstreferent)
Prof. Dr.-Ing. Markus Miettinen (Erster Koreferent)
Prof. Dr.-Ing. Thomas Schneider (Zweiter Koreferent)

Tag der Einreichung: 29. Januar 2025
Tag der Disputation: 27. März 2025

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Secure Mobile Networking Lab
Fachbereich Informatik
Technische Universität Darmstadt

Hochschulkennziffer: D17
Erscheinungsort: Darmstadt

Richard Mitev:
Information Leakage Attacks and Defenses in the IoT Domain.
Darmstadt, Technische Universität Darmstadt
Tag der mündlichen Prüfung: 27 März 2025
Jahr der Veröffentlichung der Dissertation auf TUprints: 2025

DOCTORAL REFEREES:
Prof. Dr.-Ing. Matthias Hollick (Doctoral Referee)
Prof. Dr.-Ing. Markus Miettinen (1st Doctoral Co-Referee)
Prof. Dr.-Ing. Thomas Schneider (2nd Doctoral Co-Referee)

FURTHER DOCTORAL COMMISSION MEMBERS:
Prof. Dr.-Ing. Andreas Koch
Prof. Dr. Dr. Christian Reuter

Darmstadt, Germany, 29. Januar 2025

Erklärung zur Dissertationsschrift

gemäß §8 & §9 der Allgemeinen Bestimmungen der Promotionsordnung der Technischen Universität Darmstadt vom 12. Januar 1990 (ABI. 1990, S. 658) in der Fassung der 9. Novelle vom 15. November 2023.

Hiermit versichere ich, Richard Mitev, die vorliegende Dissertationsschrift ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die Quellen entnommen wurden, sind als solche kenntlich gemacht. Außerdem versichere ich, dass die „Grundsätze zur Sicherung guter wissenschaftlicher Praxis an der Technischen Universität Darmstadt“ und die „Leitlinien zum Umgang mit digitalen Forschungsdaten an der TU Darmstadt“ in der aktuellen Version, zum Zeitpunkt der Einreichung, beachtet wurden. Eigenzitate aus vorausgehenden wissenschaftlichen Veroöffentlichungen sowie die Urheberschaften der einzelnen Beiträge sind in Anlehnung an die Hinweise des Promotionsausschusses des Fachbereichs Informatik zum Thema „Kumulative Dissertation und Eigenzitate in Dissertationen“ (CR; 01.12.2022) im Kapitel „Contributions“ auf den Seiten XIII bis XIV angegeben. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen. In der abgegebenen Dissertationsschrift stimmen die schriftliche und die elektronische Fassung überein. Ich versichere, dass zu keinem vorherigen Zeitpunkt eine Promotion versucht wurde.

Darmstadt, Germany, 29. Januar 2025

Richard Mitev

Abstract

The Internet of Things (IoT) market is experiencing rapid growth as the market is projected to double between 2020 and 2025. The surge in the adoption of IoT technologies into devices of daily life, from smartphones to satellite communication, highlights significant concerns about their security. Even with heightened security measures, vulnerabilities persist due to oversight of potential attack surfaces and scenarios, leaving entire systems vulnerable to the risk of information leakage. Therefore, numerous challenges within this field must still be addressed. For that reason, it is crucial for manufacturers and users to understand emerging attack surfaces resulting from the IoT field. IoT devices can leak information through adversarial exploitation of novel human-computer interfaces, such as voice or touch, which are often incorporated into smart speakers and smartphones. The analysis of such attack vectors plays a pivotal role in uncovering vulnerabilities across the field of IoT devices, thereby paving the way for mitigations to increase security levels to prevent information leakage. This goal can be achieved by strengthening User Interfaces (UIs) or protecting private user data by making it inaccessible to the adversary.

In this dissertation, we design, implement and evaluate novel approaches to exploit the attack surfaces of different IoT devices and to offer mitigations of exploits for upcoming technologies in the area of IoT devices. Specifically, we propose 1) novel attacks on smart speaker virtual assistant's interaction model using ultrasonic audio, 2) improvements for virtual assistant's wake-word detectors and means to detect malicious wake-word activated devices, 3) a novel attack on smart devices' capacitive touch screens using airborne electro-magnetic-interference and 4) protection against location leakage of satellite internet users using IoT long-range radio network technology.

Virtual Assistants' Interaction Model Attacks. A limited and well-defined interaction model for virtual assistants is crucial for its users to be able to depend upon their assistants and to protect them from harm, e.g., in the form of information leakage. A virtual assistant should, by design, have access to the user's private data in order to help with everyday tasks. Therefore, the interaction model, including that of third-party service applications, called "Skills" or "Actions", is very limited. Earlier works about attacks against voice-enabled virtual assistants focused on command injection attacks without taking into account possible unintended usage of the seemingly limited Skill interaction model.

In AlexaLiedtoMe, we show that a specially crafted benign-looking malicious Skill, in combination with ultrasonic voice jamming and command injection, allows the adversary to launch powerful attacks by completely hijacking the entire conversation between the virtual assistant and the benign user without his or her knowledge. We implemented two proof-of-concept attacker devices and evaluated the attack framework on an Amazon Echo Dot device aimed at two smart home security systems and the 10 top Skills from the official Skill store.

Virtual Assistants' Wake-Word Detection Defenses. While a safe interaction model of virtual assistants is one building block to protect against information leakage to an attacker, the correct operation of the voice interface is another important aspect. However, users are not always aware of when their smart home devices record and send audio to the cloud. Related works on identifying unintended data transmissions out of the user's network are unfortunately not applicable to voice data or wake-word-activated devices (i.e., devices being activated after detecting the utterance of a specific word).

In LeakyPick, we propose a small IoT device that has a statistical approach to detect if a device unexpectedly transmits audio data over the network and which can "probe" the user's smart home devices to detect if these devices respond to a wake-word the user does not expect. We evaluate our approach on eight devices where we can detect the network traffic containing their audio transmission with 94% accuracy and discover 89 unexpected wake-words of an Amazon Echo Dot.

Building on this insight, in FakeWake, we utilize Machine Learning, specifically genetic algorithms, to quickly generate a large set of wrongly recognized wake-words used to evaluate the black-box wake-word detector precision of different virtual assistants by "probing" the detector with generated words, i.e., "audio-fuzzing" the detector. To achieve this, we utilized variation methods such as mutations, recombination and crossover of candidate words. We found 965 generated words activating the eight most popular English and Chinese smart speakers. Consecutively, to identify the features of a word that contribute to its acceptance as a wake-word, we trained a tree-based binary classifier, which we can interpret to gain the decisive phonetic features of a word. Finally, we propose countermeasures to increase the precision of a wake-word detector by screening words for the identified phonetic features or retraining the model with identified wrongly accepted wake-words, enabling it to reject 97% of identified words.

Capacitive Touch Screen Attacks. Smart Internet-of-Things devices, such as smartphones equipped with a human interface, such as a capacitive touch screen, depend on the correct input provided by these interfaces, especially for security-critical tasks. Modern touch screens and devices go through thorough Electromagnetic Compatibility (EMC) tests and exhibit anti-interference designs such as shielding, frequency hopping and layout optimization to avoid the influence of environmental Electromagnetic Inter-

ference (EMI), however, no works analyze and propose defending against directed intentional EMI attacks, which can enable a malicious actor to access sensitive information.

In GhostTouch, we propose a remote EMI attack against capacitive touchscreens. We utilize airborne EMI to interfere with the capacitance measurement of the touchscreen, which is injected into the receiving electrodes integrated into the touchscreen, affecting the touch point detection. We carefully select the parameters of the signal to achieve various predictable touch events, e.g., a tap, a swipe-up or a swipe-down in targeted locations. Twenty-three smartphones are evaluated using our novel, powerful and portable attack setup, enabling attacks such as answering a phone call, unlocking the device or entering a password.

Preventing Location Leakage of Satellite Internet Users. Satellite-based internet is a crucial remedy in conflict zones, however, it exposes its users to triangulation. Modern Wi-Fi-enabled base stations imply that the internet user is in their vicinity. Therefore, these systems can leak the user's geographic location information to an attacker. Unfortunately, existing works on emergency networks and location privacy do not consider triangulation protection, utilize specialized hardware or exhibit impractical assumptions, respectively.

In AnonSat, we utilize the advancements in IoT communication technologies to establish a long-range wireless network among satellite terminals and route user's communication to distant terminals. By randomly selecting the output gateway of the user's data, our approach prevents geolocation and, therefore, location leakage. Our proof-of-concept device is implemented with widely available hardware and does not require the usage of a specific software or hardware on the user's side. In addition, our network is compatible with the Internet Protocol.

Zusammenfassung

Der IoT-Markt erlebt ein rasantes Wachstum, da sich der Markt zwischen 2020 und 2025 voraussichtlich verdoppeln wird. Die zunehmende Verbreitung von IoT-Technologien in Geräten des täglichen Lebens, von Smartphones bis zur Satellitenkommunikation, führt zu erheblichen Bedenken hinsichtlich ihrer Sicherheit. Trotz erhöhter Sicherheitsmaßnahmen bleiben Schwachstellen bestehen, da potenzielle Angriffsflächen und -szenarien übersehen werden, wodurch ganze Systeme dem Risiko eines Datenlecks ausgesetzt sind. Daher müssen in diesem Bereich noch zahlreiche Herausforderungen bewältigt werden. Aus diesem Grund ist es für Hersteller und Nutzer von entscheidender Bedeutung, neu entstehende Angriffsflächen zu verstehen, die sich aus dem IoT-Bereich ergeben. IoT-Geräte können durch die Ausnutzung neuartiger Mensch-Computer-Schnittstellen, wie z. B. Sprach- oder Berührungseingaben, die häufig in intelligenten Lautsprechern und Smartphones integriert sind, persönliche Informationen preisgeben. Die Analyse solcher Angriffsvektoren spielt eine zentrale Rolle bei der Aufdeckung von Schwachstellen im gesamten Bereich der IoT-Geräte und ebnet damit den Weg für Maßnahmen zur Erhöhung des Sicherheitsniveaus, um Informationslecks zu verhindern. Dieses Ziel kann erreicht werden, indem UIs gestärkt oder private Benutzerdaten geschützt werden, indem sie für den Angreifer unzugänglich gemacht werden.

In dieser Dissertation entwerfen, implementieren und evaluieren wir neuartige Ansätze zur Ausnutzung der Angriffsflächen verschiedener IoT-Geräte und bieten Gegenmaßnahmen von Exploits der Technologien im Bereich der IoT-Geräte. Konkret schlagen wir 1) neuartige Angriffe auf das Interaktionsmodell des virtuellen Assistenten von Smart Speakern unter Verwendung von Ultraschall-Audio, 2) Verbesserungen für die Weckwort-Detektoren der virtuellen Assistenten und Mittel zur Erkennung bösartiger Geräte, die durch Weckwörter aktiviert werden, 3) einen neuartigen Angriff auf die kapazitiven Touchscreens von Smart Devices unter Verwendung von elektromagnetischer Interferenz und 4) Schutz gegen die Preisgabe des Standorts von Satelliten-Internetnutzern unter Verwendung von IoT-Langstrecken-Funknetzwerktechnologie vor.

Angriffe auf das Interaktionsmodell virtueller Assistenten. Ein begrenztes und genau definiertes Interaktionsmodell für virtuelle Assistenten ist von entscheidender Bedeutung, damit sich die Benutzer auf ihre Assistenten verlassen können und sie vor Schaden, z. B. in Form von Informationslecks, geschützt sind. Ein virtueller Assistent sollte von vornherein Zugang zu den privaten Daten des Nutzers haben, um ihn bei alltäglichen Aufgaben zu unterstützen. Daher ist das Interaktionsmodell, einschließlich

desjenigen von Dienstanwendungen von Drittanbietern, die als "Skills" oder "Actions" bezeichnet werden, sehr begrenzt. Frühere Arbeiten über Angriffe auf sprachgesteuerte virtuelle Assistenten konzentrierten sich auf Angriffe durch Befehlsinjektion, ohne die mögliche unbeabsichtigte Nutzung des scheinbar begrenzten Skill-Interaktionsmodells zu berücksichtigen.

In AlexaLiedtoMe zeigen wir, dass ein speziell entwickelter, harmlos aussehender aber bösartiger Skill in Kombination mit Ultraschall-Störsignalen und -Befehlsinjektion dem Angreifer ermöglicht, mächtige Angriffe zu starten, indem er die gesamte Konversation zwischen dem virtuellen Assistenten und dem Benutzer ohne dessen Wissen übernimmt. Wir haben zwei Proof-of-Concept-Angriffsgeräte implementiert und das Angriffs-Framework auf einem Amazon Echo Dot-Gerät evaluiert, indem wir zwei Smart-Home-Sicherheitssysteme und die 10 Top-Skills aus dem offiziellen Skill-Store angriffen.

Schutz der Weckwort-Erkennung von virtuellen Assistenten. Während ein sicheres Interaktionsmodell für virtuelle Assistenten ein Baustein zum Schutz vor dem Abfluss von Informationen an einen Angreifer ist, ist der korrekte Betrieb der Sprachschnittstelle ein weiterer wichtiger Aspekt. Allerdings sind sich die Nutzer nicht immer bewusst, wann ihre Smart-Home-Geräte Audiodaten aufzeichnen und an die Cloud senden. Verwandte Arbeiten zur Erkennung unbeabsichtigter Datenübertragungen aus dem Netzwerk des Benutzers sind leider nicht auf Sprachdaten oder auf durch Weckwort aktivierte Geräte anwendbar (d. h. Geräte, die nach der Erkennung der Äußerung eines bestimmten Wortes aktiviert werden).

In LeakyPick schlagen wir ein kleines IoT-Gerät vor, das über einen statistischen Ansatz verfügt, um zu erkennen, ob ein Gerät unerwartet Audiodaten über das Netzwerk überträgt, und das die Smart-Home-Geräte des Benutzers "testen" kann, um zu erkennen, ob diese Geräte auf ein Weckwort reagieren, das der Benutzer nicht erwartet. Wir evaluieren unseren Ansatz an acht Geräten, bei denen wir den Netzwerkverkehr, der ihre Audioübertragung enthält, mit einer Genauigkeit von 94% erkennen und 89 unerwartete Weckwörter eines Amazon Echo Dot entdecken können.

Aufbauend auf diesen Erkenntnissen nutzen wir in FakeWake maschinelles Lernen, insbesondere genetische Algorithmen, um schnell eine große Menge falsch erkannter Weckwörter zu generieren, mit denen wir die Präzision der Blackbox-Wechwortdetektoren verschiedener virtueller Assistenten bewerten, indem wir den Detektor mit generierten Wörtern "testen", d.h. den Detektor "audio-fuzz-testen". Um dies zu erreichen, haben wir Variationsmethoden wie Mutation, Rekombination und Crossover von Wortkandidaten eingesetzt. Wir fanden 965 generierte Wörter, die die acht beliebtesten englischen und chinesischen Smart Speaker aktivieren. Um die Merkmale eines Wortes zu identifizieren, die zu seiner Akzeptanz als Weckwort beitragen, haben wir einen baumbasierten binären Klassifikator trainiert, den wir interpretieren können, um die entscheidenden

phonetischen Merkmale eines Wortes zu gewinnen. Schließlich schlagen wir Gegenmaßnahmen vor, um die Präzision eines Weckwort-Detektors zu erhöhen, indem wir Wörter auf die identifizierten phonetischen Merkmale hin überprüfen oder das Modell mit identifizierten, falsch akzeptierten Weckwörtern neu trainieren, so dass es 97% der identifizierten Wörter ablehnen kann.

Angriffe auf kapazitive Touchscreens. Intelligente IoT-Geräte, wie z. B. Smartphones, die mit einer menschlichen Schnittstelle wie einem kapazitiven Touchscreen ausgestattet sind, sind auf die korrekte Meldung der Eingaben durch diese Schnittstellen angewiesen, insbesondere bei sicherheitskritischen Aufgaben. Moderne Touchscreens und Geräte werden gründlichen EMC-Tests unterzogen und verfügen über Anti-Interferenz-Designs wie Abschirmung, Frequenzsprungverfahren und Layout-Optimierung, um den Einfluss von Umwelt-EMI zu vermeiden. Es gibt jedoch keine Arbeiten, die eine Verteidigung gegen gezielte, absichtliche EMI-Angriffe analysieren und vorschlagen, die es einem böswilligen Akteur ermöglichen können, auf sensible Informationen zuzugreifen.

In GhostTouch schlagen wir einen ferngesteuerten EMI-Angriff gegen kapazitive Touchscreens vor. Wir nutzen EMI über die Luft, um die Kapazitätsmessung des Touchscreens zu stören. Das Signal wird in die in den Touchscreen integrierten Empfangselektroden eingespeist und beeinträchtigt so die Erkennung von Berührungspunkten. Wir wählen die Parameter des Signals sorgfältig aus, um verschiedene vorhersehbare Berührungsereignisse zu erreichen, z. B. ein Tippen, ein Wischen nach oben oder ein Wischen nach unten an bestimmten Stellen. Dreiundzwanzig Smartphones werden mit unserem neuartigen, leistungsstarken und tragbaren Angriffsaufbau evaluiert, der Angriffe wie die Beantwortung eines Telefonanrufs, das Entsperren des Geräts oder die Eingabe eines Passworts ermöglicht.

Ortung von Satelliten-Internetnutzern verhindern. Das satellitengestützte Internet ist ein wichtiges Hilfsmittel in Konfliktgebieten, setzt seine Nutzer jedoch der Triangulation aus. Moderne Wi-Fi-fähige Basisstationen implizieren, dass sich der Internetnutzer in ihrer Nähe befindet. Daher können diese Systeme die geografischen Standortinformationen des Nutzers an einen Angreifer preisgeben. Leider berücksichtigen bestehende Arbeiten zu Notfallnetzen und zum Schutz der Standortdaten keinen Triangulationsschutz, verwenden spezielle Hardware oder gehen von unpraktischen Voraussetzungen aus.

In AnonSat nutzen wir die Fortschritte in der IoT-Kommunikationstechnologie, um ein drahtloses Netzwerk mit großer Reichweite zwischen Satellitenterminals einzurichten und die Kommunikation des Benutzers an entfernte Terminals weiterzuleiten. Durch die zufällige Auswahl des Ausgangsgateways der Nutzerdaten verhindert unser Ansatz die Geolokalisierung und damit die Preisgabe des Standorts. Unser Proof of Concept (PoC)-Gerät wird mit weithin verfügbarer Hardware implementiert und erfordert keine spezielle

Software oder Hardware auf Seiten des Nutzers. Darüber hinaus ist unser Netzwerk mit dem Internetprotokoll kompatibel.

Contributions

The academic publications this dissertation is based upon are the outcome of collaborations with international researchers and students, which I want to thank for their contributions. In the following, I introduce my co-authors and detail their and my contributions to each of the publications this dissertation is comprised of.

Chapter 2 is based on AlexaLiedtoMe [107] (Appendix A). This work is a collaboration with Markus Miettinen and Ahmad-Reza Sadeghi. For this publication I conducted the paper's design under the supervision of Markus Miettinen. I performed a feasibility analysis and implemented and evaluated the prototype. Preliminaries, Related Work, Countermeasures, Discussion and Future Work, were written by me. Markus Miettinen and Ahmad-Reza Sadeghi contributed to the paper writing. This paper was developed in parallel with my master thesis. The proof-of-concept attack developed in the thesis targets two Skills, while in the paper we designed and evaluated the attack on the top ten Skills in the official Skill store. Further, the paper designs and evaluates novel attack hardware based on IoT devices and provides a more streamlined attack signal generation, in comparison to the thesis.

Chapter 3 is based on LeakyPick [108] (Appendix B), a collaboration with Anna Pazii, Markus Miettinen, William Enck and Ahmad-Reza Sadeghi. For this publication, William Enck and me contributed to the discussions on the design that resulted in this publication. I conducted the paper's design under the supervision of Markus Miettinen. I performed a feasibility analysis and implemented and evaluated the prototype. Background, Related Work and Discussion were written by me. Anna Pazii partially executed evaluation experiments. Markus Miettinen, William Enck, and Ahmad-Reza Sadeghi contributed to the paper's writing.

Chapter 3 is additionally based on FakeWake [14] (Appendix C). This work is a collaboration with Yanjiao Chen, Yijie Bai, Kaibo Wang, Ahmad-Reza Sadeghi and Wenyuan Xu. For this publication, I proposed the initial idea that resulted in this publication. I performed a feasibility analysis on an implemented prototype. Yanjiao Chen and Yijie Bai conducted the paper's design under the supervision of Kaibo Wang. I engaged in discussions about the design. The Implementation was conducted by Yanjiao Chen, Yijie Bai, Kaibo Wang and me. Yanjiao Chen, Yijie Bai and Kaibo Wang evaluated the Chinese-speaking devices, whereas I evaluated the English-speaking devices. I conducted the English user study. Mitigations and Explanations were proposed by Yanjiao Chen,

Yijie Bai and Kaibo Wang. Yanjiao Chen, Yijie Bai, Kaibo Wang and I contributed to the paper's writing. All co-authors contributed genuinely to this publication.

Chapter 4 is based on GhostTouch [175] (Appendix D), a collaboration with Kai Wang, Chen Yan, Xiaoyu Ji, Ahmad-Reza Sadeghi and Wenyuan Xu. For this publication, I proposed the initial idea of using airborne EMI to inject touchpoints, which resulted in this publication. I performed an initial feasibility analysis using the ChipSHOUTER on multiple devices, finding vulnerable devices and discovering that different injection frequencies inject touches with different characteristics, before collaboration started. Kai Wang conducted the feasibility section using the Electrostatic Gun. Kai Wang and me, under the supervision of Chen Yan and Xiaoyu Ji, conducted the paper's design. Kai Wang and I independently designed and developed synchronizing the injection frequency with the screen's sensing frequency. The implementation and evaluation of the Antenna Array, Phone Locator and high-power Single Touch injector were conducted by Kai Wang. The evaluation included the User Study, Taps and Swipes as well as the entering a password scenario and the attack distance of section Factors Affecting Injection. The implementation and evaluation of the mobile Touch Injector were conducted by me. Evaluation included Taps and Swipes as well as answering the phone call, swiping up to unlock and pressing the button scenarios by creating an evaluation App and the different phone models part in the section Factors Affecting Injection. Mitigations and Explanations were proposed by Kai Wang and me. Kai Wang, Chen Yan, Xiaoyu Ji and I contributed to the paper's writing. All co-authors contributed genuinely to this publication.

Chapter 5 is based on AnonSat [77] (Appendix E). This work is a collaboration with David Koisser, Marco Chilese and Ahmad-Reza Sadeghi. For this publication, the paper's design was a collaboration between David Koisser and me. I implemented and evaluated the prototype. David Koisser processed the underlying data sets, implemented the network simulation, and evaluated the results. The security analysis and the technical consideration sections were a collaboration between David Koisser and me. Marco Chilese helped with research for the related work and contributed to the paper writing.

Contents

1

Introduction

Advances in chip production and design have enabled us to innovate and integrate technology into every aspect of our daily lives. This trend has resulted in the rise of consumer Internet of Things (IoT) devices, smart (everyday) objects equipped with sensors, software, processing capability and connectivity. Consumer IoT devices now range from personal devices such as smartphones and smart watches to smart home devices such as appliances and surveillance systems [81]. Rapid adoption has led to the IoT market reaching the size of $330.6B\$$ in 2020 and it is expected to grow to $1751.8B\$$ by 2029 [99]. However, along with this swift growth of ubiquitous and always-connected devices and the simultaneous production of increasingly diverse products comes the challenge of ensuring the security of billions of devices. In the light of IoT devices collecting and processing private data of their users, such as health-related data, consumers naturally want devices to keep their information safe. Therefore, it is crucial for manufacturers to prevent devices from leaking this information. However, the lack of effort in securing IoT devices, such as in the area of access-control, has led to large-scale attacks targeting IoT devices. A prominent attack resulting from this oversight was caused by the Mirai malware [22]. In addition, academia has published new attacks on IoT devices, repeatedly demonstrating information leakage [169]. Therefore, there is a need for identifying exploits and design mitigations for attacks against IoT devices, in order to protect their users' private information. In this dissertation, we will identify novel attacks against IoT devices and propose countermeasures to enhance their security.

Information leakage is the unintended revelation of protected data to an unauthorized party. Precisely, revealing some or all of the information inside a system which was designed to securely contain this data. This can happen through any kind of access to the system, e.g., through adversarial software attacks, side-channel attacks or even by user mistake [3, 4]. In the context of personal consumer IoT devices, information leakage can occur through a malicious entity accessing the confidential information stored or computed on IoT devices (e.g., the user's personal information) or by monitoring the user and his or her environment through compromised devices [124].

Failing to properly secure IoT devices has resulted in severe attacks, such as the Mirai botnet attack in 2016 that infected approximately 600 000 devices and caused widespread disruption through the, at the time, largest Distributed Denial of Service (DDoS) attack on the online infrastructure provider Dyn, with $1.2Tb/s$ [21]. While the method of

propagation of the malware was unsophisticated, i.e., a dictionary attack on remotely accessible services, it showed the dangers of inadequately secured IoT devices on the internet, which led to a full compromise of the device including access to all its data. However, most efforts induced by this incident related to protecting against easily scalable exploits, to protect against attacks like Mirai in the future. Though, due to the nature of IoT devices being diverse, only protecting against network-based attacks is insufficient. This led to the market size for IoT security increasing from 17.9*B*$ in 2022 to 20.9*B*$ in 2023 and is projected to grow to 59.2*B*$ in 2028 [100]. IoT devices exhibit the same vulnerabilities as other computer systems, which make them susceptible to, e.g., network-based attacks [130, 8], physical attacks [130] and malware [173]. Additionally, IoT devices come with a plethora of extra attack surfaces from protocols and technologies that are IoT specific [151] (e.g., 6LoWPAN [74, 58] or ZigBee [134]), over their sensors [156] (e.g., microphones [185] or gyroscopes [73]) to attacks that are only possible due to the resource-constrained nature (e.g., low clock speed) of embedded devices (e.g., glitching attacks [139] or attacks on primitive circuits [167]).

The impact of exploiting these vulnerabilities is exacerbated by the nature of consumer IoT devices: 1) Consumers use the devices with little to no IT security knowledge, leaving the burden of security solely on the device itself, 2) many of those devices by design sense and process private data, e.g., in the user's home, whose disclosure is a privacy concern and 3) an attacker gains the ability to cause damage not only in the cyberspace but also in the real world, as IoT devices often control other devices, such as home appliances. Additionally, IoT devices are reported "to come to market faster than they can be secured" as business models prioritize growth over security [168]. Consequently, unidentified potential attack surfaces and scenarios leave systems vulnerable to the risk of leakage of the private information of their users [124].

Preventing information leakage in the dynamic and fast-paced landscape of IoT devices is therefore a challenge that requires a proactive approach to identify and mitigate vulnerabilities instead of reactively responding to exploits, as the privacy impact of leaked information cannot be undone. We therefore address the problem of information leakage of IoT devices, which is comprised of two steps: 1) identifying the attack surfaces of these devices and 2) designing and implementing effective mitigation strategies. A comprehensive assessment of the attack surfaces and implementation of robust security measures is mandatory to significantly reduce the risk of information leakage.

Unfortunately, both academic research and industry have largely overlooked the security consequences of integrating IoT-specific features into novel devices. For instance, the deployment of virtual assistants into smart speakers and smartphones has increased their attack surface by incorporating a new interface. Adding such a User Interface (UI) may introduce significant security and privacy challenges [9]. Similarly, touch-based UIs made possible by capacitive touchscreens and new ways of connectivity, like satellite internet, have further increased the attack surface of IoT devices [110]. As a consequence,

the challenge of securing these UIs and the satellite network interface must still be addressed.

In this dissertation, we focus on these attack surfaces in particular. As the novel voice-based interfaces to control devices via virtual assistants also provide novel ways of compromising the underlying or connected devices, we advance the field of attacks leading to and defenses protecting against information leakage of these devices. Further, we found novel ways to physically and remotely attack touchscreens, which are becoming ever-present in devices including smartphones, wearables and even cars, resulting in a compromise of the device's security. Finally, in light of the recent widespread deployment, we found and addressed novel privacy concerns of satellite internet users.

1.1 NOVEL ATTACKS ON IOT DEVICES

As stated earlier, the vastly diverse IoT landscape, incorporating novel features into devices and therefore increasing the attack surface [9, 110] necessitates new and unusual approaches to analyze possible exploit entry points. Just analyzing conventional attacks on computer systems may be insufficient for IoT devices such as smart speakers and smartphones as they exhibit interfaces and sensors not available on other computing systems that can be targets of a malicious entity. In the following sections, we briefly describe our work of exploiting the attack surface created by novel features such as virtual assistants and sensors like capacitive touchscreens.

Virtual Assistants

Virtual assistants are software applications that use artificial intelligence to perform tasks and provide services for users. These tasks can range from answering questions and setting reminders to controlling smart home devices and managing schedules. Usually, virtual assistants are embedded in smart speakers, devices equipped with a microphone, loudspeaker and internet connectivity. Therefore, they can interact with users through voice commands by interpreting natural language, allowing users to communicate with them conversationally. However, they can also be integrated into various devices such as smartphones, tablets or even cars.

Popular virtual assistants include Amazon's Alexa, with a market share of 70% in US households' smart speakers [76], Apple's Siri, Google Assistant and Microsoft's Cortana. The general communication sequence with a voice-based virtual assistant starts with the user uttering a wake-word like "Alexa" or "Ok, Google", after which the device starts recording the following command of the user and forwards it to the cloud for natural

language processing and interpretation. Either a "built-in" action can be started by this command or a third-party extension, so-called "Skill" or "Action". These extensions are designed to increase the assistant's capabilities by allowing the user to invoke them by saying the corresponding name of the extension.

However, such novel voice UIs are potentially less secure than traditional UIs [120, 98], highlighting the need to analyze the created attack surface. Existing attacks on voice UIs aim to inject commands [30, 2, 171, 12, 141, 184, 160, 185, 136, 69, 183] or exploit the interaction model by utilizing the way words are pronounced [80, 186].

With the publication *Alexa Lied to Me: Skill-based Man-in-the-Middle Attacks on Virtual Assistants* (Chapter 2), we propose a man-in-the-middle attack that can take over an entire dialogue between a voice-controlled virtual assistant and a user, without making him or her suspicious.

Capacitive Touchscreens

A capacitive touchscreen is a touch-sensitive input device layered on top of a display, commonly found in, e.g., smartphones, tablets and cars. It operates by detecting capacitance variations of a touch input caused by the electrical properties of the human body. More precisely, the finger changes the capacitance at the touched point by drawing its charge to the grounded finger. This change in capacitance is quantifiable by the touchscreen's sensing microcontroller at various points in the touchscreen grid to accurately determine the touch's precise location, which is finally forwarded to the Operating System (OS). The OS then translates the coordinates into actions like opening an app or typing on a keyboard. It therefore has no other option but to trust the input of the touchscreen, posing a new attack vector.

The property of being able to sense small changes in the electromagnetic fields makes the touchscreen very precise and reliable but, in turn, also susceptible to environmental Electromagnetic Interference (EMI) [36, 86]. Existing EMI-based attacks against capacitive touchscreens utilize the emanations radiated from screens to infer its content [50] or manipulate its sensing capability [101].

With the publication *GhostTouch: Targeted Attacks on Touchscreens without Physical Touch* (Chapter 4), we present our findings that smartphones equipped with a touch UI enabled by a capacitive touchscreen sensor can be controlled by an adversary using EMI without any galvanic connection, e.g., from a distance or through matter like a table or glass.

1.2 MITIGATING INFORMATION LEAKAGE OF IOT DEVICES

Identifying new exploits by analyzing the novel attack surfaces of IoT devices is the first step in preventing the information leakage of private information of the users of said devices. The second is to propose mitigations to address the identified vulnerabilities. In the following sections, we briefly describe our work on mitigating the attack surface created by novel features such as the wake-word detector of microphone-enabled virtual assistant devices and satellite internet base stations, which are susceptible to leaking location data.

Virtual Assistants

As described earlier, voice-based virtual assistant-enabled devices such as smart speakers or smartphones utilize a so-called wake-word or hot-word detector. This ML-based system records the environmental noise and analyzes the audio for the presence of that word. Only after the word is detected is the following recorded audio (i.e., the user's command) forwarded to the cloud for natural language processing. This is done, among other things, to protect the user's privacy by not sending every recording to the manufacturer's cloud. Unlike natural-language speech recognition models, wake-word detectors only discern a specific word from all other words. However, wake word detector suffer from false positives due to the nature of probabilistic ML-based approaches. This, in consequence, mistakenly sends audio recordings to the manufacturer's cloud, outside the user's home, without notification or consent, posing an information leakage risk [17, 105, 171].

Indeed, reports show 50% of users experience unintended activation of their devices once a week [20, 144], leading to a whopping 15.3% of commands being recorded by accident [172]. To counter this, manufacturers review commands recorded by accident to increase the device's precision in detection and recognition. Unfortunately, this includes humans to listen to these recordings, being made without the user's consent [24, 172, 111, 51, 19, 43], including identifiable personal details in the recordings [172, 40].

With the publication *Leakypick: Iot Audio Spy Detector* (Chapter 3), we propose an approach to detect smart home devices that stream audio over the home network in response to voice or noise and to detect wrongly recognized wake-words which, when spoken by a human or played by, e.g., a TV, could trigger unintended information leakage to the manufacturer's cloud.

With the publication *FakeWake: Understanding and Mitigating Fake Wake-up Words of Voice Assistants*, we propose a three-fold approach to generating and evaluating wrongly accepted wake-words, identifying features contributing to the wrong acceptance and using the insights gained to mitigate this phenomenon in detector models.

Satellite Internet Base Stations

Another type of information leakage comes in the form of meta-data, e.g., the user's geographic location. Unfortunately, the versatile new satellite internet technology is susceptible to this type of leakage by triangulation. Satellite internet provides an internet connection to its users by utilizing satellites to relay data. It works by transmitting data between a user's base station and a satellite orbiting the earth, relaying the data to a ground station connected to the internet and vice versa. This technology enables internet access in remote or rural areas where traditional broadband services are unavailable.

The escalation of the Russo-Ukrainian conflict showed that citizens rely on satellite internet in conflict zones, where traditional terrestrial networks may be impaired. However, users are at risk of geolocation by triangulation of the satellite terminal signals being sent into the sky like a beacon [145], which even forced the CEO of Starlink to issue a warning to all Starlink users, as they may become targets [5].

Proposed emergency networks utilizing satellites do not protect against location privacy leakage [193, 59, 125]. Similarly, location privacy approaches are not applicable to users utilizing satellite internet [67, 177, 148, 88, 53, 68, 133, 32, 38, 39, 182, 104, 119, 70, 174, 150, 189, 162, 106, 95].

With the publication *Don't Shoot the Messenger: Localization Prevention of Satellite Internet Users* (Chapter 5), we propose an architecture that enables satellite internet users to hide their geographic position in case the origin of their satellite connection is being triangulated. This is achieved by spanning a network between base stations where the user's connection is established over a random path in the network to a distant random output gateway.

2

Attacking Virtual Assistants' Interaction Model

As outlined in the introduction, the first step towards protecting Internet of Things (IoT) devices from information leakage attacks is the analysis of the device's attack vectors. In order to pave the way for mitigations we must therefore first investigate possible exploits of the attack surface created by introducing novel features. Indeed, Li *et al.* [87] identified six novel and specific attack vectors on the Amazon Alexa virtual assistant system, ranging from the voice interface over the device's network transmissions to the device's backend in the cloud.

In this section, we propose a new attack vector on personal voice-controlled virtual assistants incorporated in, e.g., smart speakers. In a smart home scenario a user utilizes the voice User Interface (UI) to command the assistant in the smart home device. We therefore combined exploits for multiple attack vectors to create an attack of high impact, entirely hijacking the communication between the benign user and the assistant. We achieve this by exploiting the virtual assistant's interaction model on the software side and command injection and voice jamming on the hardware side. Virtual assistants, to be helpful, have access to the user's private data (e.g., calendar, home location) and are able to execute tasks on the user's behalf (e.g., controlling the heating system, locking a door) inside their ecosystem of connected devices. Unfortunately, this makes them prone to leak sensitive information or even allow the adversary to feed malicious information to the user through the assistant.

The main way to interact with such a virtual assistant is therefore by using the novel voice interface of smart speakers or smartphones. As the voice UI is currently being actively developed and deployed, it is potentially less secure and therefore increasing the attack surface of such devices [98]. In comparison, traditional UIs have been used and continuously improved over the past decades [120] .

Naturally, a new research area of attacks on voice-controlled UIs or devices hosting virtual assistants emerged. Existing voice-based attacks aimed at this interface have the goal to inject commands as stealthily as possible, utilizing synthesized [30, 2], obfuscated [171, 12], embedded [141, 184] or ultrasound-modulated [160, 185, 136] voice commands. Other attacks target the device hosting the virtual assistant by utilizing electromagnetic interference [69] or attacks leveraging the headphone jack of a device [183]. Such attacks are, however, limited in their applicability in practical smart home settings, as they

require a physical connection to the headphone jack of the victim device or relatively expensive special hardware.

In addition, modern virtual assistants have the option to extend their functionality by so-called 'Skills' or 'Actions', which can be provided by a third party. With these, the virtual assistant can, e.g., control third-party devices from this manufacturer's device ecosystem or gain access to new sources of information. By enabling the use of such third-party-provided applications, the attack surface of the virtual assistant is increased [87]. Misusing such a Skill by an adversary can, therefore, lead to information leakage or even losing control of the user's connected devices, such as his smart home devices. Recent works exploiting this interaction model make use of the way words are pronounced to, e.g., confuse the voice recognition on which Skill the user intended to invoke, in order to start a malicious Skill [80, 186].

However, there is no related work combining both approaches into a unified new attack. To be able to do so, we had to circumvent the extremely limited interaction model of Skills, by also controlling the device's audio vicinity, as Skills can only converse with the user and simultaneously communicate with third-party services that are accessible to it.

2.1 CONTRIBUTIONS

This thesis presents an attack on the virtual assistant's interaction model with the following publication, which can be found in Appendix A.

[107] Richard Mitev, Markus Miettinen, and Ahmad-Reza Sadeghi. Alexa lied to me: Skill-based man-in-the-middle attacks on virtual assistants. In *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security*, pages 465–478, 2019. CORE Rank A. Appendix A.

This paper presents our findings about how an adversary can craft a malicious virtual assistant Skill, including its backend, which can bypass the virtual assistant's limited Skill interaction model. It utilizes the malicious Skill in combination with inaudible voice injection and jamming methods. We demonstrate a man-in-the-middle attack that can take over an entire dialogue between a voice-controlled virtual assistant and a user without making him or her suspicious. This technique is particularly insidious because it operates during active user engagement with the virtual assistant, preserving the regular interaction flow from the user's point of view. This enables the attacker to execute more complex attacks than the basic command injections previously documented.

With our Skill-based approach, we are able to position the adversary between the user and the virtual assistant. In this scenario, the attacker has the capability to change or completely override the virtual assistant Skill's responses with their own, making it appear as if the responses are coming from the actual benign Skill. Our attack is able to alter legitimate data from benign Skills in real-time, making the responses seem credible. This is particularly impactful in scenarios where the information provided is used to make decisions (e.g., leaving the house after the virtual assistant said it locked the door). Our Proof of Concept (PoC) highlights the necessity for voice application developers within Skill ecosystems to consider the risks of inaudible signal injections, jamming and malicious Skills as potential attack vectors.

Design

AlexaLiedtoMe is able to serve to the user modified information from the benign Skill or totally fabricated information based on the Skill the victim wanted to invoke. In Figure 1, our attack flow is depicted.

In Step ❶, the user's command c is inaudibly jammed [136] and recorded simultaneously ❷ by the attacker. In Step ❸, the attacker then injects inaudibly the Skill's invocation command i [185], which is under his control. To prepare for Step ❹, the attacker has to analyze the recorded benign command c of the victim and decide which action the malicious Skill should commence (e.g., telling the user wrong information or modifying benign information from the intended Skill invoked by the attacker). He forwards the command c to his Skill's backend, which will then get activated by the virtual assistant's backend ❺, acting accordingly in regard to the passed benign command c and forwarding a textual response to the virtual assistant's backend ❻. The backend then converts this text using TTS ❼ and forwards it to the device, which plays the malicious response in the benign voice ❽.

Evaluation and Results

We implemented and evaluated two PoCs of AlexaLiedtoMe, a technically advanced ultrasonic-enabled evaluation setup and a low-tech real-world setup without ultrasonic capabilities. The powerful setup consist of a tweeter (i.e., ultrasound capable) speaker and a consumer hi-fi amplifier. The real-world setup is based on non-ultrasonic IoT hardware with a 2W and 3W speaker and a small amplifier board. Both are evaluated on the Amazon Echo Dot smart speaker, which incorporates the Alexa virtual assistant. We then evaluated the capability of our powerful device to reliably jam, record and invoke by injection the ten top Skills of the Alexa Skill store. Over all Skills and commands the

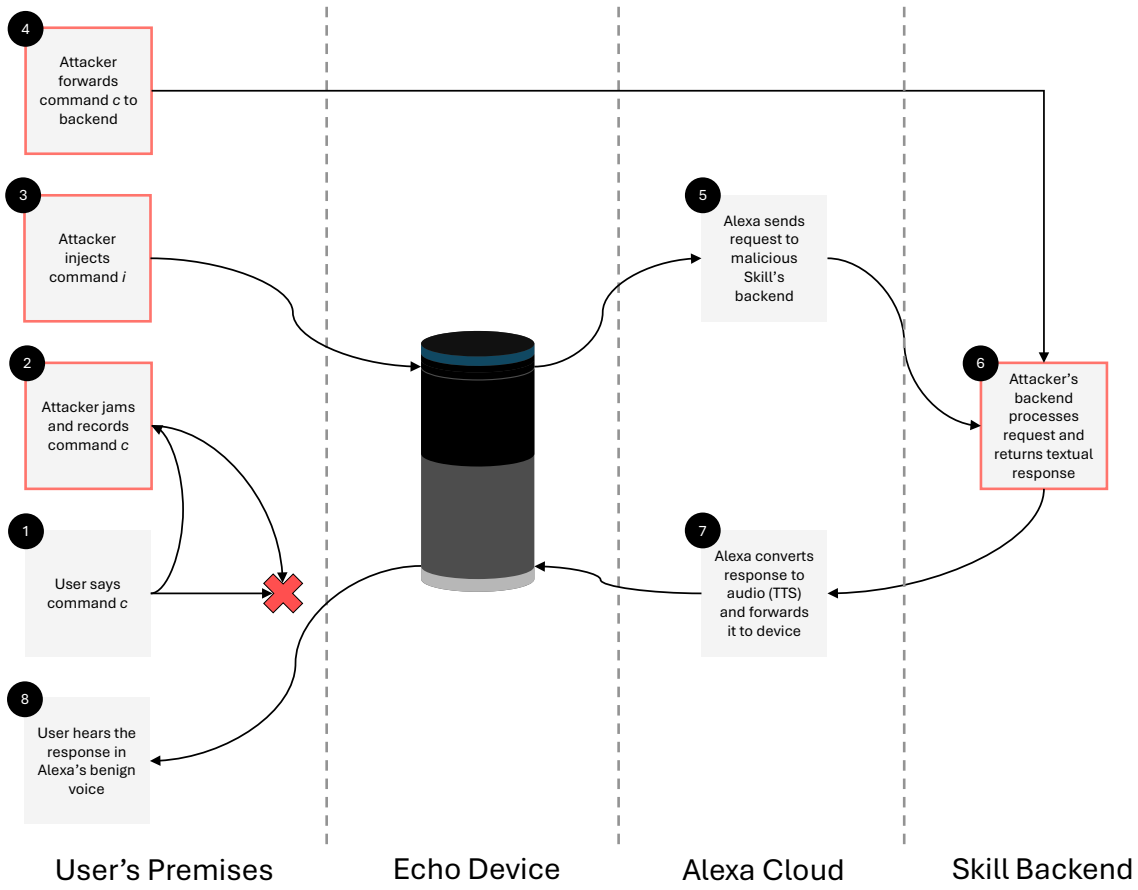


Figure 1: High-level overview of the AlexaLiedtoMe attack

probability of successfully jam, record and invoke were 95, 3%, 96, 8% and 84%. Following this, we implemented two PoC attacks on the device's ecosystem by attacking a security system and a smart lock. In both scenarios, the goal of the attacker was to prevent the victim from arming his security system or locking his door lock while at the same time being told by his benign device that the security system or door are configured as expected exhibiting an overall success rate of 75%. The third attack targeted a stock exchange Skill, which the user asked for a stock price. Our attack would perform a Man-in-the-Middle (MitM) attack by recognizing the user's intended action, querying the benign Skill in the background for the correct stock price and returning a maliciously manufactured response based on this price to trick the victim.

2.2 RELATED WORK

Related work has mainly aimed at issuing unauthorized commands to the victim's virtual assistant. However, works aimed at exploiting the assistant's interaction model have also been published. We therefore report the related work accordingly.

Command Injection

Command injection attacks typically aim to covertly inject an audio command into the victim's virtual assistant device's microphone. This can be done using synthesized, mangled, inaudible or hidden audio [9].

Audible Synthesized Voice Commands. The following attacks aim to gain unauthorized access to systems by using generated plain speech to mimic legitimate user voice commands. These attacks are, therefore, easily detectable by humans.

Diao *et al.* [30] propose voice injection attacks by using a malicious app against the Google Voice Search (GVS) app on Android smartphones. The idea of the attack is that the malicious app activates GVS and simultaneously plays audio commands over the built-in speakers. Alepis *et al.* [2] build on top of this work by proposing to infect multiple devices to issue commands to other listening devices like Amazon Echo or smartphones.

From these papers, we utilize the finding that synthesized audio can control virtual assistants and that the (synthesized speech) answer can be reliably transcribed back into text, allowing for an entire conversation between two devices.

Audible Mangled Voice Commands. More sophisticated approaches utilize mangled voice, which is recognized by a human as potential nonsense and is therefore more covert in their nature.

Vaidya *et al.* [171] propose a method for black box (i.e., without having knowledge of the inner workings) voice recognition systems. The goal is to mangle human voice by changing the audio but keeping the MFCC parameters (which represent the frequency spectrum of the audio) close to the original so that the audio is no longer comprehensible by humans but still recognizable by the voice recognition system. Carlini *et al.* [12] extended this work by proposing a similar attack with the insight a white box voice recognition system provides. By knowing the inner workings of the voice recognition

system, and therefore being able to tweak the attack to match the system, the success rate of the resulting attack is naturally higher.

Inaudible Voice Commands. The following attacks are inaudible and unrecognizable to the average human, either by not using audio at all or by using ultrasonic audio which is inaudible to humans. This way, the victim is not being made aware of an attack being conducted.

Kune *et al.* [82] show that it is possible to use electromagnetic waves to inject audio into microphone circuits. This was later refined and modeled by the work of Liu *et al.* [92]. Similarly, Sugawara *et al.* [161] utilize lasers to inject voice commands into smart speakers. Song *et al.* [160] propose exploiting the non-linear frequency-response of microphones of voice-controlled virtual assistants. The non-linearity at a specific frequency demodulates amplitude-modulated signals into the non-modulated signal, harmonics and cross-products. The low-pass filter then removes every signal except the demodulated audio. This effectively injects recorded voice in the expected frequency range without alerting humans in the vicinity as they cannot hear the ultrasonic audio.

Zhang *et al.* [185] evaluated the susceptibility of ultrasonic voice injection into Micro-Electro-Mechanical System (MEMS) and Electret Condenser (ECM) microphones of 16 devices such as smart speakers and smartphones. In addition to that, they proposed two ways to bypass devices with personalized voice by concatenating recorded phonemes into sentences. Xia *et al.* [178] showed that a similar attack is also possible using near-ultrasound played through the speaker of a victim device. An attack where the surface of a table is used to convey the ultrasonic audio containing the command is proposed by Yan *et al.* [180]. Instead of injecting voice into microphones, Roy *et al.* [135] describe an approach for injecting jamming noise into microphones using ultrasound signals.

We have utilized and extended the injection and jamming approaches described by these papers. By combining these attacks with the generated voice, we were not only able to jam the original command of the user but also inject a command to start our own Skill. However, we were also able to trick the user into believing he is speaking to the desired Skill, which allows for different types of interactive attacks apart from merely injecting unauthorized commands.

Hiding Command Injection Attacks in Audio Files. Hiding a command in an audio file can be achieved in two ways. Either by changing an audio file so that an Automated Speech Recognition (ASR) model detects a targeted result while a human still recognizes the audio contained in the file as benign, or, by injecting specific features the ASR interprets into the targeted result, into a totally different audio file. The resulting audio file is then called an adversarial example against ASR systems.

Iter *et al.* [61] showed that the Mel Frequency Cepstral Coefficients (MFCC) of an audio file can be extracted, changed and then converted back to audio to change the interpreted content of the audio by an ASR. Kreuk *et al.* [79] proposed a similar approach with a higher success rate by incorporating knowledge of the target ASM. Yuan *et al.* [184] describe a method for hiding voice commands automatically in audio files containing songs, which are detected by white-box and black-box deep neural network-based voice recognition systems but recognized by humans. Carlini *et al.* [11] showed how to create a copy of an audio file of voice that exhibits a similar waveform but transcribes differently when processed by a speech recognition system. Schönherr *et al.* [141] propose a way to create malicious perturbations in audio files that modify the recognized output by the DNN-based speech recognition system to a target output. This is achieved by adhering to a psychoacoustic model to perturb the signal below the human perception level. Liu *et al.* [93] proposed using the Audio Sequence Location (ASL) to find better perturbations while also using Total Variation Denoising (TVD) to increase the adversarial example's success rate, while the overall generation time of the adversarial example being faster than related work.

Adversarial Examples Over the Air. Most adversarial examples are generated to be fed directly into the ASR system. However, in the case of voice-enabled virtual assistants, such as the ones incorporated in smart speakers, the audio needs to be played back by a speaker and recorded by the smart speaker's microphone before being processed by the ASR system. Therefore, some adversarial examples can not be played back through speakers as they contain high or low frequencies exceeding the frequency response range of the speaker [12].

Hence, Abdullah *et al.* [1] focus on mangling voice in a way that it is still reproducible through speakers and able to being picked up by microphones. Qin *et al.* [131] propose using the psychoacoustic auditory masking principle to generate full-sentence adversarial examples that are playable over the air in simulations. Schönherr *et al.* [142] propose on the other hand embedding voice in an audio file and using the room impulse response to generate adversarial examples that can be played in arbitrary rooms.

Attacks on the Interaction Model of Skills

In AlexaLiedtoMe, we showed how to exploit the seemingly limited interaction model of Skills. In the following, we discuss other approaches presented in related work, namely skill squatting, priority override or malicious behavior change.

The attack of Kumar *et al.* [80] is based on the Alexa voice recognition system misinterpreting words. These errors are exploited by the attacker naming malicious Skills so that the speech recognition system may confuse them with benign ones. On the other hand,

Zhang *et al.* [186] introduced an attack, that by the attacker naming Skills with a similarly pronounced invocation name, may confuse the ASR. Zhang *et al.* [187] built on top of that and expanded their attack's utility by utilizing similarly pronounced Skill names, paraphrased names and voice masquerading to attack benign Skills. Snodgrass *et al.* [159] showed how a malicious Skill could exploit the internal Skill invocation priority of the voice assistant to override a benign Skill the user wanted to activate with th einvocation of a malicious one. Le *et al.* [84] show how a Skill's behavior can be maliciously changed after the Skill is approved and propose a Skill monitoring approach.

3

Defending Voice Assistants' Wake-Word Detection

In the preceding chapter, the focus lies on the attack surface created by the voice-based virtual assistant interaction model in combination with the physical properties of the microphone itself. However, even without the presence of an attacker, information leakage through the use of voice User Interfaces (UIs) can occur. Devices such as smart speakers send recorded commands to the cloud for more accurate speech recognition. However, sending voice recordings to the cloud for processing without the user's intention nor her or him noticing is potentially a severe leak of information. To enable virtual assistants to only send data to the cloud for speech recognition, after the device identified a command to be recognized, they are usually equipped with a preposed offline or hybrid (i.e., using online only if offline has a low confidence) wake-word detector. However, due to the nature of probabilistic ML being utilized for detecting specific words in the audio signal, devices can be "woken up" by similar-sounding words or even noise. This is emphasized by surveys showing 50% of users reporting unintended activation of their devices once a week and 28.5% once a day [20, 144].

Unfortunately, this unintended leakage of information to the manufacturer's cloud can be a privacy concern as virtual assistant manufacturers like Amazon [24], Google [172, 111], Apple [51], Microsoft [19] and Facebook [43] are all using third parties to increase the accuracy of voice transcription, by listening to intended as well as unintended recordings of these virtual assistants. Unfortunately, it is being reported that from 1000 Google Assistant recordings 153 (15.3%) were recorded by accident [172].

Even though the user's name is substituted by the voice assistant provider for a pseudonym [172], names, addresses and other personal details mentioned in the recording are not [40]. Contractors and journalists could find out the user's identity intentionally or unintentionally [172]. The recorded audio snippets range from talking about medical conditions over having a professional phone call to having sex or an altercation.

In addition, unintended waking of a voice-based virtual assistant can lead to information leakage in other ways, such as unintended calls being placed or voice messages being sent [10].

Earlier works have shown that background noise in a home environment can activate the wake-word detector of virtual assistant devices [60]. However, there have not been

any works using anomaly detection on network traffic to detect devices sending audio recordings out of the user's network [158, 113, 48, 126, 63].

Google stated that using humans to transcribe audio recordings is a crucial part of increasing the utility of their services; they will only temporarily hold manual transcription and continue after three months' time [35]. It is therefore crucial that the wake-word detector is as precise as possible to minimize audio snippets being sent to third parties. In this chapter, we will therefore research into two areas related to the wake-word detection mechanisms of virtual assistant-enabled devices: 1) detecting devices that react to specific audio by audio-fuzzing wake-word detectors to protect users from devices being activated without their knowledge, and 2) an ML-based approach for explaining the occurrence of these wrongly detected words as well as approaches for hardening wake-word detector models.

3.1 CONTRIBUTIONS

In the following, we present two contributions towards protecting information leakage through the virtual assistant's wake-word detector.

Detecting Devices Reacting to Audio by Fuzzing Wake-Word Detectors

This thesis addresses this problem by presenting an approach to detect devices reacting to audio by fuzzing their wake-word detectors in the following publication, which can be found in Appendix B.

[108] Richard Mitev, Anna Pazii, Markus Miettinen, William Enck, and Ahmad-Reza Sadeghi. Leakypick: Iot audio spy detector. In *Proceedings of the 36th Annual Computer Security Applications Conference*, pages 694–705, 2020. CORE Rank A. Appendix B.

This paper presents LeakyPick, an approach to detect smart home devices that stream audio over the home network in response to voice or noise. We implemented a PoC device that utilizes audio probes while statistically monitoring the network traffic for patterns that indicate an audio transmission. In addition, our device is able to conduct audio-fuzzing of virtual assistants to detect wrongly recognized wake-words which, when spoken by a human or played by, e.g., a TV, could trigger unintended information leakage to the voice assistant manufacturer's cloud.

Design

The functionality of LeakyPick is twofold: it identifies microphone-enabled devices transmitting recordings (❶- ❺) as well as identifying devices doing so unexpectedly in response to voice or noise (❻- ❼). In Figure 2 our functional overview is depicted.

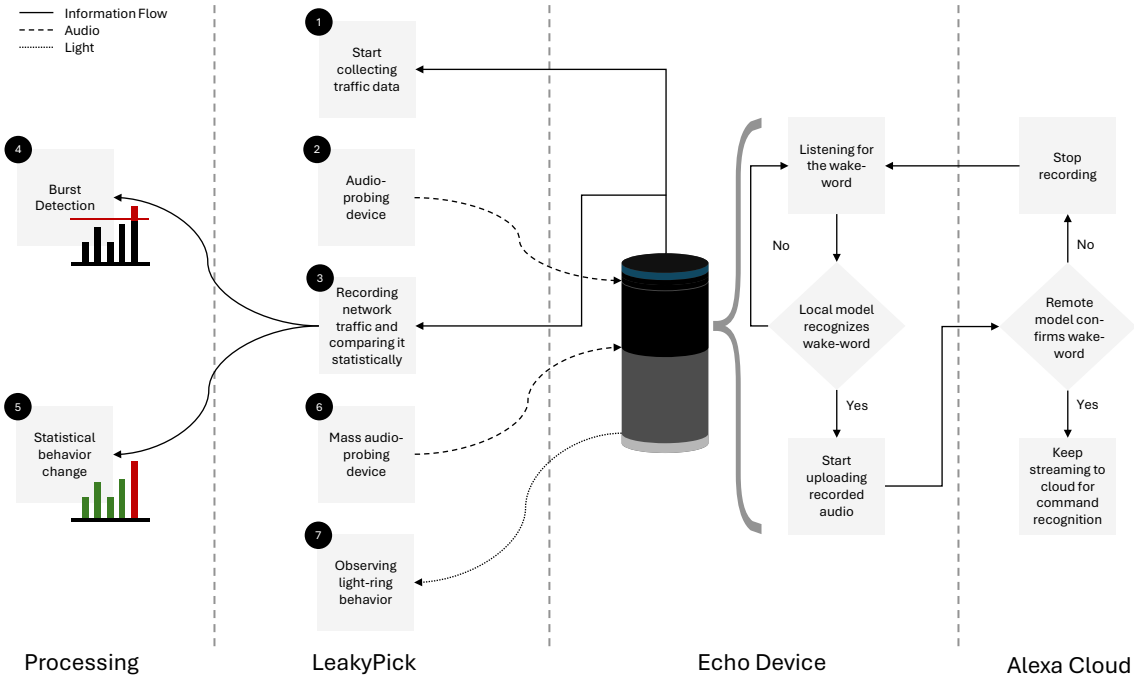


Figure 2: High-level overview of the LeakyPick architecture and Alexa wake-word behavior

The device is deployed in the user's smart home and in the audible distance of other smart-home devices. In Step ❶ LeakyPick, starts collecting traffic data (and extracting statistical features) of each device in question. Then, in Step ❷, it starts probing devices in its vicinity with generated or recorded audio snippets. Traffic recorded after probing ❸ is then statistically evaluated for abnormalities, compared to the data recorded in Step ❶, either by a burst detection algorithm ❹ and/or a binned distribution test using t-test on the histogram of inter-arrival time and packet size values, generated utilizing the Sturges and Freedman Diaconis Estimator ❺. If a device changes its network behavior, the user is immediately notified.

In addition, it may be helpful to know upon which audio cue devices react. For this, the device can use generated or previously recorded audio to fuzz (i.e., probing with a large set of arbitrary audio) the wake-word detector ❻. The device is therefore equipped with a light sensor, which is aimed at the top of the Device under Test (DuT), as devices like Amazon Echo are indicating the detection by activating its light. Typically, devices are equipped with a coarse-grained offline model to detect voices resembling the wake-word. If this model recognizes a word, it is then forwarded to the more fine-grained online

model for a final judgment. By using the light signal feedback in Step ❼ LeakyPick can discern between both by measuring the time the light stays on, as the recording will immediately stop if the offline model deems the voice not to be the wake-word.

Evaluation and Results

The LeakyPick approach was evaluated on three smart speakers incorporating virtual assistants from Amazon, Google and Apple, as well as an audio-based security system and four microphone-enabled Internet of Things (IoT) devices. We additionally conducted a real-world evaluation by collecting a 52-day dataset from the three smart speakers deployed inside a residential environment. The burst detector exhibits a True Positive Rate (TPR) of 96% and a False Positive Rate (FPR) of 4%. The the statistical test a TPR of 94% and a FPR of 6%. Both configured to use a non-overlapping rolling window size of 30*s*. In addition to being device agnostic, our statistical approach outperforms the best ML classifier, namely Random Forest, which showed, on average, a 91 F1-score in contrast to the 93 F1-score of our approach. We conducted an audio-fuzzing test to identify wrongly accepted wake-words using words from the English dictionary with a phoneme count similar to the original word (and some randomly sampled ones) spoken by a Text to Speech (TTS) engine. Our device identified 89 words that reliably activated the Alexa assistant, with words showing a phonetic distance (i.e., the Levenshtein distance between Metaphone converted words) of up to 9.

Explaining Wrong Wake-Word Recognition and Hardening Virtual Assistant's Detectors

We proposed an approach for identifying devices reacting to audio cues and identifying the audio they react to by utilizing audio-fuzzing. However, this does not explain or mitigate this phenomenon. Therefore, this thesis addresses this problem by presenting an approach to efficiently generate wrongly detected wake-words, identify features contributing to the false detection and propose strengthening the underlying ML models to increase their resiliency against this phenomenon in the following publication, which can be found in Appendix C.

[14] Yanjiao Chen, Yijie Bai, Richard Mitev, Kaibo Wang, Ahmad-Reza Sadeghi, and Wenyuan Xu. Fakewake: Understanding and mitigating fake wake-up words of voice assistants. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 1861–1883, 2021. CORE Rank A*. Appendix C.

This paper presents FakeWake, a three-fold approach to 1) generating and evaluating efficiently large amounts of wrongly accepted wake-words for popular English and Chinese

virtual assistants, 2) using the identified words to explain which features contribute to the wrong acceptance and finally 3) using the insights gained to mitigate this phenomenon in detector models by screening for these features or retraining the model utilizing these words.

Design

The FakeWake approach is three-fold; it allows for the generation and evaluation of wrongly accepted wake-words, quantization and identification of the features that play a role in the wrong acceptance and re-training of conventional models using the identified word dataset. In Figure 3, the design overview is depicted on a high level.

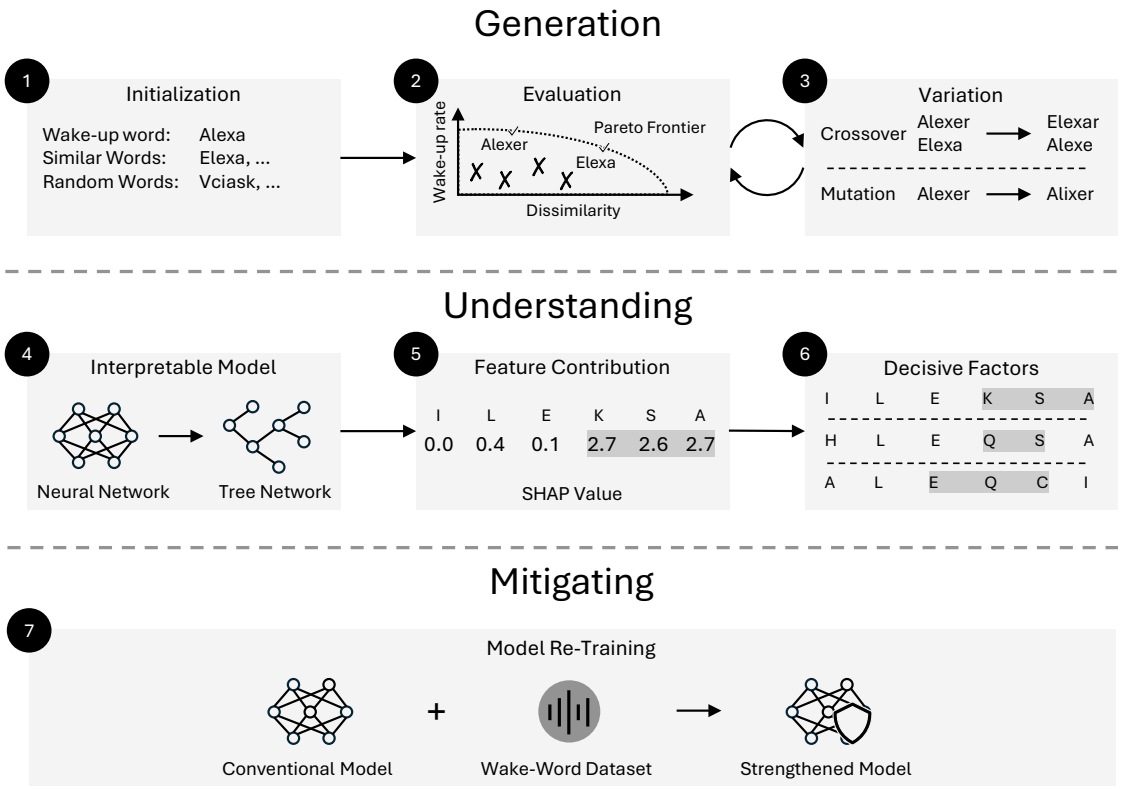


Figure 3: High-level overview of the three-fold FakeWake approach

The first step of our approach is to generate wrongly accepted wake-words that are as dissimilar as possible to the benign wake-word. For this, we utilize genetic algorithms and treat the words as a population and the letters (or pinyin) as chromosomes. We, therefore, start with an initial population of the wake-word itself, similar words and randomly generated words in Step ❶. To not evolve the population towards the benign wake-word, we sort the individuals by utilizing the Pareto Frontier [26] approach (i.e., selecting words on the pareto curve) on the wake-up rate (i.e., how often a word triggers

the wake-word detector) versus the dissimilarity to the benign wake-word ❷. Finally, we evolve the population by using crossover (combining parts of different survivors), recombination (rearranging parts of one survivor) and mutation (random alterations) on the survivors ❸ and re-evaluating them until the population is sufficiently evolved.

The next step is to understand what features of a word contribute to it being wrongly classified as a wake-word. As we have no access to the wake-word detection model, we train a gradient boosting classifier [42], an ensemble of decision trees. To mimick the behavior of the wake-word detector we use the misinterpreted generated words as positive and the generated but not misinterpreted words as negative samples, in Step ❹. This classifier interpretability allows to utilize the contribution (i.e., coefficients) of each feature (e.g., phoneme) to the prediction result of a word in order to calculate the Shapley Additive Explanation (SHAP) values [94]. The SHAP value shows the importance of each feature to the prediction but also accounts for the local importance of the features and un-scales the coefficients used. We will then calculate the values for each feature of a word in Step ❺ and construct a set of decisive factors, which are the features with a high positive contribution value, for each phoneme (i.e., for each initial and final, from the Pinyin representation in Chinese) of the word in Step ❻.

Finally, our insight into wrongly accepted wake-words can be used to mitigate this phenomenon. A straightforward approach would be to detect if a word contains substantial decisive factors and then send the audio recording to a more precise cloud-based detector. Step ❼ shows another approach, which would be to re-train a conventional model, trained on actual spoken words, with our generated wrongly accepted wake-words as additional negative samples to strengthen the decision boundary of the model.

Evaluation

We evaluate our generator with audio-fuzzing the virtual assistant's wake-word detector by the generated speech of the generated candidate words to calculate the wake-up rate used in the Pareto Frontier calculation. Each word is played ten times. We evaluated our approach on Amazon Echo, Amazon Echo Dot, Google Nest Mini, Apple HomePod, Baidu, Xiaomi, AliGenie and Tencent devices. We generated a total of 965 wrongly accepted wake-words summed up over all devices.

Analysis of the decisive factors for each wake-word showed that these features only concentrate on a small part of the words. This means it is possible to generate vastly different words by keeping only the small part, including the decisive factors. In fact, more than 80% of wrongly detected wake-words contain at least one of the three most present decisive factors, e.g., *ks* for Alexa so that even words like *ilexbs* are reliably accepted wake-words.

To evaluate the model re-training, we utilized Mycroft Precise [46]. We trained the model on a conventional dataset and re-trained it on our generated wake-word set. Training on the conventional model alone leaves the model up to a 17.67% acceptance rate of wrong wake-words (i.e., in the case of Alexa). After re-training, this rate drops to 1.18% or lower, while also slightly increasing the overall accuracy of the model, showing that this approach is a promising mitigation against wrongly detected wake-words.

3.2 RELATED WORK

In the following section, we present related work regarding LeakyPick and FakeWake. For LeakyPick we discuss approaches aimed at detecting IoT devices and network traffic analysis-based approaches for IoT device anomaly detection. For FakeWake we discuss related work regarding the shortcomings of wake-word detectors as well as literature proposing solutions to increase their precision. In addition, we detail the position of our work in the related literature.

Detecting Devices Reacting to Audio by Fuzzing Wake-Word Detectors

LeakyPick utilizes a custom approach to traffic analysis to detect if a device reacts to sound. However, related work uses traffic analysis in the realm of IoT devices primarily to identify or detect devices or detect anomalies in the network.

IoT Device Detection. IoT device detection mechanisms, such as for IP cameras, exploit the fact that cameras stream a constant recording over the network. The idea is then to change its stream (e.g., by changing the scene the camera records) and capture the changes in network traffic to identify the device.

Cheng *et al.* [15] propose that the person being allegedly monitored can alter the camera's captured scene by physically moving to modify the camera's field of view. These traffic modifications are then detected using ML. Similarly, Liu *et al.* [91] propose altering the light conditions of the monitored scene to change the stream. If the stream also changes its bitrate, these changes can even be detected by a statistical t-test. Salman *et al.* [138] removed the need to purposefully change the environment by correlating physical movement recorded by a smartphone with changes in stream bitrate using ML. Similarly, Singh *et al.* [157] use an extra device to correlate physical movement. In addition, they extended their work to use audio-probing to detect voice-activated devices and locate them using ML. Sharma *et al.* [152] propose an ML-based approach to

detecting and localizing various IoT devices, including smart speakers and IP cameras, by accessing only MAC layer data.

All of these works rely on changing the environment a device captures, requiring a-priori training or asking the user to participate in device detection actively. In addition, none of these works is able to detect which wake-words a microphone-enabled device reacts to.

Network Traffic Analysis. Network traffic-based techniques have been proposed to learn the behavior of IoT devices, distinguishing and identifying IoT devices based on their traffic profile.

Sivanathan *et al.* [158] use pre-trained supervised ML approaches to characterize network traffic corresponding to various IoT devices. Nguyen *et al.* [113] propose an ML-based self-learning system for detecting compromised IoT devices based on changes in their communication behavior. Hafeez *et al.* [48] propose a lightweight network intrusion detection approach for IoT devices, which can restrict network access for devices acting maliciously. Perdisci *et al.* [126] show that it is possible to passively detect and identify IoT devices on the internet by correlating ISP's DNS data to the device's DNS fingerprints. Jaafar *et al.* [63] extended this approach by combining traffic analysis and analysis of power consumption of IoT devices to detect compromised devices more precisely.

The contributions of these proposals are two-fold: identifying benign devices and detecting deviations of benign behavior. In contrast, LeakyPick detects smart speakers by their benign actions in response to audio events.

Explaining Wrong Wake-Word Recognition and Hardening Virtual Assistant's Detectors

FakeWake aims to protect against unintended activations of the virtual assistant. However, related work either describes the phenomenon or proposes extra features for wake-word detectors to protect against this phenomenon.

Shortcomings of Wake-Word Detectors. In the past, multiple works were published reporting that voice-activated virtual assistants such as Amazon Alexa [17] and Google Assistant [105, 171] reacting to words they are not designed to, are an information leakage hazard. In addition to words being wrongly accepted as wake-words, Islam *et al.* [60] showed that background noise (e.g., coughing, crying, sound from appliances) can exhibit the same features as the wake-word and activate virtual assistant devices.

Schönherr *et al.* [143] discovered wrongly accepted wake-words by playing an extensive collection of TV shows, news and other audio material to smart speakers. Similarly,

Dubois *et al.* [31] played 134*h* of English-speaking TV shows to virtual assistants and recorded misactivations by audio parts of the shows.

Related work relies on an audio corpus played to the devices under test to identify wrongly accepted wake-words. We solve this limitation by automatically generating and evaluating candidate words. A straight forward way to calculate the distance between words is to use the Levenshtein distance [143]. To be able to measure the distance between candidate words more precise, we propose a novel metric.

Mitigations of Adversarial Examples of Automated Speech Recognition (ASR). Adversarial examples (i.e., input crafted to manipulate the classification result) against systems understanding natural language are conceptionally similar to wrongly recognized wake-words, which can be considered as adversarial examples against the wake-word detector. However, there is no work specifically aimed at creating or defending against adversarial examples against such detectors. We, therefore, discuss defenses regarding adversarial learning-based approaches against natural ASR. See Section 2.2 for adversarial examples considering voice-based assistants.

Sun *et al.* [163] proposed generating adversarial examples and adding them to the training set of an ASR system to decrease the Word Error Rate (WER). If the perturbation introduced by the adversarial example is small, methods aimed at removing this small perturbation are applicable. Latif *et al.* [83] proposed to train the ASR model using adversarial examples, noise or a trained Generative Adversarial Network (GAN) tasked to clean incoming data to protect the model from adversarial examples. Esmaeilpour *et al.* [37] propose a Discrete Wavelet Transform (DWT) approach to preprocessing audio before the ASR processes it. Tamura *et al.* [165] propose to forward input speech to the ASR system while also feeding a dynamically downsampled and denoised version to the ASR system and comparing its output, likely detecting adversarial examples if transcriptions differ.

Hardening Wake-Word Detectors. Related work proposes a diverse set of solutions to this problem by presenting approaches enhancing the wake-word detector with extra features.

Kepuska *et al.* [72] suggests a system to distinguish between wake-words spoken in the context of a sentence or stand-alone in an alerting context with the intention to enable the voice UI. Amazon designed such a system in the context of follow-up queries [97]. Similarly, Zhang *et al.* [188] incorporated the prosody of a sentence to detect directed or non-directed usage of the wake-word. Michaely *et al.* [105] proposed to also utilize a cloud-based wake-word detector after the offline wake-word detector accepted a word as a wake-word, which is now standard for devices like Amazon Echo. Sigtia *et al.* [155]

propose the analysis of the words followed by the wake-word to identify wrongly recognized wake-ups.

Approaches aimed at detecting wake-words in a sentence or stand-alone need new speech detection mechanisms and are, therefore, incompatible with legacy systems. In addition, an additional cloud-based detection may minimize wrongly detected wake-words but does not mitigate them in a satisfactory way. We overcome these shortcomings by showing that re-training the detector model with generated wake-words strengthens the decision boundary and reduces wrong wake-word acceptance tremendously, similar to what Sun *et al.* [163] proposed for ASR systems.

4

Attacking IoT Devices through Capacitive Touchscreens

In Chapter 2, we analyzed the attack surface of the voice User Interface (UI) and proposed a novel exploit leading to information leakage attacks. However, there also exists a more popular and more mature UI in Internet of Things (IoT) devices, namely the touchscreen. It is, therefore only natural to also analyze the attack vector created by this technology. However, due to the popularity and maturity of this technology, a sophisticated exploit is required.

In this section, we propose a new attack vector on the touch UI of smartphone devices utilizing a capacitive touchscreen. Specifically, we utilize Electromagnetic Interference (EMI) to inject precisely controllable fake touch-points or swipes into a touchscreen without any physical contact and therefore enable an attacker to manipulate the underlying device's software that relies on dependable data from the touchscreen sensor. We achieve this by interfering with the beginning of the touch-sensing pipeline of a smartphone device that finally propagates the wrong information to the software (e.g., Operating System (OS) or application). Naturally, a personal device such as a smartphone contains a plethora of personal information and the ability to process privacy and security critical tasks (e.g., online banking or smart home control), making it a prime target for information leakage attacks.

Touch-enabled UIs have been trending in the last decade, specifically because of the popularity of tablets and smartphones, but also laptops, displays, vehicles and medical devices. Currently, there are an estimated 6.84 billion smartphones worldwide, of which most use a type of touchscreen as their primary human-machine interface [56]. Over all devices using a form of touchscreen, the capacitive touchscreen has a market share of around 90%, making it by far the most popular type of touchscreen, e.g., due to its high precision, low wear and multi-touch support [129]. Precise sensing of the user's gestures is crucial for reliable device control. However, capacitive touchscreens are susceptible to environmental EMI [36], such as the EMI emitted by fluorescent light [86]. Therefore, modern touchscreens go through thorough Electromagnetic Compatibility (EMC) tests [103] and exhibit anti-interference design such as shielding [34], frequency hopping [41] and layout optimization [23] to avoid the influence of environmental interference, however there are no reports of manufacturers testing against intentional EMI.

Naturally, a new research area of attacks on touch-controlled UIs emerged. Existing EMI-based attacks against capacitive touchscreens aimed to utilize the emanations radiated from screens to infer their content [50] or manipulate their sensing capability [101]. Such attacks are, however, limited in their applicability as they are either passive, imprecise or require the victim to use the device while the attack takes place.

Therefore there is a need to analyze in more depth whether an attacker can overcome the protections in place against the influence of environmental interference to reliably inject touch points into the victims' phone in order to be able to control it. In order to achieve this goal, we additionally had to design our attack without knowledge of the touchscreen specifics, which are usually undisclosed and differ between devices.

4.1 CONTRIBUTIONS

This thesis presents an attack on the virtual assistant's interaction model with the following publication, which can be found in Appendix D.

[175] Kai Wang, Richard Mitev, Chen Yan, Xiaoyu Ji, Ahmad-Reza Sadeghi, and Wenyuan Xu. Ghosttouch: Targeted attacks on touchscreens without physical touch. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 1543–1559, 2022. CORE Rank A*. Appendix D.

This paper presents our findings that smartphones equipped with a touch UI enabled by a capacitive touchscreen sensor can be controlled by an adversary using EMI without any galvanic connection, i.e., from a distance or through a table or glass. This is possible because of the capacitance sensing and the physical properties of the screen itself. The touchscreen sensor is usually layered above the actual screen and uses capacitance variation at specific points on the screen as an indicator for a finger touching it. The touch sensor is made up of a grid of transmitting (TX) and receiving (RX) electrodes, exciting the screen by putting a charge on it (TX) and sensing the differences in induced charge (RX). A finger touching the screen in the vicinity of an intersection between RX and TX lines will modify the induced charge and, therefore, be considered as a touch by the Analog Front-End (AFE) Integrated Circuit (IC) and subsequently the Microcontroller Unit (MCU) forwarding the detection to the main Central Processing Unit (CPU). The OS is then responsible for calculating a swipe from moving touch points, multi-touch or the size of a touch point. The most ubiquitous way to achieve this is by using the Scan Driving Method (SDM) [54, 55], which excites one TX electrode at a time with a specific refresh rate.

By carefully crafting specific EMI with a precise injection frequency and timing, the signal can couple to the target RX line. This induces an Alternating Current (AC) signal

into the sensing line, confusing the AFE IC to detect a touch point. By changing the parameters of the signal, predictable touch points are possible, e.g., the position and movement of the touch points on the screen. We implemented a powerful and portable Proof of Concept (PoC) attack device and showed that multiple attacks on smartphones are possible by being able to control the touch sensor. Our PoC, therefore, highlights the necessity for smartphone manufacturers to consider the risks of intentional EMI, injecting touch points into the touchscreen sensor.

Design

GhostTouch is able to fully control the touchscreen of the victim's device without a physical touch. In Figure 4, our attack flow is depicted.

In Step ❶, GhostTouch has to initially learn the characteristics of the touchscreen to be attacked. As stated earlier, the RX electrodes in a touchscreen essentially act as antennas that unintentionally pick up the electromagnetic interference. Similarly, the TX electrodes can be regarded as sending dipole antennas, as shown in ❸. Therefore, it is possible to learn the refresh rate of the screen and the currently sensed position by probing and measuring the emanated EMI from the screen. With this information, the attacker can generate a short EMI signal with a duration equal to or less than the time the screen takes to scan one RX electrode in Step ❷. By increasing this duration, multiple RX lines are excited, and the size of the injected touch point can be increased. The location of the injected touch point can be fixed by setting the injection frequency (i.e., the amount of EMI signals being sent per time frame) equal to the scanning frequency. Adding a delay to the start of the injecting procedure selects the location on the screen to inject a directed touch point ❹. Naturally, by increasing or decreasing the injection frequency, two consecutive touch points in their vicinity will be injected, which are interpreted by the system as a swipe (i.e., a moving finger). Finally, with this setup, conducting powerful attacks is possible. The attacker can inject a directed swipe to answer an eavesdropping phone call or swipe to unlock the phone. Similarly, the attacker can inject a directed single touch to press a button, such as allowing a malicious pairing request. With a powerful setup consisting of multiple antennae, even the directed input of a PIN is possible ❺.

Evaluation and Results

We implemented and evaluated two PoCs of GhostTouch, one powerful multi-antenna device with sensing capabilities and a portable device based on the ChipSHOUTER. Our evaluation comprised of testing the feasibility on 12 smart phone devices of which eight are susceptible. The precise and directed attacks were evaluated on eleven, of

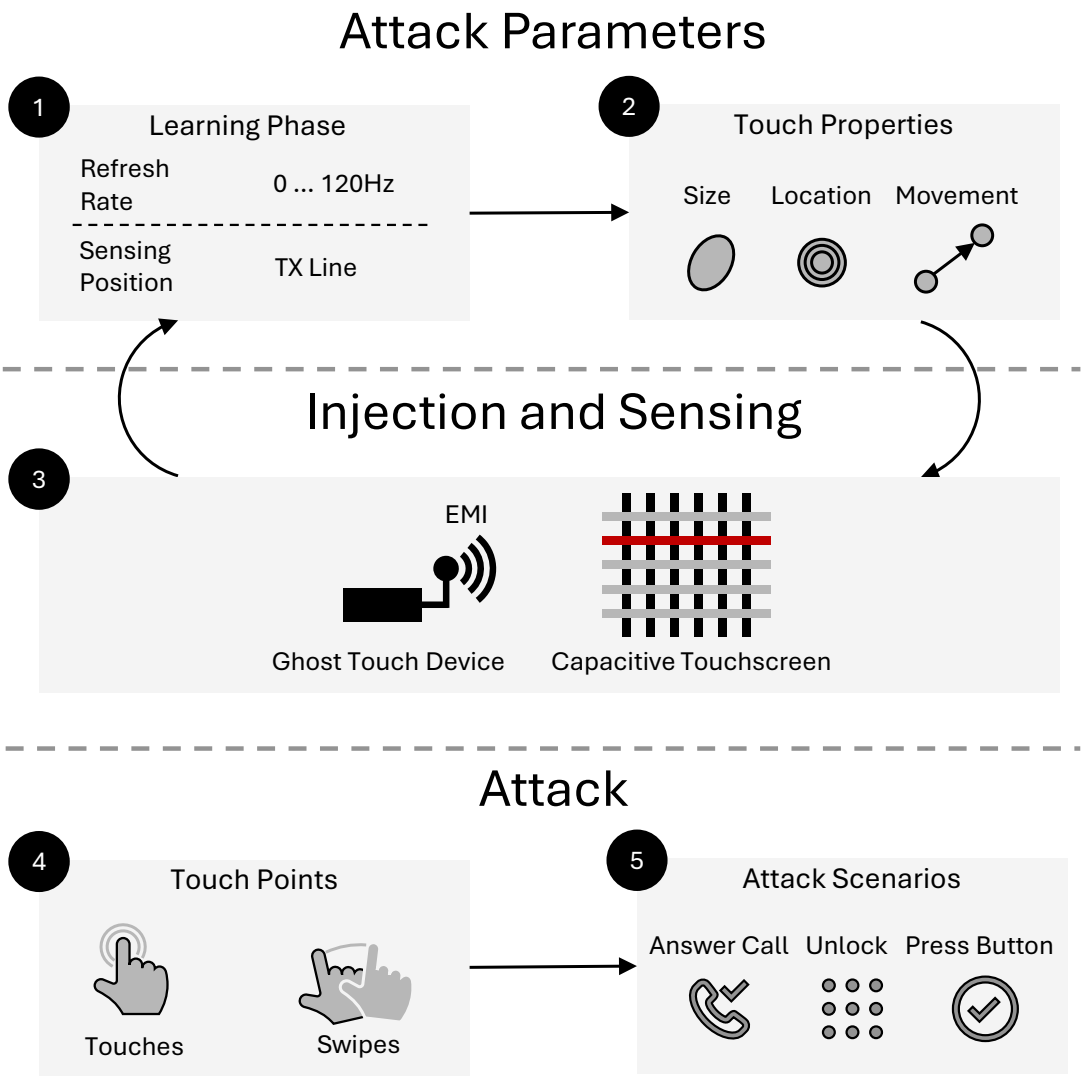


Figure 4: High-level overview of the GhostTouch attack

which we found that 9 are susceptible to the attack. GhostTouch is, therefore, able to precisely inject touch events with a mean range of 36.3px x 175.8px (ca. 2mm x 11mm) at a distance of 5mm on a Nexus 5X, which is more precise than a human finger. Similarly, our approach can inject touch points with a speed of up to 45.38$events/s$ and swipes with a precision of 62.5%. However, by increasing the attack distance the number of registered touch events decreases up to the distance of 4cm, after which no touch points are registered. In addition, we evaluated three attack scenarios: 1) answering a phone call, 2a) unlocking a phone by swiping, 2b) unlocking a phone by PIN input and 3) pressing a small button. GhostTouch is able to succeed in all these attack scenarios in a feasible amount of time. Namely, 4.1s to inject a swipe that accepts a phone call, 8.5s to inject a swipe that unlocks the victim's phone, entering a 4-digit PIN in around 20s and pressing

a 36dp x 80dp button in around 7.5s. In addition, we show that multi-touch gestures are possible by utilizing multiple antennas with the parameters computed for a single touch.

4.2 RELATED WORK

In this section, we provide an overview of EMI attacks on devices as well as general attacks on the touchscreen UI. In addition, we position our work in regard to the related literature.

Attacks Utilizing EMI

Works proposing attacks utilizing intentional EMI make up a diverse field and can result in Denial of Service (DoS), tampering with circuits or even forcing predictable faults in computing devices.

Sabath [137] categorized attacks using intentional EMI on infrastructure by criminals or terrorists, showing that intentional EMI is a real-world threat. Hayashi *et al.* [49] showed that a covert fault injection attack on the hardware Advanced Encryption Standard (AES) module of an Field-Programmable Gate Array (FPGA) board is possible using EMI. Later, Dehbaoui *et al.* [28] provided an explanation and characterization of the faults induced on AES and the coupling mechanism between the antenna and the device. Similarly, Schmidt *et al.* [140] proposed using EMI to induce faults into a device to break Rivest–Shamir–Adleman (RSA) encryption and signatures. O'Flynn [117] showed an attack combining an exploit in the USB stack as well as using EMI on a hardware key store to extract its secrets from memory. Kune at al. [82] showed that EMI could be used to impair the normal function of consumer electronics and implantable medical devices by injection of bogus signals. Selvaraj *et al.* [146] showed using EMI to inject sensor readings into a sensing circuit or introducing bit-flips into a serial communication channel. Jiang *et al.* [64] showed an approach to inject disturbances in the recorded image of a surveillance camera using EMI to confuse ML approaches of object detection. Zhang *et al.* [191] show the possibility of injecting arbitrary bits into a Differential Signalling-protected Controller Area Network (CAN) bus by using EMI. A similar attack has been proposed by Xie *et al.* [179] on IoT devices using Universal Asynchronous Receiver-Transmitter (UART).

At the same time, works addressing the attack scenario of using EMI were published.

Giechaskiel *et al.* [47] proposed a method of computing the susceptibility of sensing circuits against EMI attacks and evaluated it on the microphone circuit of smartphones.

Zhang *et al.* [190] proposed a lightweight detection mechanism against intentional EMI attacks on sensor readings.

A touchscreen can be considered a sensing circuit, which makes it susceptible to malicious intentional EMI interfering with the measurements. However, the goal of GhostTouch is not only to interfere with the measurements or inject faults into it but to force a precise and predictable outcome, which in turn allows us to control the underlying device.

Attacks on Touchscreens

We will group the area of research on attacks on touchscreens into two categories: passive attacks inferring information from a touchscreen and active ones manipulating the screen or its sensed touch points.

Passive Attacks. Published passive attacks are very diverse and analyze a large area of the attack surface of touchscreens, from the smudges left on the touchscreen to the underlying screen's emanations to even the under-display fingerprint sensor.

Aviv *et al.* [4] analyzed the feasibility of extracting an Android graphical password using the smudges left behind by the victim's fingers. Maggi *et al.* [96] propose an automatic approach for extracting keyboard presses of a smartphone using surveillance footage. Hayashi *et al.* [50] used ML to infer the contents of a tablet PC's screen through its Electromagnetic (EM) emanations. Mohamed *et al.* [109] inferred the position of a touch on a touchscreen using magnetometer readings by utilizing the finding that the magnetometer can pick up the EM emanations created by a touch. Ni *et al.* [115] presented an attack aimed to extract the user interactions of a user and his smartphone by correlating the disturbances induced by the interactions into the magnetic field created by charging the device on a wireless charger. Ni *et al.* [114] recovered the fingerprint of a user scanning his or her finger on an under-screen fingerprint sensor by recording the EM emanations of this sensor.

Active Attacks. Active attacks differ in the attack vector being exploited to gain a foothold in the system to conduct the attack. The two main categories are attacks utilizing EMI and attacks utilizing perturbance of the electrical signal through the charging cable connected to a device under attack.

Shwartz *et al.* [154] proposed an attack vector of maliciously re-flashed touchscreen controllers being able to inject touch points into the device or even force a buffer overflow to leak information of the host. Maruyama *et al.* [101] presented an attack on

the touchscreen of devices utilizing EMI to skew the touch points of a victim using the device to force him into actions desired by the attacker.

To the best of our knowledge, Maruyama *et al.* [101] was the only published work that attacked the touchscreen's touch point reading using EMI to conduct an attack. However, the attack suffers from the limitation of injecting directed touch points and only enabling the attacker to skew the touch points of a victim, which may alert her or him. After the publication of GhostTouch, a plethora of new works were published.

Shan *et al.* [149] analyzed the underlying reason for EMI attacks being able to inject touch points, as well as providing extensive evaluation and a feedback mechanism for injected touch points. Wang *et al.* [176] published a follow-up work to GhostTouch analyzing the attack in more detail and providing defenses. On the other hand, Gao *et al.* [44] propose using EMI to deny the victim the usage of his or her touchscreen by injecting a reverse current into the sensing circuit to counteract the current induced by the touch of the victim.

Jiang *et al.* [65] showed that by disturbing the power provided to a charging smartphone by injecting common-mode noise (i.e., noise injected from the ground, injecting perturbances in a signal pair in the same direction), it is possible to inject touch points into the screen of the device, similar to Jiang *et al.* [190]. To increase the attack approach as mentioned earlier, Zhu *et al.* [194] propose a genetic algorithm-based framework for finding the proper parameter for this kind of attack automatically.

5

Defending against Location Leakage of Satellite Internet Users

Advancements in satellite communications have enabled consumer satellite internet. This system uses easy-to-deploy, WiFi-enabled base stations. However, satellite communication acts as a beacon, transmitting data into space. The personal Internet of Things (IoT) base station also implies the user's presence to be nearby. As a result, the system is vulnerable to information leakage, specifically the user's geographic location. Therefore, the attack surface enabling geolocation of the users of this emerging technology is to be analyzed, and mitigations are to be found.

In this section, we propose a system to protect satellite internet users from the threat of being geolocated. Specifically, satellite communication has proven to be susceptible to being monitored or the source being triangulated by adversary satellites. We utilize the advancements in IoT long-range radio technologies to enable base stations to form a local terrestrial network. Our approach then re-routes the user's data through the network, hiding his or her (geographical) position in the network. The whole system is comprised of off-the-shelf IoT devices and is compatible with consumer devices such as smartphones without extra hardware or software.

Satellite internet has been a trending technology in the past years, from applications in remote areas to even conflict zones. In conflict zones, however, civilians utilize the internet to share, publish and forward possibly incriminating information about the conflict. After the escalation of the Ukrainian conflict, the U.S. government spent millions of dollars to enable the Ukrainians to deploy Starlink satellite internet terminals to enable a free flow of information [89]. Starlink quickly reached over 150 000 users in Ukraine for the population to stay connected even with the impaired communication infrastructure [166]. However, sharing and publishing information from an active conflict zone on social media carries risks, leading social media companies to publish articles on how to add additional security measures [71]. Unfortunately, not only the published information may be identifying, but geolocation by triangulation of satellite terminal users is possible. This is highlighted by the UN issuing warnings to journalists, that they may be targeted and in danger [170] and by a security researcher issuing a warning on Twitter, stating that Starlink users may become potential beacons for attacks due to triangulation of their signals by other satellites [145]. The CEO of Starlink promptly reacted to this information by issuing a warning to all Starlink users, that they may become targets [5]. Tracing

satellite signals to target particular people also has an unfortunate precedent with the killing of Mari Colvin, a famous journalist who was allegedly traced via her satellite phone [132]. This shows the need for a technological solution protecting satellite internet users from information leakage in the form of their geographic location.

Unfortunately, no practical solution for that problem has been proposed. Overlay networks such as Tor assume a fully connected network and are susceptible to entry point monitoring [112, 181] or entry and exit point attacks [6, 85, 121]. This, however, is even easier in satellite internet scenarios as the signal can be monitored by any adversary satellite. Works on location privacy exhibit impractical assumptions in regard to our scenario [67, 177, 148, 88, 53, 68, 133, 32, 38, 39, 182, 104, 119, 70, 174, 150, 189, 162, 106, 95]. Such as for approaches building on top of network coding [38, 39] add significant delays to the communication due to the additional usage of heavy cryptography (e.g., homomorphic encryption) or dummy traffic. Approaches relying primary on dummy traffic [182, 104] exhibit an impractical network overhead, as fake traffic floods the network and decreases its throughput. On the other hand, works aimed at re-establishing a reliable network connection for disaster zones, called emergency networks, utilize Unmanned Aerial Vehicles (UAVs) or heavy-duty vehicles such as helicopters to provide communication services to a disaster network [90, 122, 29, 123, 192, 127]. However, due to their nature, they are only applicable to short-term disasters. In addition, emergency networks utilizing satellites do not account for location privacy [193, 59, 125].

Therefore, there is a need to propose a solution to overcome the threat of remote monitoring and the possibility of malicious actors' triangulation of satellite internet users to thwart citizens' free flow of information.

5.1 CONTRIBUTIONS

This thesis presents a scheme to ensure location privacy for ground-based satellite communications with the following publication, which can be found in Appendix E.

[77] David Koisser, Richard Mitev, Marco Chilese, and Ahmad-Reza Sadeghi. Don't shoot the messenger: Localization prevention of satellite internet users. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 426–444. IEEE Computer Society, 2023. CORE Rank A*. Appendix E.

This paper presents the AnonSat architecture, which enables satellite internet users to hide their geographic position in case the origin of their satellite connection is being triangulated. In our system, base stations utilize long-range radio for IoT to span a network between base stations. The user's connection is then established over a number

of intermediate hops to a random output gateway, hiding the connection's origin. Our design is agnostic to the utilized radio technology, and the base station extension is entirely made up of cheap, off-the-shelf IoT devices that are readily available. In addition, the network is accessible by the user out-of-the-box without any extra hardware or software downloads by using standardized WiFi. Finally, our approach provides real-time internet (IP) access to be compatible with popular social networks such as Twitter or WhatsApp.

Design

AnonSat is able to protect the geographical location of the satellite internet users connected to its network. In Figure 5, a high-level overview of the AnonSat system is depicted.

The first step towards designing AnonSat is to consider the available long-range radio technologies and resulting challenges to overcome ❶ and the security features to be available in the setup ❷. In Step ❶, we analyzed the technologies used to enable remote IoT devices to be connected to the Internet or to each other. Due to the relatively high data rates, unlicensed frequency band usage and the readily available development kits and shields, we chose Long Range (LoRa) Sub-GHz as the long-range radio technology for AnonSat. However, this implies technological hurdles between LoRa and Ethernet. Namely, a mismatch between the MTU sizes makes it necessary to fragment $1500B$ Ethernet frames to $255B$ LoRa packets and vice versa. This, however, leads to the occurrence of spurious retransmissions of packets when using Transmission Control Protocol (TCP) due to sending large fragmented packets over a slow connection with a low Retransmit Timeout (RTO). We mitigate this by setting the transmission queue length of each hop to a very low value to prevent excessive, unnecessary retransmissions. As multipath connections or quick endpoint changes are not supported in our software stack, the origin gateway utilizes the TCP Reset (RST) flag to signal to the client and server to stop their session. This is necessary as one security feature ❷ is to periodically change the output gateway to prevent the origin gateway from being located over time by the attacker utilizing local meta-data. Another feature is that the data in transit is per-hop encrypted. In addition, we also introduce an Output Selection Bias to prevent a malicious actor from geolocating a user by placing all used output gateways over time on the perimeter of a circle. The center of it may be the user's location. By using the bias, the origin gateway will utilize output gateways in a random direction and hops (0, ..., *max_hops*) distance to shift the circle's center in a random direction. With these considerations, we design the general setup of AnonSat in Step ❸. This setup consists of five satellite internet base stations, each connected with a long-range radio (i.e., LoRa). The user uses his out-of-the-box device to connect to the base station's Wi-Fi to gain access to the internet. The origin gateway the user is connected to then chooses an output gateway while adhering to the security parameters mentioned earlier. In *Session 1*, the origin

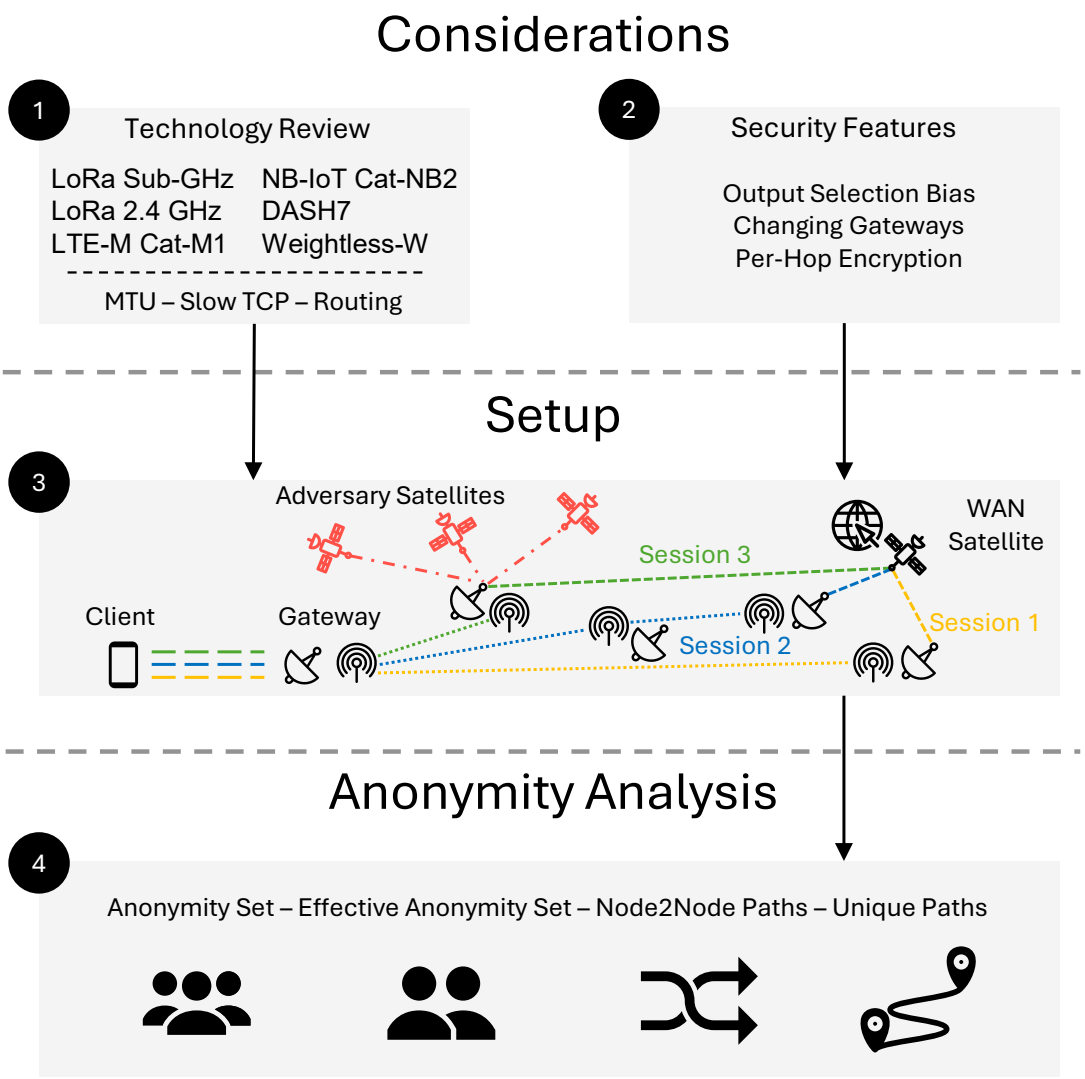


Figure 5: High-level overview of the AnonSat system

gateway forwards the user's packets to another gateway, which then utilizes its satellite internet connectivity to forward them to the internet. After a time, this session expires as described earlier, and the origin gateway chooses a new output gateway resulting in *Session 2*. This time, this gateway is reachable over an intermediate gateway, creating a 2-hop connection. This procedure is repeated indefinitely. In *Session 3* the adversary geolocated the output gateway of a connection, but cannot infer the location of the client. Finally, in Step ❹, with the technology being used and the features of our approach being fixed, we can calculate the anonymity and resiliency provided by AnonSat. We propose to calculate the Anonymity Set metrics as defined by Chaum [13], which represents the set of parties from which one is not distinguishable, as well as the Effective Set as proposed by Serjantov *et al.* [147] which shows the reachability of each node throughout

the graph for the anonymity guarantees. To quantify the resilience of AnonSat, the average number of paths between any two gateways can be calculated, which details the general connectivity among the nodes and the unique paths between any two nodes. This number shows the reliability when individual gateways fail, as it indicates the number of backup paths.

Evaluation and Results

We evaluated AnonSat on an implemented PoC and extensive simulation. The PoC comprises a Raspberry Pi 3 B+ equipped with a LoRa Sub-Ghz shield (i.e., easily connectable daughterboard) and a Starlink base station. The three Raspberry Pis are connected through LoRa, and a WiFi connection is advertised for clients to connect to. The round trip time (RTT) of a ping packet from the origin gateway to a server on the internet over three hops takes around $212ms$, including around $50ms$ of delay from the Starlink connection. Uploading or downloading a typical $200kB$ picture to an external server on the internet over three hops takes $269s$. We argue that the overhead expense is feasible for our scenario, as with three hops, the distance between the origin gateway and output gateway can be as high as $15km$, providing ample distance between the user and the satellite connection. Note that, the higher the distance, the larger the number of gateways a client may be connected to, increasing the uncertainty for the attacker.

We utilize OMNet++ 6.0 for our simulation evaluation. To evaluate real-world data rates, delays as well as anonymity and resiliency metrics, we utilized public WiFi location databases with up to 5441 nodes each. The results show that with a growing number of concurrently sending nodes in nearly all networks, the average time to send a fixed amount of data increases logarithmically, whereas the roundtrip delay of setting up a TLS session and the achieved distance when increasing the number of hops increases relatively linearly. This shows the applicability of AnonSat even in big network constellations with many clients using the network at the same time. Similarly, we calculated the anonymity and resiliency metrics on the real-world datasets. In the case of the most extensive dataset of Hong Kong, consisting of 5441 nodes, of which with a one-hop distance of $5km$, up to 866 nodes are reachable. The average anonymity set is, therefore, 417 and the effective set is 278. On average, 6080 different paths exist between two nodes, and 54 unique paths exist. These calculations show that for a malicious actor, it is improbable to identify the client protected by AnonSat, as well as challenging to disrupt the network in an impactful manner.

5.2 RELATED WORK

In this section, we provide an overview of the related work regarding Location Privacy in Mesh and Wireless Sensor Networks and Emergency Communication Networks utilizing satellites and other technologies. In addition, we detail the position of our work in the related literature.

Location Privacy in Mesh and Wireless Sensor Networks

Due to the nature of, e.g., long-range radio communication between devices of a mesh or sensor network, eavesdropping by an adversary is trivial. An adversary can correlate metadata or even read the (unencrypted) intercepted data to geolocate nodes in these networks. To counter this, the research area of Location Privacy emerged. Approaches for location privacy can be categorized into eleven groups [18], which we re-group into four.

Routing-based. These approaches modify the routes of packages to avoid or confuse the attacker. *Random Walk*-based approaches route packets through a random path to the sink (i.e., the target node), making the chosen path unpredictable to the adversary [67, 177]. Similarly, *Geographic Routing* creates a (mostly) random path through the network. However, it also considers the nodes' geographic distribution to increase its provided efficiency and security [148, 88]. These approaches can also utilize other approaches in combination to increase security guarantees on location privacy. These may be pseudonyms, reputation- or mix-networks.

Delay-based. These approaches aim to confuse the attacker trying to correlate packet timings by introducing delays into the communication channel, potentially scrambling the chronological order of the packets arriving and modifying the traffic pattern [53, 68]. Another way of using delays to protect geolocation of nodes is to *limit node detectability*. This is achieved by temporarily limiting the transmitting power of devices or even powering them off to hide from malicious actors, based on events or at random [133, 32]. In addition, data aggregation at selected intermediary nodes introduces delays in communication. The nodes in the network can use homomorphic encryption to aggregate data of multiple nodes to hide individual traffic flows into one emanating from the aggregating node [38, 39].

Fake or Extra Traffic. These traffic generating approaches may create realistic-looking traffic or route benign traffic in unusual ways to create a traffic overhead that thwarts

the attacker's capability to correlate data. One technique to achieve this are *Dummy Data Sources*-based approaches, that employ nodes creating fake traffic which hinders the attacker from identifying the benign traffic routed through the network [182, 104]. Another way to create extra traffic is to utilize *Cyclic Entrapment*, which routes benign traffic in circular patterns [119, 70]. Similarly, *Separate Path Routing* fragments data into smaller packets and routes them through multiple different paths. The packets must then be combined at the sink, which is impossible for the adversary to achieve as he needs to monitor all utilized paths simultaneously [174].

Hiding Traffic or Identity. Location Privacy can also be achieved by completely hiding the traffic of the nodes from the attacker in the first place or hiding the node's identity in the network. *Cross Layer Routing* hides the nodes' traffic on multiple Open Systems Interconnection (OSI) layers, assuming the adversary is unaware of this approach being deployed [150]. *In network location anonymization* approaches based on the hierarchical nature of *Wireless Mesh Networks* aim to hide or anonymize the traffic by utilizing hierarchical aggregation of data or by using pseudonyms for the nodes [106, 95]. In order to achieve this, cryptographic secrets need to be agreed on or distributed, which may be achieved by an authority [189] or self-generation in accordance with a domain authority [162].

Unfortunately, all related work regarding location privacy is incompatible with the scenario of AnonSat. Routing-based approaches employ scenarios that limit the amount of sources and sinks in the network. Delay-based and fake or extra traffic-based solutions create an extra load on the network or even, by design, turn the network into a high-latency network, which makes it impossible to be connected to the internet, which is one of the main requirements of AnonSat. Hiding traffic is only possible when the adversary is unaware of the approach being used and identity-based approaches are only available to hierarchical network structures that AnonSat does not possess.

Emergency Communication Networks

Emergency Communication Networks aim to provide reliable means of communication after the conventional infrastructure fails to operate in a conflict or emergency zone. Such networks should exhibit the security guarantees of privacy, data integrity, authentication and access control [127]. Networks consist of locally deployed devices, satellites or ad-hoc networks, of which only a subset has the means for external communication while the rest spans the network forwarding traffic [128, 75]. We will, therefore, group the related work in this area into approaches for short-term relief of a disaster and approaches utilizing satellites, as it is the closest work in relation to AnonSat.

Short-Term Relief. Short-term relief can be quickly provided by deploying Unmanned Aerial Vehicles (UAVs) [27]. It was proposed to use drones to either directly act as a base station [192] or to span a mesh network to connect to terrestrial base stations using WiFi [123], Long Term Evolution (LTE) [29], 5G [45] or LoRa [122].

Instead of drones, balloons can be deployed in a similar fashion. The network between the balloons either consists of an ad-hoc IEEE 802.11j [153], multihop WiFi [164] or WiFi mesh network [118]. Independently of the technology used, recent works propose optimizations of the network traffic handling, such as load balancing [90, 116].

In the case of a partial power outage or destruction at specific cell towers, related work proposes to utilize vehicle-mounted or vehicle-powered router devices used to bridge the communication gap resulting from the out-of-order tower [52, 102].

Satellite-based. Satellite-based emergency communication networks span a terrestrial (mesh) network, and specially equipped nodes have the capability to connect to satellites, providing external network access. Zhou *et al.* [193] proposes to use a self-organizing mesh network based on WiFi and custom devices for the nodes from which some are connected to a satellite internet base station. Iapichino *et al.* [59] and Patricelli *et al.* [125] propose the usage of mobile vehicle-mounted satellite base stations to be deployed in an emergency situation, providing internet access to users in its vicinity through, e.g., WiFi.

Both types of emergency networks, for short-term relief or satellite-based, assume purpose-built vehicles (UAVs, cars, balloons) to be readily available and easily deployable. In addition, airborne vehicles exhibit limited flight times and are susceptible to meteorological events. Approaches spanning networks or providing their user's connection via WiFi have the restriction of users needing to be present in the vicinity of the gateways or requiring a lot of relay nodes, which are both the target scenario of AnonSat. In addition, satellite-based approaches do not take the unique problem of a conflict scenario into consideration and, therefore, do not provide adequate triangulation protection.

6

Conclusion & Outlook

In this chapter, we summarize the contributions of this dissertation on information leakage attacks and defenses of Internet of Things (IoT) devices by proposing exploits and mitigations aimed at virtual assistant's Skill ecosystem and wake-word detector, physical remote attacks against capacitive touchscreens and mitigations of location leakage of satellite internet users and give an outlook on future research directions.

6.1 CONCLUSION

The primary way to interact with virtual assistants incorporated in smart speakers or smartphones is by using the novel and convenient voice interface. However, the seemingly limited virtual assistant's interaction model can be exploited on the software side. In addition, on the hardware side, command injection and voice jamming are possible attack scenarios.

In AlexaLiedtoMe [107] (Appendix A), we presented a new attack vector on personal voice-controlled virtual assistants incorporated in, e.g., smart speakers by combining exploits for multiple attack vectors to create an attack of high impact, entirely hijacking the communication between the benign user and the assistant. We leveraged a crafted malicious virtual assistant Skill, including a corresponding backend, to demonstrate a man-in-the-middle attack that can take over an entire dialogue between a voice-controlled virtual assistant and a user. The attacker is able to change or completely override the virtual assistant Skill's responses with their own, making it appear as if the responses are coming from the actual benign Skill. AlexaLiedtoMe evaluation is based on two Proof of Concepts (PoCs), one powerful using a tweeter speaker and a consumer hi-fi amplifier and one based on non-ultrasonic IoT hardware with a 2W and 3W speaker and a small amplifier board. We showed that our attack can successfully attack the ten most popular Skills as well as Skills of the Alexa ecosystem, controlling third-party hardware.

To preserve the users' privacy and minimize resource usage on speech recognition, voice-based virtual assistant devices are equipped with a wake-word detector, which needs

to be triggered by the user saying this specific word. Even worse, audio recorded by wrongly understood commands may be reviewed by contractors to increase the precision of the device, but at the same time, posing a privacy leak. Previous work and reports showed that most users suffer from unintended activation on a regular basis but did not identify devices being susceptible to it, to which words in detail or proposed easily applicable solutions against this phenomenon.

We presented an approach to detect devices reacting to audio by fuzzing their wake-word detectors in order to warn the user from unintended recording in LeakyPick [108] (Appendix B). We implemented a PoC device that utilizes audio probes while statistically monitoring the network traffic for patterns that indicate an audio transmission. In addition, our device is able to conduct audio-fuzzing of virtual assistants to detect wrongly recognized wake-words which, when spoken by a human or played by, e.g., a TV, could trigger unintended information leakage to the manufacturer's cloud. We evaluated our approach on three smart speakers incorporating virtual assistants, as well as an audio-based security system and four microphone-enabled IoT devices. In addition, we identified 89 words that reliably activated the Alexa assistant.

Building on top of that, in our work FakeWake [14] (Appendix C), we explained and mitigated this phenomenon by efficiently generating wrongly detected wake-words, identifying features contributing to the false detection and proposing strengthening the underlying ML models to increase their resiliency. We evaluated our generator by audio-fuzzing the virtual assistant's wake-word detector on eight devices, generating a total of 965 wrongly accepted wake-words over all devices. We showed that re-training the detector model on our generated wake-word set drastically lowers the acceptance rate of wrong wake-words.

Aside from the voice User Interface (UI), IoT devices also exhibit other UIs, such as a touch-based UI enabled by a touchscreen. Therefore, there is a need to analyze in more depth if an attacker can manipulate the touchscreen of, e.g., a victim's smartphone in order to control it. The ability of capacitive touchscreens to sense small changes in the electromagnetic fields makes intentional Electromagnetic Interference (EMI)-based attacks a promising candidate to investigate into.

In GhostTouch [175] (Appendix D), we presented our findings that smartphones equipped with a touch UI enabled by a capacitive touchscreen sensor can be controlled by an adversary using EMI without any galvanic connection, e.g., from a distance or through matter like a table or glass. By carefully crafting specific EMI with a precise injection frequency and timing, we forced the touchscreen to report a touch point. By changing the parameters of the signal, predictable touch points are possible, e.g., the position and movement of the touch points on the screen. We implemented and evaluated two PoCs, one powerful multi-antenna device with sensing capabilities and a portable device based

on the ChipSHOUTER, and tested our approach on nine smartphone devices. We showed the possibility of our attack to conduct powerful attack scenarios: answering a phone call, unlocking a phone by swiping, unlocking a phone by PIN input as well as pressing a small button.

Aside from UIs, communication interfaces can also exhibit information leakage. Specifically, in satellite communication, personal IoT base stations, acting as beacons, sending data into space and implying the user of this terminal to be in its vicinity, are vulnerable to information leakage in the form of the user's geographic location. Therefore, the attack surface enabling geolocation of the users of this emerging technology is to be analyzed, and mitigations are to be identified.

In AnonSat [77] (Appendix E), we, therefore, proposed a system to protect satellite internet users from the threat of being geolocated. We utilized IoT long-range radio technologies to enable base stations to form a local terrestrial network and re-route a user's data through the network, hiding his or her position in the network. In addition, the whole system is comprised of off-the-shelf IoT devices and is compatible with consumer devices such as smartphones without extra hardware or software. It also provides real-time internet (IP) access. We evaluated our approach on an implemented PoC and extensive simulation to compute real-world data rates, delays as well as anonymity and resiliency metrics. We showed the applicability of AnonSat even in big networks with many clients using the network simultaneously. Similarly, we calculated the anonymity and resiliency metrics on the real-world datasets, showing that for a malicious actor it is highly unlikely to identify the client, as well as that it is difficult to disrupt the network in an impactful manner.

6.2 OUTLOOK

More and more new IoT devices will come to market in the coming years, diversifying the IoT landscape even more. In turn, devices will be equipped with novel sensors and features, creating new or increasing the existing attack surfaces of said devices. While the contributions of this dissertation tackle many challenges in identifying and mitigating novel attack vectors leading to information leakage of IoT devices, this research field will continue to grow, with IoT devices becoming more ubiquitous. Thus, in the following, we further discuss potential research areas of IoT device information leakage attacks and defenses.

In AlexaLiedtoMe, we proposed to minimize the requirements of the attack, e.g., the adversary being able to record the audio in the vicinity of the victim's device, by further

exploiting the interaction model of Skills. One possible optimization could be to jam only the Skill's name and forward the rest of the command to our malicious Skill's catch-all utterance, making the backend transcribe the command for the attacker. However, this requires precise jamming and approximation of the original Skill's name. However, after we published AlexaLiedtoMe, new related work attacking the Skill's interaction model emerged. Works proposing novel ways to redirect the interaction flow to a malicious Skill [187, 159] could be utilized by future work to remove the necessity of jamming and Skill name injection, streamlining AlexaLiedtoMe even more.

For audio-fuzzing the wake-word detector of devices, we mainly utilized dictionary words spoken by a TTS generator. The limitation here is obviously that noise, music, and other spoken words can also activate a wake-word detector. However, in our follow-up work FakeWake, we improved on this by generating non-existing words. In addition, the LeakyPick device could be advanced by additional features such as automatically cutting the internet connection for devices sending audio out of the user's network. In addition, future work may research into a more privacy-preserving way to do traffic analysis using a statistical approach. Upcoming related work after LeakyPick was published showed that it is possible only to use MAC layer data to identify or detect devices reacting to audio [157, 152]. However, resource-heavy and pre-trained ML methods may be needed. It seems, therefore, promising to research into making these approaches more lightweight by utilizing a statistical approach.

In FakeWake, we generated and explained wrongly detected wake-words on a phoneme, letter or pinyin level, as our approach aimed to protect against wake-word detectors misidentifying words spoken by a human. However, noise or music was also shown to activate detectors [60]. To harden wake-word detectors, future work may also aim to mitigate this phenomenon. We showed that adding wrongly understood wake-words to the negative training set increases the model's precision. Related work showed that this is also possible with adversarial examples [163]. Therefore, a promising direction of research is generating generic adversarial examples (that can be played over the air) activating wake-word detectors [131, 141] and also using them for re-training in order to harden the model even against non-human voice misinterpretations.

GhostTouch injects touchpoints based on the timing the touchscreen is sensed. Usually, the Scan Driving Method (SDM) is used in one direction of the screen, horizontal or vertical. Therefore, by changing the timing, the attacker can only reliably move the injected touch point around the x or y-axis. GhostTouch and upcoming related work after publishing utilized multiple antennas to circumvent this constraint [149]. However, a straightforward approach that may also increase the attack distance of GhostTouch may utilize EMI beam-forming. If reliable beam-forming can be used to inject at a specific location of the screen perpendicular to the scanning direction, more precise long-range touch points may be possible.

For AnonSat, optimizations including extra hardware or software a user or gateway owner has to implement or download contradict our goals of providing an out-of-the-box experience for users and only using off-the-shelf hardware for gateway owners. However, the reliance on custom software for the network nodes can be decreased by advancements in network standards. For example, instead of modifying the Transmission Control Protocol (TCP) transmission queue to counter retransmissions, QUIC [62] may be used after it finds enough adoption. In addition, instead of killing TCP sessions and recomputing a new path to an output gateway, QUIC multipath [25] or connection migration [62] may be used if available. Additionally, to increase the security of AnonSat, future research may propose an encryption scheme protecting the connection's meta-data, e.g., based on onion-routing [33].

7

List of Own Publications

Part of this Dissertation

Richard Mitev, Markus Miettinen, and Ahmad-Reza Sadeghi. Alexa lied to me: Skill-based man-in-the-middle attacks on virtual assistants. In *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security*, pages 465–478, 2019. CORE: A.

Richard Mitev, Anna Pazii, Markus Miettinen, William Enck, and Ahmad-Reza Sadeghi. Leakypick: Iot audio spy detector. In *Proceedings of the 36th Annual Computer Security Applications Conference*, pages 694–705, 2020. CORE: A.

Yanjiao Chen, Yijie Bai, Richard Mitev, Kaibo Wang, Ahmad-Reza Sadeghi, and Wenyuan Xu. Fakewake: Understanding and mitigating fake wake-up words of voice assistants. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 1861–1883, 2021. CORE: A*.

Kai Wang, Richard Mitev, Chen Yan, Xiaoyu Ji, Ahmad-Reza Sadeghi, and Wenyuan Xu. Ghosttouch: Targeted attacks on touchscreens without physical touch. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 1543–1559, 2022. CORE: A*.

David Koisser, Richard Mitev, Marco Chilese, and Ahmad-Reza Sadeghi. Don't shoot the messenger: Localization prevention of satellite internet users. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 426–444. IEEE Computer Society, 2023. CORE: A*.

Additional Publications

Lars Baumgärtner, Alexandra Dmitrienko, Bernd Freisleben, Alexander Gruler, Jonas Höchst, Joshua Kühlberg, Mira Mezini, Richard Mitev, Markus Miettinen, Anel Muhamedagic, et al. Mind the gap: Security & privacy risks of contact tracing apps. In *2020 IEEE 19th international conference on trust, security and privacy in computing and communications (TrustCom)*, pages 458–467. IEEE, 2020. CORE: B.

Yan Jiang, Xiaoyu Ji, Kai Wang, Chen Yan, Richard Mitev, Ahmad-Reza Sadeghi, and Wenyuan Xu. Wight: Wired ghost touch attack on capacitive touchscreens. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 984–1001. IEEE Computer Society, 2022. CORE: A*.

Xhani Marvin Saß, Richard Mitev, and Ahmad-Reza Sadeghi. Oops..! i glitched it again! how to multi-glitch the glitching-protections on arm trustzone-m. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 6239–6256, 2023. CORE: A*.

Yan Jiang, Xiaoyu Ji, Kai Wang, Chen Yan, Richard Mitev, Ahmad-Reza Sadeghi, and Wenyuan Xu. Marionette: Manipulate your touchscreen via a charging cable. *IEEE Transactions on Dependable and Secure Computing*, 2023. CORE: A.

Philipp Pütz, Richard Mitev, Markus Miettinen, and Ahmad-Reza Sadeghi. Unleashing iot security: Assessing the effectiveness of best practices in protecting against threats. In *Proceedings of the 39th Annual Computer Security Applications Conference*, pages 190–204, 2023. CORE: A.

Kai Wang, Richard Mitev, Chen Yan, Xiaoyu Ji, Ahmad-Reza Sadeghi, and Wenyuan Xu. Analyzing and defending ghosttouch attack against capacitive touchscreens. *IEEE Transactions on Dependable and Secure Computing*, 2024. CORE: A.

Marco Chilese, Richard Mitev, Meni Orenbach, Robert Thorburn, Ahmad Atamli, and Ahmad-Reza Sadeghi. One for all and all for one: Gnn-based control-flow attestation for embedded devices. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 203–203. IEEE Computer Society, 2024. CORE: A*.

David Koisser, Richard Mitev, Nikita Yadav, Franziska Vollmer, and Ahmad-Reza Sadeghi. Orbital trust and privacy: Sok on pki and location privacy challenges in space networks. In *33nd USENIX Security Symposium (USENIX Security 24)*, 2024. CORE: A*.

Under Submission

Jonas Huellsieck, Richard Mitev, Markus Miettinen, and Ahmad-Reza Sadeghi. Hear me out: making audio-based pairing resilient against eavesdroppers. In *Under Submission*.

Bibliography

Die vorliegende Arbeit wurde eigenständig erstellt und die deutsche Übersetzung der Zusammenfassung wurde mit Hilfe des DeepL Dienstes übersetzt.

[1] Hadi Abdullah, Washington Garcia, Christian Peeters, Patrick Traynor, Kevin RB Butler, and Joseph Wilson. Practical hidden voice attacks against speech and speaker recognition systems. 2019. URL `https://www.ndss-symposium.org/ndss-paper/practical-hidden-voice-attacks-against-speech-and-speaker-recognition-systems/`.

[2] Efthimios Alepis and Constantinos Patsakis. Monkey says, monkey does: security and privacy on voice assistants. *IEEE Access*, 5:17841–17851, 2017.

[3] Sultan Alneyadi, Elankayer Sithirasenan, and Vallipuram Muthukkumarasamy. A survey on data leakage prevention systems. *Journal of Network and Computer Applications*, 62:137–152, 2016.

[4] Adam J Aviv, Katherine Gibson, Evan Mossop, Matt Blaze, and Jonathan M Smith. Smudge attacks on smartphone touch screens. In *4th USENIX workshop on offensive technologies (WOOT 10)*, 2010.

[5] Thomas Barrabi. Elon Musk warns Starlink users in Ukraine could be Russian targets, March 2022. URL `https://nypost.com/2022/03/04/elon-musk-warns-starlink-users-in-could-be-russian-targets`.

[6] Kevin Bauer, Dirk Grunwald, and Douglas Sicker. Predicting tor path compromise by exit port. In *2009 IEEE 28th International Performance Computing and Communications Conference*, pages 384–387. IEEE, 2009.

[7] Lars Baumgärtner, Alexandra Dmitrienko, Bernd Freisleben, Alexander Gruler, Jonas Höchst, Joshua Kühlberg, Mira Mezini, Richard Mitev, Markus Miettinen, Anel Muhamedagic, et al. Mind the gap: Security & privacy risks of contact tracing apps. In *2020 IEEE 19th international conference on trust, security and privacy in computing and communications (TrustCom)*, pages 458–467. IEEE, 2020.

[8] H Felcia Bel and S Sabeen. A survey on iot security: attacks, challenges and countermeasures. *Webology*, 19(1):3741–3763, 2022.

[9] Mary K. Bispham, Ioannis Agrafiotis, and Michael Goldsmith. The speech interface as an attack surface: An overview. *International Journal On Advances in Security*, 12(1 and 2), 2019.

[10] Katie Canales. Amazon alexa records private conversation and sends it to a friend, May 2018. URL `https://www.businessinsider.com/amazon-alexa-records-private-conversation-2018-5`.

[11] Nicholas Carlini and David Wagner. Audio adversarial examples: Targeted attacks on speech-to-text. In *2018 IEEE security and privacy workshops (SPW)*, pages 1–7. IEEE, 2018.

[12] Nicholas Carlini, Pratyush Mishra, Tavish Vaidya, Yuankai Zhang, Micah Sherr, Clay Shields, David Wagner, and Wenchao Zhou. Hidden voice commands. In *25th USENIX security symposium (USENIX security 16)*, pages 513–530, 2016.

[13] David Chaum. The dining cryptographers problem. *Journal of Cryptology*, 1:65–75, 1988.

[14] Yanjiao Chen, Yijie Bai, Richard Mitev, Kaibo Wang, Ahmad-Reza Sadeghi, and Wenyuan Xu. Fakewake: Understanding and mitigating fake wake-up words of voice assistants. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 1861–1883, 2021.

[15] Yushi Cheng, Xiaoyu Ji, Tianyang Lu, and Wenyuan Xu. Dewicam: Detecting hidden wireless cameras via smartphones. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, pages 1–13, 2018.

[16] Marco Chilese, Richard Mitev, Meni Orenbach, Robert Thorburn, Ahmad Atamli, and Ahmad-Reza Sadeghi. One for all and all for one: Gnn-based control-flow attestation for embedded devices. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 203–203. IEEE Computer Society, 2024.

[17] Hyunji Chung, Michaela Iorga, Jeffrey Voas, and Sangjin Lee. Alexa, can i trust you? *Computer*, 50(9):100–104, 2017.

[18] Mauro Conti, Jeroen Willemsen, and Bruno Crispo. Providing source location privacy in wireless sensor networks: A survey. *IEEE Communications Surveys & Tutorials*, 15(3):1238–1280, 2013.

[19] Joseph Cox. Revealed: Microsoft Contractors Are Listening to Some Skype Calls, August 2019. URL `https://www.vice.com/en/article/xweqbq/microsoft-contractors-listen-to-skype-calls`.

[20] Toby Cox. Siri and Alexa Fails: Frustrations With Voice Search, June 2024. URL https://themanifest.com/digital-marketing/resources/siri-alexa-fails-frustrations-with-voice-search.

[21] Smita Dange and Madhumita Chatterjee. IoT botnet: The largest threat to the IoT network. In *Advances in Intelligent Systems and Computing*, pages 137–157. Springer Singapore, 2019.

[22] Smita Dange and Madhumita Chatterjee. Iot botnet: The largest threat to the iot network. In *Data Communication and Networks: Proceedings of GUCON 2019*, pages 137–157. Springer, 2019.

[23] Chris Daskalou. How conducted emi influences touch sensors, May 2020. URL https://fieldscale.com/blog/eminoise-touch-sensors/.

[24] Matt Day, Giles Turner, and Natalia Drozdiak. Is anyone listening to you on alexa? a global team reviews audio, April 2019. URL https://www.bloomberg.com/news/articles/2019-04-10/is-anyone-listening-to-you-on-alexa-a-global-team-reviews-audio.

[25] Quentin De Coninck and Olivier Bonaventure. Multipath extension for quic. Technical report, 2017.

[26] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

[27] Sanjoy Debnath, Wasim Arif, Sourav Roy, Srimanta Baishya, and Debarati Sen. A comprehensive survey of emergency communication network and management. *Wireless Personal Communications*, pages 1–47, 2021.

[28] Amine Dehbaoui, Jean-Max Dutertre, Bruno Robisson, and Assia Tria. Electromagnetic transient faults injection on a hardware and a software implementations of aes. In *2012 Workshop on Fault Diagnosis and Tolerance in Cryptography*, pages 7–15. IEEE, 2012.

[29] Margot Deruyck, Jorg Wyckmans, Wout Joseph, and Luc Martens. Designing uav-aided emergency networks for large-scale disaster scenarios. *EURASIP Journal on Wireless Communications and Networking*, 2018(1):1–12, 2018.

[30] Wenrui Diao, Xiangyu Liu, Zhe Zhou, and Kehuan Zhang. Your voice assistant is mine: How to abuse speakers to steal information and control your phone. In *Proceedings of the 4th ACM Workshop on Security and Privacy in Smartphones & Mobile Devices*, pages 63–74, 2014.

[31] Daniel J Dubois, Roman Kolcun, Anna Maria Mandalari, Muhammad Talha Paracha, David Choffnes, and Hamed Haddadi. When speakers are all ears: Characterizing misactivations of iot smart speakers. *Proceedings on Privacy Enhancing Technologies*, 2020.

[32] Rania El-Badry, Ahmed Sultan, and Moustafa Youssef. Hyberloc: providing physical layer location privacy in hybrid sensor networks. In *2010 IEEE International Conference on Communications*, pages 1–5. IEEE, 2010.

[33] Amr El Mougy and Sandra Sameh. Preserving privacy in wireless sensor networks using onion routing. In *2018 International Symposium on Networks, Computers and Communications (ISNCC)*, pages 1–6. IEEE, 2018.

[34] Electronic Products. Shielding touchscreens from EMI - Electronic Products, May 2008. URL `https://www.electronicproducts.com/shielding-touchscreens-from-emi/`.

[35] Jennifer Elias. Google suspends transcriptions of recordings from its voice assistant in Europe, August 2019. URL `https://www.cnbc.com/2019/08/01/google-suspends-transcriptions-of-recordings-from-voice-assistant-in-eu.html`.

[36] Embedded Staff. Understanding electromagnetic interference sources in touchscreens, November 2011. URL `https://www.embedded.com/understanding-electromagnetic-interference-sources-in-touchscreens`.

[37] Mohammad Esmaeilpour, Patrick Cardinal, and Alessandro Lameiras Koerich. A robust approach for securing audio classification against adversarial attacks. *IEEE Transactions on information forensics and security*, 15:2147–2159, 2019.

[38] Yanfei Fan, Yixin Jiang, Haojin Zhu, and Xuemin Shen. An efficient privacy-preserving scheme against traffic analysis attacks in network coding. In *IEEE INFOCOM 2009*, pages 2213–2221. IEEE, 2009.

[39] Yanfei Fan, Jiming Chen, Xiaodong Lin, and Xuemin Shen. Preventing traffic explosion and achieving source unobservability in multi-hop wireless networks using network coding. In *2010 IEEE Global Telecommunications Conference GLOBECOM 2010*, pages 1–5. IEEE, 2010.

[40] Flanders News. The analysis of our voices by Google is perfectly acceptable, but not listening in to our most intimate moments without our knowledge, July 2019. URL `https://www.vrt.be/vrtnws/en/2019/07/10/the-analysis-of-our-voices-by-google-is-perfectly-acceptable-bu`.

[41] FOCUS LCDs. Capacitive Touch Noise Prevention. *FOCUS LCDs*, August 2023. URL https://focuslcds.com/wp-content/uploads/2023/08/Capacitive-Touch-Noise-Prevention.pdf.

[42] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[43] Sarah Frier. Facebook (FB) Paid Contractors to Transcribe User Audio Files, August 2019. URL https://www.bloomberg.com/news/articles/2019-08-13/facebook-paid-hundreds-of-contractors-to-transcribe-users-audio.

[44] Ming Gao, Fu Xiao, Wentao Guo, Zixin Lin, Weiran Liu, and Jinsong Han. Practical emi attacks on smartphones with users' commands cancelled. *IEEE Transactions on Dependable and Secure Computing*, 2023.

[45] Yuan Gao, Jiang Cao, Ping Wang, Junsong Yin, Ming He, Ming Zhao, Mugen Peng, Su Hu, Yunchuan Sun, Jing Wang, et al. Intelligent uav based flexible 5g emergency networks: Field trial and system level results. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 138–143. IEEE, 2020.

[46] Kris Gesling. Precise | Mycroft AI, 2021. URL https://mycroft-ai.gitbook.io/docs/mycroft-technologies/precise.

[47] Ilias Giechaskiel, Youqian Zhang, and Kasper B Rasmussen. A framework for evaluating security in the presence of signal injection attacks. In *Computer Security–ESORICS 2019: 24th European Symposium on Research in Computer Security, Luxembourg, September 23–27, 2019, Proceedings, Part I 24*, pages 512–532. Springer, 2019.

[48] Ibbad Hafeez, Markku Antikainen, Aaron Yi Ding, and Sasu Tarkoma. Iot-keeper: Detecting malicious iot network activity using online traffic analysis at the edge. *IEEE Transactions on Network and Service Management*, 17(1):45–59, 2020.

[49] Yu-ichi Hayashi, Naofumi Homma, Takeshi Sugawara, Takaaki Mizuki, Takafumi Aoki, and Hideaki Sone. Non-invasive emi-based fault injection attack against cryptographic modules. In *2011 IEEE International Symposium on Electromagnetic Compatibility*, pages 763–767. IEEE, 2011.

[50] Yuichi Hayashi, Naofumi Homma, Mamoru Miura, Takafumi Aoki, and Hideaki Sone. A threat for tablet pcs in public space: Remote visualization of screen images using em emanation. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 954–965, 2014.

[51] Alex Hern. Apple contractors 'regularly hear confidential details' on Siri recordings, August 2019. URL https://www.theguardian.com/technology/2019/jul/26/apple-contractors-regularly-hear-confidential-details-on-siri-recordings.

[52] Ai Hua Ho, Yao Hua Ho, and Kien A Hua. Handling high mobility in next-generation wireless ad hoc networks. *International Journal of Communication Systems*, 23(9-10):1078–1092, 2010.

[53] Xiaoyan Hong, Pu Wang, Jiejun Kong, Qunwei Zheng, et al. Effective probabilistic approach protecting sensor traffic. In *MILCOM 2005-2005 IEEE Military Communications Conference*, pages 169–175. IEEE, 2005.

[54] Steve Hotelling, Joshua A Strickon, and Brian Q Huppi. Multipoint touch surface controller, February 2006. US Patent US8279180B2.

[55] Steve Hotelling, Joshua A Strickon, and Brian Q Huppi. Multipoint touchscreen, February 2013. US Patent US9035907B2.

[56] Josh Howarth. How Many People Own Smartphones? (2024-2029), June 2024. URL https://explodingtopics.com/blog/smartphone-stats.

[57] Jonas Huellsieck, Richard Mitev, Markus Miettinen, and Ahmad-Reza Sadeghi. Hear me out: making audio-based pairing resilient against eavesdroppers. In *Under Submission*.

[58] René Hummen, Jens Hiller, Hanno Wirtz, Martin Henze, Hossein Shafagh, and Klaus Wehrle. 6lowpan fragmentation attacks and mitigation mechanisms. In *Proceedings of the sixth ACM conference on Security and privacy in wireless and mobile networks*, pages 55–66, 2013.

[59] Giuliana Iapichino, Christian Bonnet, Oscar del Rio Herrero, Cedric Baudoin, and Isabelle Buret. Advanced hybrid satellite and terrestrial system architecture for emergency mobile communications. In *26th international communications satellite systems conference (ICSSC)*, 2008.

[60] Md Tamzeed Islam, Bashima Islam, and Shahriar Nirjon. Soundsifter: Mitigating overhearing of continuous listening devices. In *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*, pages 29–41, 2017.

[61] Dan Iter, Jade Huang, and Mike Jermann. Generating adversarial examples for speech recognition. 2017. URL https://api.semanticscholar.org/CorpusID:20789384.

[62] Jana Iyengar, Martin Thomson, et al. Quic: A udp-based multiplexed and secure transport. In *RFC 9000*. Internet Engineering Task Force (IETF) Fremont, CA, USA, 2021.

[63] Fehmi Jaafar, Darine Ameyed, Amine Barrak, and Mohamed Cheriet. Identification of compromised iot devices: Combined approach based on energy consumption and network traffic analysis. In *2021 IEEE 21st International Conference on Software Quality, Reliability and Security (QRS)*, pages 514–523. IEEE, 2021.

[64] Qinhong Jiang, Xiaoyu Ji, Chen Yan, Zhixin Xie, Haina Lou, and Wenyuan Xu. Glitchhiker: Uncovering vulnerabilities of image signal transmission with iemi. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 7249–7266, 2023.

[65] Yan Jiang, Xiaoyu Ji, Kai Wang, Chen Yan, Richard Mitev, Ahmad-Reza Sadeghi, and Wenyuan Xu. Wight: Wired ghost touch attack on capacitive touchscreens. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 984–1001. IEEE Computer Society, 2022.

[66] Yan Jiang, Xiaoyu Ji, Kai Wang, Chen Yan, Richard Mitev, Ahmad-Reza Sadeghi, and Wenyuan Xu. Marionette: Manipulate your touchscreen via a charging cable. *IEEE Transactions on Dependable and Secure Computing*, 2023.

[67] Pandurang Kamat, Yanyong Zhang, Wade Trappe, and Celal Ozturk. Enhancing source-location privacy in sensor network routing. In *25th IEEE international conference on distributed computing systems (ICDCS'05)*, pages 599–608. IEEE, 2005.

[68] Pandurang Kamat, Wenyuan Xu, Wade Trappe, and Yanyong Zhang. Temporal privacy in wireless sensor networks: Theory and practice. *ACM Transactions on Sensor Networks (TOSN)*, 5(4):1–24, 2009.

[69] Chaouki Kasmi and Jose Lopes Esteves. Iemi threats for information security: Remote command injection on modern smartphones. *IEEE Transactions on Electromagnetic Compatibility*, 57(6):1752–1755, 2015.

[70] Leonidas Kazatzopoulos, Constantinos Delakouridis, Giannis F Marias, and Panagiotis Georgiadis. ihide: Hiding sources of information in wsns. In *Second International Workshop on Security, Privacy and Trust in Pervasive and Ubiquitous Computing (SecPerU'06)*, pages 8–pp. IEEE, 2006.

[71] Heather Kelly. Social media companies push Ukrainian users to add safeguards, February 2022. ISSN 0190-8286. URL `https://www.washingtonpost.com/technology/2022/02/26/protecting-identity-socialmedia`.

[72] Veton Z Këpuska and TB Klein. A novel wake-up-word speech recognition system, wake-up-word recognition task, technology and evaluation. *Nonlinear Analysis: Theory, Methods & Applications*, 71(12):e2772–e2789, 2009.

[73] Shadi Khazaaleh, Georgios Korres, Mohammed Eid, Mahmoud Rasras, and Mohammed F Daqaq. Vulnerability of mems gyroscopes to targeted acoustic attacks. *IEEE Access*, 7:89534–89543, 2019.

[74] HyunGon Kim. Protection against packet fragmentation attacks at 6lowpan adaptation layer. In *2008 International conference on convergence and hybrid information technology*, pages 796–801. IEEE, 2008.

[75] Vasani Yash Kishorbhai and Nagekar Nainesh Vasantbhai. Aon: a survey on emergency communication systems during a catastrophic disaster. *Procedia computer science*, 115:838–845, 2017.

[76] John Koetsier. Amazon Echo, Google Home Installed Base Hits 50 Million; Apple Has 6% Market Share, Report Says, August 2018. URL `https://www.forbes.com/sites/johnkoetsier/2018/08/02/amazon-echo-google-home-installed-base-hits-50-million-apple-has-6-market-share-report-says`.

[77] David Koisser, Richard Mitev, Marco Chilese, and Ahmad-Reza Sadeghi. Don't shoot the messenger: Localization prevention of satellite internet users. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 426–444. IEEE Computer Society, 2023.

[78] David Koisser, Richard Mitev, Nikita Yadav, Franziska Vollmer, and Ahmad-Reza Sadeghi. Orbital trust and privacy: Sok on pki and location privacy challenges in space networks. In *33nd USENIX Security Symposium (USENIX Security 24)*, 2024.

[79] Felix Kreuk, Yossi Adi, Moustapha Cisse, and Joseph Keshet. Fooling end-to-end speaker verification with adversarial examples. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1962–1966, 2018. doi: 10.1109/ICASSP.2018.8462693.

[80] Deepak Kumar, Riccardo Paccagnella, Paul Murley, Eric Hennenfent, Joshua Mason, Adam Bates, and Michael Bailey. Skill squatting attacks on amazon alexa. In *27th USENIX security symposium (USENIX Security 18)*, pages 33–47, 2018.

[81] Deepak Kumar, Kelly Shen, Benton Case, Deepali Garg, Galina Alperovich, Dmitry Kuznetsov, Rajarshi Gupta, and Zakir Durumeric. All things considered: An analysis of iot devices on home networks. In *28th USENIX security symposium (USENIX Security 19)*, pages 1169–1185, 2019.

[82] Denis Foo Kune, John Backes, Shane S Clark, Daniel Kramer, Matthew Reynolds, Kevin Fu, Yongdae Kim, and Wenyuan Xu. Ghost talk: Mitigating emi signal injection attacks against analog sensors. In *2013 IEEE Symposium on Security and Privacy*, pages 145–159. IEEE, 2013.

[83] Siddique Latif, Rajib Rana, and Junaid Qadir. Adversarial machine learning and speech emotion recognition: Utilizing generative adversarial networks for robustness. *arXiv preprint arXiv:1811.11402*, 2018.

[84] Tu Le, Dongfang Zhao, Zihao Wang, XiaoFeng Wang, and Yuan Tian. Alexa, is the skill always safe? uncover lenient skill vetting process and protect user privacy at run time. In *Proceedings of the 46th International Conference on Software Engineering: Software Engineering in Society*, ICSE-SEIS'24, page 34–45, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400704994. doi: 10.1145/3639475.3640102. URL https://doi.org/10.1145/3639475.3640102.

[85] Stevens Le Blond, Pere Manils, Abdelberi Chaabane, Mohamed Ali Kaafar, Claude Castelluccia, Arnaud Legout, and Walid Dabbous. One bad apple spoils the bunch: Exploiting p2p applications to trace and profile tor users. In *4th USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET 11)*, 2011.

[86] Lenovo. Touch function of Touch Screen cannot run normally under fluorescent light environment, February 2018. URL https://support.lenovo.com/us/en/solutions/ht100489-touch-function-of-touch-screen-cannot-run-normally-under-fluorescent-light-environment.

[87] Yanyan Li, Sara Kim, and Eric Sy. A survey on amazon alexa attack surfaces. In *2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC)*, pages 1–7. IEEE, 2021.

[88] Yun Li and Jian Ren. Preserving source-location privacy in wireless sensor networks. In *2009 6th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, pages 1–9. IEEE, 2009.

[89] Cristiano Lima-Strong and Aaron Schaffer. U.S. quietly paying millions to send Starlink terminals to Ukraine, contrary to SpaceX claims, April 2022. ISSN 0190-8286. URL https://www.washingtonpost.com/politics/2022/04/08/us-quietly-paying-millions-send-starlink-terminals-ukraine-contrary-spacexs-claims.

[90] Na Lin, Yuheng Liu, Liang Zhao, Dapeng Oliver Wu, and Yifan Wang. An adaptive uav deployment scheme for emergency networking. *IEEE Transactions on Wireless Communications*, 21(4):2383–2398, 2021.

[91] Tian Liu, Ziyu Liu, Jun Huang, Rui Tan, and Zhen Tan. Detecting wireless spy cameras via stimulating and probing. In *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services*, pages 243–255, 2018.

[92] Tiantian Liu, Feng Lin, Zhangsen Wang, Chao Wang, Zhongjie Ba, Li Lu, Wenyao Xu, and Kui Ren. Magbackdoor: Beware of your loudspeaker as a backdoor for magnetic injection attacks. In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 3416–3431. IEEE, 2023.

[93] Xiaolei Liu, Kun Wan, Yufei Ding, Xiaosong Zhang, and Qingxin Zhu. Weighted-sampling audio adversarial example attack. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):4908–4915, Apr. 2020. doi: 10.1609/aaai.v34i04.5928. URL `https://ojs.aaai.org/index.php/AAAI/article/view/5928`.

[94] Scott M Lundberg, Gabriel Erion, Hugh Chen, Alex DeGrave, Jordan M Prutkin, Bala Nair, Ronit Katz, Jonathan Himmelfarb, Nisha Bansal, and Su-In Lee. From local explanations to global understanding with explainable ai for trees. *Nature machine intelligence*, 2(1):56–67, 2020.

[95] Xi Luo, Xu Ji, and Myong-Soon Park. Location privacy against traffic analysis attacks in wireless sensor networks. In *2010 International Conference on Information Science and Applications*, pages 1–6. IEEE, 2010.

[96] Federico Maggi, Alberto Volpatto, Simone Gasparini, Giacomo Boracchi, and Stefano Zanero. A fast eavesdropping attack against touchscreens. In *2011 7th International Conference on Information Assurance and Security (IAS)*, pages 320–325. IEEE, 2011.

[97] Sri Harish Mallidi, Roland Maas, Kyle Goehner, Ariya Rastrow, Spyros Matsoukas, and Björn Hoffmeister. Device-directed Utterance Detection. In *Proc. Interspeech 2018*, pages 1225–1228, 2018. doi: 10.21437/Interspeech.2018-1531.

[98] Pratyusa K Manadhata, Kymie MC Tan, Roy A Maxion, and Jeannette M Wing. An approach to measuring a system's attack surface. *Pittsburgh, PA, Tech. Rep*, 2007.

[99] Market Data Forecast. Global internet of things (iot) market size, share, trends, covid-19 impact & growth forecast report – segmented by software (data management, remote monitoring, network management and security solutions), hardware (sensors and camera), services, organization type, end-user and region (north america, europe, asia pacific, latin america, and middle east & africa) - industry analysis from 2024 to 2029, 2024. URL `https://www.marketdataforecast.com/market-reports/internet-of-things-iot-market`.

[100] Markets And Markets. Iot security market by type (network security, endpoint, application security, & cloud security), offerings (solutions & services), application area (smart manufacturing, connected logistics & transportation), data sensitivity - global forecast to 2028, 2023. URL `https://www.marketsandmarkets.com/Market-Reports/iot-security-market-67064836.html`.

[101] Seita Maruyama, Satohiro Wakabayashi, and Tatsuya Mori. Tap'n ghost: A compilation of novel attack techniques against smartphone touchscreens. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 620–637. IEEE, 2019.

[102] Kenichi Mase and Jing Gao. Electric vehicle-based ad-hoc networking for large-scale disasters design principles and prototype development. In *2013 IEEE Eleventh International Symposium on Autonomous Decentralized Systems (ISADS)*, pages 1–6. IEEE, 2013.

[103] Manisha Mathur, JK Rai, and N Sridhar. Electromagnetic compatibility analysis of projected capacitive touch technology based panel computer for military application. *Journal of ElEctromagnEtic WavEs and applications*, 30(13):1689–1701, 2016.

[104] Kiran Mehta, Donggang Liu, and Matthew Wright. Location privacy in sensor networks against a global eavesdropper. In *2007 IEEE International Conference on Network Protocols*, pages 314–323. IEEE, 2007.

[105] Assaf Hurwitz Michaely, Xuedong Zhang, Gabor Simko, Carolina Parada, and Petar Aleksic. Keyword spotting for google assistant using contextual speech recognition. In *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 272–278. IEEE, 2017.

[106] Satyajayant Misra and Guoliang Xue. Efficient anonymity schemes for clustered wireless sensor networks. *International Journal of Sensor Networks*, 1(1-2):50–63, 2006.

[107] Richard Mitev, Markus Miettinen, and Ahmad-Reza Sadeghi. Alexa lied to me: Skill-based man-in-the-middle attacks on virtual assistants. In *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security*, pages 465–478, 2019.

[108] Richard Mitev, Anna Pazii, Markus Miettinen, William Enck, and Ahmad-Reza Sadeghi. Leakypick: Iot audio spy detector. In *Proceedings of the 36th Annual Computer Security Applications Conference*, pages 694–705, 2020.

[109] Reham Mohamed, Habiba Farrukh, Yidong Lu, He Wang, and Z Berkay Celik. Istelan: Disclosing sensitive user information by mobile magnetometer from finger touches. *Proceedings on Privacy Enhancing Technologies*, 2023.

[110] Mujahid Mohsin, Zahid Anwar, Ghaith Husari, Ehab Al-Shaer, and Mohammad Ashiqur Rahman. Iotsat: A formal framework for security analysis of the internet of things (iot). In *2016 IEEE conference on communications and network security (CNS)*, pages 180–188. IEEE, 2016.

[111] David Monsees. More information about our processes to safeguard speech data, July 2019. URL `https://blog.google/products/assistant/more-information-about-our-processes-safeguard-speech-data`.

[112] Steven J Murdoch and Piotr Zieliński. Sampled traffic analysis by internet-exchange-level adversaries. In *International workshop on privacy enhancing technologies*, pages 167–183. Springer, 2007.

[113] Thien Duc Nguyen, Samuel Marchal, Markus Miettinen, Hossein Fereidooni, N Asokan, and Ahmad-Reza Sadeghi. Dïot: A federated self-learning anomaly detection system for iot. In *2019 IEEE 39th International conference on distributed computing systems (ICDCS)*, pages 756–767. IEEE, 2019.

[114] Tao Ni, Xiaokuan Zhang, and Qingchuan Zhao. Recovering fingerprints from in-display fingerprint sensors via electromagnetic side channel. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, pages 253–267, 2023.

[115] Tao Ni, Xiaokuan Zhang, Chaoshun Zuo, Jianfeng Li, Zhenyu Yan, Wubing Wang, Weitao Xu, Xiapu Luo, and Qingchuan Zhao. Uncovering user interactions on smartphones via contactless wireless charging side channels. In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 3399–3415. IEEE, 2023.

[116] Haibin Niu, Xinyu Zhao, and Jing Li. 3d location and resource allocation optimization for uav-enabled emergency networks under statistical qos constraint. *IEEE Access*, 9:41566–41576, 2021.

[117] Colin O'Flynn. Min()imum failure: Emfi attacks against usb stacks. In *13th USENIX Workshop on Offensive Technologies (WOOT 19)*, 2019.

[118] Hiraku Okada, Hironori Oka, and Kenichi Mase. Network construction management for emergency communication system skymesh in large scale disaster. In *2012 IEEE Globecom Workshops*, pages 875–880. IEEE, 2012.

[119] Yi Ouyang, Xhengyi Le, Guanling Chen, James Ford, and Fillia Makedon. Entrapping adversaries for source protection in sensor networks. In *2006 International symposium on a world of wireless, mobile and multimedia networks (WoWMoM'06)*, pages 10–pp. IEEE, 2006.

[120] Andy Ozment and Stuart E Schechter. Milk or wine: does software security improve with age? In *USENIX Security Symposium*, volume 6, pages 10–5555, 2006.

[121] Francesco Palmieri. A distributed flow correlation attack to anonymizing overlay networks based on wavelet multi-resolution analysis. *IEEE Transactions on Dependable and Secure Computing*, 18(5):2271–2284, 2019.

[122] Miaoxin Pan, Chongcheng Chen, Xiaojun Yin, and Zhengrui Huang. Uav-aided emergency environmental monitoring in infrastructure-less areas: Lora mesh networking approach. *IEEE Internet of Things Journal*, 9(4):2918–2932, 2021.

[123] Kirtan Gopal Panda, Shrayan Das, Debarati Sen, and Wasim Arif. Design and deployment of uav-aided post-disaster emergency network. *IEEE Access*, 7:102985–102999, 2019.

[124] Mookyu Park, Haengrok Oh, and Kyungho Lee. Security risk measurement for information leakage in iot-based smart homes from a situational awareness perspective. *Sensors*, 19(9):2148, 2019.

[125] Frédéric Patricelli, James E Beakley, Angelo Carnevale, Marcello Tarabochia, and Dag KJE Von Lubitz. Disaster management and mitigation: the telecommunications infrastructure. *Disasters*, 33(1):23–37, 2009.

[126] Roberto Perdisci, Thomas Papastergiou, Omar Alrawi, and Manos Antonakakis. Iotfinder: Efficient large-scale identification of iot devices via passive dns traffic analysis. In *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 474–489. IEEE, 2020.

[127] Marius Portmann and Asad Amir Pirzada. Wireless mesh networks for public safety and crisis management applications. *IEEE Internet computing*, 12(1):18–25, 2008.

[128] P Devi Pradeep, B Anil Kumar, et al. A survey of emergency communication network architectures. *International Journal of u-and e-Service, Science and Technology*, 8(4):61–68, 2015.

[129] Premio Inc. Capacitive vs Resistive Touchscreen Panel Computers | Which one to use?, April 2022. URL `https://premioinc.com/blogs/blog/capacitive-vs-relative-touchscreen-panel-pcs`.

[130] Philipp Pütz, Richard Mitev, Markus Miettinen, and Ahmad-Reza Sadeghi. Unleashing iot security: Assessing the effectiveness of best practices in protecting against threats. In *Proceedings of the 39th Annual Computer Security Applications Conference*, pages 190–204, 2023.

[131] Yao Qin, Nicholas Carlini, Garrison Cottrell, Ian Goodfellow, and Colin Raffel. Imperceptible, robust, and targeted adversarial examples for automatic speech recognition. In *International conference on machine learning*, pages 5231–5240. PMLR, 2019.

[132] Rfe/rl. Marie Colvin's Death Raises Concerns About Use Of Satellite Phones, February 2012. URL `https://www.rferl.org/a/marie_colvin_death_concerns_about_safe_use_satelite_phones/24495230.html`.

[133] Ruben Rios and Javier Lopez. Exploiting context-awareness to enhance source-location privacy in wireless sensor networks. *The Computer Journal*, 54(10):1603–1615, 2011.

[134] Eyal Ronen, Adi Shamir, Achi-Or Weingarten, and Colin O'Flynn. Iot goes nuclear: Creating a zigbee chain reaction. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 195–212. IEEE, 2017.

[135] Nirupam Roy, Haitham Hassanieh, and Romit Roy Choudhury. Backdoor: Making microphones hear inaudible sounds. In *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*, pages 2–14, 2017.

[136] Nirupam Roy, Haitham Hassanieh, and Romit Roy Choudhury. Backdoor: Sounds that a microphone can record, but that humans can't hear. *GetMobile: Mobile Computing and Communications*, 21(4):25–29, 2018.

[137] Frank Sabath. What can be learned from documented intentional electromagnetic interference (iemi) attacks? In *2011 XXXth URSI General Assembly and Scientific Symposium*, pages 1–4. IEEE, 2011.

[138] Muhammad Salman, Nguyen Dao, Uichin Lee, and Youngtae Noh. Csi: Despy: enabling effortless spy camera detection via passive sensing of user activities and bitrate variations. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 6(2):1–27, 2022.

[139] Xhani Marvin Saß, Richard Mitev, and Ahmad-Reza Sadeghi. Oops..! i glitched it again! how to multi-glitch the glitching-protections on arm trustzone-m. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 6239–6256, 2023.

[140] Jörn-Marc Schmidt and Michael Hutter. *Optical and em fault-attacks on crt-based rsa: Concrete results*. na, 2007.

[141] Lea Schönherr, Katharina Kohls, Steffen Zeiler, Thorsten Holz, and Dorothea Kolossa. Adversarial attacks against automatic speech recognition systems

via psychoacoustic hiding. *arXiv preprint arXiv:1808.05665*, 2019. URL https://www.ndss-symposium.org/ndss-paper/adversarial-attacks-against-automatic-speech-recognition-systems-via-psychoacoustic-hiding/.

[142] Lea Schönherr, Thorsten Eisenhofer, Steffen Zeiler, Thorsten Holz, and Dorothea Kolossa. Imperio: Robust over-the-air adversarial examples for automatic speech recognition systems. In *Annual Computer Security Applications Conference (ACSAC)*, 2020.

[143] Lea Schönherr, Maximilian Golla, Thorsten Eisenhofer, Jan Wiele, Dorothea Kolossa, and Thorsten Holz. Exploring accidental triggers of smart speakers. *Computer Speech & Language*, 73:101328, 2022.

[144] Eric Hal Schwartz. Voice Assistants Very Prone to Accidentally Waking Up and Recording Long Audio Clips: Study - Voicebot.ai, February 2020. URL https://voicebot.ai/2020/02/21/voice-assistants-very-prone-to-accidentally-waking-up-and-recording-long-audio-clips-study.

[145] John Scott-Railton. Archived Tweet of John Scott-Railton. https://web.archive.org/web/20220301105339/twitter.com/jsrailton/status/1497745011932286979, February 2022.

[146] Jayaprakash Selvaraj, Gökçen Yılmaz Dayanıklı, Neelam Prabhu Gaunkar, David Ware, Ryan M Gerdes, and Mani Mina. Electromagnetic induction attacks against embedded systems. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, pages 499–510, 2018.

[147] Andrei Serjantov and George Danezis. Towards an information theoretic metric for anonymity. In *Privacy Enhancing Technologies: Second International Workshop, PET 2002 San Francisco, CA, USA, April 14–15, 2002 Revised Papers 2*, pages 41–53. Springer, 2003.

[148] Riaz Ahmed Shaikh, Hassan Jameel, Brian J d'Auriol, Heejo Lee, Sungyoung Lee, and Young-Jae Song. Achieving network level privacy in wireless sensor networks. *Sensors*, 10(3):1447–1472, 2010.

[149] Haoqi Shan, Boyi Zhang, Zihao Zhan, Dean Sullivan, Shuo Wang, and Yier Jin. Invisible finger: Practical electromagnetic interference attack on touchscreen-based electronic devices. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1246–1262. IEEE, 2022.

[150] Min Shao, Wenhui Hu, Sencun Zhu, Guohong Cao, Srikanth Krishnamurth, and Tom La Porta. Cross-layer enhanced source location privacy in sensor networks. In

2009 6th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, pages 1–9. IEEE, 2009.

[151] Gaurav Sharma, Stilianos Vidalis, Niharika Anand, Catherine Menon, and Somesh Kumar. A survey on layer-wise security attacks in iot: Attacks, countermeasures, and open-issues. *Electronics*, 10(19):2365, 2021.

[152] Rahul Anand Sharma, Elahe Soltanaghaei, Anthony Rowe, and Vyas Sekar. Lumos: Identifying and localizing diverse hidden iot devices in an unfamiliar environment. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 1095–1112, 2022.

[153] Yoshitaka Shibata, Yosuke Sato, Naoki Ogasawara, and Go Chiba. A disaster information system by ballooned wireless adhoc network. In *2009 international conference on complex, intelligent and software intensive systems*, pages 299–304. IEEE, 2009.

[154] Omer Shwartz, Amir Cohen, Asaf Shabtai, and Yossi Oren. Shattered trust: When replacement smartphone components attack. In *11th USENIX Workshop on Offensive Technologies (WOOT 17)*, 2017.

[155] Siddharth Sigtia, John Bridle, Hywel Richards, Pascal Clark, Erik Marchi, and Vineet Garg. Progressive voice trigger detection: Accuracy vs latency. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6843–6847. IEEE, 2021.

[156] Amit Kumar Sikder, Giuseppe Petracca, Hidayet Aksu, Trent Jaeger, and A Selcuk Uluagac. A survey on sensor-based threats to internet-of-things (iot) devices and applications. *arXiv preprint arXiv:1802.02041*, 2018.

[157] Akash Deep Singh, Luis Garcia, Joseph Noor, and Mani Srivastava. I always feel like somebody's sensing me! a framework to detect, identify, and localize clandestine wireless sensors. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 1829–1846, 2021.

[158] Arunan Sivanathan, Daniel Sherratt, Hassan Habibi Gharakheili, Adam Radford, Chamith Wijenayake, Arun Vishwanath, and Vijay Sivaraman. Characterizing and classifying iot traffic in smart cities and campuses. In *2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 559–564. IEEE, 2017.

[159] Michael Snodgrass and Yanyan Li. An empirical study of alexa skill system from malicious skill developers. In *2023 IEEE World AI IoT Congress (AIIoT)*, pages 0481–0486, 2023. doi: 10.1109/AIIoT58121.2023.10174411.

[160] Liwei Song and Prateek Mittal. Poster: Inaudible voice commands. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 2583–2585, 2017.

[161] Takeshi Sugawara, Benjamin Cyr, Sara Rampazzi, Daniel Genkin, and Kevin Fu. Light commands: Laser-based audio injection attacks on voice-controllable systems. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 2631–2648, 2020.

[162] Jinyuan Sun, Chi Zhang, Yanchao Zhang, and Yuguang Fang. Sat: A security architecture achieving anonymity and traceability in wireless mesh networks. *IEEE Transactions on Dependable and Secure Computing*, 8(2):295–307, 2010.

[163] Sining Sun, Ching-Feng Yeh, Mari Ostendorf, Mei-Yuh Hwang, and Lei Xie. Training augmentation with adversarial examples for robust speech recognition. *arXiv preprint arXiv:1806.02782*, 2018.

[164] Hirokazu Suzuki, Youichiro Kaneko, Kenichi Mase, Shigemitsu Yamazaki, and Hideo Makino. An ad hoc network in the sky, skymesh, for large-scale disaster recovery. In *IEEE vehicular technology conference*, pages 1–5. IEEE, 2006.

[165] Keiichi Tamura, Akitada Omagari, and Shuichi Hashida. Novel defense method against audio adversarial example for speech-to-text transcription neural networks. In *2019 IEEE 11th international workshop on computational intelligence and applications (IWCIA)*, pages 115–120. IEEE, 2019.

[166] Sam Tonkin. SpaceX Starlink has 150,000 daily users in Ukraine just five weeks after being activated, May 2022. URL `https://www.dailymail.co.uk/sciencetech/article-10781461/SpaceX-Starlink-150-000-daily-users-Ukraine-just-five-weeks-activated.html`.

[167] Yazhou Tu, Sara Rampazzi, Bin Hao, Angel Rodriguez, Kevin Fu, and Xiali Hei. Trick or heat? manipulating critical temperature-based control systems using rectification attacks. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 2301–2315, 2019.

[168] TUV Rheinland. TÜV Rheinland: Cybersecurity decides on the stability of societies, February 2020. URL `https://insights.tuv.com/blog/cybersecurity-decides-on-the-stability-of-societies`.

[169] Amit Kumar Tyagi and Deepti Goyal. A survey of privacy leakage and security vulnerabilities in the internet of things. In *2020 5th International conference on communication and electronics systems (ICCES)*, pages 386–394. IEEE, 2020.

[170] UN. Ukraine: Journalists targeted and in danger, warn top rights experts, May 2022. URL https://news.un.org/en/story/2022/05/1117462.

[171] Tavish Vaidya, Yuankai Zhang, Micah Sherr, and Clay Shields. Cocaine noodles: exploiting the gap between human and machine speech recognition. In *9th USENIX Workshop on Offensive Technologies (WOOT 15)*, 2015.

[172] Tim Verheyden, Denny Baert, Lente Van Hee, and Ruben Van Den Heuvel. Google employees are eavesdropping, even in your living room, VRT NWS has discovered, July 2019. URL https://www.vrt.be/vrtnws/en/2019/07/10/google-employees-are-eavesdropping-even-in-flemish-living-rooms.

[173] Aohui Wang, Ruigang Liang, Xiaokang Liu, Yingjun Zhang, Kai Chen, and Jin Li. An inside look at iot malware. In *Industrial IoT Technologies and Applications: Second EAI International Conference, Industrial IoT 2017, Wuhu, China, March 25–26, 2017, Proceedings 2*, pages 176–186. Springer, 2017.

[174] Haodong Wang, Bo Sheng, and Qun Li. Privacy-aware routing in sensor networks. *Computer Networks*, 53(9):1512–1529, 2009.

[175] Kai Wang, Richard Mitev, Chen Yan, Xiaoyu Ji, Ahmad-Reza Sadeghi, and Wenyuan Xu. Ghosttouch: Targeted attacks on touchscreens without physical touch. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 1543–1559, 2022.

[176] Kai Wang, Richard Mitev, Chen Yan, Xiaoyu Ji, Ahmad-Reza Sadeghi, and Wenyuan Xu. Analyzing and defending ghosttouch attack against capacitive touchscreens. *IEEE Transactions on Dependable and Secure Computing*, 2024.

[177] Yong Xi, Loren Schwiebert, and Weisong Shi. Preserving source location privacy in monitoring-based wireless sensor networks. In *Proceedings 20th IEEE International Parallel & Distributed Processing Symposium*, pages 8–pp. IEEE, 2006.

[178] Qi Xia, Qian Chen, and Shouhuai Xu. Near-ultrasound inaudible trojan (nuit): Exploiting your speaker to attack your microphone. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 4589–4606, Anaheim, CA, August 2023. USENIX Association. ISBN 978-1-939133-37-3. URL https://www.usenix.org/conference/usenixsecurity23/presentation/xia.

[179] Zhixin Xie, Chen Yan, Xiaoyu Ji, and Wenyuan Xu. Bitdance: Manipulating uart serial communication with iemi. In *Proceedings of the 26th International Symposium on Research in Attacks, Intrusions and Defenses*, pages 63–76, 2023.

[180] Qiben Yan, Kehai Liu, Qin Zhou, Hanqing Guo, and Ning Zhang. Surfingattack: Interactive hidden attack on voice assistants using ultrasonic guided waves. In

27th Annual Network and Distributed System Security Symposium, NDSS 2020, San Diego, California, USA, February 23-26, 2020. The Internet Society, 2020. URL https://www.ndss-symposium.org/ndss-paper/surfingattack-interactive-hidden-attack-on-voice-assistants-using-ultrasonic-guided-waves/.

[181] Ming Yang, Xiaodan Gu, Zhen Ling, Changxin Yin, and Junzhou Luo. An active de-anonymizing attack against tor web traffic. *Tsinghua Science and Technology*, 22 (6):702–713, 2017.

[182] Yi Yang, Min Shao, Sencun Zhu, and Guohong Cao. Towards statistically strong source anonymity for sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, 9(3):1–23, 2013.

[183] Park Joon Young, Jo Hyo Jin, Samuel Woo, and Dong Hoon Lee. Badvoice: Soundless voice-control replay attack on modern smartphones. In *2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN)*, pages 882–887. IEEE, 2016.

[184] Xuejing Yuan, Yuxuan Chen, Yue Zhao, Yunhui Long, Xiaokang Liu, Kai Chen, Shengzhi Zhang, Heqing Huang, Xiaofeng Wang, and Carl A Gunter. Commandersong: A systematic approach for practical adversarial voice recognition. In *27th USENIX security symposium (USENIX security 18)*, pages 49–64, 2018.

[185] Guoming Zhang, Chen Yan, Xiaoyu Ji, Tianchen Zhang, Taimin Zhang, and Wenyuan Xu. Dolphinattack: Inaudible voice commands. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, pages 103–117, 2017.

[186] Nan Zhang, Xianghang Mi, Xuan Feng, XiaoFeng Wang, Yuan Tian, and Feng Qian. Understanding and mitigating the security risks of voice-controlled third-party skills on amazon alexa and google home. *arXiv preprint arXiv:1805.01525*, 2018.

[187] Nan Zhang, Xianghang Mi, Xuan Feng, XiaoFeng Wang, Yuan Tian, and Feng Qian. Dangerous skills: Understanding and mitigating security risks of voice-controlled third-party functions on virtual personal assistant systems. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 1381–1396. IEEE, 2019.

[188] Xinlei Zhang, Zixiong Su, and Jun Rekimoto. Aware: intuitive device activation using prosody for natural voice interactions. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, pages 1–16, 2022.

[189] Yanchao Zhang and Yuguang Fang. Arsa: An attack-resilient security architecture for multihop wireless mesh networks. *IEEE Journal on Selected areas in communications*, 24(10):1916–1928, 2006.

[190] Youqian Zhang and Kasper Rasmussen. Detection of electromagnetic signal injection attacks on actuator systems. In *Proceedings of the 25th International Symposium on Research in Attacks, Intrusions and Defenses*, pages 171–184, 2022.

[191] Youqian Zhang and Kasper Rasmussen. Electromagnetic signal injection attacks on differential signaling. In *Proceedings of the 2023 ACM Asia Conference on Computer and Communications Security*, pages 314–325, 2023.

[192] Nan Zhao, Weidang Lu, Min Sheng, Yunfei Chen, Jie Tang, F Richard Yu, and Kai-Kit Wong. Uav-assisted emergency networks in disasters. *IEEE Wireless Communications*, 26(1):45–51, 2019.

[193] Jianguo Zhou, Changjia Zhou, Yuqin Kang, and Shenghui Tu. Integrated satellite-ground post-disaster emergency communication networking technology. *Natural Hazards Research*, 1(1):4–10, 2021.

[194] Huifeng Zhu, Zhiyuan Yu, Weidong Cao, Ning Zhang, and Xuan Zhang. Powertouch: A security objective-guided automation framework for generating wired ghost touch attacks on touchscreens. In *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*, pages 1–9, 2022.

List of Figures

Acronyms

AC Alternating Current

AES Advanced Encryption Standard

AFE Analog Front-End

ASL Audio Sequence Location

ASR Automated Speech Recognition

CAN Controller Area Network

CPU Central Processing Unit

DDoS Distributed Denial of Service

DoS Denial of Service

DuT Device under Test

DWT Discrete Wavelet Transform

ECM Electret Condenser

EMC Electromagnetic Compatibility

EMI Electromagnetic Interference

EM Electromagnetic

FPGA Field-Programmable Gate Array

FPR False Positive Rate

GAN Generative Adversarial Network

GVS Google Voice Search

IC Integrated Circuit

IoT Internet of Things

LoRa Long Range

LTE Long Term Evolution

MCU Microcontroller Unit

MEMS Micro-Electro-Mechanical System

MFCC Mel Frequency Cepstral Coefficients

MitM Man-in-the-Middle

OSI Open Systems Interconnection

OS Operating System

PoC Proof of Concept

RSA Rivest–Shamir–Adleman

SDM Scan Driving Method

TCP Transmission Control Protocol

TPR True Positive Rate

TTS Text to Speech

TVD Total Variation Denoising

UART Universal Asynchronous Receiver-Transmitter

UAV Unmanned Aerial Vehicle

UI User Interface

WER Word Error Rate

APPENDICES

A

Alexa Lied to Me: Skill-based Man-in-the-Middle Attacks on Virtual Assistants (AsiaCCS'19, CORE: A)

---

[107] Richard Mitev, Markus Miettinen, and Ahmad-Reza Sadeghi. Alexa lied to me: Skill-based man-in-the-middle attacks on virtual assistants. In *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security*, pages 465–478, 2019. https://doi.org/10.1145/3321705.3329842.

Alexa Lied to Me: Skill-based Man-in-the-Middle Attacks on Virtual Assistants

Richard Mitev
richard.mitev@trust.tu-darmstadt.de
Technische Universität Darmstadt
Germany

Markus Miettinen
markus.miettinen@
trust.tu-darmstadt.de
Technische Universität Darmstadt
Germany

Ahmad-Reza Sadeghi
ahmad.sadeghi@trust.tu-darmstadt.de
Technische Universität Darmstadt
Germany

ABSTRACT

Voice-based virtual personal assistants such as Amazon's *Alexa* or *Google Assistant* have become highly popular and are used for diverse daily tasks ranging from querying on-line information, shopping, smart home control and a variety of enterprise application scenarios[1]. Capabilities of virtual assistants can be enhanced with so-called *Skills*, i.e., programmatic extensions that allow third-party providers to integrate their services with the respective voice assistant.

In this paper, we show that specially crafted *malicious Skills* can use the seemingly limited Skill interaction model to cause harm. We present novel man-in-the-middle attacks against benign Skills and Virtual Assistant functionalities. Our attack uses loopholes in the Skill interface to redirect a victim's voice input to a malicious Skill, thereby hijacking the conversation between Alexa and the victim. To the best of our knowledge this is the *first man-in-the-middle attack targeting the Skill ecosystem*. We present the design of our attack and demonstrate its feasibility based on a proof-of-concept implementation attacking the Alexa Skills of a smart lock as well as a home security system.

ACM Reference Format:
Richard Mitev, Markus Miettinen, and Ahmad-Reza Sadeghi. 2019. Alexa Lied to Me: Skill-based Man-in-the-Middle Attacks on Virtual Assistants. In *ACM Asia Conference on Computer and Communications Security (AsiaCCS '19), July 9–12, 2019, Auckland, New Zealand.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3321705.3329842

1 INTRODUCTION

Personal voice-controlled virtual assistants are getting more and more popular and ubiquitous. There are numerous virtual assistants available on the market from different vendors like, e.g., *Amazon Alexa*, *Google Assistant*, *Apple Siri* and *Microsoft Cortana*. Virtual assistants can be integrated into other programs, built into OSs or into IoT devices like smart speakers, smart watches, appliances, cars or even clothing.

[1]https://aws.amazon.com/alexaforbusiness/

AsiaCCS '19, July 9–12, 2019, Auckland, New Zealand

ACM ISBN 978-1-4503-6752-3/19/07…$15.00
https://doi.org/10.1145/3321705.3329842

Recently, multiple new attacks against devices utilizing voice control-based UIs have emerged, mainly focused on issuing unauthorized commands without the legitimate user noticing this. The attacks typically utilize synthesized [1, 6], obfuscated [3, 21], embedded [16, 24] or ultrasound-modulated voice commands [13, 19, 25]. Besides sound, also other attack vectors exist, e.g., using electromagnetic interference [10] or attacks leveraging the headphone jack of a device [23]. There are also attacks exploiting the way we pronounce words to invoke a similar sounding Skill [12, 26].

What makes the voice-control interface of virtual assistants particularly interesting is that it can be augmented with various third-party service applications, called 'Skills', or 'Actions' that provide extended functionality beyond the basic functions of the virtual assistant, like, e.g., the possibility to control smart home appliances, or, access to specific information services. It is obvious that by misusing such 'Skills' an adversary could potentially cause much harm to the user. Therefore the interaction model of Skills is very limited: Skills can only be used to retrieve information from a third-party service or invoke actions explicitly exposed by the service.

Our goals and contributions. In this paper, we show that carefully crafted malicious back-end Skill functionality can be combined with known inaudible attack techniques to circumvent the seemingly limited Skill interaction model of a virtual assistant like Amazon Alexa to allow an adversary to *arbitrarily control and manipulate interactions* between the user and other benign Skills. We present a man-in-the-middle attack that *completely hijacks* a full conversation between a voice-controlled virtual assistant and the targeted user and is very hard to detect. The advantage of our approach using a malicious back-end Skill for the attack is that it works *in the context of an active user interaction* with the virtual assistant, completely *maintaining the interaction semantics* from the user's perspective. This allows the adversary to launch much more powerful attacks than simple command injection as presented in earlier work [3, 14, 16, 19, 21, 24, 25].

In our attack the adversary can arbitrarily modify the virtual assistant's responses, or, can entirely replace its functionality by her own, returning any responses of her choice to the victim that seem to be coming from genuine personal virtual assistant. Furthermore, our attack can make the responses seem plausible even to a suspicious and particularly vigilant user by maliciously *modifying genuine, plausible data* provided by benign Skills *on-the-fly*. With this Proof-of-Concept we show that designers of voice controlled applications in Skill ecosystems must take inaudible injection and jamming attacks into account.

To implement our attack we tackle a number of technical challenges: (1) How to capture user commands and redirect them to

a malicious Skill so that the original intention of the user is overridden? (2) How to exfiltrate genuine information from legitimate Skills so that it can be used to form fabricated responses to the user that seem plausible? (3) Finally, how to orchestrate the technical components required by the attack seamlessly together to create the illusion that the user is conversing with the legitimate Alexa functionality or Skills, while it in reality is interacting with a malicious Skill.

Contributions Our main contributions are as follows:

- We present *Lyexa*, the (to the best of our knowledge) *first man-in-the-middle attack against personal virtual assistants* (Sect. 3).
- The implementation of a *malicious Skill framework* that can *convincingly impersonate* the behavior of virtual assistants and benign Skills associated with it, while simultaneously being able to modify this behavior as desired by the adversary (Sect. 4).
- A Proof-of-concept evaluation of the Lyexa attack framework in different rooms and environments, on two smart home security systems and the 10 top Skills of the U.S. Skill store controlled by an *Amazon Echo Dot* virtual assistant device (Sect. 5).

In this paper, we focus on the most popular virtual assistant, i.e., Amazon Alexa. However, we stress that our attack can be modified to cover also other similar virtual assistants from other vendors in a similar fashion. The implementation of the attack components is described in Sect. 4, and suggestions for countermeasures against this attack are given in Sect. 7.

2 PRELIMINARIES

The most popular voice-controlled virtual assistant utilising a smart speaker device placed in users' homes is *Alexa*[2], short for Alexa Voice Service (AVS), from Amazon, with a market share of ca. 70% in US households in 2018. [11]Typically Alexa is integrated into a smart speaker device like the *Amazon Echo* or its smaller variant *Amazon Echo Dot*. A similar set-up applies also to Google's comparable service, the *Google Assistant*. These smart speaker devices are equipped with microphones and loudspeakers in order to facilitate voice-based interaction with the user of the device. As shown in Fig. 1 users issue voice commands to their virtual assistants by uttering a so called *wake word* like "Alexa", or, "Ok, Google". When the corresponding smart speaker recognizes the wake word, it starts recording audio to capture any subsequent commands given by the user (1). The recorded audio is forwarded to the corresponding back-end service like Amazon Voice Service or Google Assistant for speech-to-text translation (2). Depending on the issued commands, the back-end service will look up information or initiate actions (3) and respond with a reply (4) that the smart speaker will play back to the user (5).

In addition to this basic usage scenario, virtual assistant functionality can also be built into dedicated third-party devices. Examples include *BMW cars*[3], where Alexa is able to control the air conditioning system of the car, or, *LG ThinQ Smart fridges and ovens*[4] where Alexa can for instance be used to preheat the oven.

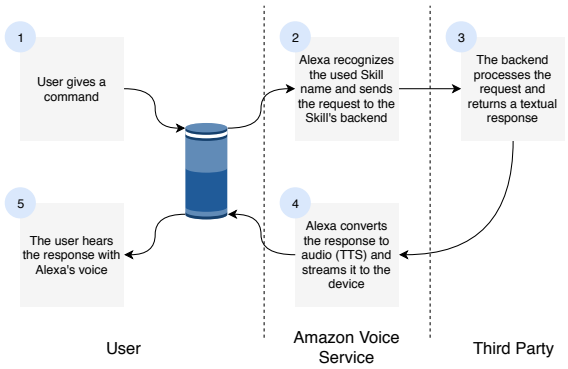


Figure 1: User interaction flow while using a Skill

2.1 Skills and Actions

The functionality of Alexa can be extended by so-called *Skills*[5] (called *Actions*[6] in Google Assistant). These are third-party service plug-ins that extend the virtual assistant's functionality and allow it to interact with third-party services, generating an entire ecosystem around AVS. Any developer (private or organization) can develop Skills free of charge. Examples of third-party Skills include, e.g., ordering pizza through Domino's Pizza Skill or calling a Uber ride through Uber's Skill, to name a few. Utilizing appropriate Skills, Alexa can also be used for controlling user's smart home IoT devices given that the manufacturers of these devices provide the appropriate Skill for controlling them through Alexa.

In contrast to apps on popular smartphone platforms like Android or iOS, which are essentially pieces of software running on the host smartphone, Skills are different. They don't contain executable code that could be run on devices and they can't thus change the behavior of the Alexa-enabled device itself. Skills are merely third-party-provided service extensions to AVS and can only react to invocations, i.e. telling Alexa what to echo to the user in response to specific requests to the Skill. The Lyexa attack framework utilizes a specially-crafted malicious Skill to make Alexa speak to the user what the adversary wants, mimicking the behavior of genuine Skills or Alexa functionalities. The adversary can thereby exploit the Alexa ecosystem for different attacks without ever having to compromise the Alexa-enabled device itself.

Skills need to be registered in Amazon's Skill repository, from where users can activate them to be used. The activation of Skills can happen through the Alexa app, webpage, or, using a voice command. To be listed in the repository each Skill has to pass a certification process in which Amazon tests the Skill to verify that it works as specified. However, as the Skill is running on a remote

[2]https://www.amazon.com/Amazon-Echo-And-Alexa-Devices/b?node=9818047011

[3]https://www.gearbrain.com/which-cars-have-amazon-alexa-2525958778.html

[4]https://www.cnet.com/news/lg-instaview-thinq-alexa-fridge-clever-kitchen-tricks-ces-2018/

[5]https://developer.amazon.com/alexa-skills-kit

[6]https://developers.google.com/actions/

server, AVS can't control whether a Skill is modified after it has passed the certification.

2.2 Inaudible Speech Injection

Recent attacks against virtual assistants by Song *et al.* [19] and Zhang *et al.* [25] utilize the non-linearity of microphones beyond the audible frequency spectrum to issue commands to a voice-controlled virtual assistant that can't be heard by the user. This is achieved by exploiting physical characteristics of the microphones typically used in such devices. By carefully injecting selected signals at high frequencies that lie beyond the hearing capacity of humans, it is possible to create 'shadow' signals in the microphone within the audible frequency spectrum [14] and use this for issuing inaudible commands. These works assume a malicious device in proximity to the targeted Voice Assistant injecting the inaudible audio signal. We think this to be a realistic assumption and adopt the same adversarial scenario, assuming that the attacker is able to inject signals into the victim's audio environment. To show that Skills can be used for attacks we realise the Lyexa attack where we will take use of this inaudible command injection technique together with inaudible jamming of user commands to redirect the user interaction from the Skill the user intended to the adversary's malicious Skill as described below.

3 OUR ATTACK LYEXA

Our goal with the Lyexa attack is to show that even though the functionality and interaction model of Skills is very limited, it is possible to craft Skills with malicious functionality that can be used to construct harmful attacks against users by utilizing compromised IoT devices in the vicinity of the Alexa device for realizing the attack. The Lyexa attack has been designed under the assumption that the Alexa devices and the AVS ecosystem have no exploitable vulnerabilities and that the traffic is properly secured (e.g., TLS, no ARP spoofing). We think this to be a realistic assumption since Amazon spends considerable resources in designing and testing their devices and would also have the necessary means for quickly updating or patching their devices in case security vulnerabilities affecting them were to be found. With this attack we want to show how the AVS ecosystem can be exploited using Skills for unintended purposes *without* compromising individual components as such.

However, as recent numerous reports about IoT devices with security vulnerabilities suggest, many IoT devices in the smart homes of users can be exploited relatively easily by automated security attacks like those performed by IoT malware like *Mirai* [2], *Hajime* [7] or *Persirai* [22]. Especially devices like IP cameras seem to be often affected by such attacks, as many of them can be easily exploited due to insecure security configurations like easy-to-guess administrator passwords.

3.1 Attack Overview

We developed and verified the Lyexa attack on Amazon's AVS ecosystem, as it is currently the most popular voice-controlled virtual assistant platform for smart speakers. [11] However, as virtual assistants are conceptually similar, our attack can likely be extended also to other virtual assistants. A conceptual overview of the Lyexa attack scenario is shown in Fig. 2. It involves a smart home user U, a voice-controlled virtual assistant device E like the Amazon Echo connected to the Alexa Voice Service (AVS), a malicious IoT device D and a malicious Skill S in the AVS ecosystem. The adversary is a remote attacker controlling both malicious components S and D and able to take use of a speech-to-text service or library STT, many of which are readily available.

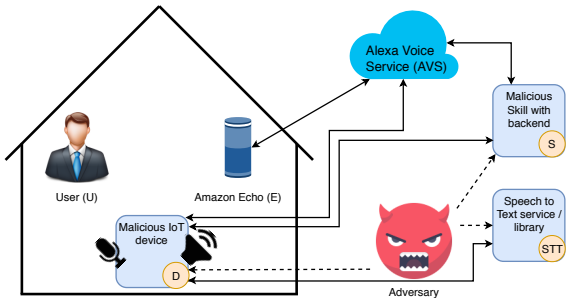


Figure 2: Overview of the Lyexa attack, dotted lines represent a "can control" or "has access to" relation, solid lines show that objects have a connection

3.1.1 Assumptions. The malicious IoT device D is located in the vicinity of E (e.g., on the same living room table) and is equipped with a microphone and a loudspeaker capable of emitting ultrasound signals. We will show in Sect. 5.2.1 that numerous different kinds of entry-level IoT devices like IP cameras are equipped with hardware that are likely to satisfy these requirements.

Device D is accompanied by a malicious Skill S under the control of the adversary. This Skill has to be activated on the victim's AVS account. Note that since it is possible to enable Skills by voice[7], it is possible that device D could use inaudible speech injection as described in Sect. 2.2 to activate Skill S by itself.

3.2 Attack Components

Our attacks consists of four distinct components, as depicted in Fig. 3 and discussed below.

3.2.1 Command jamming. Figure 3(a) shows how commands issued by user U are inaudibly jammed and simultaneously recorded by malicious device D. When U speaks the wake word ("Alexa") both the benign Alexa-enabled device E and malicious device D are activated (1). Benign device E starts listening for subsequent commands and D starts jamming this command with ultrasound modulated noise, which is inaudible for humans as evaluated by Roy et al. [14] (2). This way the command issued by U can't be understood by E. D simultaneously records the command.

3.2.2 Malicious Skill invocation. The second attack component is depicted in Fig. 3(b). When user U finishes speaking the command (1), D immediately stops jamming and inaudibly injects a Skill invocation command with malicious Skill S's invocation name to E which is still listening for a command (2). E will forward the injected audio command to AVS (3), which interprets it and invokes malicious Skill S (4). Malicious Skill S is now started and can return

[7] https://www.amazon.com/gp/help/customer/display.html?nodeId=201848700

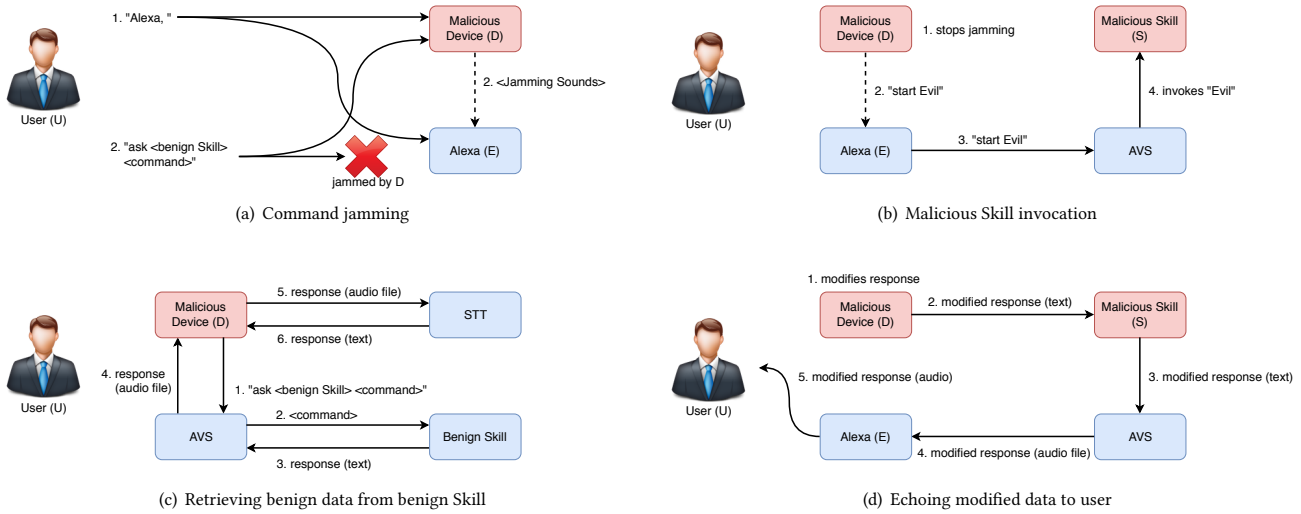


Figure 3: Depiction of the four attack components needed for the Man-in-the-Middle attack

arbitrary text to be echoed through E in Alexa's voice. The inaudible command injection is realized by utilizing the amplitude modulation technique discussed in Sect. 2.2.

3.2.3 Retrieving benign data from benign Skill. The purpose of the third attack component is to fetch benign data from a benign Skill that user U wanted to invoke with his command. This is shown in Fig. 3(c). D first starts a new session with AVS and forwards U's recorded command to it, just like a benign Alexa-enabled device would do (1). AVS will then invoke the requested Skill and pass the command to it (2). Subsequently, the benign Skill will return a textual representation of the response to AVS (3) which will then use the Alexa TTS engine to create a voice audio file out of it and forward it to D (4). Now the malicious device D will send the audio file containing the response to a *STT* service (5) for it to be transcribed back into text (6). Now D knows what data the victim wanted to fetch from the requested Skill.

3.2.4 Echoing modified data to user. The fourth component, depicted in Fig. 3(d), shows how modified data is finally echoed to the victim user U. After arbitrarily modifying the response data received from the benign Skill in the previous step (1), D passes this text to the backend of Skill S (2). S then passes the received text to the already established AVS session (3) which will then create an audio file with the help of Alexa's TTS engine. This audio file is passed to E (4) to be played back to user U (5).

3.3 Hijacking Built-In Alexa Commands

The most straightforward attack variant aims at hijacking built-in Alexa commands like "Alexa, will it rain today?" (i.e., commands containing no Skill name, as shown in Fig. 4 segment 0). The command can be redirected to malicious Skill S by associating the utterance "will it rain today" with an intent of Skill S, and letting malicious IoT device D inject an inaudible voice command "using *Evil*" (where *Evil* is the invocation name of Skill S) when the user stops speaking to Alexa, as shown in Fig. 4 segment 1. This will cause AVS to invoke Skill S using an intent corresponding to this utterance. The malicious Skill S can then return any desired response to AVS, which will be echoed back to user U using Alexa device E, thereby making the user believe he is talking to Alexa's built-in functionality and not to Skill S.

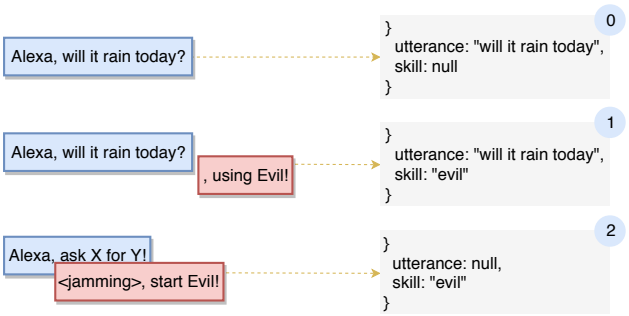


Figure 4: Abstract representation of how injected voice commands lead to different interpretations of the intended command. Segment 0 depicts a command activating a built-in feature, segment 1 and 2 are depicting redirections to the Skill "Evil".

3.4 Skill Redirection Attack

One drawback of the above attack is that it can't be used to redirect commands that are targeted at a particular benign Skill. For example, if the user wants to lock his smart lock *front door* with the help of the *Security* Skill, he will issue the command "Alexa, ask *Security* to lock the *front door*". Just injecting the Skill S's invocation name after the command ("using *Evil*") would not be sufficient, as the

resulting command heard by E would contain the invocation name of two different Skills, resulting in AVS ignoring this command.

In order to redirect Skill invocations, we need to enhance the attack by *preventing Alexa from understanding* the genuine Skill's invocation. The obvious way to achieve this is by device D injecting inaudible noise, as depicted in Fig. 4 segment 2, at the same time the user is speaking the command, essentially jamming the user's input so that Alexa will not be able to understand it (a similar jamming attack, however not targeting voice assistants, has recently been published independently to our work by Roy *et al.* [14]).

Our evaluation of jamming of user issued commands (cf. Sect. 5.1.2) showed that to be effective, D needs to jam the user command starting immediately after the wake-word, up until the command is finished. This means D needs to start jamming even *before* it can know which command the user is going to issue. To still be able to provide meaningful responses to the user's command the malicious device D needs to independently record the command of the user while *simultaneously* jamming it, as shown in Fig. 3(a). This is feasible because different microphones are sensitive to different AM carrier wave frequencies. Device D's microphone will therefore not pick up the emitted inaudible jamming signal as it is not sensitive to the same frequency as the microphone of the targeted Alexa device. After recording the command, D uses the speech-to-text service STT to convert the recorded audio to text and forwards it to Skill S's backend. After the user command has been jammed, D executes the Malicious Skill invocation attack component (Fig. 3(b)) by inaudibly injecting the redirection command to invoke Skill S and cause AVS to start a session with it.

If the transcribed user command text returned by STT contains a Skill name or utterance the adversary wants to hijack, the malicious Skill S returns a reply to AVS to be played back to the user, as depicted in Fig. 3(d) (Echoing modified data to user component). This essentially denies the user access to this Skill while simultaneously making him think the response is coming from the genuine Skill.

However, if the user command does not contain Skill names or utterances the adversary wants to hijack, it still needs to provide a plausible response to the user in order to avoid the user realizing that he is the victim of an attack. For realizing this the adversary can utilize the benign functionality of the AVS by playing a *Skill-in-the-Middle* attack as discussed below.

3.5 Skill-in-the-Middle Attack

In this variant of our attack the adversary stages a full man-in-the-middle attack between the user and benign Alexa Skills. After performing the Command jamming and Malicious Skill invocation attack components as described above, device D will use the Retrieving benign data from benign Skill attack component as shown in Fig. 3(c) over a *separate session* with AVS to get benign data from the Skill that user U wanted to invoke. After obtaining the benign data, device D can process it and modify it in any way it wants and use the Echoing modified data to user attack component (Fig. 3(d)) to return a forged response to the user with the help of malicious Skill S. The Skill-in-the-middle attack gives the adversary thus full control over the conversation of the victim user U and any Alexa functionalities, in particular, Skills.

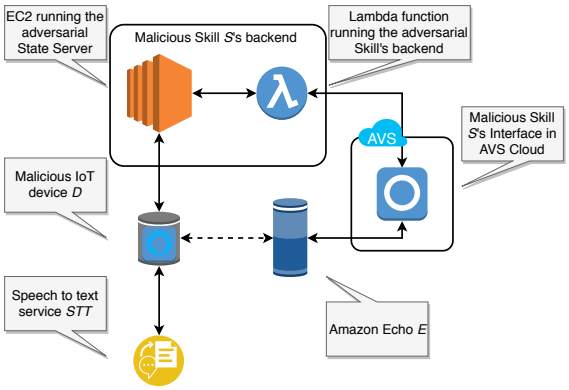


Figure 5: Components of the attack setup

The Skill-in-the-Middle attack is particularly devastating, as an adversary can use it, e.g., to modify arbitrary responses of benign Skills. For example by manipulating the responses of a Skill providing the latest quotes on share prices it can falsify the quotes of particular companies' stock in a way that may trick the victim into making valuable financial transactions based on false information with potentially far-reaching consequences.

Note that all attack variants presented above use Alexa's genuine voice and device (which the user typically trusts) to deliver the modified or wrong information, giving the adversary the potential to substantially mislead the user or fool him to take specific actions as desired by the adversary.

4 IMPLEMENTATION

The overall design of our system is shown in Fig. 5. It consists of a malicious IoT device D and Skill S. The Skill is implemented with the help of a *lambda function* and a state server that are both under control of the adversary. Our implementation also takes use of an online speech-to-text service STT.

4.1 Malicious IoT Device

To evaluate our attack we constructed a prototype of a malicious IoT device comprising following relatively inexpensive off-the-shelf hardware components shown in Fig. 6: a Fostex FT17H wide-range tweeter speaker[8] (ca. US$ 70) connected to a YAMAHA R-S202D [9] amplifier (ca. US$ 200). With this setup we were able to produce signals in the frequency range of 500 Hz to 50 kHz. The amplifier amplifies signals with frequencies up to 100 kHz without much distortion (±3 dB), this makes it suitable for our purpose.

For recording sounds and measurements we used a *MiniDSP UMIK-1*[10] microphone. To run the software we used a *Lenovo ThinkPad T430*[11] laptop. This laptop has a sound card (and drivers) capable of generating signals with frequencies of up to 192 kHz.

[8] https://fostexinternational.com/docs/speaker_components/pdf/FT17H_rev3.pdf
[9] https://europe.yamaha.com/en/products/audio_visual/hifi_components/r-s202d/index.html
[10] https://www.minidsp.com/products/acoustic-measurement/umik-1
[11] https://www3.lenovo.com/us/en/laptops/thinkpad/t-series/t430/

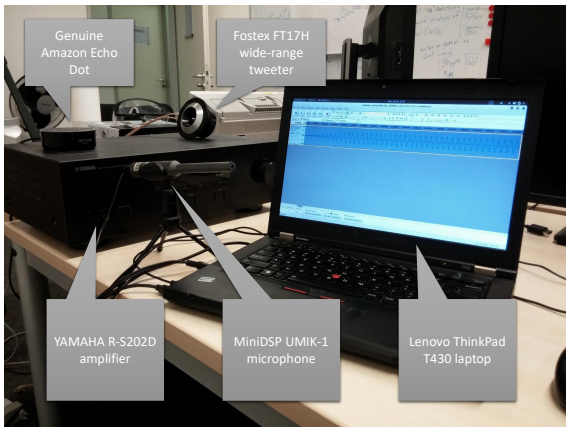


Figure 6: Photo of the hardware setup mimicking the malicious IoT device. The laptop is connected to the microphone and the amplifier. The tweeter is connected to the amplifier and aligned in the direction of the Amazon Echo Dot.

The malicious IoT device's software consists of a hotword detection engine called Snowboy[12], a library for communicating with AVS and means to connect to the STT service. The Snowboy engine was modified to also be able to save recorded audio, and return the status of detection. With these modifications it is possible to start the hotword detector with the pre-trained Alexa detection model, which is shipped with Snowboy, and record audio after the hotword is detected.

4.2 Malicious Skill

The malicious Skill S consists of two components: a state server and lambda function realizing the functionality of the Skill.

4.2.1 Adversarial State Server. When the lambda function of Skill S is invoked by AVS in a Skill redirection or Skill-in-the-middle attack, it initially can't know how to correctly react to the user command, as the invocation name of the intended Skill and any command parameters are jammed by D and thus not understood by AVS. Device D needs therefore a way to notify S the hijacked Skill's name and possible command parameters. For this it uses S's RESTful state server, which is continuously polled by S's lambda function. Device D uses POST instructions to store the instructions of what Skill to mimic or what text to reply on the state server, from where the lambda function will receive it by polling the state server. Polling of the state server is performed by the lambda function using GET requests once every second.

4.2.2 Adversarial Skill's Backend Lambda Function. Skill S's lambda function is implemented using *alexa-sdk* and *Node.js* on *AWS Lambda*[13], which is a fee-based Function as a Service (FaaS). AWS Lambda is the platform recommended by Amazon for Skill backends. To get a malicious Skill certified and into the official Alexa Skill store the adversary has to create a functional and apparently benign Skill that can pass the screening performed by Amazon. This needs to be done in a way that the Skill still provides a hidden entry point for the adversary to trigger the malicious functionality after the Skill has passed certification. This can be done either by directly triggering a dedicated intent for the malicious functionality, or, it can be activated when the issued command contains a specific keyword as a command parameter.

Naturally, the utterance triggering a malicious intent should be known only to the adversary, which means that it needs to be hidden from regular users. This is, however, not difficult, as there is no apparent way (as a normal user) to get a list of all utterances a Skill can understand or react to. In the certification process only three different commands have to be declared to be listed in the official interface description of a Skill in Amazon's Skill repository. Any additional commands can be left 'invisible' to regular users.

All commands and intents the Skill understands are tested by Amazon during certification for their functionality, but after passing certification the adversary can change their behavior in its backend system without having to re-certify the Skill. Similarly, the adversary can change the behavior of an intent in its backend in response to specific Slot values triggering malicious functionality. Effectively, malicious Skill S will therefore have two handlers: one for its public 'benign' functionality and a second one realizing malicious functionality. The second handler is triggered by the first handler when it receives a malicious functionality-triggering intent or a specific keyword value as a command parameter.

4.2.3 Malicious Skill's Interaction Model. A Skill's Interaction Model is a mapping of the user's input to intents, which are passed to the Skill's backend. The user's input consists of utterances which can include Slots. From the available Skill types[14] we utilize so-called Custom Skills for our attack since they provide the most flexible invocation options and functionality. An example command to invoke a Custom Skill could be: "Alexa, ask *Recipes* how do I make an omelet." where *Recipes* is the invocation name and "how do I make an omelet" is an utterance for the Recipes Skill to process[15, 16].

Our malicious Skill's Interaction Model is quite simple. There are only four intents of which three provide benign functionality and are publicly advertised as well as one malicious intent that is not disclosed to users. This intent consists of a genuine looking utterance and an utterance consisting solely of two Slots for which we do not provide sample values. Normally a Slot would match exactly one word (or more with provided samples), but this utterance consisting of two times the same Slot separated by a space will match any number of words. This way Alexa can be tricked into passing S the full transcript of the user command as a Slot value and match every possible sentence the user says (as we might not know what the impersonated Skill wants the user to say).

4.2.4 Skill Certification. As stated in Sects. 2.1 and 4.2.2, malicious Skill S needs to be certified to be listed in the Alexa Skill Store so that it can be enabled on every Alexa-enabled device via companion app or voice command. Our Skill S disguises itself as a Skill for

[12] https://snowboy.kitt.ai/
[13] https://aws.amazon.com/lambda/features/
[14] https://developer.amazon.com/docs/ask-overviews/understanding-the-different-types-of-skills.html
[15] https://developer.amazon.com/docs/custom-skills/understanding-how-users-invoke-custom-skills.html
[16] https://developer.amazon.com/docs/custom-skills/understanding-custom-skills.html

retrieving statistics about YouTube channels. It is called "YouTube Statistics (Unofficial)" and it's invocation name is "you statistics". We were able to get our Skill certified and added to the Store.

The benign functionality of Skill S returns statistics about YouTube channels. It can be queried to return for a given channel name the number of subscribers, total number of views, the title of the latest activity or the title of the latest playlist. We didn't disclose the corresponding utterance for querying playlist titles, so this intent can be used for triggering the malicious functionality. In addition, if an utterance with a Slot value "evil" is passed to the skill, a state variable is set, which will cause the malicious handlers to handle all intents instead of the handlers providing the benign functionality.

We minimized the potential exposure of users to our malicious Skill functionality by disabling listening for the utterance or Slot value when no evaluation was done. In addition, the Skill was available only briefly in the Skill store, as development work was done using an unlisted development Skill. We are therefore certain that no user enabled the malicious functionality.

4.3 Attack Signal Generation

We implemented the method for shifting voice samples of utterances and the jamming sound into the ultrasonic spectrum with MATLAB. The details of the used signal modulation are presented in Appendices A and B. We used for low-pass filtering the Equiripple single-rate FIR filter[17] and we were able to automatically create ultrasonic audio files with different carrier wave frequencies from 20 kHz to 40 kHz for evaluation purposes. We encoded the audio files with the Free Lossless Audio Codec (FLAC), a lossless audio format suitable for storing also ultrasonic frequencies.

Stealthiness improvements. With our initial setup a crackling/pop sound could be heard when playback of the ultrasonic audio file was started and stopped. We removed this unwanted noise with a linear fade-in of 0.16 ms corresponding to 30 samples at a sampling rate of 192 kHz and a linear fade-out of one second (192000 samples) applied on an added one second of silence to the original audio file. As a result, the starting and ending of the ultrasound injection can't be heard as a clearly audible sound.

4.4 Putting it All Together

A user typically starts an interaction by uttering the wake-word "Alexa". When the wake-word is detected by the malicious IoT device D, it immediately starts jamming the user's subsequent command by using a pre-recorded and modulated audio file with a duration of > 7 seconds and consisting of sawtooth signal noise. While D is jamming the user command, it also records it. This is possible because D will not record its own jamming sound, as the microphone of D is not sensitive to demodulating signals with the same carrier wave frequency as the Alexa-enabled device's microphone. When the user stops speaking, D stops jamming and inaudibly injects malicious Skill S's invocation name from a pre-recorded and modulated audio file.

To handle the recorded user command D sends it to a speech-to-text Service STT which transcribes it and returns the most probable transcriptions as text. We are using the Bing Speech API[18]. If the transcription contains a command or invocation name that D wants to hijack, it sends an instruction to the state server to mimic this specific Skill and command. Otherwise D sends the recorded audio to AVS. The AVS backend will then process the command and return a JSON payload containing a Speak Directive with a binary attachment containing MP3 encoded audio. To get the response in textual form this audio has to be transcribed by STT as well. After D gets the transcription its sends it as an instruction to the state server telling S to return the transcribed audio text via AVS to the user. If the genuine Skill expects a response from the user, the Skill's reply will prompt the user to provide this response. In this case D will proceed to recording the user's response, send it response to STT for transcription, forward this to the genuine Skill through AVS and, receive the response, transcribe it again through STT and finally send the transcription as an instruction to S for playback to the user. In this way the adversary is are able to conduct a complete MITM attack on the user's genuine Alexa-enabled device and the requested genuine Skill.

5 EVALUATION

The evaluation of our attack was carried out in an isolated office room measuring about 4 times 6 meters. The background noise was around 50 dB. The ultrasonic speaker was placed about 30 cm above an Amazon Echo Dot, our Alexa device, to avoid coupling effects.

5.1 Attack Components' Performance

5.1.1 Adversarial Skill invocation. First we needed to find the carrier wave frequency on which Amazon Echo Dot demodulates the strongest signal. We injected the command "start *Evil*", where *Evil* is our uncertified Skill's invocation name. Our evaluation in App. C shows that using carrier wave frequencies from 22 kHz to 23 kHz provides the best performance in terms that AVS is able to optimally understand the injected voice command.

5.1.2 What to jam. To test how AVS and the Alexa-enabled device (e.g. Echo Dot) react to jamming, we used white noise to jam parts of a command in order to find the best jamming strategy.

Setup. To evaluate what we have to jam and how AVS reacts to it we used the (syntactically incorrect) command: "Alexa, Ask X for Y, ask Evil for Z", with the invocation names X and Evil and utterances Y and Z. For testing purposes we used audible white noise and added it to parts of the command audio sample recorded from a male person. We experimented with jamming different parts of the command in order to see which parts of the command needed to be jammed to make AVS not understand the benign command anymore and instead understand our injected command. The results are shown in Tab. 1.

Jamming benign Skill name. Jamming the benign Skill's invocation name "X", resulted in the effective command "Alexa, Ask for Y, ask Evil for Z". AVS started "Z" unreliably (in $< 50\%$ of trials) if it was also a built-in functionality like "tell me a joke". If "Z" was only an utterance for "Evil" we were not even able to make AVS launch "Evil" with the utterance "Z".

[17] https://www.mathworks.com/help/signal/ref/equiripple.html

[18] https://azure.microsoft.com/en-us/services/cognitive-services/speech/

Table 1: Results of jamming different parts of the command.

Part to jam	Result
Only X	Alexa starts unreliably Z if it is also a built-in function
Ask and first part of X	Alexa starts unreliably Evil with Z
Ask X for Y	Alexa starts reliably Evil with Z

Command: "Alexa, Ask X for Y, ask Evil for Z"

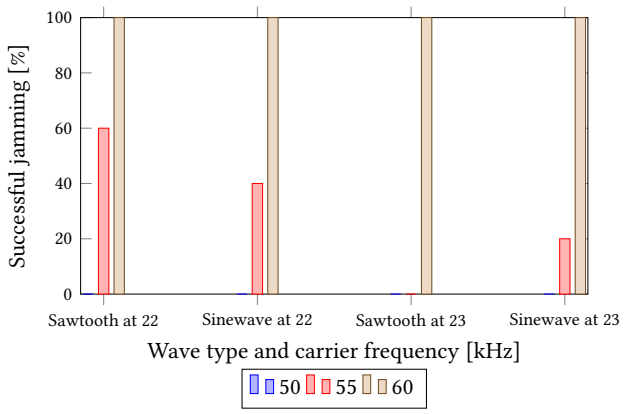


Figure 7: Results of jamming the voice sample "tell me a joke" recorded from a male person, arriving with an air pressure of 73 dB at the Amazon Echo Dot, with a sawtooth and sine wave signal at 1000 Hz, modulated with different carrier waves, from 30 cm distance, at different volume levels (50-60/100)

Jamming Ask and first part of X. We only jammed "Ask" and first part of "X" leaving the recognizable part of the command as "for Y, ask Evil for Z". If the added noise was loud enough so that AVS was not able to understand the jammed part correctly it interprets it as "ask Evil for Z" unreliably (in < 50% of trials). This shows that the leftover utterance "for Y" is still confusing AVS.

Jamming the whole benign command. Finally we added noise to the complete benign command "Ask X for Y". AVS was now able to understand "ask Evil for Z" reliably, and therefore started our malicious Skill (Evil) with the utterance "Z".

Result. The results show that jamming has to start immediately after the wake-word and last just until the inaudible command injection starts (i.e., the entire benign command has to be jammed). This means that jamming needs to start *before* the attacker knows which command the user will issue, otherwise Alexa will recognize the genuine Skill name and will ignore the redirection to the malicious Skill S or will be confused by those parts of the benign command that have not been jammed.

5.1.3 User command jamming. To evaluate command jamming, we recorded a human male saying "tell me a joke". This command starts the built-in functionality of Alexa to tell a joke and therefore should be easy for Alexa to understand correctly, because it uses no arbitrary Skill names. This sample was played with a sound pressure of 73 dB at the Echo Dot. We modulated a 1000 Hz sawtooth and a 1000 Hz sine wave signal with a carrier wave with a frequency of 22 kHz and 23 kHz to induce a noise signal in Echo Dot's microphone. The results of jamming with 10 trials each are shown in Fig. 7. At volume level 60 all signal types are able to jam the voice sample successfully. The sawtooth signal was barely audible whereas the sine wave-based signal was even less audible.

5.1.4 Attack performance. We evaluated injection and jamming at a distance of 30 cm. The greater the distance between D and E, the higher sound pressure [14, 25] is required. With our setup the maximum distance for reliable injection and jamming we could achieve was 140 cm between the ultrasonic speaker and Echo Dot with a success rate of 100% at a volume level of 70/100 for injection and 60% for jamming with 10 trials each.

We evaluated the Skill redirection attack on 10 "top skills" of the US Alexa Skill store and two smart home security Skills "Nuki" and "Homematic IP" in an isolated room. We invoked each Skill using the 1-3 example utterances displayed on each Skill's store page. Every Skill was invoked 10 times by two different persons. Each attack was executed as described in Sect. 4.4, i.e., by jamming the user command after the wake-word and injecting a command to invoke the malicious Skill after the user stopped speaking. This results in an attack success rate of 75 %. Tab. 2 shows the percentage of failure for each module (jamming, injection and STT). The most prevalent reason for the attack to fail was due to a failure in the injection of the malicious Skill's invocation name. We observed that even when the recorded audio of the injected invocation-name was very good (the audio captured by AVS can be inspected through the "history" tab in the Alexa Apps settings), AVS still sometimes misheard the command, resulting in a failure to start the malicious skill. Jamming and speech-to-text translation (STT) of the command were almost always successful. An asterisk next to the STT value in Tab. 2 indicates minor transcription errors (e.g. "new key" instead of "Nuki", or "jurassic park" instead of "jurassic bark") which were systematic and can therefore easily be corrected by malicious device D using an extended keyword list to capture desired utterances. They were thus not counted as STT module failures.

We also tested how accurately Skills are triggered without adversarial influence by invoking each Skill ten times, as described above. In the benign setting the requested Skill was successfully started with the correct command in 87 % of the trials. An invocation trial was considered successful, if the command was correctly transcribed by AVS as displayed by the "history" tab in the Alexa app and the correct Skill was started. As can be seen, the attack deteriorates the accuracy of Alexa's responses from a user point of view, but only slightly (from 87 % to 75 %). It is therefore questionable whether a user would be able to notice the presence of attacks only based on the changed behavior of AVS.

To evaluate whether different room layouts or interiors affect the attack we carried out tests in a conference room, different office rooms with up to 6 persons in it and even a crowded exhibition hall. We found no degradation of the attack success rate based on these factors, however, a large enough background noise pressure like in the crowded exhibition hall can cause our attack to fail (when the injected command sound level is less than the background noise

Table 2: Results of attacking the 10 "top skills" from the US Skill store (Nov. 2018) and two smart home security Skills, using example commands (1-3) listed on the store page

Skill Name	Jamming	Injection	STT
Flash Briefing Skills			
Reuters TV (U.S.) Fox News NPR News Now	7 %	10 %	0 %*
Custom Skills			
Jeopardy!	5 %	0 %	15 %
Find My Phone	0 %	22 %	6 %
Jurassic Bark	0 %	22 %	0 %*
Question of the Day	0 %	15 %	0 %*
Sleep Sounds: Ocean Sounds	5 %	20 %	0 %
Sleep Sounds: White Noise	25 %	25 %	0 %*
Sleep and Relaxation Sounds	0 %	11 %	6 %
Nuki	5 %	20 %	0 %*
Smart Home Skills			
Homematic IP	0 %	15 %	5 %*

level) but such noise levels are not common in private homes where Alexa devices are typically deployed.

5.2 Proof-of-Concept Attack

We implemented two proof-of-concept implementations of our attack for two Alexa-controlled smart-home devices, the Homematic IP Security System [19] and the Nuki smart lock[20]. Both security Skills can be activated or locked through Alexa. The goal of the attack is to prevent the user from locking his smart lock or arming of the smart home security system while simultaneously leaving him believing Alexa has armed or locked it properly. The Homematic IP Security System uses a Smart Home Skill for arming the home security system. When the Skill-redirection attack is run and the user says "Turn on absence mode", the command is redirected to the malicious Skill which will return a reply that is identical with that of the benign Skill, requesting further information on what device is to be armed: "Sorry what device?". The user can reply to this anything, the malicious Skill will always just return "Okay." without notifying or contacting the genuine Skill at any stage. The result of the attack is that the security system remains disarmed, even though the user thinks it is armed, because the exact same wording is used by the malicious Skill than what the genuine Skill would use. The Nuki Smart Lock uses a Custom Skill. If a user wants to close a lock and says "Ask Nuki to lock *front door*.", where *front door* is the name of the lock to be locked, the malicious Skill will directly return "Smart lock front door is locking.", again using a wording that is identical with the benign Skill's reply.

[19] https://www.homematic-ip.com/
[20] https://nuki.io/en/smart-lock/

In a Skill-in-the-middle scenario if the user wants to hear recent stock prices for a company using a popular stock market Skill called *Stock Prices by Opening Bell*[21] and asks "Ask *Opening Bell* for the price of Amazon", malicious device *D* forwards this command to Alexa's backend, and gets a response which is formed like "Amazon is trading at 1599.01. Down 1.97%.". Device *D* can modify these values at will and return this forged response with the help of malicious Skill *S*. Again the adversary used the exact wording of the genuine Skill to make the user believe the provided information is real because it comes from his trusted stock Skill.

5.2.1 Feasibility on IoT Hardware. To evaluate the susceptibility of smart home devices to the Lyexa attack, we examined its feasiblity given the typical hardware set-up of IoT devices. The used malicious device could be a cheap two-way security camera like the Apexis J011-WS. The J011-WS is equipped with a generic 1 W, 8 Ω small loudspeaker and the Cirrus WM8731 driver[22], which is able to process PCM audio with a sampling rate of up to 96 kHz. We tested the speaker and were able to inaudibly inject commands into the Echo Dot from a distance of up to ≈ 30 cm and inaudibly jam commands from up to ≈ 15 cm, with an AM carrier frequency of up to 25 kHz even though the speaker is not rated for frequencies beyond 20 kHz (this is the maximum we could safely achieve without overheating the speaker, for a one time attack, even more power could be used to extend the range of attack). We also tested a 3 W and 2 W generic small form factor non-tweeter speaker sold for less than $5. We were able to successfully perform command injection at a distance of up to 45 cm and 40 cm and jam at up to 30 cm and 20 cm, respectively. This shows that no special hardware is needed to accomplish this attack and even cheap off the shelf IoT devices can be used.

Cheap IP cameras, including devices from Apexis, are reported to be vulnerable to many attacks. This (generic) IP camera family is branded by bigger companies like Logilink, which sells the model WC0030A[23]. Remote code execution vulnerabilities for 1250 different camera models, including brandings of this generic camera model (Apexis, Logilink, etc.) were published[24]. In addition to that our model automatically gets a public hostname on power on which also suffers from an enumeration vulnerability.

All this combined could enable an attacker to enumerate devices, access them remotely and upload a malicious firmware or ultrasonic modulated audio files to the devices to attack Alexa devices in vicinity. This shows that many devices, even without special hardware like a tweeter speaker may be used remotely by an attacker to emit ultrasonic frequencies and attack other devices.

6 RELATED WORK

To the best of our knowledge there are no published man-in-the-middle attacks against Virtual Personal Assistants like Amazon Alexa, especially attacks utilizing service plug-ins like Skills. We are also not aware of approaches combining both jamming and injecting of inaudible commands to implement more complex attacks.

[21] https://www.amazon.com/Stock-Prices-by-Opening-Bell/dp/B01E9Z3UVC
[22] https://www.cirrus.com/products/wm8731/
[23] http://www.logilink.com/WLAN_Indoor_Pan-Tilt_IP_Kamera_mit_Nachtsicht-Bewegungsmelder_2-Way_Audio.htm
[24] https://seclists.org/fulldisclosure/2017/Mar/23

We focus therefore on reviewing existing approaches that aim at issuing commands to Virtual Private Assistants without victim users noticing.

Audible Synthesized Voice Commands. Diao *et al.* [6] discuss attacks against the Google Voice Search (GVS) app on Android smart phones. To carry out these attacks, a malicious app must be installed on the device. The malicious app activates GVS and simultaneously plays a recorded or synthesized command over the built-in speakers which is then picked up by the microphone, on which GVS is listening. This way the malicious app can access restricted resources by using GVS without asking the user for permission. The app can also call the attacker through GVS, enabling the attacker to issue commands via the call. This attack, however, does not work anymore on Android 7, as it uses now voice cancellation techniques to improve voice calls by reducing recorded noise. Alepis *et al.* [1] therefore extend the work of Diao *et al.* [6] by proposing to use multiple devices to perform the attack and to use Firebase to return transcribed audio replies.

These attacks only work on Android smart phones and are limited to injecting commands when the victim user is not listening. It is therefore not applicable in our scenario. Nevertheless we utilize the idea of using synthesized audio to control Virtual Private Assistants and that the (synthesized speech) answer can be reliably transcribed back into text, essentially allowing us to flexibly interact with a voice-controlled assistant via text.

Audible Mangled Voice Commands. Vaidya *et al.* [21] propose a method to mangle human voice so that it is no longer human comprehensible but still recognizable by the Voice Recognition Systems. The audio mangler takes raw voice and MFCC parameters as input and returns mangled voice which is hard for humans to understand but Voice Recognition Systems detect the same text in this mangled audio as in the raw audio.

Carlini *et al.* [3] extended the work of Vaidya *et al.* [21]. In addition to the black box approach Carlini *et al.* evaluate a white box approach against the open-source CMU Sphinx where the underlying mechanics are known to the attacker. Because the inner workings of CMU Sphinx are known, a more precise attack is possible. Our attack does not use command mangling for injecting or jamming because if the attack is repeated several times this would raise the suspicions of the victim as the mangled sounds would always need to occur at the same time he interacts with Alexa.

Inaudible Voice Commands. Song *et al.* [19] describe an attack using the non-linearity of microphones to induce low frequency sounds with the use of ultrasonic frequencies in microphones of voice-controlled virtual assistants (cf. Sect 2.2). Independently Zhang *et al.* [25] evaluated the demodulation properties of MEMS and ECM microphones for a single modulated frequency and voice, which consists of multiple frequencies. We adopted techniques in these papers for realizing command injection performed by malicious device D and combining it with inaudible command jamming to realize a complete attack framework against the Alexa virtual assistant. In contrast to their attack that is limited to simple command injection, our attack framework allows to hijack the entire interaction between user and Alexa or Skills allowing for much more powerful attack scenarios than mere issuing of single commands.

Independently to our work Roy *et al.* [14] have developed an approach for realizing jamming of spy microphones using ultrasound signals while not interfering with human conversations. Our method builds on similar findings than what also they report.

Voice Commands over the headset jack. Kasmi *et al.* [10] were able to induce voice into a headset connected to a smart phone using intentional electromagnetic interference (IEMI) and control the listening Virtual Personal Voice Assistant this way. Another attack requiring physical access to devices is described by Young *et al.* [23]. The attack uses a RaspberryPi 2 with an audio board, which is connected to the victim's device over the headphone jack to activate various functions on the smartphone. However, since the typical virtual assistants targeted by our attack are not equipped with headphone jacks, we did not consider these approaches in constructing our attack framework.

Skill Squatting. Kumar et al. [12] identified utterances that the Alexa STT engine misinterprets systematically and created an attack, dubbed *Skill Squatting* utilizing these misinterpretations to trick the user to open a malicious Skill. Zhang et al. [26] attack uses Skills with a similar pronounced or paraphrased invocation-name to hijack the command meant for that Skill. They do however not elaborate on how their attack would work in practice, as for it to be effective, e.g., for Skills requiring pairing to a device, both the benign and the malicious Skills would need to be activated on the victim's Alexa account (the malicious after the benign Skill was paired). Note that these kinds of attacks can only be used to redirect the command flow to a different Skill, but not a full man-in-the-middle attack, where communications between users and benign Skills are modified on-the-fly, like Lyexa does.

Hide commands in other audio. Schönherr et al. [16] and Yuan et al. [24] describe a method for hiding commands recognizable by deep neural network-based voice recognition algorithms but not by humans inside other audio files. Carlini et al. [4] showed how to create an audio file with a similar waveform which transcribes into a totally different sentence when processed by Mozilla's DeepSpeech.

IoT attacks. Fernandes et al. [8] did a security analysis of the smart home framework for Samsung's SmartThings which allows third parties to write apps for this service. Ho et al. [9] reviewed the security of smart home locks against physically-present attackers and relay attacks. Our attack differs from these works in that it does not attack IoT devices directly, but achieves malicious functionality through weaknesses in the AVS Skill ecosystem.

Other audio based attacks. Numerous works have used audio-based components as parts of security attacks similar to our approach. Son et al. [18] used sound played at the resonance frequency of MEMS gyroscopes to crash drones. Trippel et al. [20] used an acoustic injection attack to control the output of a MEMS gyroscope which made them able to inject fake steps into a Fitbit. Cisse et al. [5] published an approach on how to create malicious (perturbed) inputs for image classification and speech recognition. Finally, Schlegel et al. [15] created a low-permission Android app that can extract PINs and passwords from phone calls.

7 COUNTERMEASURES

Our attack is enabled by the non-linearity of microphones used in Amazon Echo and other Voice Assistant devices and loopholes in the Skill model. In this section we will discuss countermeasures against both vulnerabilities.

Non-linearity. An effective countermeasure to defend against Lyexa would be to change the hardware of microphones to not pick up ultrasonic frequencies or to implement a hardware module between amplifier and LPF to detect AM signals and silence the baseband of the demodulated signal, to suppress the demodulated voice commands. Furthermore, using machine-learning classifiers can be used to distinguish between demodulated and normally recorded audio. However, realizing such defenses in already deployed devices is not feasible. For new devices, this would also imply significant changes in device and in particular microphone designs, making it unlikely to be adopted in the near future. A feasible option could therefore be to resort to a dedicated device that would monitor the audio environment and alarm the user or AVS or even jam the injection, if it detects an abnormally high sound pressure in the ultrasonic sound spectrum. After-market solutions like this are not uncommon as similar products for securing the home network[25, 26] are emerging.

Skill model. Our attack works because Alexa continues to listen even when it hears only noise. Alexa's logic could therefore be changed to ignore commands that are issued after extremely loud noise in order to prevent jamming of commands. In addition, Alexa could audibly announce which Skill is activated. This would allow users to notice if an unintended Skill is invoked.

The recommended way of running a Skills backend is to use the Amazon Web Services (AWS) Lambda service which is called directly by AVS when a user invokes the Skill. Amazon could force Skill developers to use Lambda and certify also the Skills' backend source code to make sure the Skill backend doesn't contain malicious functionality. As Lambda is under control of Amazon, it could require re-certification when the source code on Lambda is changed, essentially preventing post-certification code changes. While this countermeasure may not be feasible (e.g. increased cost of certification, developers are limited in possibilities) it could be used to prevent backdoors in Skills.

8 DISCUSSION

Note that the attacks described above are not achievable by previous attack approaches that only inject commands (e.g. Dolphin [25], which records voice and replays it inaudibly) as the invoked Skill will return an audio reply which the user will be able to hear and which will make him suspicious. Without the use of a malicious Skill it is not possible to return modified or false information to the user. Skill Squatting attacks [12, 26] could in principle be used to realize our Skill redirection attack, but have a significant practical obstacle. To hijack an interaction with a specific Skill the adversary would need to bring the user to activate the malicious Skill with an invocation name that is similar to the benign one *in addition to* the already used Skill, which is unlikely.

[25] https://www.bitdefender.com/box/

[26] https://www.f-secure.com/en_US/web/home_us/sense

Using our man-in-the-middle attack it is also possible to realize attacks requiring explicit entry of passwords or PIN codes, as the user can be fooled to reveal these during the interaction with the attacked virtual assistant. Previous approaches utilizing only clandestine command injection will not be able to do this, as these are not known to the adversary.

In the Alexa companion app there are two lists where a user can see the history of commands issued to Alexa, possibly revealing the attack to the user. The first is the main window, where all previous 'home cards' issued by Skills are listed. Issuing home cards is, however not mandatory for Skills, so that the malicious Skill can simply omit issuing home cards for malicious functionality. The second is the "true" history of commands, which is located on a third submenu level among all other companion app settings, making it unlikely that regular users would actively monitor it.

Loud background noise or drastic changes in background noise intensity can lead to unsuccessful command injections. However, this does not impact the effectiveness of the attack in the most common deployment environments of virtual assistants, which is at home. There the environment is mostly silent, making the setting favorable for our attack.

There may finally be commands we cannot successfully hijack without the user noticing it. An example could be an interaction like "Alexa, close my last used lock!" where Alexa would respond with "Okay, your lock front door is now locked.". Without the information which lock was locked last, the adversary cannot reliably return a satisfying response to this command. Getting this information from Alexa is also not possible, as using the command would have the side effect of closing the lock.

9 FUTURE WORK

To be able to attack a victim's device reliably without a manual setup, our malicious device has to be able to set up and calibrate itself automatically, according to the physical characteristics of the targeted victim device. There are different Amazon Echo variants (Dot, Echo, Echo Plus) and two different generations of these devices. Even two identically looking devices of the same generation and version could use different microphone hardware. In addition to that, the distance between the two devices requires a signal of specific volume level to be used. If it is too low, no demodulation of the signal is possible. If it is too high, we jam the microphone by suppressing (quieter) voice by the automatic gain control [14], instead of injecting voice. To adjust the correct volume level it is possible to play, e.g., the Alexa wake-word at different volumes and modulated with different carrier frequencies until the recognition audio cue is played by the Echo device, indicating that the injection is working.

Even though we eliminated the crackling sound when an ultrasonic audio file starts playing or ends, we still get a slightly audible crack if we stop playing it halfway through. This happens when the jamming audio stops abruptly because the injection audio has to be played. We could use a pure tweeter speaker instead of a wide-range one, to get rid of the crackling sound. A software approach (a one second fade out) could be too long and Alexa might stop listening.

We could use also modify the attack to use Alexa itself as the STT service. If we have an utterance consisting solely of slots (which

are able to catch everything) and a user says e.g. "Alexa, ask Nuki to lock garden door" we could jam only "ask Nuki" and append ", using Evil". "to lock garden door" will then be transcribed by Alexa and passed to our Skill's backend as the Slot value. Drawbacks are that we don't know exactly which Skill the user wanted to invoke (we could guess it from the rest of the command) and would need a very precise way to jam only the correct part of the command. In this case we wouldn't need to record the user's command and transcribe it using a third party service.

Currently we are looking into Google Assistant and its Actions. Samsung's Bixby and Apple's Siri have at the time of writing no comparable add-on functionality like Alexa.

10 SUMMARY

We presented and evaluated the Lyin' Alexa attack. We showed that with this attack it is possible to completely hijack a full conversation between Alexa and the user while simultaneously maintaining the interaction semantics from the point of view of the user. We were able to deny access to Skills or built-in functionality of Alexa and return false but plausible looking data to the user in the name of Alexa. Furthermore we suggest countermeasures and limitations of this attack as well as possible modifications as future work.

Acknowledgements. This work was founded in part by the German Research Foundation (DFG) within CRC 1119 CROSSING (S2 and P3) and the Intel Collaborative Institute for Collaborative Autonomous and Resilient Systems (ICRICARS). We would also like to thank Cisco Systems, Inc. for their support of this work.

11 RESPONSIBLE DISCLOSURE

We have contacted Amazon and notified them about our findings. We are currently engaged in discussions with their security team.

REFERENCES

[1] E. Alepis and C. Patsakis. 2017. Monkey Says, Monkey Does: Security and Privacy on Voice Assistants. *IEEE Access* 5 (2017), 17841–17851. https://doi.org/10.1109/ACCESS.2017.2747626

[2] Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J. Alex Halderman, Luca Invernizzi, Michalis Kallitsis, Deepak Kumar, Chaz Lever, Zane Ma, Joshua Mason, Damian Menscher, Chad Seaman, Nick Sullivan, Kurt Thomas, and Yi Zhou. 2017. Understanding the Mirai Botnet. In *26th USENIX Security Symposium (USENIX Security 17)*. USENIX Association, Vancouver, BC, 1093–1110.

[3] Nicholas Carlini, Pratyush Mishra, Tavish Vaidya, Yuankai Zhang, Micah Sherr, Clay Shields, David Wagner, and Wenchao Zhou. 2016. Hidden Voice Commands. In *25th USENIX Security Symposium (USENIX Security 16)*. USENIX Association, Austin, TX, 513–530. https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/carlini

[4] Nicholas Carlini and David Wagner. 2018. Audio adversarial examples: Targeted attacks on speech-to-text. *arXiv preprint arXiv:1801.01944* (2018).

[5] Moustapha M Cisse, Yossi Adi, Natalia Neverova, and Joseph Keshet. 2017. Houdini: Fooling deep structured visual and speech recognition models with adversarial examples. In *Advances in Neural Information Processing Systems*. 6977–6987.

[6] Wenrui Diao, Xiangyu Liu, Zhe Zhou, and Kehuan Zhang. 2014. Your Voice Assistant is Mine: How to Abuse Speakers to Steal Information and Control Your Phone. In *Proceedings of the 4th ACM Workshop on Security and Privacy in Smartphones & Mobile Devices (SPSM '14)*. ACM, New York, NY, USA, 63–74. https://doi.org/10.1145/2666620.2666623

[7] Sam Edwards and Ioannis Profetis. 2016. *Hajime: Analysis of a decentralized internet worm for IoT devices.* Technical Report. Rapidity Networks.

[8] Earlence Fernandes, Jaeyeon Jung, and Atul Prakash. 2016. Security analysis of emerging smart home applications. In *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE, 636–654.

[9] Grant Ho, Derek Leung, Pratyush Mishra, Ashkan Hosseini, Dawn Song, and David Wagner. 2016. Smart Locks: Lessons for Securing Commodity Internet of Things Devices. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security (ASIA CCS '16)*. ACM, New York, NY, USA, 461–472. https://doi.org/10.1145/2897845.2897886

[10] C. Kasmi and J. Lopes Esteves. 2015. IEMI Threats for Information Security: Remote Command Injection on Modern Smartphones. *IEEE Transactions on Electromagnetic Compatibility* 57, 6 (Dec 2015), 1752–1755. https://doi.org/10.1109/TEMC.2015.2463089

[11] John Koetsier. 2018. Amazon Echo, Google Home Installed Base Hits 50 Million; Apple Has 6% Market Share, Report Says. https://www.forbes.com/sites/johnkoetsier/2018/08/02/amazon-echo-google-home-installed-base-hits-50-million-apple-has-6-market-share-report-says.

[12] Deepak Kumar, Riccardo Paccagnella, Paul Murley, Eric Hennenfent, Joshua Mason, Adam Bates, and Michael Bailey. 2018. Skill squatting attacks on amazon alexa. In *27th USENIX Security Symposium (USENIX Security 18)*. USENIX Association, 33–47.

[13] Nirupam Roy, Haitham Hassanieh, and Romit Roy Choudhury. 2018. BackDoor: Sounds That a Microphone Can Record, but That Humans Can'T Hear. *GetMobile: Mobile Comp. and Comm.* 21, 4 (Feb. 2018), 25–29. https://doi.org/10.1145/3191789.3191799

[14] Nirupam Roy, Haitham Hassanieh, and Romit Roy Choudhury. 2017. BackDoor: Making Microphones Hear Inaudible Sounds. In *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '17)*. ACM, New York, NY, USA, 2–14. https://doi.org/10.1145/3081333.3081366

[15] Roman Schlegel, Kehuan Zhang, Xiao-yong Zhou, Mehool Intwala, Apu Kapadia, and XiaoFeng Wang. 2011. Soundcomber: A Stealthy and Context-Aware Sound Trojan for Smartphones. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 2011, San Diego, California, USA, 6th February - 9th February 2011*. http://www.isoc.org/isoc/conferences/ndss/11/pdf/1_1.pdf

[16] Lea Schönherr, Katharina Kohls, Steffen Zeiler, Thorsten Holz, and Dorothea Kolossa. 2018. Adversarial Attacks Against Automatic Speech Recognition Systems via Psychoacoustic Hiding. *arXiv preprint arXiv:1808.05665* (2018).

[17] Claude Elwood Shannon. 1949. Communication in the presence of noise. *Proceedings of the IRE* 37, 1 (1949), 10–21.

[18] Yun Mok Son, Ho Cheol Shin, Dong Kwan Kim, Young Seok Park, Ju Hwan Noh, Ki Bum Choi, Jung Woo Choi, and Yong Dae Kim. 2015. Rocking drones with intentional sound noise on gyroscopic sensors. In *24th USENIX Security symposium*. USENIX Association.

[19] Liwei Song and Prateek Mittal. 2017. Inaudible Voice Commands. *CoRR* abs/1708.07238 (2017). arXiv:1708.07238 http://arxiv.org/abs/1708.07238

[20] Timothy Trippel, Ofir Weisse, Wenyuan Xu, Peter Honeyman, and Kevin Fu. 2017. WALNUT: Waging doubt on the integrity of MEMS accelerometers with acoustic injection attacks. In *Security and Privacy (EuroS&P), 2017 IEEE European Symposium on*. IEEE, 3–18.

[21] Tavish Vaidya, Yuankai Zhang, Micah Sherr, and Clay Shields. 2015. Cocaine Noodles: Exploiting the Gap between Human and Machine Speech Recognition. In *9th USENIX Workshop on Offensive Technologies (WOOT 15)*. USENIX Association, Washington, D.C. https://www.usenix.org/conference/woot15/workshop-program/presentation/vaidya

[22] Tim Yeh, Dove Chiu, and Kenney Lu. [n. d.]. Persirai: New Internet of Things (IoT) Botnet Targets IP Cameras. https://blog.trendmicro.com/trendlabs-security-intelligence/persirai-new-internet-things-iot-botnet-targets-ip-cameras/.

[23] Park Joon Young, Jo Hyo Jin, Samuel Woo, and Dong Hoon Lee. 2016. BadVoice: Soundless voice-control replay attack on modern smartphones. In *2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN)*. 882–887. https://doi.org/10.1109/ICUFN.2016.7537163

[24] Xuejing Yuan, Yuxuan Chen, Yue Zhao, Yunhui Long, Xiaokang Liu, Kai Chen, Shengzhi Zhang, Heqing Huang, Xiaofeng Wang, and Carl A Gunter. 2018. CommanderSong: A Systematic Approach for Practical Adversarial Voice Recognition. *arXiv preprint arXiv:1801.08535* (2018).

[25] Guoming Zhang, Chen Yan, Xiaoyu Ji, Tianchen Zhang, Taimin Zhang, and Wenyuan Xu. 2017. DolphinAttack: Inaudible Voice Commands. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS '17)*. ACM, New York, NY, USA, 103–117. https://doi.org/10.1145/3133956.3134052

[26] Nan Zhang, Xianghang Mi, Xuan Feng, XiaoFeng Wang, Yuan Tian, and Feng Qian. 2018. Understanding and Mitigating the Security Risks of Voice-Controlled Third-Party Skills on Amazon Alexa and Google Home. *arXiv preprint arXiv:1805.01525* (2018).

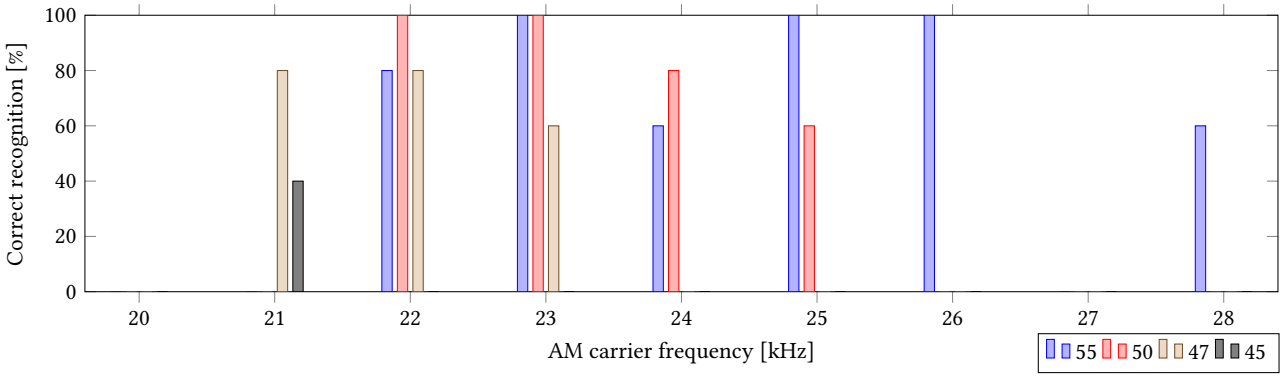


Figure 8: Results of injecting the voice sample "start Evil" recorded from a male human, modulated with different carrier waves and at different volumes (45-55/100), into an Amazon Echo Dot from 30 cm distance.

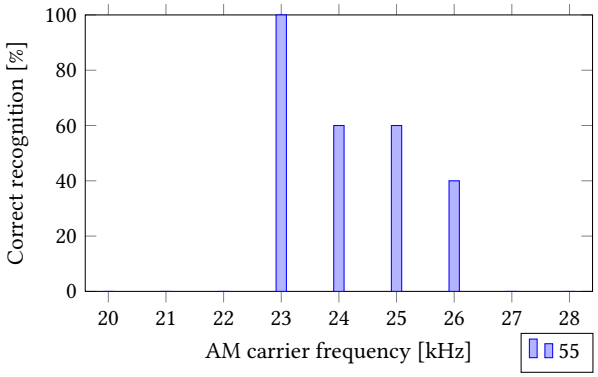


Figure 9: Results of injecting the voice sample "start Evil" generated by a female voice TTS service, modulated with different carrier waves, into an Amazon Echo Dot from 30 cm distance.

APPENDIX

A CRAFTING INAUDIBLE VOICE SIGNALS

The non-linearity of the transducer (the system consisting of a microphone, amplifier, LPF and analog-digital converter (ADC)) will output a signal consisting of more terms as the input. If S_{in} is the input signal transported over the air (voice, music, single frequency e.g. $cos(2\pi ft)$), the recorded signal, due to the non-linearity, would be:

$$S_{out} = \sum_{\infty}^{i=1} G_i S_{in}^i = G_1 S_{in} + G_2 S_{in}^2 + \cdots \quad (1)$$

where G_i is the gain for each term ($G_i S_{in}^i$), decreasing rapidly with increasing i. The attack uses the demodulation properties of the second term (the most powerful except the linear term). First we need to low-pass filter our audio signal at $8kHz$. Voice is mainly concentrated on the lower frequencies and therefore we can remove all frequencies above $8kHz$. This way our frequency spectrum reaches from $0 - 8kHz$. After that we need to upsample it to be able to convert it to ultrasound ($> 20kHz$) later on. The upsampling frequency should be the maximum frequency our device can process but no less than $56kHz$. We denote this signal as S_{up}.

The next step is to amplitude modulate it onto an ultrasonic carrier wave with frequency f_c to shift the signal into the ultrasonic spectrum and make it inaudible for humans. We get:

$$S_{modu} = n_1 S_{up} cos(2\pi f_c t) \quad (2)$$

where n_1 is a normalization factor. This will create sidebands ranging from $f_c - 8kHz$ to $f_c + 8kHz$. This is why we need at least an upsampling frequency of $58kHz$. Given the Nyquist-Shannon sampling theorem [17] we are able to sample a signal with a frequency of up to $28kHz$ without loss using a sampling rate of 56kHz. With the lower sideband ranging down to $f_c - 8kHz$ f_c should be at least $28kHz$ to make it inaudible. After that we need to add the carrier wave to the signal, so it can be demodulated by the non-linearity of the speaker. We get:

$$S_{added} = n_2(S_{modu} + cos(2\pi f_c t)) \quad (3)$$

where n_2 is another normalization factor.

Suppose $S_m = cos(2\pi f_m t)$ then the attack signal would be:

$$S_{in} = n_2(n_1 cos(2\pi f_m t) cos(2\pi f_c t) + cos(2\pi f_c t)) \quad (4)$$

After demodulation we get the frequency components f_m, $2(f_c - f_m)$, $2(f_c + f_m)$, $2f_c$, $2f_m$, $2f_c - f_m$ and $2f_c + f_m$. Every component is above $20kHz$ and will be removed by the LPF of the microphone except f_m which is $8kHz$ at the maximum. That way the original S_m is perceived by the microphone without ever played and thus inaudible for humans.

B ATTACK SIGNAL PARAMETERS

The ultrasonic attack description of Song *et al.* [19] and Zhang *et al.* [25] leaves many implementation and parameterization questions open. To craft the ultrasonic signal, we have to normalize it with the variables n_1 and n_2, as described in Sect. A. Obviously we cannot set them to 1 otherwise we will get our signal clipped at the maximum amplitude of 1. We have to scale one or both amplitudes down. Fortunately we don't have to evaluate it but we can calculate it easily as follows:

Applying Eq. 4 to Eq. 1 to calculate the second term of S_{out} which we denominate S_{out_2}, we get:

$$\begin{aligned} S_{out_2} = G_2 \frac{1}{8}(&n_2^2 n_1^2 cos(4\pi f_c t - 4\pi f_m t) \\ &+ n_2^2 n_1^2 cos(4\pi f_c t + 4\pi f_m t) + 2n_2^2 n_1^2 cos(4\pi f_c t) \\ &+ 2n_2^2 n_1^2 cos(4\pi f_m t) + 2n_2^2 n_1^2 + 4n_2^2 n_1 cos(4\pi f_c t - 2\pi f_m t) \\ &+ 4n_2^2 n_1 cos(4\pi f_c t + 2\pi f_m t) + 8n_2^2 n_1 cos(2\pi f_m t) \\ &+ 4n_2^2 cos(4\pi f_c t) + 4n_2^2) \end{aligned} \tag{5}$$

With this we we see that if we want to maximize the amplitude of the demodulated S_m we have to maximize $n_2^2 n_1$ while the maximum possible amplitude of S_{in} is 1 to avoid clipping of the constructed signal. Suppose S_m has an amplitude of 1 like the carrier wave and the added wave. With Eq. 3 we get that the amplitude of S_{in} is $n_2(n_1 + 1)$. We have to find n_1 and n_2 of:

$$max \left\{ n_1 n_2^2 | (n_1 + 1) n_2 = 1 \right\} \tag{6}$$

which is at $(n_1, n_2) = (1, \frac{1}{2})$.

C EVALUATION OF PARAMETERS

The Echo Dot was activated by unmuting it by pressing the button on top of the device, after which the modulated recording of a male person saying "start Evil" was played back at different volumes and modulated with different carrier waves of different frequencies. The results are displayed in Fig. 8. We see that lower sound pressure signals modulated with a lower frequency are successfully demodulated. Unsuccessful injections usually mean that Alexa does not understand anything, but it occasionally also understood "Even", "Not", "Not you" and only "Evil", as reported by the Alexa App. As stated earlier this can be mitigated by using an invocation word composed of two words which do not occur anywhere else. Evaluating with the same setup but using a voice sample generated by a female TTS service is depicted in Fig. 9. Song *et al.* [19] came to the same conclusion, that this yields much worse results, as the female voice is centered much higher than that of a male and is clipped by the low-pass filter.

B

LeakyPick: IoT Audio Spy Detector (ACSAC 2020, CORE: A)

---

[108] Richard Mitev, Anna Pazii, Markus Miettinen, William Enck, and Ahmad-Reza Sadeghi. Leakypick: Iot audio spy detector. In *Proceedings of the 36th Annual Computer Security Applications Conference*, pages 694–705, 2020. https://doi.org/10.1145/3427228.3427277.

LeakyPick: IoT Audio Spy Detector

Richard Mitev
richard.mitev@trust.tu-darmstadt.de
Technical University of Darmstadt

Anna Pazii
anna.pazii@inria.fr
University of Paris Saclay

Markus Miettinen
markus.miettinen@trust.tu-darmstadt.de
Technical University of Darmstadt

William Enck
whenck@ncsu.edu
North Carolina State University

Ahmad-Reza Sadeghi
ahmad.sadeghi@trust.tu-darmstadt.de
Technical University of Darmstadt

ABSTRACT

Manufacturers of smart home Internet of Things (IoT) devices are increasingly adding voice assistant and audio monitoring features to a wide range of devices including smart speakers, televisions, thermostats, security systems, and doorbells. Consequently, many of these devices are equipped with microphones, raising significant privacy concerns: users may not always be aware of when audio recordings are sent to the cloud, or who may gain access to the recordings. In this paper, we present the LeakyPick architecture that enables the detection of the smart home devices that stream recorded audio to the Internet in response to observing a sound. Our proof-of-concept is a LeakyPick device that is placed in a user's smart home and periodically "probes" other devices in its environment and monitors the subsequent network traffic for statistical patterns that indicate audio transmission. Our prototype is built on a Raspberry Pi for less than USD $40 and has a measurement accuracy of 94% in detecting audio transmissions for a collection of 8 devices with voice assistant capabilities. Furthermore, we used LeakyPick to identify 89 words that an Amazon Echo Dot misinterprets as its wake-word, resulting in unexpected audio transmission. LeakyPick provides a cost effective approach to help regular consumers monitor their homes for sound-triggered devices that unexpectedly transmit audio to the cloud.

CCS CONCEPTS

• **Security and privacy → Domain-specific security and privacy architectures**; **Intrusion/anomaly detection and malware mitigation**.

ACM Reference Format:
Richard Mitev, Anna Pazii, Markus Miettinen, William Enck, and Ahmad-Reza Sadeghi. 2020. LeakyPick: IoT Audio Spy Detector. In *Annual Computer Security Applications Conference (ACSAC 2020), December 7–11, 2020, Austin, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3427228.3427277

ACSAC 2020, December 7–11, 2020, Austin, USA

ACM ISBN 978-1-4503-8858-0/20/12…$15.00
https://doi.org/10.1145/3427228.3427277

1 INTRODUCTION

Consumer Internet of Things (IoT) devices have emerged as a promising technology to enhance home automation and physical safety. While the smart home ecosystem has a sizeable collection of automation platforms, market trends in the US [44] suggest that many consumers are gravitating towards Amazon Alexa and Google Home. These two platforms are unique from the other automation platforms (e.g., Samsung SmartThings, WeMo) in that they focused much of their initial smart home efforts into smart speaker technology, which allows users to speak commands to control smart home devices (e.g., light switches), play music, or make simple information queries. This dominance of Amazon Alexa and Google Home might suggest that consumers find voice commands more useful than complex automation configurations.

For many privacy-conscious consumers, having Internet connected microphones scattered around their homes is a concerning prospect. This danger was recently confirmed when popular news media reported that Amazon [27], Google [23, 31], Apple [24], Microsoft [13], and Facebook [18] are all using contractors to manually analyze the accuracy of voice transcription. The news reports include anecdotes from contractors indicating they listened to many drug deals, domestic violence, and private conversations. Perhaps more concerning is that many of the recordings were the result of false positives when determining the "wake-word" for the platform. That is, the user never intended for the audio to be sent to the cloud.

Unfortunately, avoiding microphones is not as simple as not purchasing smart speakers. Microphones have become a pervasive sensor for smart home devices. For example, it is difficult to find a smart television that does not support voice controls via the display or the handheld remote control. Smart thermostats (e.g., Ecobee) commonly advertise that they have dual function as a smart speaker. Surveillance cameras (e.g., Ring, Wyze) are designed to notify users of events, but are in fact always recording. Perhaps most concerning was the report that the Nest security system includes a microphone [27], despite no packing material or product documentation reporting its existence. While the manufacturers of these devices might argue that users can disable microphone functionality in device software, history has repeatedly demonstrated that software can and will be compromised. Furthermore, mass collection and storage of audio recordings increases concerns over the potential for a "surveillance state" (e.g., Ring has recently been criticized for working with local police [22]).

Our research seeks to answer the question: *Can a user effectively detect if a smart home device expectantly transmits audio recordings to Internet servers in response to a sound trigger?* Such failures can occur in two types of situations: (1) devices that are not expected to have recording capability or are expected to have the capability disabled transmit user audio, or, (2) devices that are expected to have recording capability enabled, but transmit audio in response to unexpected stimuli (e.g., unexpected or misunderstood wake-words). In both cases we are primarily concerned with the benign, but hidden, recording and immediate transmission of audio to cloud services, as such behaviors can potentially lead to mass surveillance. We believe there is significant utility in detecting this subset of misbehaving devices, particularly given the limited storage capacity of many low-level IoT devices. Additionally, we only consider audio transmission that occur in response to a sound-based trigger event. Devices that continuously stream audio are detectable by monitoring bandwidth use.

Existing approaches for identifying unintended data transmissions out of the user's network [11, 29] focus on other modalities (e.g., video) and rely on assumptions that do not apply to audio transmissions (e.g., some devices require an utterance of specific wake-words). Furthermore, while traffic analysis approaches targeting IoT devices have been proposed [33, 41], to the best of our knowledge there are no earlier approaches specifically targeting traffic of microphone-enabled IoT devices. Additionally, prior work attacking voice assistants and voice recognition [3, 7, 8, 14, 28, 30, 40, 46, 49–51] focuses on maliciously issuing commands or changing the interaction flow without the victim noticing.

In this paper, we present the LeakyPick architecture, which includes a small device that can be placed in various rooms of a home to detect the existence of smart home devices that stream recorded audio to the Internet. LeakyPick operates by periodically "probing" an environment (i.e., creating noise) and monitoring subsequent network traffic for statistical patterns that indicate the transmission of audio content. By using a statistical approach, LeakyPick's detection algorithm is generalizable to a broad selection of voice-enabled IoT devices, eliminating the need for time-consuming training required by machine learning.

We envision LeakyPick being used in two scenarios. First, LeakyPick can identify devices for which the user does not know there is a microphone, as well as smart home devices with smart speaker capabilities (e.g., an Ecobee thermostat) that the user was not aware of, or thought were disabled (e.g., re-enabled during a software update). To evaluate this scenario, we studied eight different microphone-enabled IoT devices and observed that that LeakyPick can detect their audio transmission with 94% accuracy. Second, LeakyPick can be used to determine if a smart speaker transmits audio in response to an unexpected wake-word. To evaluate this scenario, we used LeakyPick to perform a wake-word fuzz test of an Amazon Echo Dot, discovering 89 words that unexpectantly stream audio recordings to Amazon. For both scenarios, LeakyPick can run when the user is not home (to avoid annoyance), since this behavior is generally not contextual the users' presence.

This paper makes the following contributions:

- *We present the* LeakyPick *device for identifying smart home devices that unexpectedly record and send audio to the Internet in response to observing a sound.* The device costs less than USD $40 and can be easily deployed in multiple rooms of a home.
- *We present a novel audio-based probing approach for estimating the likelihood that particular devices react to audio.* Our approach has a 94% accuracy for a set of devices known to transmit audio to the cloud. We also show that the approach is generalizable to different device types without the need to pre-train costly device-type-specific detection profiles.
- *We show that* LeakyPick *can identify hidden wake-words that cause unexpected audio transmission.* Our analysis of an Amazon Echo Dot identified 89 incorrect wake-words.

Finally, LeakyPick uses human-audible noises, which may be annoying to physically present users. Prior work has suggested the use of inaudible sound to control voice assistants using ultrasound audio [42, 50]. However, these approaches are specific to the technical characteristics of the targeted devices. Therefore, they are not immediately applicable to our goal of identifying unknown devices streaming audio. We leave the task of creating generic models of transmitting audio via ultrasonic sound as a topic for future work.

The remainder of this paper proceeds as follows. Section 2 provides background on devices with voice control and audio interfaces. Section 3 overviews our architecture. Section 4 describes our design and implementation. Section 5 evaluates the accuracy of LeakyPick. Section 6 discusses our approach and security considerations. Section 7 overviews related work. Section 8 concludes.

2 BACKGROUND

IoT devices increasingly use audio sensing for enabling voice-based control by the user or for other audio-based use cases. Examples of such devices include smart audio security systems [25], smart audio event-detecting IP cameras [20], vacuum cleaner robots equipped with microphones and nightvision [6], and smart fire alarms with a self-testing siren [19]. Due to the abundance of devices with audio sensing capabilities, the user may not always be aware of when a particular device will record audio and send it to the cloud. Sending audio to the cloud is frequently required for voice-control based user interfaces, as speech-to-text translation often needs more computational resources than are available on IoT devices.

Devices with voice-control interfaces typically use local speech recognition for detecting a specific set of "wake-words" (i.e., utterances meant to be used by the user to invoke the voice-control functionality of the device). When the local model detects the utterance of a potential wake-word, the device starts sending audio to back-end servers for voice-to-text translation. In order to not miss speech commands uttered by users, the local model needs to be configured to recognize any utterance that resembles the intended wake-word. In case of the Alexa voice assistant, it is then the task of the back-end service to verify whether the observed utterance really represents a wake-word or not, as it is equipped with a more comprehensive speech recognition model and is not limited by the potentially scarce computational capabilities of the IoT device recording the audio. Figure 1 overviews this typical approach.

Problems arise when the local or online detection model mistakenly classifies specific audio inputs as the wake-word and consequently starts sending the recorded audio to the cloud, thereby

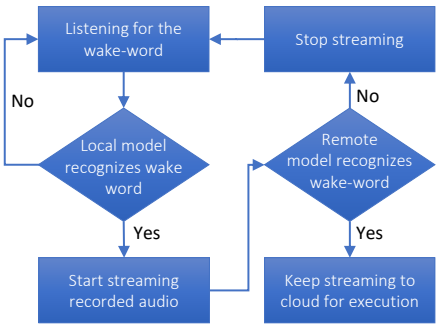


Figure 1: Overview of wake-word detection process in voice-controlled devices such as the Amazon Echo

potentially leaking sensitive information. Private information may also be leaked unintentionally when the user is unaware that a device will react to specific wake-words.

Finally, attacks targeting voice assistants can use malicious audio signal injection to trick the assistant to perform actions. In these attacks, the adversary either waits for the user to be asleep or not present [3, 14] or uses inaudible [30, 42, 50] or unrecognizable [7, 40, 46, 49] signals to stage the attack, making it very difficult for the victim user to realize that the device is being attacked.

Our goal is to provide tools that enable users to detect and identify 1) devices that are not expected to have audio recording transmission capability, 2) devices for which audio recording transmission is expected to be disabled, and 3) unexpected wake-words that cause devices to unexpectedly transmit audio recordings.

Threat Model and Assumptions: In this paper, we are concerned with threats related to IoT devices that stream recorded audio over the Internet using Wi-Fi or a wired LAN connection in response to audio signals recognised by the device as potential voice commands or different sounds the device reacts to. As processing voice commands is (except for the detection of the device's wake-word) implemented on the server-side, we assume that recorded audio is transmitted instantaneously to the cloud to allow the voice-controlled device to promptly react to user commands. We consider three main threat scenarios:

(1) Benign IoT devices that may have undocumented microphones and audio-sensing capabilities, devices for which audio-sensing capabilities are unexpectedly enabled (e.g., via a software update), or devices whose wake-word detection is inaccurate, leading to audio being sent to the cloud without the users intent and knowledge.
(2) Application-level attacks that cause a benign IoT device to send audio without the user's knowledge. For example, the Amazon Echo contained a vulnerability [21] that allowed a malicious skill to silently listen. More recently, security researchers have reported [32] the existence of eavesdropping apps targeting Alexa and Google Assistant. Note that in this scenario, the IoT device is benign, but it supports third-party applications that may be malicious.
(3) Active attacks by an external adversary where the adversary tricks the targeted device to initiate transmission of audio data by *injection of audio signals* to the device's environment so that the device will identify these as its wake-word. The main threat in this scenario is the *unauthorized invocation* of device or service functionalities.

We do not consider scenarios in which an IoT device is modified by the adversary to transform the device to act as a bugging device that records and stores audio locally without sending it to the network. While feasible, such attacks are much less scalable, as they must be tailored for each targeted device and are only applicable for devices with sufficient storage space. We note, however, that LeakyPick is applicable to settings where an adversary has compromised an IoT device and immediately transmits audio data.

3 SOLUTION OVERVIEW

The goal of this paper is to devise a method for regular users to reliably identify IoT devices that 1) are equipped with a microphone, 2) send recorded audio from the user's home to external services without the user's awareness, and 3) do so unexpectantly in response to observing a sound (e.g., unexpected wake-word, software update re-enabling smart speaker functionality). If LeakyPick can identify which network packets contain audio recordings, it can then inform the user which devices are sending audio to the cloud, as the source of network packets can be identified by hardware network addresses. Achieving this goal requires overcoming the following research challenges:

- *Device traffic is often encrypted.* A naïve solution that simply looks for audio codecs in network traffic will fail to identify most audio recordings.
- *Device types are not known a priori.* Devices transmit audio in different ways. We need to identify generic approaches that work with previously unseen devices.

Due to these challenges, our solution cannot passively monitor network traffic with the goal of differentiating the transmission of audio recordings from other network traffic. While prior approaches such as HomeSnitch [34] are able to classify the semantic behavior of IoT device transmissions (e.g., voice command), they require *a priori* training for each manufacturer or device model. Since we seek to identify this behavior for potentially unknown devices, we cannot rely on supervised or semi-supervised machine learning.

At a high level, LeakyPick overcomes the research challenges by periodically transmitting audio (potentially prepended with wake-words) into a room and monitoring the subsequent network traffic from devices. As shown in Figure 2, LeakyPick's main component is a probing device that emits audio probes into its vicinity. By temporally correlating these audio probes with observed characteristics of subsequent network traffic, LeakyPick identifies devices that have potentially reacted to the audio probes by sending audio recordings.

LeakyPick identifies network flows containing audio recordings using two key ideas. First, it looks for traffic bursts following an audio probe. Our observation is that voice-activated devices typically do not send much data unless they are active. For example, our analysis shows that when idle, Alexa-enabled devices periodically send small data bursts every 20 seconds, medium bursts every 300 seconds, and large bursts every 10 hours. We further found that

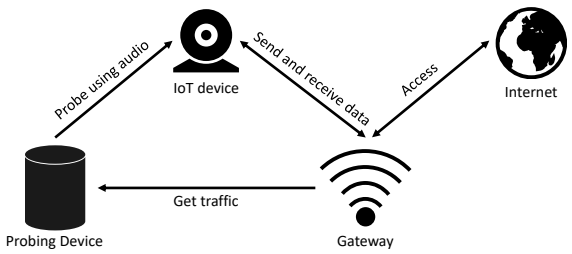


Figure 2: System set-up of LeakyPick

when it is activated by an audio stimulus, the resulting audio transmission burst has distinct characteristics. However, using traffic bursts alone results in high false positive rates.

Second, LeakyPick uses statistical probing. Conceptually, it first records a baseline measurement of idle traffic for each monitored device. Then it uses an *independent two-sample t-test* to compare the features of the device's network traffic while being idle and of traffic when the device communicates after the audio probe. This statistical approach has the benefit of being inherently device agnostic. As we show in Section 5, this statistical approach performs as well as machine learning approaches, but is not limited by *a priori* knowledge of the device. It therefore outperforms machine learning approaches in cases where there is no pre-trained model for the specific device type available.

Finally, LeakyPick works for both devices that use a wake word and devices that do not. For devices such as security cameras that do not use a wake word, LeakyPick does not need to perform any special operations. Transmitting any audio will trigger the audio transmission. To handle devices that use a wake word or sound, e.g., voice assistants, security systems reacting on glass shattering or dog barking, LeakyPick is configured to prefix its probes with known wake words and noises (e.g., "Alexa", "Hey Google"). It can also be used to fuzz test wake-words to identify words that will unintentionally transmit audio recordings.

4 LEAKYPICK DESIGN

This section describes the central aspects of LeakyPick's design. We primarily focus on audio event detection. We then describe our system implementation on a Raspbery Pi 3B.

4.1 Audio Event Detection

Due to the encryption between devices and back-end cloud systems, it is not possible to detect audio-related events by inspecting packet payloads. Instead, LeakyPick identifies audio transmissions by observing sudden outgoing traffic rate increases for specific devices, or significant changes in the device's communication behavior, both of which can indicate the transmission of audio recordings. We consider two possible complementary approaches to audio event detection: (1) a simple baseline method based on network traffic burst detection, and (2) a statistical approach for detecting apparent changes in the communication characteristics of monitored devices in response to performed audio probes. Both audio event detection mechanisms can be used either while actively probing devices, or, without active probing (i.e., when the user is at home) to detect when a device reacts to the noise the user makes (e.g., speaking, playing music) and notifying the user about such activations.

Table 1: Parameters and packet features used by our Burst Detection and Statistical Probing approaches

Approach	Parameters	Packet Features
Burst Detection	Window size s_w Traffic rate B_{audio} Consecutive detections n	Packet size MAC/IP
Statistical Probing	Bin count k Packet sequence duration d P-value threshold t	Packet size Interarrival time MAC/IP

4.1.1 Burst Detection. Our baseline approach for detecting audio transmissions is based on burst detection in the observed network traffic of devices. To do this, we need to first identify the characteristics of potential audio transmissions. We therefore analyzed the invocation process and data communication behavior of popular microphone-enabled IoT devices when they transmit audio.

Our traffic analysis was based on data of popular IoT devices with integrated virtual assistant support: (1) Echo Dot (Amazon Alexa), (2) Google Home (Google Assistant), (3) Home Pod (Apple Siri), and (4) an audio-activated home security system (Hive Hub 360). Our analysis showed that these devices do not typically send much traffic during normal standby operation. Therefore, it is possible to detect audio transmissions through the increase in traffic rate they cause. Our approach is generic in the sense that it is applicable to all devices sending audio. We chose these microphone-enabled devices, as they are popular devices produced for a broad range of use cases. To determine the parameters for burst detection, we monitored the network traffic of devices in response to audio probes emitted into the devices' environment.

We perform audio event detection by observing sudden increases in the traffic rate emitted by a device that is sustained for a specific amount of time. This is because an audio transmission will inevitably cause an increase in the data rate that will typically last at least for the duration of the transmitted audio sample. This is consistent with how most voice-controlled IoT devices utilizing cloud-based back-ends function (Section 2), where local wake-word detection causes subsequent audio to be streamed to the back-end.

Specifically, LeakyPick performs burst detection by dividing the network traffic originating from a device into time windows $W = (w_1, w_2, \ldots)$ of size s_w and calculating for each time window w_i the sum of packet payload sizes of the packets falling within the window. We then calculate the average traffic rate B_i during the time window w_i in bytes per second. If the traffic rate B_i is above a threshold B_{audio} during at least n consecutive time windows

$$W_i = (w_i, w_{i+1}, \ldots, w_{i+k-1})$$

where $k \geq n$, detection is triggered. Empirically, we found that $B_{audio} = 23kbit/s$ is sufficient to separate audio bursts from background traffic. Therefore, it is reasonable to assume our approach will work also for systems using audio codecs with lower bandwidth requirements than Alexa.

As shown in Table 1, LeakyPick uses predefined parameters and packet features. We extract the packet sizes, the corresponding MAC or IP (depending on the layer), and (implicitly) the direction

of the packet (leaving or entering the network). To evaluate the optimal consecutive detection threshold n (Section 5.2.1), we used a fixed window size $s_w = 1s$, as common voice commands rarely take less than a second. For the traffic rate threshold B_{audio}, we chose $23kbit/s$. This value is a sufficiently low threshold to capture any audio and sufficiently high to discard anything else, as voice recognition services need good voice recording quality (e.g., Alexa's Voice Service uses $256kbit/s$, 16-bit PCM at $16kHz$ [4]).

4.1.2 Statistical Probing. LeakyPick uses statistical probing to refine audio transmission detection by eliminating cases where traffic bursts result from non-audio transmission. Most importantly, the approach is generic and does not need *a priori* knowledge of a device's behavior. It also can determine if a device's communication behavior changes significantly in response to audio probes.

To detect devices that react to audio, we monitor network traffic for significant changes in the device's communication behavior in response to audio probes. This is done by determining whether the distribution of the properties of communication packets transmitted by a device after the emission of an audio probe is statistically different from the distribution of packets observed before the probe injection. Specifically, LeakyPick uses a t-test [9], which is one of the most commonly used statistical tests. Given two data samples, the test computes a t-score by determining the data samples' distributions' means and standard deviations, and mapping this to a p-value. If the p-value is below a specified threshold, the distributions are considered statistically different and therefore indicate that the device reacted to the audio probe. The p-value threshold is therefore a system parameter which can be tweaked to produce a trade-off between sensitivity and false alarm rate. However, this threshold is independent of the device type, i.e., a system-wide threshold value is used. The evaluation of this parameter is described in Section 5.2.2.

First, the probing device monitors idle device traffic while it is not emitting audio probes. It captures a sequence

$$T_s = (pck_1, pck_2, \ldots, pck_n)$$

of duration d seconds of data packets pck_i and calculates a packet size (or inter-arrival time) distribution vector

$$\vec{F_s} = (f_1, f_2, \ldots, f_k)$$

by binning the packets $p_i \in T_s$ into k bins based on the size (or inter-arrival time) of these packets[1] and where f_i denotes the number of packets assigned to the i-th bin.

The probing device then emits multiple audio probes and captures associated data traffic

$$T_{pr} = (pck_1, pck_2, \ldots, pck_n)$$

of duration d seconds and extracts a packet size (time) distribution vector $\vec{F_{pr}}$ using the same binning as for $\vec{F_s}$. The packet size vectors $\vec{F_s}$ and $\vec{F_{pr}}$ are then used in the t-test to determine a p-value (p) indicating the likelihood that both frequency samples originate from the same distribution (i.e., from the distribution the device produces while in idle mode).

[1] To determine the binning automatically, we use `numpy.histogram` with the bin option `auto` which uses the maximum of the Sturges and Freedman Diaconis Estimator.

If the p-value is below a specified threshold t (i.e., $p < t$), we assume the traffic samples are not from the same distribution. That is, the device has reacted in some way and changed its communication behavior. To further refine the results, the p-value resulting from the packet size distribution is combined with the p-value of the inter-arrival time distribution. However, as shown in Section 5.2.2, only using the p-value of the packet size distribution is sufficient.

We collected idle data samples T_s from multiple voice-controlled devices and discovered that they contained frequently occurring smaller data bursts (possibly related to, e.g., heartbeat messages) and infrequently occurring larger data bursts (e.g., state synchronization). This observation indicates it is possible to capture a large data burst in one of the two samples (T_s or T_{pr}) while missing it in the other. Therefore, when encountering a p-value indicating a possible reaction of the IoT device, the probing test must be repeated several times to ensure the observed device behavior change is caused by the audio probe and not by background traffic bursts.

Table 1 shows the parameters and packet features used for statistical probing. As with burst detection, we extract the packet sizes, the corresponding MAC or IP, and (implicitly) the direction of the packets from the recorded traffic. Additionally, we extract packet inter-arrival times. As discussed in Section 5.2.2, we set the p-value threshold t to be 0.42-0.43 to achieve an optimal precision while fixing the packet sequence duration to $d = 60$ seconds.

4.1.3 Voice User Interface. LeakyPick offers also a voice-based user interface (UI) for conveniently controlling the probing device and its functions. Our current implementation uses a custom Amazon Alexa Skill. With this interface, the user can control when the device is allowed to probe to avoid annoyance while the user is at home. Additionally, the user can query the results of the last probing session to learn which devices responded to the probing and streamed audio to the cloud. We present this Skill-based method as an example of interacting with LeakyPick. However, optimizing the usability of the user interactions for obtaining the report is not in the core focus of this paper.

4.2 Wake-Word Selection

Users are not always aware of IoT devices that require a wake-word before transmitting audio recordings to the cloud. While it is possible enumerate the wake-words for known voice assistants, recent reports of third-party contractors reviewing voice assistant accuracy [13, 18, 23, 24, 27, 31] highlight the significance of false voice assistant activation. Therefore, LeakyPick identifies other wake-words that will trigger the voice detection. Note that this approach is different than using mangled voice samples [7, 46] or other means to attack the voice recognition process [8, 40, 49]. We also do not want to limit LeakyPick to words sounding similar to the known wake-words in order to confuse the voice assistant [28, 51].

Using a full dictionary of the English language is impractical. It would take roughly 40 days to test a voice assistant with the entire dictionary of 470,000 words [48] at a speed of one word every seven seconds. However, by only testing words with a phoneme count similar to the known wake-word, the subset of viable words is manageable. Our intuition is that a benign device will more likely confuse words with a similar structure. Therefore, we select all words in a phoneme dictionary [39] with the same or similar

Table 2: Devices used for evaluation

Device Type	Device Name
Smart Speaker	Echo Dot (Amazon Alexa)
	Google Home (Google Assistant)
	Home Pod (Apple Siri)
Security System	Hive Hub 360
Microphone-Enabled IoT Device	Netatmo Welcome
	Netamo Presence
	Nest Protect
	Hive View

phoneme count than the actual wake-word. We also used random words from a simple English word list [26]. These words are spoken using a text-to-speech (TTS) engine.

4.3 System Implementation

The LeakyPick probing device injects audio probes into the user's environment and analyzes the resulting device network traffic. Our current implementation achieves this functionality using the following hardware set-up. The probing device consists of a Raspberry Pi 3B [36] connected via Ethernet to the network gateway. It is also connected via the headphone jack to a PAM8403 [15] amplifier board, which is connected to a single generic 3W speaker.

To capture network traffic, we use a TP-LINK TL-WN722N [45] USB Wifi dongle to create a wireless access point using `hostapd` and `dnsmasq` as the DHCP server. All wireless IoT devices connect to this access point. To provide Internet access, we activate packet forwarding between the `eth` (connected to the network gateway) and `wlan` interfaces. Alternatively the device could sniff Wi-Fi packets without being connected to the network using packet size and MAC address as the features. This approach would also work for foreign Wi-Fi networks, as it is not required to have the decrypted traffic, i.e. our device does not need to be connected to that network at all: package size information is sufficient.

Finally, LeakyPick is written in Python. It uses `tcpdump` to record packets on the `wlan` interface. We use Google's text-to-speech (TTS) engine to generate the audio played by the probing device.

5 EVALUATION

This section evaluate LeakyPick's ability to detect when audio recordings are being streamed to cloud servers. Specifically, we seek to answer the following research questions.

- **RQ1** What is the detection accuracy of the burst detection and statistical probing approaches used by LeakyPick?
- **RQ2** Does audio probing with a wrong wake-word influence the detection accuracy?
- **RQ3** How well does LeakyPick perform on a real-world dataset?
- **RQ4** How does LeakyPick's statistical probing approach compare to machine learning-based approaches?
- **RQ5** Can LeakyPick discover unknown wake-words?

5.1 Experimental Setup

Our evaluation considers 8 different wireless microphone-enabled IoT devices: 3 smart speakers, 1 security system that detects glass breaking and dogs barking, and 4 microphone-enabled IoT devices, namely the audio event detecting smart IP security cameras Netatmo Welcome, Netatmo Presence and Hive View as well as the smart smoke alarm Nest Protect. Table 2 lists the specific devices. We now describe our dataset collection and evaluation metrics.

5.1.1 Datasets. We used four techniques to collect datasets for our evaluation. Table 3 overviews these four collection methodologies, as well as to which devices the datasets apply. The following discussion describes our collection methodology.

Idle Datasets: The idle dataset was collected in an empty office room. It consists of network traffic collected over six hours during which the device was not actively used and no audio inputs were injected. We also made sure to record at least one occurrence of every traffic pattern the devices produce (e.g., for Echo devices every type of periodic bursts).

Controlled Datasets - Burst Detection: The controlled datasets for burst detection were collected in an empty office room while injecting audio probes approximately 100 times for each of the studied devices. In all cases, the injected probe was the known wake-word for the device in question. The Hive 360 Hub device does not use a wake-word, but is activated by specific noise like dog barking and glass shattering. For this device we therefore used recordings of dog barking sounds to trigger audio transmission. For each device, three different datasets were collected by varying the wake-word invocation interval between 1, 5, and 10 minutes.

Controlled Datasets - Statistical Probing: The collection of the controlled dataset for statistical probing was performed in a way similar to the burst detection dataset. However, the experiment collected six datasets for each device. Each dataset consisted of six hours of invoking the wake-word at intervals ranging from two minutes to two hours. Thereby resulting in datasets with varying ratios of audio-related and idle background traffic.

Online Probing Datasets: Using live traffic of the 8 different devices listed in Table 2 we randomly selected a set of 50 words out of the 1000 most used words in the English language [16] combined with a list of known wake-words of voice-activated devices as possible wake-words to test. We configured our probing device to alternatingly record silence traffic T_s and probing traffic T_{pr} of one minute duration each for every wake-word in the list. T_{pr} was recorded immediately after the device started playing a word from the list repeating the word every 10 seconds in this minute.

Real-World Datasets: To create a realistic dataset for evaluation, we collected data from the three smart speakers over a combined period of 52 days in three different residential environments (houses). The times for each smart speaker are listed in Table 4. During this time period, humans used the assistants as intended by the manufacturer.

In order to evaluate the accuracy of LeakyPick, the dataset was labeled by recording the timestamps of when the device was recording audio. This was accomplished by taking advantage of the visual indicator (e.g., a light ring that glows) that Smart speakers use to alert the user when the voice assistant is activated in response to voice inputs. We therefore automated the labeling process in the real-world environment by creating a small specialized device with a light sensor to measure the visual indicator. Our device consisted of a Raspberry Pi and a Light Dependent Resistor (LDR) in conjunction with a LM393 [43] analogue-digital comparator. The LDR

Table 3: Datasets for Burst Detection, Statistical Probing, Online Probing and Machine Learning

Dataset	Frequency	Devices
Idle	-	Echo Dot, Google Home, Home Pod, Hive 360 Hub
Controlled - Burst Detection	1min, 5min, 10min	Echo Dot, Google Home, Home Pod, Hive 360 Hub
Controlled - Statistical Probing	2min, 5min, 10min, 30min, 1h, 2h	Echo Dot, Google Home, Home Pod, Hive 360 Hub
Online Probing	10s during probing windows	all, cf. Table 2
Real-World	real-world, cf. Table 4	Echo Dot, Google Home, Home Pod

Table 4: Duration of collected data in different residential environments (households) while used by humans

Amazon Echo Dot	Google Home	Apple Home Pod
31d	15d	15d

sensor was then attached to the smart speaker's visual indicator and protected from environmental luminosity with an opaque foil. This setup allowed the Raspberry Pi to record a precise timestamp each time the device was activated and allowed us to label periods with audio activity in the dataset accordingly.

5.1.2 Evaluation metrics. We evaluate the performance of our detection approach in terms of true positive rate (TPR) and false positive rate (FPR). The true positive rate is defined as $TPR = \frac{TP}{TP + FN}$, where TP is true positives and FN false negatives, resulting in the fraction of audio events correctly identified as such. Similarly, false positive rate is defined as $FPR = \frac{FP}{TN + FP}$, where TN is the true negatives and FP the false positives. It denotes the fraction of non-audio events falsely identified as audio events. Ideally we would like our system to maximize TPR, i.e., the capability to identify devices sending audio to the cloud, while minimizing FPR, i.e., generating as few as possible false detections of audio transmissions.

5.2 RQ1: Detection Accuracy

In this section we evaluate the detection accuracy of our two approaches: (1) burst detection and (2) statistical probing.

5.2.1 Burst Detection. To evaluate the performance of Burst Detection for detecting audio transmissions, we used the controlled dataset for burst detection (Table 4) to determine its ability to detect audio transmissions correctly.

Figure 3 shows the receiver operating characteristic (ROC) curve for the burst detection approach. The ROC curve varies the parameter n, which defines the number of consecutive windows with high data throughput required for triggering detection (cf. Section 4.1.1),

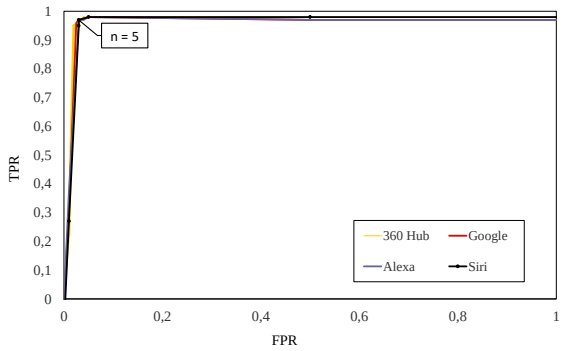


Figure 3: Results of BurstDetector using known wake-words detecting outgoing audio transmissions of Echo Dot, Google Home, Home Pod and Hive 360 Hub on the controlled data set

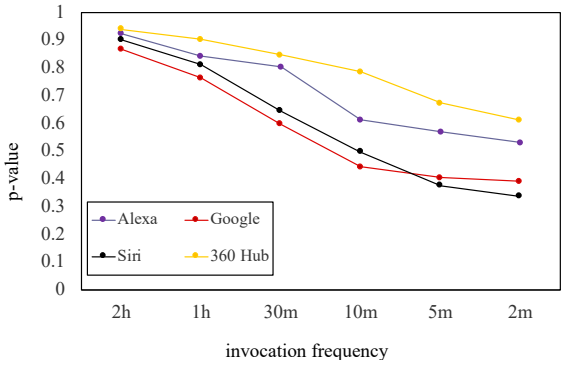


Figure 4: The resulting p-value when traffic of devices being invoked in intervals from 2 minutes to 2 hours compared to known silence, showing that the p-value decreases with an increasing number of audio bursts in the traffic

from $n_{min} = 1$ to $n_{max} = 8$. As can be seen, with $n = 5$ consecutive time windows, detection is triggered with a TPR of 96% and an FPR of 4% (averaged over all devices). This is explained by the fact that as mentioned in Section 3, the voice-activated devices typically send only a small amount of data unless they are active: medium bursts every few minutes and large bursts only every few hours when idle. This allows Burst Detection to identify nearly all audio invocations as they are clearly distinguishable from idle traffic, making this approach practical for devices with such behavioral characteristics.

5.2.2 Statistical Probing. To evaluate the ability of LeakyPick to detect whether a device reacts to audio events, we first determine whether the statistical properties of data traffic of IoT devices when in idle mode (i.e., not in relation to audio events) is statistically different from the devices' behavior when transmitting audio to the cloud. For this, we calculate the statistical difference of the packet size distributions in the Idle dataset to the packet distributions in the controlled datasets for statistical probing (Table 3) using the t-test as discussed in Section 4.1.2. The results are shown

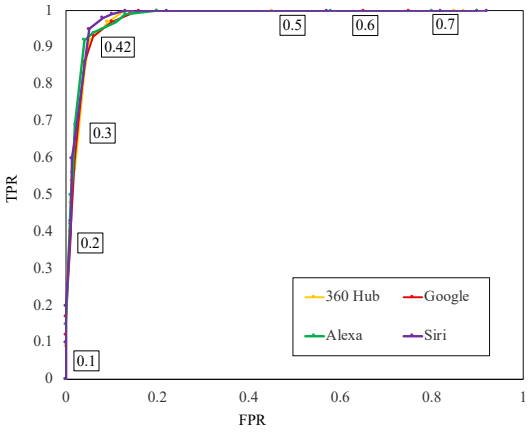


Figure 5: ROC graph of comparing consecutive windows of 30 seconds of traffic of the Controlled - Statistical probing dataset using the t-test for different p-value thresholds and comparing the output to the actual labels of the traffic

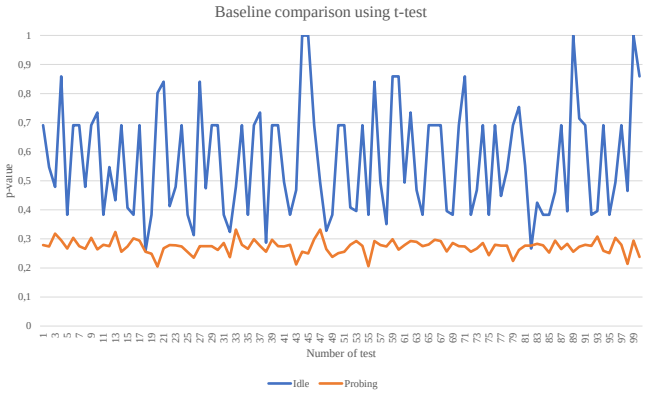


Figure 6: LeakyPick t-test p-values for probing Amazon Echo during 100 alternating windows of 1 minute of idle traffic and probing with wake-word "Alexa" at 10-second intervals, respectively

in Figure 4, showing the resulting p-value in dependence of the frequency of invocations of the corresponding device's wake-word. As can be seen, for all tested voice-controlled devices, the p-value decreases the more often the wake-word is injected, i.e., the more audio-transmissions the dataset contains. This suggests that the distributions of packet sizes related to audio transmission indeed are different to the distribution of packets in the background traffic and can be thus utilized to identify audio transmissions.

Figure 5 shows the ROC curve for our approach on the controlled dataset for statistical probing (Table 4) for different p-value thresholds. We use a sliding window approach and compare two consecutive windows of 30 seconds duration using the test, moving the window for 30 seconds to get the new window. We compare the result with the actual label of this traffic region to assess if our approach can reliably find exactly the device sending audio data. As can be seen, for a p-value threshold of 0.42 or 0.43 a True Positive Rate of 94% with a simultaneous False Positive Rate of 6% averaged over all devices can be achieved for these datasets.

5.3 RQ2: Wake-Word Sensitivity

LeakyPick is designed to detect devices reacting to a specific set of wake-words. However, as this set may be different for different device types, a relevant question is to what degree the detection accuracy is dependent on the presence of the correct wake-words in the audio probes. To evaluate this aspect, we first tested LeakyPick on the Online Probing dataset representing a live operative setting in which a number of audio probes containing actual wake-words were injected into the environment of an Amazon Echo device with the target of trying to trigger a wake-word induced audio transmission. We used the t-test as discussed in Sect. 4.1.2 to calculate the p-value between consecutive samples of packet sequences T_s and T_{pr} of duration $d = 60$ seconds each. The result of this for 100 time window pairs is shown in Figure 6. As can be seen, the p-values for the non-probing (i.e., "idle" time windows) range between approximately 0.3 and 1, whereas the p-values for time windows containing audio probes remain mostly below 0.3. This shows that given an appropriate p-value threshold LeakyPick is able to distinguish between "idle" traffic and audio transmissions.

To further evaluate how sensitive LeakyPick is to the use of the right wake-words, we compiled a set of audio probes consisting of the 50 most used English words and a set of nine known wake-words used by the IoT devices used in our evaluation (shown in Table 2). The set of audio probes was injected into the devices' environment and the resulting p-values for each device evaluated. We evaluated all devices at the same time with the same parameters, exactly as it would occur in a smart home scenario where the user has many devices in listening range. The resulting p-values for two representative examples of used audio probes are shown in Figure 7. The shown audio probes are the randomly-selected word "major", which does not correspond to any known wake-word of any of the tested devices and the Google Home wake-word "Hey Google". While these examples are provided to demonstrate the discriminative ability of our approach, similar results apply also to other words in the list of tested audio probes. As one can see, with a p-value threshold of, e.g., 0.5 the word "major" would not be considered to activate any of the devices, whereas one can clearly see that the p-value for "Hey Google" indicates a reaction by the Google Home device. From the results we can see that only devices responsive to a wake-word react to it which in turn can be detected using the statistical t-test employed by LeakyPick. This means, that the same p-value threshold can be used for any device tested. It shows that only the device actually reacting to the wake word exhibits a low enough p-value to be classified as sending audio across all other devices. Note that Nest Protect is not shown in Figure 7, as it was not activated by any of the examined words and therefore did not transmit any data at all.

5.4 RQ3 and RQ4: Real-World Performance

We evaluated LeakyPick on our real-world dataset containing 52 days of operation in residential environments (households) (Table 4). In addition to using this dataset to measure the accuracy of

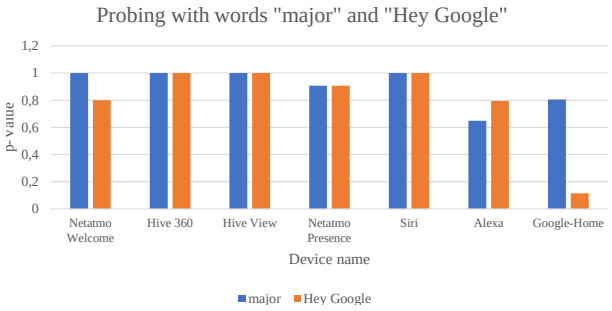


Figure 7: Representative examples of LeakyPick *p*-values for audio probes. None of the devices react to the non-wake-word probe "major" while only the Google Home device shows reaction for its wake-word "Hey Google"

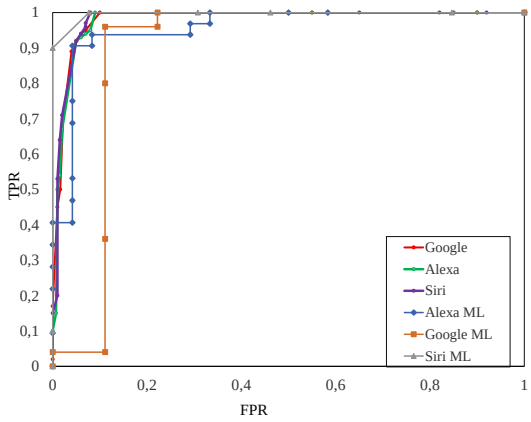


Figure 8: ROC curves of the ML-based and LeakyPick approaches on the real-world dataset

LeakyPick (**RQ3**), we also compare LeakyPick's accuracy to that of machine learning algorithms (**RQ4**). Recall from Section 3 that a key research challenge is being able to operate for unknown devices. Since machine learning algorithms require training on known devices, they are not appropriate to achieve our goals, as our approach needs to be able to handle also previously unseen device-types. That said, we use a trained machine learning algorithm as a baseline, hypothesizing that LeakyPick can perform at least as well, but without the need for training.

5.4.1 ML Implementation. We tested the performance of several commonly-used machine learning (ML) algorithms for detecting audio events in the real-world dataset. We then selected the classifier with the best performance to compare against the statistical detection approach used by LeakyPick. We consider both simple ML algorithms as well as more advanced ensemble (i.e., Bagging and Boosting) and majority voting-based classifiers. The ML algorithms tested include XGboost [10], Adaboost [17], RandomForest [5], SVM with RBF kernel [47], K-NN [2], Logistic Regression, Naïve Bayes, and Decision Tree classifiers as provided by the the Scikit-learn ML package [1]. For each classifier, the used hyper-parameters were tuned using the provided Grid-search and Cross-validation processes. For constructing features for training we extracted the sequence of packet lengths (SPL) from the traffic flow and utilized the `tsfresh` tool [12] that automatically calculates a large number of statistical characteristics from a time-ordered sequence of packets. All experiments were conducted on a laptop that runs Ubuntu Linux 18.04 with an Intel i7-9750H CPU with 32 GB DDR4 Memory.

5.4.2 Evaluation. For the ML approach, we used 90% of the dataset for training and 10% for testing. In addition, we conducted a 10-fold Cross-Validation (CV) on the training data to better evaluate the performance of the ML classifiers. According to our experiments, based on CV accuracy, the Random Forest Classifier provided the best performance on our dataset, achieving 91.0% accuracy (f1-score) on test data while 10-fold CV accuracy was 90.5%.

We also evaluated LeakyPick as described in Sect. 5.2.2 on the same real world dataset in order to compare its performance to the ML-based approach. The results are displayed in Figure 8, showing the ROC curves for both approaches on the Google Home, Siri Home Pod and Alexa Echo devices. For *p*-value threshold 0.43 LeakyPick achieves a TPR of 93% with a simultaneous FPR of 7% averaged over all devices, compared to a best-case TPR of 95% and FPR of 9.7% for the ML-based classifier for Alexa Echo Dot. We also found that models are not transferable between voice assistants. For example, training on Alexa voice traffic and using the model to identify Siri voice traffic had around 35% precision.

As our evaluation results in Figure 8 show, ML-based models are indeed able to detect audio events based on the traffic the devices send out of the network. However, the evaluation also shows that similar or even better performance can be achieved using a device-agnostic approach as taken by LeakyPick.

Since applying this kind of a profiling approach requires dedicated traffic models to be trained and included in the system for each device type considered, its practicality in real-world scenarios is questionable. Due to the very large and ever-growing number of different types of voice-activated devices, this approach seems impractical. The approach adopted in LeakyPick can achieve similar performance without the need to employ pre-trained device-type-specific detection models for audio event detection, providing it much wider applicability in a wider range of environments with diverse audio-activated device types.

5.5 RQ5: Identifying Unknown Wake-Words

To demonstrate LeakyPick's ability to identify unknown wake-words, we performed a systematic experiment with Amazon's Alexa-enabled Echo Dot. As voice assistants are conceptually similar, we believe the results can be generalized to other voice-controlled devices. We configured the Echo Dot to use the standard "Alexa" wake word (other options include "Computer", "Amazon", and "Echo"). The experiment played different audio inputs, waiting for two seconds for the visual light-ring indicator of the device to light up, indicating the device reacted to the input. For each tested audio input, we recorded the number of attempts that triggered a reaction. Recall from Section 2 that Alexa-enabled devices have two states of detection: (1) an offline model on the device, and (2) an online model. We classify a word to be mistaken as a wake-word when

Table 5: Full results of testing Alexa with English words

Probability of activating Alexa	Wake-Word
2/10	alita, baxa, elater, hexer, liker, ochna, taxer
3/10	bertha, electroceramic, excern, oxer, taxir
4/10	electrohydraulic, electropathic, wexler
5/10	blacksher, electic, hoaxer
6/10	bugsha, elatha, elator, electrodissolution, electrostenolytic, eloper, eluted, fluxer, huerta, hurter, irksome, lecher, lefter, lepre, lesser, letter, licker, lipper, loasa, loker, lotor, lyssa, maloca, maxillar, melosa, meta, metae, muleta, paxar, rickner
7/10	alexy, crytzer, electroanalytical, hyper, kleckner, lecture, likker, volupte, wexner
8/10	electroreduction, hiper, wechsler
9/10	aleta, alexa, alexia, annection, elatcha, electre, kreitzer
10/10	alachah, alexipharmic, alexiteric, alissa, alosa, alyssa, barranca, beletter, elector, electra, electroresection, electrotelegraphic, elissa, elixir, gloeckner, lechner, lecter, lictor, lxi, lxx, mixer, olexa, walesa

the word triggers at least the offline model, since this transmits recorded audio to the cloud.

Results. The Alexa-enabled Echo Dot reliably reacted to 89 words across multiple rounds of testing. Table 5 (Appendix) shows the full list of words. To estimate the phonetic distance between these words and the true wake-word, we used the Metaphone algorithm [35] to convert the words into a phonetic alphabet based on their pronunciation. The resulting words were then compared with the Levenshtein distance to "Alexa." Among the 89 words, 52 have a phonetic distance of 3 or more. We found that 3 words had a phonetic distance of 0, 9 a distance of 1, 25 a distance of 2, 29 a distance of 3, 14 a distance of 4, 2 a distance of 5 and 6, 4 of 7 and one even a distance of 8. These distances shows that the Echo Dot reliably reacted to words that are phonetically very different than "Alexa."

Some of the found wake-words can also be spoken by a human even as part of a sentence and Alexa will be activated. In a smart home scenario users speaking a sentence including such a word can mistakenly activate Alexa and therefore stream the following sentences out of the users home. This shows that those identified words are one cause of misactivations and therefore lead to recorded audio from the users home being sent to the cloud and processed by computers or even other humans. Based on these findings, it is unsurprising that Alexa-enabled devices are often triggered unintentionally, leading to private conversations and audio being transmitted outside the user's home.

The full results of testing the Alexa wake-word (Alexa) with words of the English language dictionary with 6 and 5 phonemes as well as some random words, is shown in Table 5. The results shown are the last round of 10 tests for each word. The left column shows the probability of the device being activated while replaying 10 times the word in question.

6 DISCUSSION

Burst Detector: A malicious audio bug device whose sole purpose is to eavesdrop on a victim may use extensive lossy audio compression to keep the traffic rate below the detection threshold of $23kbit/s$. However, such audio may not be suitable for automated voice recognition as many features of the voice are deleted or exchanged with noise which impairs the scalability of such an attack dramatically. However, our statistical probing approach would still detect a significant difference in the traffic and detect the sent audio.

Statistical Probing: As mentioned in Section 2, attacks that issue commands to a victim's voice assistant can be detected by LeakyPick. To achieve that, increasing the time traffic samples are acquired as well as disabling audio probing is needed. By disabling the audio probing mechanism, every invocation of the device must be done by an external entity (e.g., the user or an attacker). By increasing the sample size, it is also possible to distinguish reliably between an actual invocation and background traffic spikes, even without the knowledge of when audio is played or not as the p-values are different for an invocation and background noise (cf. Figure 4). With this tweak, LeakyPick would also be able to warn the user of such attacks. Currently we are investigating into the influence of varying levels of background noise on the Statistical Probing approach.

Countermeasures against Devices sending Audio: Depending on whether LeakyPick acts as the gateway of the home network or is sniffing passively the (encrypted) Wi-Fi traffic, there are different approaches to prevent a device from recording and sending audio without the user's permission. If our device is replacing the gateway, traffic identified as containing audio recordings can be simply dropped at the network layer. If our device can only passively sniff encrypted MAC layer traffic, inaudible microphone jamming techniques could be used to prevent the device from recording unsuspecting users private conversations [30, 37, 38, 42, 50].

Wake-Word Identification: We found that some of the identified wake-words for Alexa are only effective if spoken by Google's TTS voice, and that we were unable to replicate the effect when spoken by a human. We believe this may result from features that differ between the TTS service audio file and natural voice. However, the goal of the experiment was to demonstrate the extent to which words are incorrectly interpreted as wake-words, rather than determining the actual words triggering incorrect wake-word detection. There may also be wake-words, sounds, or noise our approach could not find. We are currently investigating whether wrongly recognized wake-words could be used to attack voice assistants and other voice recognition applications.

7 RELATED WORK

Existing works discussing detection of inadvertent data transmissions out of a user network have focused on IP cameras. To the best of our knowledge, there are no published approaches for detecting outgoing audio traffic for voice assistants and other audio-activated devices, in particular approaches utilizing audio probing. We are also not aware of publications utilizing audio probes to determine if devices react to audio inputs.

The following discussion of related work focuses on existing attacks on voice assistants and traffic analysis approaches for IoT

device identification and IP camera detection. We also review approaches to microphone jamming, which can be used by LeakyPick to prevent microphone-enabled IoT devices to record meaningful audio when the user is not aware of it.

IP Camera Detection: IP camera detection approaches usually extract features from packet headers. Wireless cameras in operation continuously generate camera traffic flows that consist of video and audio streams. The resulting traffic patterns of IP cameras are likely to be different and easily distinguishable from that of other network applications. Furthermore, to save bandwidth, IP cameras utilize variable bit rate (VBR) video compression methods, like H264. Because of the use of VBR, by changing the scene the camera monitors a change in the bitrate of the video can be enforced. Finally, by correlating scene changes and traffic bitrate changes cameras, monitoring can be identified.

Cheng et al. [11] propose using the person being monitored to change the scene by letting them move around. The resulting traffic is then classified using machine learning. Similarly Liu et al. [29] focus on altering the light condition of a private space to manipulate the IP camera's monitored scene. The resulting stream also changes its bitrate and can therefore be distinguished from non-altered streams, e.g., by using the statistical t-test. The above proposals are completely orthogonal to our approach, as they are customized for cameras. In addition, they make assumptions that are not applicable to microphone-enabled IoT devices, e.g., utilizing a variable bit rate encoding (VBR) and continuous data transmission.

Traffic Analysis: Numerous classification techniques have been proposed to learn the behavior of IoT devices, distinguishing and identifying IoT devices based on their traffic profile. Sivanathan et al. [41] use network traffic analysis to characterize the traffic corresponding to various IoT devices. They use the activity pattern (traffic rate, burstiness, idle duration) and signalling overheads (broadcasts, DNS, NTP) as features to distinguish between IoT and non-IoT traffic. However, the approach requires training. Nguyen et al. [33] propose an autonomous self-learning distributed system for detecting compromised IoT devices. Their system builds on device-type-specific communication profiles without human intervention nor labeled data which are subsequently used to detect anomalous deviations in devices' communication behavior, potentially caused by malicious adversaries. However, these proposals focus on detecting anomalous behavior not consistent with benign device actions. In contrast, our goal is to detect benign actions in response to audio events, which may or may not be falsely detected. Also our approach does not require the system to identify IoT devices based on their traffic.

Eavesdropping Avoidance: Microphone, and more specifically, voice assistant jamming attacks have been proposed by several prior works. Roy et al. [37] present an approach for inaudibly injecting audio to jam spy microphones using ultrasonic frequencies and ultrasound modulated noise. As it is inaudible to humans, the jamming is not interfering with human conversations. Zhang et al. [50] build upon this work to inaudibly inject commands into voice assistants, demonstrating that voice assistants and possibly other commodity IoT devices are susceptible to the proposed ultrasonic control. Mitev et al. [30] further build upon these findings to precisely jam human voice and inject recorded voice into voice assistants. As discussed in Section 6, inaudible jamming approaches could be used by LeakyPick to prevent a device from recording meaningful audio when the user is not aware of it. In future work we aim to use these approaches as an additional component of LeakyPick, further increasing the privacy gains of our approach.

Voice Assistant Attacks: Voice assistants using voice recognition are fairly new and many security and privacy aspects are still to be improved.The common goal of such attacks is to control the voice assistant of a user without him noticing.Diao et al. [14] present attacks against the Google Voice Search (GVS) app on Android. A malicious app on the smart phone can activate GVS and simultaneously play back a recorded or synthesized command over the built-in speakers which is then picked up by the microphone, to control the victim's voice assistant. Alepis et al. [3] extend upon this attack. They then proceed to use use multiple devices to overcome implemented countermeasures by showing that infected devices can issue commands to other voice-activated devices such as the Amazon Echo or other smart phones.

Vaidya et al. [46] present a method to change a recording of human voice so that it is no longer comprehensible by humans but still correctly recognizable by voice recognition systems. Carlini et al. [7] extended this work by presenting voice mangling on a voice recognition system where the underlying mechanics are known, resulting in a more precise attack. Since a mangled voice may alert nearby users, Schönherr et al. [40] and Yuan et al. [49] propose methods for hiding commands inside other audio files (e.g., music files) such that they are not recognizable by humans. Similarly, Carlini et al. [8] create audio files with similar waveforms, which Mozilla's DeepSpeech interprets as different sentences.

Voice assistant extensions have also been attacked. Kumar et al. [28] showed that utterances exist such that Alexa's speech-to-text engine systematically misinterprets them. Using these findings they proposed *Skill Squatting*, which tricks the user into opening a malicious Skill. Simultaneously, Zhang et al. [51] proposed using malicious Skills with a similarly pronounced or paraphrased invocation-name to re-route commands meant for that Skill.

These attacks show that an attacker is able to manipulate the interaction flow with a voice assistant by, e.g., issuing commands without the victim noticing, turning voice assistants into a potential privacy and security risk for the user. LeakyPick can warn the user if their voice assistant is under attack without him noticing it. When combined with eavesdropping avoidance (e.g., jamming), the attacks could be mitigated or even prevented.

8 CONCLUSION

As smart home IoT devices increasingly adopt microphones, there is a growing need for practical privacy defenses. In this paper, we presented the LeakyPick architecture that enables detection of smart home devices that unexpectantly stream recorded audio to the Internet in response to observing a sound. Conceptually, LeakyPick periodically "probes" other devices in its environment and monitors the subsequent network traffic for statistical patterns that indicate audio transmission. We built a prototype of LeakyPick on a Raspberry Pi and demonstrate an accuracy of 94% in detecting audio transmissions from eight different devices with voice assistant capabilities without any *a priori* training. It also identified 89 words

that could unknowingly trigger an Amazon Echo Dot to transmit audio to the cloud. As such, LeakyPick represents a promising approach to mitigate a real threat to smart home privacy.

ACKNOWLEDGMENTS

We thank our anonymous reviewers for their valuable and constructive feedback. This work was funded by the Deutsche Forschungsgemeinschaft (DFG) – SFB 1119 – 236615297.

REFERENCES

[1] 2020. *Scikit-learn Python machine learning library.* https://github.com/scikit-learn/
[2] David W Aha, Dennis Kibler, and Marc K Albert. 1991. Instance-based learning algorithms. *Machine learning* 6, 1 (1991).
[3] Efthimios Alepis and Constantinos Patsakis. 2017. Monkey says, monkey does: security and privacy on voice assistants. *IEEE Access* 5 (2017).
[4] Amazon. [n.d.]. *Alexa Built-in Products with AVS - SpeechRecognizer.* https://developer.amazon.com/de-DE/docs/alexa/alexa-voice-service/speechrecognizer.html
[5] Leo Breiman. 2001. Random forests. *Machine learning* 45, 1 (2001).
[6] Dell Cameron. 2018. *Hack Can Turn Robotic Vacuum Into Creepy Rolling Surveillance Machine.* https://gizmodo.com/hack-can-turn-robotic-vacuum-into-creepy-rolling-survei-1827726378
[7] Nicholas Carlini, Pratyush Mishra, Tavish Vaidya, Yuankai Zhang, Micah Sherr, Clay Shields, David Wagner, and Wenchao Zhou. 2016. Hidden voice commands. In *Proceedings of the USENIX Security Symposium.*
[8] Nicholas Carlini and David Wagner. 2018. Audio adversarial examples: Targeted attacks on speech-to-text. *arXiv preprint arXiv:1801.01944* (2018).
[9] G. Casella and B. Roger. 199. *Statistical Inference.* Duxbury, 2nd edition.
[10] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (KDD).*
[11] Yushi Cheng, Xiaoyu Ji, Tianyang Lu, and Wenyuan Xu. 2018. DeWiCam: Detecting Hidden Wireless Cameras via Smartphones. In *Proceedings of the ACM Asia Conference on Computer and Communications Security (ASIACCS).*
[12] Maximilian Christ, Nils Braun, Julius Neuffer, and Andreas W. Kempa-Liehr. 2018. Time Series FeatuRe Extraction on basis of Scalable Hypothesis tests (tsfresh – A Python package). *Neurocomputing* 307 (2018). https://doi.org/10.1016/j.neucom.2018.03.067
[13] Joseph Cox. [n.d.]. *Revealed: Microsoft Contractors Are Listening to Some Skype Calls.* https://www.vice.com/en_us/article/xweqbq/microsoft-contractors-listen-to-skype-calls
[14] Wenrui Diao, Xiangyu Liu, Zhe Zhou, and Kehuan Zhang. 2014. Your voice assistant is mine: How to abuse speakers to steal information and control your phone. In *Proceedings of the ACM Workshop on Security and Privacy in Smartphones & Mobile Devices (SPSM).*
[15] DIODES Incorporated. [n.d.]. *FILTERLESS 3W CLASS-D STEREO AUDIO AMPLIFIER.* https://www.diodes.com/assets/Datasheets/PAM8403.pdf
[16] Education First. [n.d.]. *1000 most common words in English.* https://www.ef.com/wwen/english-resources/english-vocabulary/top-1000-words/
[17] Yoav Freund, Robert E Schapire, et al. 1996. Experiments with a new boosting algorithm. In *Proceedings of the International Conference on Machine Learning (ICML)*, Vol. 96.
[18] Sarah Frier. 2019. *Facebook Paid Contractors to Transcribe Users' Audio Chats.* https://www.bloomberg.com/news/articles/2019-08-13/facebook-paid-hundreds-of-contractors-to-transcribe-users-audio
[19] Google. [n.d.]. *Learn about Nest Protect's automatic Sound Check test.* https://support.google.com/googlenest/answer/9242130?hl=en
[20] Google. [n.d.]. *Learn how Nest cameras and Nest Hello detect sound and motion.* https://support.google.com/googlenest/answer/9250426?hl=en
[21] Andy Greenberg. 2017. *This hack lets Amazon Echo 'remotely snoop' on users.* https://www.wired.co.uk/article/amazon-echo-alexa-hack
[22] Caroline Haskins. 2019. *Amazon Is Coaching Cops on How to Obtain Surveillance Footage Without a Warrant.* https://www.vice.com/en_us/article/43kga3/amazon-is-coaching-cops-on-how-to-obtain-surveillance-footage-without-a-warrant
[23] Lente Van Hee, Ruben Van Den Heuvel, Tim Verheyden, and Denny Baert. [n.d.]. *Google employees are eavesdropping, even in your living room, VRT NWS has discovered.* https://www.vrt.be/vrtnws/en/2019/07/10/google-employees-are-eavesdropping-even-in-flemish-living-rooms/
[24] Alex Hern. 2019. *Apple contractors 'regularly hear confidential details' on Siri recordings.* https://www.theguardian.com/technology/2019/jul/26/apple-contractors-regularly-hear-confidential-details-on-siri-recordings
[25] Hive. [n.d.]. Hive Hub 360. https://www.hivehome.com/products/hive-hub-360 (Accessed June 2020).
[26] Infochimps.com. [n.d.]. *350000 simple english words.* http://www.infochimps.com/datasets/word-list-350000-simple-english-words-excel-readable
[27] Business Insider. [n.d.]. *Google says the built-in microphone it never told Nest users about was 'never supposed to be a secret'.* https://www.businessinsider.com/nest-microphone-was-never-supposed-to-be-a-secret-2019-2
[28] Deepak Kumar, Riccardo Paccagnella, Paul Murley, Eric Hennenfent, Joshua Mason, Adam Bates, and Michael Bailey. 2018. Skill squatting attacks on amazon alexa. In *Proceedings of the USENIX Security Symposium.*
[29] Tian Liu, Ziyu Liu, Jun Huang, Rui Tan, and Zhen Tan. 2018. Detecting Wireless Spy Cameras Via Stimulating and Probing. In *Proceedings of the ACM International Conference on Mobile Systems, Applications, and Services (MobiSys).*
[30] Richard Mitev, Markus Miettinen, and Ahmad-Reza Sadeghi. 2019. Alexa Lied to Me: Skill-based Man-in-the-Middle Attacks on Virtual Assistants. In *Proceedings of the ACM Asia Conference on Computer and Communications Security (ASIACCS).*
[31] David Monsees. [n.d.]. *More information about our processes to safeguard speech data.* https://www.blog.google/products/assistant/more-information-about-our-processes-safeguard-speech-data/
[32] Alfred Ng. 2019. *Alexa and Google Assistant fall victim to eavesdropping apps.* https://www.cnet.com/news/alexa-and-google-voice-assistants-app-exploits-left-it-vulnerable-to-eavesdropping/
[33] Thien Duc Nguyen, Samuel Marchal, Markus Miettinen, Minh Hoang Dang, N. Asokan, and Ahmad-Reza Sadeghi. 2018. DÏoT: A Crowdsourced Self-learning Approach for Detecting Compromised IoT Devices. *CoRR* abs/1804.07474 (2018). arXiv:1804.07474 http://arxiv.org/abs/1804.07474
[34] TJ OConnor, Reham Mohamed, Markus Miettinen, William Enck, Bradley Reaves, and Ahmad-Reza Sadeghi. 2019. HomeSnitch: behavior transparency and control for smart home IoT devices. In *Proceedings of the ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec).*
[35] Lawrence Philips. 1990. Hanging on the metaphone. *Computer Language* 7, 12 (1990).
[36] Raspberry Pi Foundation. [n.d.]. *Raspberry Pi 3 Model B.* https://www.raspberrypi.org/products/raspberry-pi-3-model-b/
[37] Nirupam Roy, Haitham Hassanieh, and Romit Roy Choudhury. 2017. BackDoor: Making Microphones Hear Inaudible Sounds. In *Proceedings of the ACM International Conference on Mobile Systems, Applications, and Services (MobiSys).*
[38] Nirupam Roy, Sheng Shen, Haitham Hassanieh, and Romit Roy Choudhury. 2018. Inaudible voice commands: The long-range attack and defense. In *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI).*
[39] Alex Rudnicky. [n.d.]. *The CMU Pronouncing Dictionary.* http://www.speech.cs.cmu.edu/cgi-bin/cmudict
[40] Lea Schönherr, Katharina Kohls, Steffen Zeiler, Thorsten Holz, and Dorothea Kolossa. 2018. Adversarial Attacks Against Automatic Speech Recognition Systems via Psychoacoustic Hiding. *arXiv preprint arXiv:1808.05665* (2018).
[41] Arunan Sivanathan, Daniel Sherratt, Hassan Habibi Gharakheili, Adam Radford, Chamith Wijenayake, Arun Vishwanath, and Vijay Sivaraman. 2017. Characterizing and classifying IoT traffic in smart cities and campuses. *Proceedings of the IEEE INFOCOM Workshop on Smart Cities and Urban Computing (SmartCity)* (2017).
[42] Liwei Song and Prateek Mittal. 2017. Inaudible voice commands. *arXiv preprint arXiv:1708.07238* (2017).
[43] Texas Instruments. [n.d.]. *Low-Power, Low-OffsetVoltage, Dual Comparators.* https://www.ti.com/lit/ds/symlink/lm393-n.pdf
[44] Kevin C. Tofel. 2018. *Here's why smart home hubs seem to be dying a slow, painful death.* https://staceyoniot.com/heres-why-smart-home-hubs-seem-to-be-dying-a-slow-painful-death/
[45] tp-link. [n.d.]. *150Mbps High Gain Wireless USB Adapter - TL-WN722N.* https://www.tp-link.com/en/home-networking/adapter/tl-wn722n/
[46] Tavish Vaidya, Yuankai Zhang, Micah Sherr, and Clay Shields. 2015. Cocaine noodles: exploiting the gap between human and machine speech recognition. In *Proceedings of the USENIX Workshop on Offensive Technologies (WOOT).*
[47] Vladimir Vapnik. 2013. *The nature of statistical learning theory.* Springer science & business media.
[48] Merriam Webster. [n.d.]. .
[49] Xuejing Yuan, Yuxuan Chen, Yue Zhao, Yunhui Long, Xiaokang Liu, Kai Chen, Shengzhi Zhang, Heqing Huang, Xiaofeng Wang, and Carl A Gunter. 2018. CommanderSong: A Systematic Approach for Practical Adversarial Voice Recognition. *arXiv preprint arXiv:1801.08535* (2018).
[50] Guoming Zhang, Chen Yan, Xiaoyu Ji, Tianchen Zhang, Taimin Zhang, and Wenyuan Xu. 2017. Dolphinattack: Inaudible voice commands. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS).*
[51] Nan Zhang, Xianghang Mi, Xuan Feng, XiaoFeng Wang, Yuan Tian, and Feng Qian. 2018. Understanding and Mitigating the Security Risks of Voice-Controlled Third-Party Skills on Amazon Alexa and Google Home. *arXiv preprint arXiv:1805.01525* (2018).

C

FakeWake: Understanding and Mitigating Fake Wake-up Words of Voice Assistants (CCS'21, CORE: A*)

[14] Yanjiao Chen, Yijie Bai, Richard Mitev, Kaibo Wang, Ahmad-Reza Sadeghi, and Wenyuan Xu. Fakewake: Understanding and mitigating fake wake-up words of voice assistants. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 1861–1883, 2021. https://doi.org/10.1145/3460120.3485365.

FakeWake: Understanding and Mitigating Fake Wake-up Words of Voice Assistants

Yanjiao Chen
Zhejiang University
chenyanjiao@zju.edu.cn

Yijie Bai
Zhejiang University
baiyj@zju.edu.cn

Richard Mitev
Technical University of Darmstadt
richard.mitev@trust.tu-darmstadt.de

Kaibo Wang
Zhejiang University
kaibo@zju.edu.cn

Ahmad-Reza Sadeghi
Technical University of Darmstadt
ahmad.sadeghi@trust.tu-darmstadt.de

Wenyuan Xu
Zhejiang University
xuwenyuan@zju.edu.cn

Abstract

In the area of Internet of Things (IoT), voice assistants have become an important interface to operate smart speakers, smartphones, and even automobiles. To save power and protect user privacy, voice assistants send commands to the cloud only if a small set of pre-registered wake-up words are detected. However, voice assistants are shown to be vulnerable to the *FakeWake* phenomena, whereby they are inadvertently triggered by innocent-sounding fuzzy words. In this paper, we present a systematic investigation of the *FakeWake* phenomena from three aspects. To start with, we design the first fuzzy word generator to automatically and efficiently produce fuzzy words instead of searching through a swarm of audio materials. We manage to generate 965 fuzzy words covering 8 most popular English and Chinese smart speakers. To explain the causes underlying the *FakeWake* phenomena, we construct an interpretable tree-based decision model, which reveals phonetic features that contribute to false acceptance of fuzzy words by wake-up word detectors. Finally, we propose remedies to mitigate the effect of *FakeWake*. The results show that the strengthened models are not only resilient to fuzzy words but also achieve better overall performance on original training datasets.

CCS Concepts

• **Security and privacy** → **Privacy protections**; • **Computing methodologies** → **Heuristic function construction**; • **Human-centered computing** → **Mobile devices**.

Keywords

voice assistants, fuzzy words, interpretable machine learning, security

ACM Reference Format:
Yanjiao Chen, Yijie Bai, Richard Mitev, Kaibo Wang, Ahmad-Reza Sadeghi, and Wenyuan Xu. 2021. *FakeWake*: Understanding and Mitigating Fake Wake-up Words of Voice Assistants. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security (CCS '21), November 15–19, 2021, Virtual Event, Republic of Korea.* ACM, New York, NY, USA, 23 pages. https://doi.org/10.1145/3460120.3485365

CCS '21, November 15–19, 2021, Virtual Event, Republic of Korea

ACM ISBN 978-1-4503-8454-4/21/11…$15.00
https://doi.org/10.1145/3460120.3485365

1 Introduction

Voice assistants are popular interfaces embedded in smart Internet of Things (IoT) devices (e.g., smart speakers), which enable us to use voice commands to execute various operations, e.g., send messages, make calls, and even control (e.g., open the door) their IoT ecosystem (e.g., smart home appliances). Despite the recession under the influence of COVID-19, the global smart speaker market is expected to grow by 21% in 2021 [5]. With the omnipresence of voice assistants in the near future, the potential threats to user privacy and security regarding misconduct of voice assistants have to be addressed.

Almost all voice assistants adopt the wake-up mechanism. Before being triggered for receiving voice commands, voice assistants actively listen to the surrounding environment for wake-up words, which are usually short and catchy words chosen by manufacturers to brand their products[1]. Once the lightweight local detection model believes that it has detected a wake-up word, the voice assistant will record and send audio to the cloud for further analysis.

Unfortunately, voice assistants suffer from the *FakeWake* phenomena, whereby they can be wrongly activated by words that are not wake-up words. We define the words that are not wake-up words but induce the *FakeWake* phenomena as *fuzzy words*, and the ones that do not activate voice assistants as non-fuzzy words. The *FakeWake* phenomena is fairly prevalent: Recent surveys show that 50% of users wake their voice assistants up by mistake once a week and 28.5% of them even experience daily accidental wake-up [11, 39]. As shown in Figure 1, the *FakeWake* phenomena can be incurred by sources such as human conversation, TV shows [13, 37], and TTS-spoken texts [29, 37]. The *FakeWake* phenomena pose privacy and security risks, e.g., uploading audio with sensitive information to the cloud or accepting malicious commands without be noticing. The Amazon Echo has been reported to be activated mistakenly and sent the recorded private conversation of family members to various random contacts [10]. Prior efforts have found several fuzzy words [13, 29, 37] without understanding why and how to defend against them. Thus, in this paper, we aim to systematic study the root causes and mitigation of the *FakeWake* problems.

[1] A few voice assistants, e.g., Xiaomi, allow users to customize their own wake-up words, which may even aggravate the *FakeWake* phenomena if users choose convenient but commonly-used words.

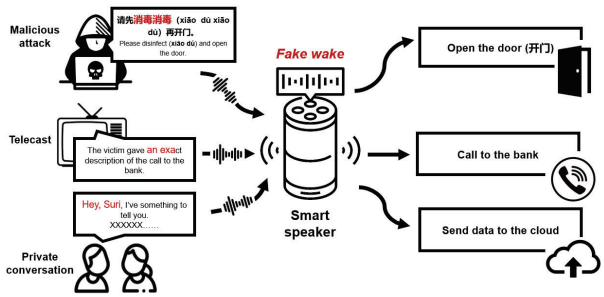


Figure 1: An illustration of the *FakeWake* phenomena. Various sources from the attacker-generated audios, TV shows and private conversations may incur the *FakeWake* phenomena, and result in privacy leakage and security threats.

Particularly, we focus on studying the *FakeWake* phenomena, aiming to answer the following questions.

- *How to efficiently generate a large collection of fuzzy words for a given voice assistant?*
- *What are the causes that lead to false acceptance of fuzzy words by wake-up detectors?*
- *How to strengthen wake-up detectors of voice assistants to be resilient against fuzzy words?*

Generating. First of all, we target at generating large quantities of fuzzy words for a given voice assistants in an efficient manner, which provides the samples for analyzing the causes of the *FakeWake* phenomena and to strengthen the wake-up word detector. A naive solution is to continuously play audio materials and record whether the smart speakers are activated or not, which takes days or even weeks to find a few dozens of mis-activation incidents [13, 29, 37], and in many cases, the triggers are the real wake-up words themselves articulated in audio materials. For the sake of security, we are interested in fuzzy words that are not only able to activate the voice assistant but also sound dissimilar to the real wake-up word to avoid being detected by users. The task is made challenging because commodity voice assistants are typically black-box and we have little information of the AI-based wake-up word detection model. To address this challenge, we carefully design a framework for fuzzy word generation, which mutates the best candidates for fuzzy words to quickly create new fuzzy words through multiple evolutionary generations, and balances the wake-up rate and the dissimilarity distance.

Additionally, we investigate voice assistants for both English and Chinese, which have the most speakers worldwide [20]. To customize the generation framework for English and Chinese, we need to encode the English and Chinese words into vectors and quantify the dissimilarity distance between two English or Chinese words. Nonetheless, the word composition and pronunciation rules of English and Chinese are different, making it difficult to apply the same encoding system and dissimilarity measurement to the two languages. After carefully investigating the word structure and pronunciation patterns of English and Chinese, we tailor the generation framework to cater to the linguistic features of the two languages respectively.

For 8 popular English and Chinese smart speakers, i.e., Amazon Echo, Echo Dot, Google Assistant, Apple Siri, Baidu, Xiaomi, AliGenie, and Tencent, we manage to generate a total of 965 fuzzy words within 4 hours, while previous effort found merely 194 fuzzy words in 13 days [37]. In particular, we have generated 577 Chinese fuzzy words, 30 times as many as the Chinese fuzzy words found in [37]. By generating more fuzzy words, our methods greatly increase the attack success probability, which poses a real threat to the security and privacy of smart speakers. Moreover, the rich corpus of generated fuzzy words deepens our understanding of these words and therefore provides the training data needed for our mitigation approach. The subjective tests with human volunteers verify that the generated fuzzy words sound far from the real wake-up words, which means that these fuzzy words may be used to wrongly activate voice assistants in a more surreptitious way.

Understanding. Given the generated fuzzy words, we target at revealing why wake-up word detectors wrongly accept these fuzzy words. Under the black-box settings, explaining the *FakeWake* phenomena is challenging since we have no knowledge of the internal structure and parameters of wake-up word detectors, thus unable to gauge the cause of *FakeWake* at the model level. A possible way of explanation is to measure the Levenshtein distance between fuzzy words and the real wake-up words [37]. However, experiments show that our generated fuzzy words have similar Levenshtein distance as non-fuzzy words to the real wake-up words. To address this problem, we develop a more sophisticated explanation framework. To start with, we train an interpretable tree-based binary classifier to distinguish fuzzy words from non-fuzzy words, based on which we deduce a dissimilarity score that can well separate fuzzy words and non-fuzzy words. Then, we pinpoint the features that contribute the most to the false acceptance of fuzzy words based on the SHAP value [27]. It is demonstrated that the decisive factors usually concentrate on a small snippet of the word, e.g., *ks* in Alexa and *ai* in Xiaomi (the wake-up word is xiǎo ài tóng xué, i.e., 小爱同学). We show that a wake-up word detector that concentrates on fewer decisive factors will have more fuzzy words.

Knowing the decisive factors that lead to false acceptance of fuzzy words is helpful in two aspects. On the one hand, we can quickly construct fuzzy words by keeping the decisive factors and alter the other parts of the words. On the other hand, wake-up word detectors may be strengthened against fuzzy words by paying special attention to the decisive factors.

Mitigating. After understanding the causes of false acceptance of fuzzy words, we can leverage the findings to help defend against the *FakeWake* phenomena, which is an unexplored territory, possibly due to a lack of access to commercial models. In regard to this, we propose two potential remedies. The first approach is to screen input audios for decisive factors, e.g., *ks*. If there is no decisive factor, the audio is fed into the lightweight wake-up word detector for decision-making; otherwise, the audio will be scrutinized by more complicated speech recognition models. The second method is to strengthen wake-up word detectors by retraining with the generated fuzzy words. As the wake-up word detectors on commercial voice assistants are unavailable, we resort to the open-source GRU recurrent network model Precise [19]. Surprisingly, our experiments on five wake-up word detectors of "Alexa", "Computer", "Athena", "Hi Xiaowen" and "Hi Mia" show that the strengthened

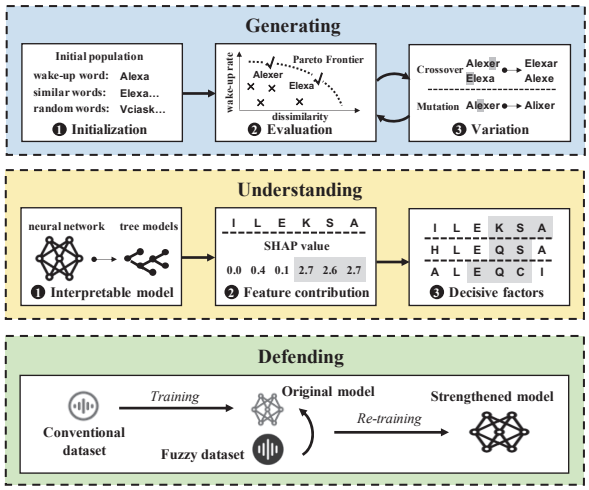


Figure 2: A systematic study on the *FakeWake* phenomena. We generate large quantities of fuzzy words for both English and Chinese smart speakers, and then analyze these fuzzy words to unveil the causes of their false acceptance by wake-up word detectors, based on which we propose remedies to mitigate the *FakeWake* phenomena.

models not only reject more than 97% of the fuzzy words, but also become better at distinguishing non-fuzzy words. The possible reason is that the fuzzy words are near the decision boundaries of wake-up word detectors, which helps the detectors to learn the decision boundaries in a more efficient and more precise way. The main flow of our paper is summarized in Figure 2.

In summary, our main contributions are as follows.

(1) We propose a systematic and automatic generation framework for producing fuzzy words and customize the framework for both English and Chinese voice assistants. We conduct extensive evaluations on eight most popular English and Chinese smart speakers, and find a total of 965 fuzzy words. The sheer volume of our generated fuzzy words reveals the vulnerability of the wake-up word detectors used by voice assistants. In addition, the rich corpus of generated fuzzy words enables us to conduct in-depth analysis to understand the fuzzy words and provide the data needed for mitigating the *FakeWake* phenomena.
(2) We build an explanation framework for the *FakeWake* phenomena, which locates the decisive factors that lead to false acceptance of fuzzy words.
(3) We present countermeasures to strengthen wake-up word detectors against fuzzy words, which improve the overall performance of wake-up word detectors.

Currently, manufacturers choose wake-up words with more concerns on commercial interests than on security. With our effort on dissecting the *FakeWake* problem, we hope to raise the attention on potential risks of wake-up words and motivate future works on improving the security of wake-up words and the robustness of wake-up word detectors.

2 Background

2.1 Voice Assistant

Almost all popular smart speakers and most smartphones are equipped with embedded voice assistants, e.g., Amazon Echo, Apple Siri, and Google Home. To reduce power consumption and protect user privacy, nearly every voice assistant uses a wake-up mechanism, i.e., no uploading recorded audios to the cloud until the wake-up word is detected. The wake-up word(s) for a voice assistant are usually unique or limited to a pre-registered set of words. For instance, Amazon Echo uses "Alexa" as its wake-up word and Google Home can be woken up by either "OK Google" or "Hey Google". Some Chinese voice assistants use their brand names as the wake-up words, e.g., "tiān māo jīng líng" for AliGenie (named 天猫精灵) and "xiǎo dù xiǎo dù" for Baidu smart speakers (named 小度).

It is known that voice assistants can be mistakenly woken up by fuzzy words other than the authentic wake-up words [11, 39], which raises security and privacy concerns. If such fuzzy words occur inadvertently in conversations or if malicious attackers play innocent-sounding audio files containing fuzzy words, smart speakers may be activated by mistake and transmit the recorded voice afterwards to the cloud or to specific contacts (e.g., the attacker). To make matters worse, attackers may issue malicious commands to the activated voice assistants, e.g., open the door or turn off the alarm system, which poses great threat to user safety. Therefore, to protect voice assistants from being wrongly activated by fuzzy words is of great importance.

2.2 Wake-up Word Detection

Wake-up word detection is essential to voice assistants. During the standby mode, voice assistants listen to the environment and record snippets of audio samples (e.g., 3s) to check the presence of wake-up words. A voice-activity detection module confirms the presence of voice, and then extract features, e.g., the mel-frequency cepstrum coefficients (MFCC), to feed into detection models (e.g., GMM, HMM, DNN) to determine the existence of wake-up words. For most voice assistants, there is a lightweight local detector model deployed on the smart devices and a more complicated model deployed on the cloud. Only the audio samples that are believed to contain wake-up words by the local model will be sent to the remote model for further examination. Different from speech recognition models, wake-up word detectors are keyword-spotting models that only focus on differentiating a specific keyword from all other words rather than translating the texts for any audio content. The output of wake-up word detectors (accept or reject) is only known to the manufacturer but will not be fed back to the users.

2.3 Threat Model

Considering searching for fuzzy words for commercial voice assistants, we make the following assumptions.

No access to the wake-up word detection model (black-box). The attacker has no knowledge of the detection model, including model structure, parameters and hyperparameters[2]. The

[2]Reverse-engineering the wake-up word detector, e.g., using model extraction methods, is possible. However, most recent model extraction methods achieve only about 70% agreement rate between the substitute model and commercial APIs [45].

output labels and confidence scores of the wake-up word detector are unavailable. The attacker can only interact with the voice assistant and observe whether it is activated or not, e.g., LED on/off.

No access to the training dataset. The training datasets of wake-up word detectors are privately collected by manufacturers with regard to the unique wake-up word of their products. The attacker has no access to the training dataset, thus cannot obtain the exact wake-up word detection model or infer any deficiency of the training process.

Attacker's ability. We assume that the attacker can acquire smart devices (e.g., smart speakers, smartphones) equipped with the targeted voice assistant. The attacker can query the smart devices for unlimited times, and indicate whether the voice assistant is activated or not. The attacker has speakers to play the generated fuzzy word candidates. Ultimately, the goal of the attacker is to insert the generated fuzzy words into innocent-sounding music or video clips to activate the voice assistant without users noticing, then to issue hidden commands to conduct malicious operations, e.g., upload private conversations to the cloud or open the door.

3 Generating Fuzzy Words

3.1 Framework Overview

In the strict black-box settings, no information about the wake-up word detector is available, thus gradient-based optimization methods cannot be used to generate fuzzy words. Therefore, we resort to heuristic algorithms, which stochastically search for solutions without the gradient information. Among commonly-used heuristic algorithms, genetic algorithm is the most suitable one to solve the problem of fuzzy word generation. Simulated annealing suffers from slow convergence, and it is difficult to apply ant colony optimization (ACO) or particle swarm optimization (PSO) to generate fuzzy words, since ACO deals with problems that can be converted into shortest path finding problems on a graph, while PSO requires the position information in order to move a group of particles in a search-space towards the optimal solutions. Genetic algorithms treat each candidate solution as an individual that contains several chromosomes, and these chromosomes can be mutated and crossed-over to evolve into new individuals. For example, we can regard "alexa" as an individual consisting of chromosomes "a", "l", "e", "x" and "a". If we mutate the chromosome "a" to "i", we get a new individual "alexi", and if we cross "alexa" with "olive" at "e" and "i", we obtain two new individuals "alive" and "olexa".

The key to utilizing genetic algorithm to generate fuzzy words is how to create a diversified initial batch of words that can efficiently evolve into fuzzy words and how to measure whether a word is "good" in terms of its ability to activate the voice assistants and its dissimilarity to the real wake-up word. To tackle these problems, we design the fuzzy word generation framework as follows.

(1) **Initialization.** To achieve both diversity and fast convergence, we include three groups of individuals in the initial batch: the real wake-up word itself, words that are similar to the wake-up word (measurement of similarity will be given in 3.2), and randomly-generated words. Note that we have tried to adopt an entirely random initial population, but found that most random words are non-fuzzy words, and will be killed in the first generation, leaving few to breed useful offspring.

(2) **Evaluation.** If we only evaluate an individual in terms of its wake-up rate, the algorithm will end up producing individuals that are almost identical to the wake-up word to achieve high wake-up rate. To prevent this, we formulate the fuzzy word generation as a multi-objective optimization problem, which aims to find fuzzy words that have both high wake-up rate and high dissimilarity distance from the real wake-up word. Instead of simply using weighted sum to combine the two objectives, we leverage the concept of Pareto frontier to select non-dominated individuals [12], which preserves as many words as possible to improve diversity of the next generation of descendants. We rank individuals in a non-increasing order according to their wake-up rate and dissimilarity distance respectively, and non-dominated individuals are maintained for reproduction. An individual x_1 is dominated by x_2 if for all objective functions $f_i(x),\ i = 1, ..., N$, we have

$$f_i(x_1) \leq f_i(x_2), \quad \forall i = 1, ..., N \tag{1}$$

If there is no individual dominating x_i, x_i is said to be non-dominated. In our problem, a word is non-dominated if there is no other word that has both higher wake-up rate and larger dissimilarity distance than the word.

(3) **Variation**. A new population is created by varying the survived individuals to maintain important pronunciation units and adjust other pronunciation units to find more fuzzy word candidates in the problem space. Commonly-used variation methods include crossover, recombination and mutation.

To customize the generation framework to different languages, there are two aspects that require specific design. First, we need to encode words by determining the number of variables representing each word and the range of each variable, thus an individual can be easily evaluated and transformed into a new individual that represents a valid word. For instance, how to encode "Alexa" or "xiǎo dù xiǎo dù" so that we can perform mutation or cross-over operations to generate new individuals that represent valid English or Chinese words? Secondly, we need to define the dissimilarity distance between two individuals. It is difficult to obtain the dissimilarity distance directly from the encoding, since the encoded vectors consist of numbers that do not carry the pronunciation information of the English or Chinese words. For example, "a", "e" and "f" may be encoded as "1", "5", and "6" respectively in the alphabetic order, but the dissimilarity distance between "e" and "a" are obviously smaller than that between "e" and "f".

English and Chinese voice assistants cover more than 85% of the global market [4]. In particular, Chinese smart speakers have occupied more than 51% market share in 2019 [34], but Chinese fuzzy words is less well studied. Mandarin Chinese and English are different as they belong to different language families. Chinese belongs to the Sino-Tibetan language family, while English belongs to the Indo-European language family. Unlike English, Chinese words are not made up with letters as in an alphabetic system, and the pronunciation of Chinese words cannot be inferred directly from the Chinese characters as Chinese is not a phonetic language. Moreover, Chinese and English vary greatly in pronunciation. Chinese is a tone language with four different tones, and pronouncing the same syllable in different tones indeed lead to different meanings. In contrast, English uses stress (rising or falling tones) to emphasize

or express emotions, without changing the meaning of a word. In summary, Chinese and English have different word formations, pronunciation units and pronunciation rules, and we need to customize the generation framework for the two languages in appropriate ways. In the following subsection, we first present the design for the Chinese language, which is less well studied, then we present the design for the English language.

3.2 Generating Chinese Fuzzy Words

A Chinese word is made up of Chinese characters, also known as sinogram or "hanzi". Chinese characters are very different from English letters. Each Chinese character is both the smallest meaning unit and the smallest pronunciation unit. The pronunciation of a Chinese character is represented by *pinyin*. The pinyin of a character consists of initials (e.g., x), finals (e.g., ai or iao) also known as vowels, and a tone. The pronunciation of a Chinese character is determined by its initial, final and tone. The pronunciation of a Chinese word is the combination of the pronunciation of each character. There is at most one initial in the pinyin of a character. Some characters do not have an initial, e.g., ài (爱). There are 23 initials in total [17, 44]. There are one to two finals in the pinyin of a character. Some finals can be combined together, e.g., i and ao form iao, but some finals can not, e.g., i and ou. By considering valid final combinations as special finals, we have a total of 37 finals [17, 44]. There are four tones in Chinese, denoted by a diacritical mark on the finals, i.e., ā, á, ǎ, à for tone 1, 2, 3, 4 respectively. The same initial-final combination with different tones have different meanings, e.g., xiǎo (小) and xiào (笑) mean "small" and "laugh" respectively. Some initial-final combinations are invalid (cannot be pronounced), e.g., xang, no matter what the tone is. Some initial-final-tone combinations have no corresponding Chinese characters, e.g., jīng (精) is valid but there is no Chinese character that pronounces as jíng (the second tone). Such invalid combinations need to be culled during the evolution in the genetic algorithm.

Encoding. Since the pinyin of a Chinese character normally comprises of the initial, the final and the tone, we use three variables to encode a character. The range of each variable is the number of possible initials/finals/tones, which are 24 (23 initials and zero-initial), 37 and 4 respectively. Without loss of generality, we use the lexicographic order of initials/finals/tones [44] as the value of the variable. As far as we know, all Chinese wake-up words are composed of four characters. Hence, we encode each individual as a 12-dimension vector.

Dissimilarity distance. We cannot use the difference between encodings of two words to represent their dissimilarity, since the lexicographic order of initials/finals/tones does not reflect their pronunciation resemblance, e.g., "a", "o" and "an" are encoded as 1, 2 and 8 respectively, but "a" pronounces more closely to "an" than to "o". To capture the phonetic similarity between initials and finals, we leverage the high-dimensional embedding [26]. Let $W_1 = [c_1^{(1)}, ..., c_1^{(n)}]$ and $W_2 = [c_2^{(1)}, ..., c_2^{(n)}]$ denote two Chinese words, where $c^{(i)}$ is the i-th character of a word. The dissimilarity distance is calculated as $\text{dist}(W_1, W_2) = \sum_i \text{dist}(c_1^{(i)}, c_2^{(i)})$, where $\text{dist}(c_1^{(i)}, c_2^{(i)})$ is the distance between two characters at the same offset. The distance calculated in this way increases with the number of characters, thus we normalize the distance to $[0, 1)$ using $\tanh(\cdot)$, a commonly-used sigmoidal function that normalizes the activation of neural networks [25].

$$\text{dist}(W_1, W_2) = \frac{1}{n} \sum_i \tanh(\text{dist}(c_1^{(i)}, c_2^{(i)})/A). \quad (2)$$

The distance is divided by constant A to attain a more evenly distribution. In our experiment, we set $A = 100$.

3.3 Generating English Fuzzy Words

An English word can be divided into graphemes (a letter or a letter combination) which correspond to different pronunciation units, i.e., phonemes. It is worth noting that people can pronounce words that they have never seen (e.g., foreign names) based on experience, and Text to Speech (TTS) engines can pronounce "non-dictionary words" [33], which increases the space of words for an attacker to generate fuzzy words.

Encoding. There are two special challenges facing the design for the English language. First, we need to decide whether to encode an English word based on its letter composition or phoneme composition. An intuitive thought is to encode an English word according to its phoneme composition. Nevertheless, we have found that this is not applicable due to several reasons. Firstly, it is not always possible to convert a combination of phonemes into a word in letter, which makes it difficult to produce meaningful fuzzy words. Secondly, existing TTS services cannot pronounce phoneme combinations as naturally as letter combinations. The pronunciation of phoneme combinations sounds mechanical and incoherent, and cannot even wake up the voice assistants by saying the phoneme combinations of their real wake-up words. Therefore, we choose to encode English words according to their letter compositions.

The second challenge is that English words have varied length. An English word can be as short as 1 letter and as long as 17-18 letters. Two English words with different lengths may sound similar, e.g., loose and lose. Moreover, a combination of two English words may sound like one English word, e.g., a lot and allot. Among the top three English voice assistants [28], Amazon uses one word as the wake-up word, while Google and Apple use two. To address this problem, for a voice assistant with a specific wake-up word, we first use one variable to represent one letter, and then insert spaces (" ") between letters as a place holder to increase the overall length of an individual. The length of an individual is set as r times that of the original wake-up word (with a length of n), where $r > 1$ is generally set to 1.5. In this way, we not only address the problem of encoding wake-up words that are composed of two words, but also increase the diversity of the generated fuzzy words. To sum up, we use an $r \times n$-dimension vector to represent an individual, where each variable represents a letter or a space, and the variable value ranges from 1 to 27.

Dissimilarity distance. Similar to the Chinese language, the encoding of English words also does not carry pronunciation information, thus we choose phonemes instead of letters to quantify dissimilarity distance. Two same-length English words may have different numbers of phonemes, e.g., animal and beauty. To tackle this difficulty, we use Levenshtein distance between the phoneme composition of two English words to calculate their dissimilarity distance. Let $W_1 = [p_1^{(1)}, ...p_1^{(m)}]$ and $W_2 = [p_2^{(1)}, ...p_2^{(n)}]$ denote

Table 1: Generated fuzzy words for Chinese and English smart speakers.

	Baidu	Xiaomi	AliGenie	Tencent	Amazon Echo	Echo Dot	Google	Apple Siri
Wake-up word	xiǎo dù xiǎo dù	xiǎo ài tóng xué	tiān māo jīng líng	jiǔ sì èr líng	Alexa	Alexa	Hey Google	Hey Siri
Total number	63	108	322	84	127	130	79	52
Mean dissimilarity	5.35%	6.26%	2.37%	2.66%	15.28%	15.29%	11.98%	9.92%
Top-ten examples wake-up rate	xiāo lǒng xiāo lǒng	qiāo bāi dōng hè	yān mēn jīng líng	jiōng niào èr líng	ilebser	ureqssr	heii googerl	hey sserea
	20%	90%	50%	30%	70%	10%	60%	80%
	piāo dòu piāo dòu	qiāng bāi dōng sè	yān māng jīng líng	jīn sì ào líng	ileqsur	arleqsr	heiigoogaa	heai ssuree
	10%	100%	100%	10%	60%	100%	30%	50%
	tiāo dōu tiāo dōu	qiāo bāi tōu shè	wán māng jīng líng	jiōng sì èr lián	ileqcer	ilekcer	heay gugal	hay scir e
	60%	90%	40%	10%	90%	100%	100%	60%
	tiāo dòng tiāo dòng	qiāo cāi dōng sè	wáng mào jīng lǐn	jiǒng shì èr lián	ilekcer	ilexcer	hey gooogov a	haiiasciree
	20%	100%	10%	20%	70%	10%	100%	60%
	shāo dōu shāo dōu	xiāo cāi dōng sè	yán māo jīng lǐng	jiǒng sì èr lián	ileqsar	ilexsur	heii googurl	heyisyree
	20%	100%	90%	100%	100%	100%	70%	20%
	jiāo dōu jiāo dōu	qiāo ē dū sè	yān māo jīng líng	jiǒng sì è rliáng	ilexsar	ileqsar	heii gugurl	heii sirea
	40%	100%	100%	30%	100%	100%	60%	70%
	qiáo dōu qiáo dōu	qiāo āi dū sè	wān māo jīng líng	jiǒng zì èr liáo	ilexsur	ileksar	hea gougll	haiy cire
	30%	100%	90%	90%	100%	100%	50%	80%
	qiāo dōu qiāo dōu	qiàng āi dōng sàng	yán māo jīng líng	jiōng sì èr mín	ileksur	ileksur	hei googll a	hey sirr e
	50%	100%	90%	10%	80%	100%	100%	100%
	qiāo dòng qiāo dòng	qiàng āi chōu lè	wán māo jīng líng	jiǒng sì èr mín	alexoer	alekcir	hei gooo r	hei suru r a
	20%	100%	20%	30%	60%	70%	20%	70%
	xiāo dōu xiāo dōu	qiāo ā dǒng sà	wáng māo jīng líng	jiǒng sì er liāo	ilexcer	ileqser	heiy googow l	hay syrrie e
	50%	100%	10%	30%	100%	100%	50%	100%

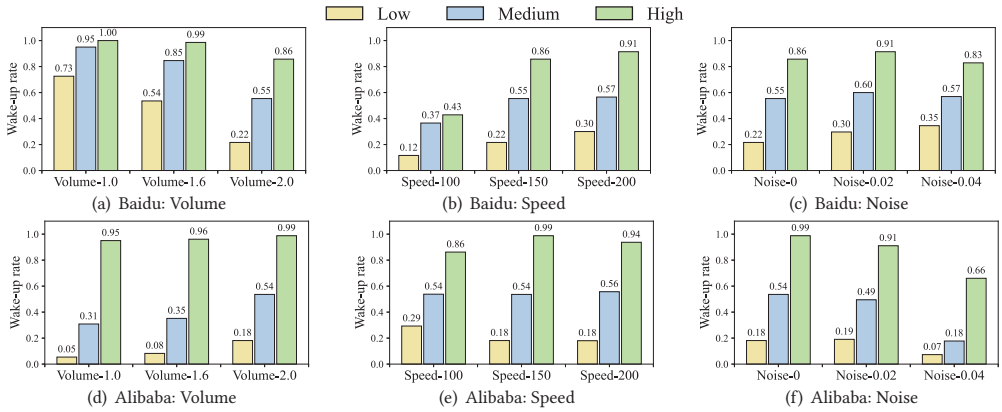


Figure 3: Wake-up rate of fuzzy words under different environments for Chinese voice assistants.

two English words, where $p^{(i)}$ is the i-th phoneme of a word. The conventional Levenshtein distance [31] assumes that the distance between any pair of different phonemes is 1, while some phonemes sound similar and some phonemes sound far apart. Therefore, we integrate phonetic dissimilarity of phonemes [30] into the Levenshtein distance to quantify the dissimilarity distance between two words. Let $\text{dist}(p^{(i)}, p^{(j)}) \in [0, 1]$ denote the dissimilarity of two phonemes. We have

$$\text{dist}(W_1, W_2) = \frac{D + I + 2 \sum\limits_{(i,j)\in\mathcal{S}} \text{dis}(p_1^{(i)}, p_2^{(j)})}{m + n}, \tag{3}$$

where D and I denote the number of deletions and insertions respectively, $\mathcal{S}$ is the set of substitutions which replaces phoneme $p_1^{(i)}$ with $p_2^{(j)}$, and m and n are the length of W_1 and W_2 respectively. The dissimilarity between space and any phoneme is set as 1.

3.4 Experiment Results

We conduct experiments to answer the following questions:

- **(Q1)** How does our proposed generation framework perform in producing fuzzy words for different Chinese and English smart speakers?
- **(Q2)** How do environmental factors influence the robustness of the generated fuzzy words?
- **(Q3)** Do the generated fuzzy words sound different from the real wake-up words from the human perspective?
- **(Q4)** How do the generated fuzzy words perform when articulated by real human individuals with different accents?
- **(Q5)** How do the generated fuzzy words perform when inserted in conversations?

We will answer these questions after presenting the experiment settings.

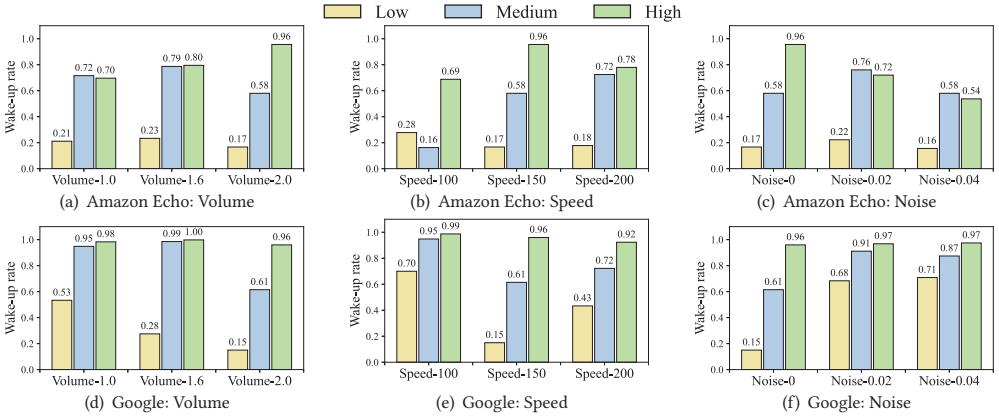


Figure 4: Wake-up rate of fuzzy words under different environments for English voice assistants.

Evaluated voice assistants. For English voice assistants, we conduct experiments on Amazon Echo, Amazon Echo Dot, Google Nest Mini and Apple HomePod. Amazon Echo and Amazon Echo Dot can be woken up by "Alexa", "Amazon", "Echo" or "Computer", and we focus on generating fuzzy words of "Alexa". Google Nest Mini can be woken up by "Hey Google" or "Ok Google", and we focus on generating fuzzy words of "Hey Google". Apple HomePod's wake-up word is "Hey Siri", the same as iPhone and iPad.

For Chinese voice assistants, we conduct experiments on Baidu, Xiaomi, AliGenie, and Tencent. Baidu smart speakers can be woken up by "xiǎo dù xiǎo dù" (小度小度), a repetition of the nickname of Baidu (百度). Xiaomi smart speakers can be triggered by "xiǎo ài tóng xué" (小爱同学), the Chinese name of Xiaomi's virtual assistant. AliGenie, also known as Tmall Genie, can be woken up by "tiān māo jīng líng" (天猫精灵), the Chinese name for the smart speaker. AliGenie can also be woken up by "nǐ hǎo tiān māo" (你好天猫), where "nǐ hǎo" means "hello" in Chinese. Tencent smart speakers can be activated by "jiǔ sì èr líng" (九四二零), which sounds similar to "I just love you" (就是爱你) in Chinese.

Experiment setup. As shown in Figure 10, our experiment setup consists of a laptop, a Raspberry Pi, a stereo and a light sensor. The laptop runs the generation algorithm to produce fuzzy words, and the stereo plays the audio of each generated fuzzy word to test its wake-up rate. The Raspberry Pi is equipped with a light sensor to detect whether the smart speaker is activated or not. The Raspberry Pi returns the wake-up rate of the fuzzy words to the generation algorithm to evaluate their fitness.

Our experiments are carried out in a quiet laboratory room. We employ pyttsx3 to generate the audio samples of fuzzy words by using TTS, which can articulate non-dictionary words [33]. The distance between the stereo and the tested smart speaker is 20 centimeters. We play the audio samples with the default male voice at a moderate volume. The play speed is set to 150 in pyttsx3 by default, which approximates the average speed of human speakers. Each word is played 10 times for wake-up rate computation. The crossover rate and the the mutation rate selected for the genetic algorithms are 1 and 0.1 respectively.

Performance of generation framework (Q1). We display the results of fuzzy word generation in Table 1. A full list of generated fuzzy words is in Section A.12. Note that we consider the fuzzy words generated by the genetic algorithm as out of distribution, since subjective tests show that users perceive the fuzzy words as different from the real wake-up word. Furthermore, we leverage the generated fuzzy words to strengthen the wake-up word detector, which improves its performance regarding both fuzzy words and non-fuzzy words.

For Chinese voice assistants, Baidu has the fewest fuzzy words while AliGenie has the most fuzzy words. This indicates that repetition in xiǎo dù xiǎo dù may indeed mitigate the *FakeWake* phenomena. Tencent also has a small number of fuzzy words since the wake-up word jiǔ sì èr líng contains rich combinations of initials, finals and tones. AliGenie has a significantly larger number of fuzzy words since its wake-up word detector relies heavily on *ian* to detect the wake-up word, which we will explain in Section 4. For English voice assistants, longer wake-up words, e.g., Hey Google and Hey Siri, have fewer fuzzy words. Also, Siri is a less commonly-used word with distinctive pronunciation, making it more difficult to produce its fuzzy words. In total, we have generated 965 fuzzy words, which not only reveals the severity of the *FakeWake* phenomena in voice assistants, but also lays the foundation for understanding and mitigating the *FakeWake* phenomena. By retraining the wake-up word detector using the generated fuzzy words, it can distinguish between fuzzy words and the real wake-up words more precisely (c.f. Section 5). We show the wake-up rate distribution of fuzzy words for different voice assistants in Figure 11 in the Appendix.

Environmental impact on fuzzy words (Q2). We investigate the impact of volume, speed, noise level and speaker gender on the wake-up rate of generated fuzzy words. For volume, we change the volume to 1.6 and 2 times of the original volume. For speed, the original speed parameter is 150, and we change it to 100 (slower) and 200 (faster). For noise level, we add Gaussian noise to simulate environmental noises. The parameter of the Gaussian noise is set to 0.02 and 0.04. At 0.02, the signal-to-noise ratio (SNR) is around 40db (office, library), and at 0.04, the SNR is around 30db (soft music, whisper) [15, 23]. For speaker gender, the original audio samples use a male voice, and we change the voice to female. As the TTS we use does not support Chinese female voice, we test the impact of speaker gender only on English voice assistants.

Insight 1

Fuzzy words with high wake-up rate mostly maintain their wake-up rate when the environment changes.

We demonstrate the experiment results of Chinese and English voice assistants in Figure 3 and Figure 4 respectively. Due to space limitations we only show the results of four mainstream voice assistants and put the rest of the results in Figure 13 in the appendix.

- *Volume.* With increased volume, the wake-up rate rises for most fuzzy words of most voice assistants. But for Baidu and Google, the wake-up rate decreases with a higher volume.
- *Speed.* Speed has a mixed influence on the wake-up rate. Generally, as speed increases, the wake-up rate first goes up then goes down, especially for the English voice assistants, e.g., Echo and Apple Siri. This may be because a slightly faster speed boosts coherence of the TTS, but a super fast speed makes the speech intelligible for the voice assistants.
- *Noise level.* In general, noises degrade the wake-up rate. But for Xiaomi, the wake-up rate for fuzzy words with medium wake-up rate grows to as high as 0.91 at a high noise level. Similar trend is also observed in Google.
- *Speaker gender.* After changing the speaker voice from male to female, the mean wake-up rate for fuzzy words with high wake-up rate decreases, while the mean wake-up rate for fuzzy words with medium and low wake-up rate increases. Due to page limitation, the results of the influence of speaker gender is in Section A.3.

Perceptual difference of fuzzy words (Q3). We conduct a subjective test to investigate whether the generated fuzzy words with high dissimilarity distance according to the generation algorithm indeed sound different from the real wake-up words to human ears.

We have recruited 33 volunteers from our campus, including 26 males and 7 females aged from 20 to 40. Among the 33 volunteers, 18 people are native Chinese speakers, and 15 people are English speakers. The Chinese native speakers come from different provinces of China with different dialects, and they all had English as a second language. The English speakers originate from different countries with English as their first, second or third language. We provide the volunteers with audio recordings of the real wake-up words and the fuzzy words to enable the volunteers to evaluate the dissimilarity between the fuzzy words and the real wake-up words. All volunteers have good or very good experience of using smart speakers. Before the user study, we have given the volunteers introduction to the *FakeWake* phenomena and given them instructions on the experiments.

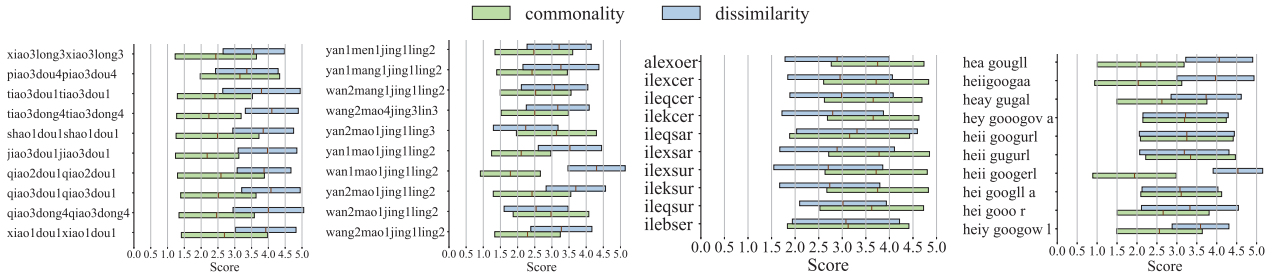
For each evaluated smart speaker, we choose the top 20 fuzzy words with the largest dissimilarity distance. We ask each volunteer to listen to the audio of each fuzzy word for 3 times and then evaluate the dissimilarity between the fuzzy word and the real wake-up word on a scale from 1 (very similar) to 5 (very dissimilar). We also ask each volunteer to score whether the fuzzy word is common in daily life on a scale from 1 (not common) to 5 (very common). We show the results of the top 10 fuzzy words in Figure 5 and Figure 14, and the results of the remaining fuzzy words are in Figure 15.

Insight 2

Fuzzy words with large perceptual difference from the real wake-up words can wake up the voice assistants.

As shown in Figure 5, for most voice assistants, the generated fuzzy words have an average perceptual difference of more than 3.0. Specifically, the fuzzy words for Xiaomi have high perceptual difference (more than 4.0), while the fuzzy words for Echo have relatively low perceptual difference (around 3.0). The fuzzy words with high perceptual difference tend to be regarded as less common in daily life. There is large variance for certain fuzzy words due to individual differences in hearing experiences. The user study confirms that the dissimilarity distance metric is able to quantify the perceptual differences between the fuzzy words and the real wake-up words. Therefore, it is confirmed that fuzzy words with large dissimilarity distance are also perceptually more distinctive from the real wake-up words. The fact that the fuzzy words with large perceptual difference from the real wake-up word can wake up the voice assistants shows the severity of the security and privacy threat posed by the *FakeWake* phenomena.

Real human audio of fuzzy words (Q4). In reality, different individuals may pronounce the same word in different ways (e.g., in different accents). Therefore, we conduct experiments with humans articulating the generated fuzzy words. We have recruited 30 Chinese native speakers, including 20 males and 10 females. The Chinese speakers come from 17 different provinces in China with different dialects. We have recruited 30 English speakers from Amazon Mechanical Turk, including 21 males and 9 females. The English speakers come from 6 countries, i.e., the United States, the United Kingdom, Germany, Italy, Brazil, and India, with English as their first or second language. The volunteers have different occupations and backgrounds in society. We randomly selected 20 fuzzy words for each voice assistant. Each Chinese/English speaker is instructed to articulate and record the audios of 20 fuzzy words for the 4 Chinese/English voice assistants. In total, we have $20 * 4 * 30 * 2 = 4,800$ audio samples spoken by humans. We play each audio sample 10 times to each voice assistant and calculate the mean wake-up rate of each fuzzy word across the 30 volunteers and display the cumulative distribution function (CDF) of the wake-up rate for each voice assistant in Figure 6. Although the differences between human pronunciation and TTS pronunciation lead to a decline in the wake-up rate, we can observe that almost all fuzzy words spoken by real humans have positive wake-up rates, which proves these fuzzy words also pose a threat in real life. For the Chinese voice assistants, 50% of the fuzzy words for Baidu, AliGenie, and Tencent have a wake-up rate of more than 0.5, except for Xiaomi which has lower wake-up rates. Based on our experiments the reason for this may lie in the fact that fuzzy words for Xiaomi are perceived as being less common in daily language (Figure 14(a)) such that the volunteers may have difficulty articulating these fuzzy words fluently. For English voice assistants, the fuzzy words for Amazon Echo have higher wake-up rates than those for Echo Dot, probably because the former has more high-quality microphones and thus is more susceptible to fuzzy words (Echo has a seven-microphone array, and the Echo Dot has a four-microphone array). The fuzzy words for Siri have the lowest wake-up rates since it also has the



(a) Baidu (b) AliGenie (c) Amazon Echo (d) Google

Figure 5: The results of subjective tests for the top-10 fuzzy words for Baidu, AliGenie, Amazon Echo, and Google. The perceptual dissimilarity between fuzzy words and the real wake-up words ranges from 1 (very similar) to 5 (very dissimilar). The commonality of a fuzzy word in daily life ranges from 1 (not common) to 5 (very common).

Table 2: Wake-up rate of fuzzy words in conversations. The columns *b.*, *m.*, and *e.* present the mean wake-up rate of the 10 fuzzy words when inserted at the beginning, the middle and the end of the conversations respectively.

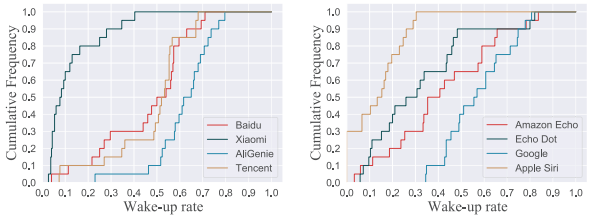
Chinese	b.	m.	e.	English	b.	m.	e.
Baidu	0.81	0.73	0.72	Amazon Echo	0.88	0.77	0.70
Xiaomi	0.77	0.66	0.74	Echo Dot	0.79	0.74	0.60
AliGenie	0.95	0.86	0.92	Google	0.90	0.81	0.71
Tencent	0.94	0.76	0.82	Apple Siri	0.76	0.50	0.61

fewest generated fuzzy words and their wake-up rates when spoken through TTS are originally lower. The full statistics of the mean wake-up rate of each fuzzy word by human audio are provided in Section A.7.

Performance of fuzzy words in conversations (Q5). Different from previous works that play TV shows or conversations to voice assistants to exhaustively search for fuzzy words, we automatically generate fuzzy words using a genetic algorithm and test them word by word. To evaluate whether the generated words can also wake up the voice assistants when used in conversations, we select 10 fuzzy words for each voice assistant and create 3 sentences for each fuzzy word. The fuzzy word appears at the beginning, the middle, and the end of the 3 sentences respectively. We play each sentence for 10 times to the voice assistant to compute the wake-up rate of the corresponding fuzzy word. The mean wake-up rate of the 10 words for different voice assistants is listed in Table 2. The results demonstrate that being embedded in conversations only slightly decrease the wake-up rate of the fuzzy words. Fuzzy words at the beginning of a conversation have the highest wake-up rates since other parts of the conversation have a smaller influence on the detection process. The fuzzy words of Siri have the lowest wake-up rates in conversations. This may be because the words are less commonly used in everyday conversations. The full statistics of the wake-up rate of each fuzzy word in conversations are listed in Section A.9.

4 Understanding Fuzzy Words

To understand the cause of the *FakeWake* phenomena, we aim to find the decisive factors that lead to false acceptance of fuzzy words.



(a) Chinese voice assistants (b) English voice assistants

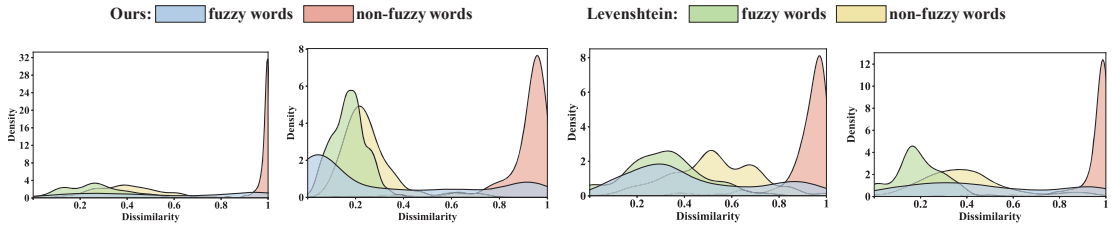
Figure 6: Cumulative distribution function of the wake-up rate of fuzzy words spoken by real humans.

In this section, we first present a black-box explanation of the fuzzy words of commercial voice assistants since we have no access to their wake-up word detector models. Then, we conduct a white-box explanation of the fuzzy words based on an open-source keyword spotting model.

4.1 Black-box Explanation

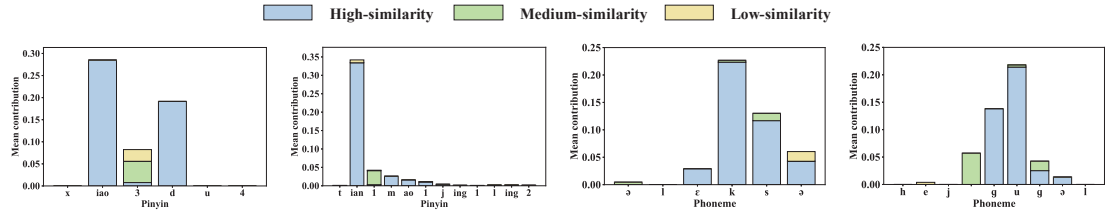
We have no access to the commercial wake-up word detector, e.g., Google or Baidu, which makes it impossible to analyze the causes of false acceptance of our generated fuzzy words. Therefore, we try to explain the *FakeWake* phenomena of commercial voice assistants by analyzing the generated fuzzy words themselves. A fuzzy word is considered as the wake-up word because the wake-up word detector is fooled by certain "similarities" between the two words. Measuring the similarities based on conventional metrics (e.g., calculating the Levenshtein distance between two words [37]) may not be appropriate since our generated fuzzy words have an expanded distribution of Levenshtein distance from the real wake-up words (as shown in Figure 7). Therefore, we need to dig deeper into the root causes of false acceptance of fuzzy words.

To address this challenge, we propose to build an interpretable tree-based classifier as a proxy to the inaccessible and inexplicable black-box wake-up word detector. Based on the classifier, we propose a dissimilarity score that can predict whether a word is likely to be accepted by the wake-up word detector or not. After that, we pinpoint phonetic features with the highest contributions to the classification results as decisive factors that have the most influence on the decision of wake-up word detectors.



(a) Baidu (b) AliGenie (c) Amazon Echo (d) Google

Figure 7: The distribution of dissimilarity between fuzzy-words/non-fuzzy-words and real wake-up words for Baidu, AliGenie, Amazon Echo, and Google. Both fuzzy words and non-fuzzy words have similar distribution of Levenshtein distance, but have distinctively different distributions of our proposed dissimilarity score.



(a) Baidu (b) AliGenie (c) Amazon Echo (d) Google

Figure 8: Black-box explanation for private wake-up word detectors of commercial voice assitants. The contributions of decisive factors at different positions of the real wake-up word for Baidu, AliGenie, Amazon Echo, and Google. The legend *High-similarity* refers to decisive factors that are phonetically similar to their counterparts in the real wake-up word. *Medium-similarity* and *Low-similarity* are defined accordingly.

Dissimilarity score. We train an interpretable model with the generated fuzzy words as positive samples and non-fuzzy words as negative samples, which simulates the behavior of the wake-up word detector. To extract features of each fuzzy word as the input to the model, we do not use the encoding of the genetic algorithm, since it consists only of numbers that do not carry pronunciation information. For Chinese, we leverage the high-dimensional embedding [26], which encodes each initial and each final into a two-dimensional vector. This embedding characterizes the phonetic features of a Chinese word. For English, since there is no high-dimensional embedding for phonemes, we use multi-dimensional scaling [22] to find an encoding that preserves the dissimilarity between phonemes [30]. We also encode each phoneme into a two-dimensional vector. In summary, each initial/final/phoneme is encoded into two features.

We use gradient boosting classifier [16], an ensemble of weak classifiers (we use decision tree as the weak classifier), as the interpretable model. We perform 10-fold cross validation, and the results show that the interpretable models have a prediction accuracy of more than 90% for most voice assistants. There are various reasons why the prediction accuracy may fluctuate between voice assistants. In particular, it is not possible to exhaustively search for fuzzy words. The training set naturally contains a subset of samples that belong to fuzzy words and non-fuzzy words, which is common for all machine learning tasks. Needless to say, the decision boundaries learned by the classifier is imperfect, and the prediction accuracy is unable to reach 100%. Interestingly, we observe that the prediction accuracy for voice assistants with more fuzzy words is lower. For instance, the prediction accuracy for Baidu is the highest (93.81%), while Baidu has the second smallest number of fuzzy words (63). The prediction accuracy for AliGenie is the lowest among Chinese voice assistants (87.09%), even though AliGenie has the largest number of fuzzy words (322). This indicates that with an increasing amount of fuzzy words it gets more difficult to distinguish the fuzzy words from the real wake-up word of this voice assistant. In addition, the tree-based classifier is relatively inaccurate. A possible remedy is to increase the number of tree-based classifiers in the gradient boosting classifier, but this will greatly increase the computational complexity without much improvement in the accuracy according to our experiments. The prediction accuracy, false positive and false negative rates of the models for different voice assistants are displayed in Table 6. We use the output confidence score of a certain fuzzy word as its similarity score γ. The dissimilarity score equals $1 - \gamma$.

Insight 3

Our constructed dissimilarity score can better separate fuzzy words and non-fuzzy words than Levenshtein distance.

Figure 7 displays the distribution of dissimilarity scores between fuzzy-words/non-fuzzy-words and real wake-up words. We can observe that fuzzy words and non-fuzzy words have similar expanded distributions in terms of Levenshtein distance, which indicates that Levenshtein distance is not a proper metric to predict whether a word is similar to the real wake-up word. In contrast, with our approach, non-fuzzy words have significantly higher dissimilarity scores than fuzzy words, which means that our constructed dissimilarity score can better predict whether a word is likely to be accepted by the wake-up word detector.

Decisive factors. In order to evaluate the contribution of each feature, we calculate their SHAP values [27]. We regard each feature as a "contributor", and add up contributions of all features to obtain the prediction result.

$$y_i = y_0 + \sum_j f(\alpha_i^{(j)}), \tag{4}$$

where y_i is the prediction result for sample x_i, y_0 is the mean prediction results of all samples, $\alpha_i^{(j)}$ is the j-th feature of x_i, and $f(\alpha_i^{(j)})$ is the contribution of $\alpha_i^{(j)}$ to the prediction result.

Note that the contribution $f(\alpha_i^{(j)})$ can be positive or negative. A positive contribution means that the feature is helpful for accurate prediction of whether a sample is a fuzzy word or not. For a certain fuzzy word x_i, let $\mathcal{A}_i^+$ denote the set of features with positive contributions. We rank the elements in $\mathcal{A}_i^+$ according to their contributions in a non-increasing order. We construct a set $\mathcal{D}_i$ to represent important features for x_i, and then add elements in $\mathcal{A}_i^+$ to $\mathcal{D}_i$ sequentially until the ratio of the contribution of all elements in $\mathcal{D}_i$ to the contribution of all elements in $\mathcal{A}_i^+$ is more than a threshold β.

$$\frac{\sum_{k\in\mathcal{D}_i} f(\alpha_i^{(k)})}{\sum_{k'\in\mathcal{A}_i^+} f(\alpha_i^{(k')})} \geq \beta. \tag{5}$$

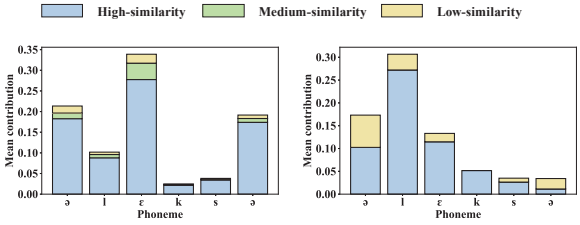
We consider a(n) initial/final/phoneme as a decisive factor for x_i if at least one of its two features is in $\mathcal{D}_i$. In this way, we can obtain the set of decisive factors and their contributions. The contribution of a(n) initial/final/phoneme is the sum of contributions of its features in $\mathcal{D}_i$.

We construct the set of decisive factors for each fuzzy word that is correctly classified by the interpretable model. We compare the decisive factors and their counterparts in the original wake-up word by computing the absolute differences of their encoding. We obtain mean and variance δ of all computed differences for normalization. According to the similarity measurement, we categorize the decisive factors into three groups: high-similarity (the normalized difference is within δ), medium-similarity (the normalized difference is within 2δ), and low-similarity (the normalized difference is beyond 2δ). After that, we compute the average contribution of each group of decisive factors at each position of the original wake-up word, as shown in Figure 8. The threshold β is set as 0.8.

Insight 4

Wake-up words with fewer decisive factors have more fuzzy words.

Figure 8 demonstrates that the decisive factors which determine false acceptance of fuzzy words usually concentrate on a short snippet of the wake-up word, e.g., *ks* for Alexa and *ài* for xiǎo ài tóng xué. By keeping the decisive factors and alter other parts of the wake-up word it is possible to create new fuzzy words. If a wake-up word has fewer decisive factors, there is more space for altering other parts of the word to generate fuzzy words, which renders the voice assistants more vulnerable. For instance, AliGenie relies disproportionately on *ian* to recognize the wake-up word, thus the number of its fuzzy words is the highest among all voice assistants. The fact that more than 80% of the fuzzy words contain



(a) Alexa (White box) (b) Alexa (Black box)

Figure 9: White-box explanation for open-source wake-up word detectors. *High-similarity*, *Medium-similarity*, and *Low-similarity* are defined the same as in Figure 8.

at least one of the top-3 decisive factors (Table 9) indicates that the wake-up word detector may put disproportionate emphasis on the decisive factors and not enough attention to other parts of the phonemes to identify the wake-up words. Developers can apply our interpretation framework to test the decisive factors of their wake-up words, then fine-tune the wake-up word detector to expand the decisive factors across the entire wake-up word, i.e., pay attention to all parts of the words. In this way, the wake-up word detector will be more robust to the *FakeWake* phenomena.

4.2 White-box Explanation

To explain the *FakeWake* phenomena and the existence of fuzzy words at the AI model level, we adopt a lightweight, open-source keyword-spotting model, named Mycroft Precise [19]. The system works as follows. The input of the model is an audio file, e.g., .wav, which contains the semantic information, i.e., phoneme or pinyin. The audio file is not directly fed to the neural network, but mel-frequency cepstral coefficients (MFCCs) are extracted from the audio file as features. MFCCs capture the frequency sensitivity of the human auditory system. The model processes the MFCCs of the audio through a series of complicated nonlinear operations in a convolutional neural network, and outputs the final results.

Deep learning models are notorious for its non-interpretable nature due to its highly nonlinear structures and massive parameters. Instead of getting entangled in the complicated model structures and parameters, we resort to the gradient information, which reflects the sensitivity of the model output to the input and is used by mainstream approaches for machine learning model interpretation. The greater the gradient of a certain feature is, the more significant is the impact of the feature on the prediction results of the model. However, the gradient may disappear, especially if the activation function is the ReLU function ($\mathrm{ReLU}(x) = \max(x, 0)$). Moreover, the gradient only characterizes the local features of the input but cannot reflect the entire decision-making process of the model. To tackle these shortcomings, we leverage the integral gradient method [41], which improves the traditional gradient method and satisfies the axioms of sensitivity and implementation invariance.

Similar to Equation (4), the integral gradient method attributes the output of the model to the sum of the contributions of each feature, but the contributions are estimated in a different way as

$$g(x_i^{(j)}) = \left[\frac{1}{n}\sum_{k=1}^{n}\left(\nabla_\gamma y_i(\gamma(\alpha))\big|_{\gamma(\alpha)=(1-\alpha)x_i+\alpha\bar{x},\alpha=\frac{k}{n}}\right)\right]_j [\bar{x}-x_i]_j$$

Table 3: Mitigation results

Wake-up word	Evaluation metric	Original model	Strengthened model
Alexa	False positive rate	0.00%	0.00%
	False negative rate	6.85%	5.48%
	Detection accuracy	96.73%	97.39%
	Fuzzy rate	**17.67%**	**1.18%**
Hi Xiaowen	False Positive rate	0.00%	0.00%
	False negative rate	2.13%	2.13%
	Detection accuracy	99.36%	99.36%
	Fuzzy rate	**11.65%**	**0.20%**
Hi Mia	False Positive rate	0.00%	0.00%
	False negative rate	0.00%	0.00%
	Detection accuracy	100%	100%
	Fuzzy rate	**5.62%**	**0.00%**

where $g(x_i^{(j)})$ is the contribution of the j-th feature of sample x_i to the prediction result y_i, and $y_0 = y(\bar{x})$ is the output of the model to the mean value of all training samples. The contribution of each feature is calculated as the integral gradient from $\bar{x}$ to x_i. We can regard $g(x_i^{(j)})$ as the contribution of the j-th feature to transforming a trivial decision y_0 to a positive decision y_i by exerting a greater gradient to the model decision-making process.

After obtaining $g(x_i^{(j)})$, we use the same approach as Equation (5) to derive the decision factors by replacing $f(x_i^{(j)})$ with $g(x_i^{(j)})$. The only difference is that the decision factors here are frames of the audio file of the fuzzy words rather than the pinyin or the phoneme composition of the word.

To visualize the decisive factors in a more intelligible way, we sum the contributions of the frames that belong to the same pinyin or phoneme. Note that we do not have access to the training datasets of the wake-up word detectors of commercial voice assistants, so we use public datasets to train our local model. We obtain a white-box wake-up word detector for "Alexa". Unfortunately, we did not find public datasets of other voice assistants.

We present the decisive factors of these three models in Figure 9(a). The results confirm that the wake-up detector pays disproportionate attention to some parts of the word, making it vulnerable to fuzzy words that contain these decisive parts. Note that the decisive factors for Alexa derived by the white-box explanation framework in Figure 9(a) seem to be different from those derived by the black-box explanation framework in Figure 8(c). The former shows that the model pays more attention to *əlɛ*, while the latter shows that the model pays more attention to *ks*. This is because our open-source wake-up word detector is different from the actual wake-up word detector of Amazon Echo. We apply our black-box explanation framework to the open-source wake-up word detector of Alexa and present the results in Figure 9(b).

Summary. We can observe that the black-box explanation framework yields similar decisive factors as the white-box explanation framework for the same wake-up word detector. This verifies that our black-box explanation framework is effective in extracting decisive factors of unknown wake-up word detectors of commercial voice assistants.

5 Mitigating Fuzzy Words

In this section, we present two potential approaches to strengthen wake-up word detectors against fuzzy words. The first approach is to scrutinize words that contain decisive factors. We quantify the proportion of generated fuzzy words that contain decisive factors, as shown in Table 9. We can observe that for most voice assistants, more than 90% of fuzzy words contain at least one of the top-3 decisive factors. In particular, 100% of the fuzzy words of Google and Baidu contain top-3 decisive factors. The wake-up word detector can send audio samples that involve decisive factors to more discreet examination by complicated speech recognition models.

The second remedy is to strengthen wake-up word detectors by retraining them with the generated fuzzy words. Since we have no access to wake-up word detectors of commercial voice assistants, we test our idea on Mycroft Precise [19]. We also experiment with TC-ResNet [9], another open-source keyword-spotting model, but the trained model has poor generalization ability on our datasets. The prediction accuracy on the test dataset is as low as 85.64%. For a specific wake-up word, we collect three types of datasets: a conventional dataset, a fuzzy word dataset, and a collective dataset. The conventional dataset consists of samples of the wake-up word (positive) and non-fuzzy words (negative). We divide the conventional dataset into a training dataset and a test dataset. To begin with, we train an original wake-up word detector using the conventional training dataset on Precise. The fuzzy word dataset consists of the generated fuzzy words of the wake-up word. We use the fuzzy word dataset to retrain the original model to obtain the strengthened model. The collective dataset consists of a large number of words from a dictionary. There is no overlap between the collective dataset and the conventional dataset, and there is no overlap between the collective dataset and the fuzzy word dataset. The words in the collective datasets include fuzzy words and non-fuzzy words.

We test the original and the strengthened models on the conventional test dataset to obtain the false positive rate, false negative rate, and accuracy. We further test the original model and the strengthened model on the collective dataset to measure the fuzzy rate, which is defined as the ratio of wrongly accepted words among all words in the collective dataset. A larger fuzzy rate means that more fuzzy words in the dictionary are accepted by detectors indicating that the model is more vulnerable to fuzzy words.

Since we do not have access to the training datasets of the wake-up word detectors of commercial voice assistants, we use public datasets as the conventional dataset to train wake-up word detectors. We obtain five public datasets of English words, including "Alexa", "Athena", "Computer", "Hi Xiaowen" and "Hi Mia". Unfortunately, we did not find public datasets of Chinese words. We split each conventional dataset into a training dataset with 3/4 of the samples and a test dataset with the remaining samples. We keep a relatively balanced ratio of negative samples to positive samples in the training dataset to avoid bias. For Alexa's training dataset, we have 296 positive samples and 399 negative samples. The positive samples are collected from kaggle [1] and the negative samples are collected from the human noise and the environment noise in the dataset of mycroft-precise [19]. For Athena's training dataset, we have 386 positive samples and 839 negative samples. For Computer's training dataset, we have 87 positive samples and

161 negative samples. We collect the data of Athena and Computer from mycroft-precise [19]. For Hi Xiaowen's training dataset, we have 380 positive samples and 693 negative samples, all collected from MobvoiHotwords [24]. For Hi Mia's training dataset, we have 800 positive samples and 1,000 negative samples, all collected from AISHELL [35]. For the fuzzy word dataset, we search for fuzzy words using a dictionary of 498 words from the Google TTS library [21]. The collective dataset is also from the Google TTS library with 49,822 words.

As shown in Table 3 and Table 10, the fuzzy rate of the original model can be as high as more than 20%, but drops to almost 0 after being strengthened. Surprisingly, for all strengthened models, their accuracy is even higher than the original model on the conventional test dataset. This shows that our mitigation approach not only defends against fuzzy words but also improves the performance of the original model. The possible reason is that the fuzzy words are close to the decision boundary, and using fuzzy words to train the detector provides a more effective way to learn the decision boundary more precisely. Even if an adversary re-applies the genetic algorithm to generate fuzzy words for the strengthened wake-up word detector, the discovered fuzzy words will naturally be fewer and their wake-up rate will be lower. This is because the strengthened wake-up detector has learned a better decision boundary between the wake-up words and non-wake-up words, and will reject almost all previously generated fuzzy words (c.f. Table 3, the fuzzy rate is reduced to under 0.25%). Therefore, our mitigation strategy can greatly reduce the number of fuzzy words and the attack success rate of the adversary.

The proposed two mitigation strategies help manufacturers improve their products in terms of security and privacy. It may be possible but inconvenient to address the *FakeWake* problem at the user side. A possible solution is to equip users with a local device that can detect the possible fuzzy words and inform users of the potential danger of privacy leakage through smartphone alerts. This solution, however, depending entirely on the user to mitigate shortcomings of a product, generates cost and places the burden of managing an additional device on him. The manufacturer should not shift the burden of improving the device to the user.

6 Related Work

Wake-up word detection. Earlier works leverage Hidden Markov Models (HMM) and Gaussian Mixture Models (GMM) for acoustic modeling [18, 48], which suffers from high computational complexity. Hence, the traditional HMM-GMM models are now replaced by more efficient Deep Neural Networks (DNN) [32, 40], Convolutional Neural Networks (CNN) [8, 36], Recurrent Neural Networks (RNN) [14], Convolutional Recurrent Neural Networks (RCNN) [2], Gated Recurrent Units (GRUs) [43], and Long Short-Term Memory (LSTM) units [3].

Wake-up word security. As far as we know, there is only a few works on wake-up word security. Schönherr et al. [37] investigated *accidental triggers* for English, Chinese and German smart speakers. Accidental triggers were exhaustively searched by playing a large corpus of media audios to smart speakers, following which a dictionary is used to find more accidental triggers based on Levenshtein distance. The Chinese speakers are tested with English audio samples, and the influence of distance, volume, speed and ambient noises are not considered. Similarly, Dubois et al. [13] played UK and US television shows to voice assistants to characterize the sources and the number of mis-activations. Mitev et al. [29] utilized a phoneme dictionary to fuzz voice assistants and scanned network traffic for mis-activated devices. All existing methods seek for fuzzy words by exhaustive search, which takes days or even weeks to find a limited set of fuzzy words. Our approach automatically generates a large set of fuzzy words without manual intervention in a much shorter time. In addition, we propose a more precise metric than the Levenshtein distance to predict fuzzy words. Finally, we are the only one presenting a practical mitigation method to strengthen wake-up detectors.

Covert commands. Covert command attacks aim to inject malicious commands for the voice assistant to execute certain operations without users noticing. Vaidya et al. [42] proposed to modify the MFCCs of voice samples (e.g., OK Google) so that the mangled samples are unintelligible to humans (e.g., electronic-sounding noise) but will be identified as a command by voice assistants. Carlini et al. [6] built on top of this work by presenting a more precise attack on a white-box voice recognition system. Carlini et al. [7] created an audio sample from another one containing a spoken sentence, preserving similar waveforms but misleading voice recognition algorithms to yield false interpretations. Schönherr et al. [38] and Yuan et al. [46] proposed to insert intended commands inside an innocent audio sample, e.g., a music file, which is only recognizable by voice recognition algorithms but not by humans. Zhang et al. [47] used ultrasonic audio to inaudibly inject commands into voice assistants.

Our work is different from audio adversarial examples since fuzzy words are natural audio samples instead of synthetic ones including carefully-crafted noise. Compared with music [38, 46] or electronic-sounding noise [6, 42], fuzzy words are more covert with less alarm. Our work is different from dolphin attacks [47] as the fuzzy words are audible. We do not intend to activate voice assistants using out-of-band voice signals. The frequency range of our fuzzy words lies within the hearing range of human ears.

7 Conclusion

In this paper, we present a systematic framework to generate, understand and mitigate fuzzy words that can falsely activate voice assistants. Using our search framework customized for different languages, we have managed to pinpoint 965 fuzzy words covering 8 most popular English and Chinese smart speakers. Our explanation of decisive factors and mitigation methods help strengthen wake-up word detectors against fuzzy words. As such, our approach presents a promising way to find, understand and mitigate privacy and security issues in voice assistants.

8 Acknowledgement

We thank all anonymous reviewers for their insightful comments on this paper. This work is funded in part by China NSFC Grant 61925109 and 61941120, as well as the German Research Foundation (DFG) within CRC 1119 CROSSING (S2 and P3) and the Intel Private AI Center.

References

[1] Amir Anhari. 2021. Alexa Dataset: Build voice-first applications. https://www.kaggle.com/aanhari/alexa-dataset

[2] Sercan O Arik, Markus Kliegl, Rewon Child, Joel Hestness, Andrew Gibiansky, Chris Fougner, Ryan Prenger, and Adam Coates. 2017. Convolutional recurrent neural networks for small-footprint keyword spotting. *arXiv preprint arXiv:1703.05390* (2017).

[3] Pallavi Baljekar, Jill Fain Lehman, and Rita Singh. 2014. Online word-spotting in continuous speech with recurrent neural networks. In *IEEE Spoken Language Technology Workshop*.

[4] BusinessWire. 2020. Strategy Analytics: Global Smart Speaker Sales Cross 150 Million Units for 2020 Following Robust Q4 Demand. https://smallurl.net/businesswire

[5] Canalys. 2020. Global smart speaker market 2021 forecast. https://www.canalys.com/newsroom/canalys-global-smart-speaker-market-2021-forecast

[6] Nicholas Carlini, Pratyush Mishra, Tavish Vaidya, Yuankai Zhang, Micah Sherr, Clay Shields, David Wagner, and Wenchao Zhou. 2016. Hidden voice commands. In *25th USENIX Security Symposium*.

[7] Nicholas Carlini and David Wagner. 2018. Audio adversarial examples: Targeted attacks on speech-to-text. In *IEEE Security and Privacy Workshops*.

[8] Guoguo Chen, Carolina Parada, and Georg Heigold. 2014. Small-footprint keyword spotting using deep neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing*.

[9] Seungwoo Choi, Seokjun Seo, Beomjun Shin, Hyeongmin Byun, Martin Kersner, Beomsu Kim, Dongyoung Kim, and Sungjoo Ha. 2019. Temporal convolution for real-time keyword spotting on mobile devices. *arXiv preprint arXiv:1904.03814* (2019).

[10] CNET. 2018. Alexa sent private audio to a random contact, Portland family says. https://www.cnet.com/home/smart-home/alexa-sent-private-audio-to-a-random-contact-portland-family-says/

[11] Toby Cox. 2020. Siri and Alexa Fails: Frustrations With Voice Search. https://themanifest.com/digital-marketing/resources/siri-alexa-fails-frustrations-with-voice-search

[12] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* 6, 2 (2002), 182–197.

[13] Daniel J. Dubois, Roman Kolcun, Anna Maria Mandalari, Muhammad Talha Paracha, David Choffnes, and Hamed Haddadi. 2020. When Speakers Are All Ears: Characterizing Misactivations of IoT Smart Speakers. *Proceedings on Privacy Enhancing Technologies* (2020).

[14] Santiago Fernández, Alex Graves, and Jürgen Schmidhuber. 2007. An application of recurrent neural networks to discriminative keyword spotting. In *International Conference on Artificial Neural Networks*.

[15] Center for Hearing and Communication. 2020. Common environmental noise levels. https://chchearing.org/noise/common-environmental-noise-levels/

[16] Jerome H. Friedman. 2000. Greedy Function Approximation: A Gradient Boosting Machine. *Annals of Statistics* (2000).

[17] FutureLearn. 2021. Introduction to Pinyin. https://www.futurelearn.com/info/courses/chinese-pronunciation-tone/0/steps/64892.

[18] Alvin Garcia and Herbert Gish. 2006. Keyword spotting of arbitrary words using minimal speech resources. In *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, Vol. 1. IEEE, I–I.

[19] Kris Gesling. 2021. Precise. https://mycroft-ai.gitbook.io/docs/mycroft-technologies/precise.

[20] Iman Ghosh. 2020. Ranked: The 100 Most Spoken Languages Around the World. https://www.visualcapitalist.com/100-most-spoken-languages/

[21] Google. 2018. Google Speech. https://pypi.org/project/google-speech/

[22] J. C. GOWER. 1966. Some distance properties of latent root and vector methods used in multivariate analysis. *Biometrika* 53, 3-4 (1966), 325–338.

[23] HealthLinkBC. 2020. Harmful Noise Levels. https://www.healthlinkbc.ca/health-topics/tf4173

[24] Jingyong Hou, Yangyang Shi, Mari Ostendorf, Mei-Yuh Hwang, and Lei Xie. 2019. Region Proposal Network Based Small-Footprint Keyword Spotting. *IEEE Signal Processing Letters* 26, 10 (2019), 1471–1475.

[25] B. L. Kalman and S. C. Kwasny. 1992. Why tanh: choosing a sigmoidal function. In *IEEE International Joint Conference on Neural Networks*, Vol. 4. 578–581.

[26] Min Li, Marina Danilevsky, Sara Noeman, and Yunyao Li. 2018. DIMSIM: An Accurate Chinese Phonetic Similarity Algorithm Based on Learned High Dimensional Encoding. In *22nd Conference on Computational Natural Language Learning*.

[27] Scott M. Lundberg, Gabriel Erion, Hugh Chen, Alex DeGrave, Jordan M. Prutkin, Bala Nair, Ronit Katz, Jonathan Himmelfarb, Nisha Bansal, and Su-In Lee. 2020. From local explanations to global understanding with explainable AI for trees. *Nature machine intelligence* 2, 1 (2020), 56–67.

[28] Markets and Markets. 2020. Smart Speaker Market with COVID-19 Impact Analysis by IVA (Alexa, Google Assistant, Siri, DuerOS, Ali Genie), Component (Hardware (Speaker Driver, Connectivity IC, Processor, Audio IC, Memory, Power IC, Microphone) and Software), Application, and Region - Global Forecast to 2025. https://www.marketsandmarkets.com/Market-Reports/smart-speaker-market-44984088.html

[29] Richard Mitev, Anna Pazii, Markus Miettinen, William Enck, and Ahmad-Reza Sadeghi. 2020. LeakyPick: IoT Audio Spy Detector. In *Annual Computer Security Applications Conference*.

[30] David R. Mortensen, Patrick Littell, Akash Bharadwaj, Kartik Goyal, Chris Dyer, and Lori Levin. 2016. PanPhon: A Resource for Mapping IPA Segments to Articulatory Feature Vectors. In *26th International Conference on Computational Linguistics: Technical Papers*.

[31] Gonzalo Navarro. 2001. A guided tour to approximate string matching. *Comput. Surveys* 33, 1 (2001), 31–88.

[32] Sankaran Panchapagesan, Ming Sun, Aparna Khare, Spyros Matsoukas, Arindam Mandal, Björn Hoffmeister, and Shiv Vitaladevuni. 2016. Multi-task learning and weighted cross-entropy for DNN-based keyword spotting. In *Interspeech*.

[33] Jongseok Park, Kyubyong Kim. 2019. g2pE: A Simple Python Module for English Grapheme To Phoneme Conversion. https://github.com/Kyubyong/g2p.

[34] Sarah Perez. 2019. China overtakes US in smart speaker market share. shorturl.at/bBGOW

[35] Xiaoyi Qin, Hui Bu, and Ming Li. 2020. Hi-mia: A far-field text-dependent speaker verification database and the baselines. In *IEEE International Conference on Acoustics, Speech and Signal Processing*.

[36] Tara N Sainath and Carolina Parada. 2015. Convolutional neural networks for small-footprint keyword spotting. In *Sixteenth Annual Conference of the International Speech Communication Association*.

[37] Lea Schönherr, Maximilian Golla, Thorsten Eisenhofer, Jan Wiele, Dorothea Kolossa, and Thorsten Holz. 2020. Unacceptable, where is my privacy? Exploring Accidental Triggers of Smart Speakers. *arXiv preprint arXiv:2008.00508* (2020).

[38] Lea Schönherr, Katharina Kohls, Steffen Zeiler, Thorsten Holz, and Dorothea Kolossa. 2018. Adversarial Attacks Against Automatic Speech Recognition Systems via Psychoacoustic Hiding. *arXiv preprint arXiv:1808.05665* (2018).

[39] Eric Hal Schwartz. 2020. Voice Assistants Very Prone to Accidentally Waking Up and Recording Long Audio Clips: Study. shorturl.at/bdCEY

[40] Ming Sun, David Snyder, Yixin Gao, Varun K Nagaraja, Mike Rodehorst, Sankaran Panchapagesan, Nikko Strom, Spyros Matsoukas, and Shiv Vitaladevuni. 2017. Compressed Time Delay Neural Network for Small-Footprint Keyword Spotting.. In *Interspeech*.

[41] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *International Conference on Machine Learning*. PMLR, 3319–3328.

[42] Tavish Vaidya, Yuankai Zhang, Micah Sherr, and Clay Shields. 2015. Cocaine noodles: exploiting the gap between human and machine speech recognition. In *9th USENIX Workshop on Offensive Technologies*.

[43] Martin Woellmer, Bjoern Schuller, and Gerhard Rigoll. 2013. Keyword spotting exploiting long short-term memory. *Speech Communication* 55, 2 (2013), 252–265.

[44] Yellowbridge. 2021. Learn Chinese Pinyin Rules: Initials, Finals, and Tones. https://www.yellowbridge.com/chinese/pinyin-rules.php.

[45] Honggang Yu, Kaichen Yang, Teng Zhang, Yun-Yun Tsai, Tsung-Yi Ho, and Yier Jin. 2020. Cloudleak: Large-scale deep learning models stealing through adversarial examples. In *Network and Distributed Systems Security Symposium*.

[46] Xuejing Yuan, Yuxuan Chen, Yue Zhao, Yunhui Long, Xiaokang Liu, Kai Chen, Shengzhi Zhang, Heqing Huang, Xiaofeng Wang, and Carl A Gunter. 2018. Commandersong: A systematic approach for practical adversarial voice recognition. In *27th USENIX Security Symposium*.

[47] Guoming Zhang, Chen Yan, Xiaoyu Ji, Tianchen Zhang, Taimin Zhang, and Wenyuan Xu. 2017. Dolphinattack: Inaudible voice commands. In *ACM Conference on Computer and Communications Security*.

[48] Yaodong Zhang and James R Glass. 2009. Unsupervised spoken keyword spotting via segmental DTW on Gaussian posteriorgrams. In *2009 IEEE Workshop on Automatic Speech Recognition & Understanding*. IEEE, 398–403.

A Appendix

A.1 Experiment Setup

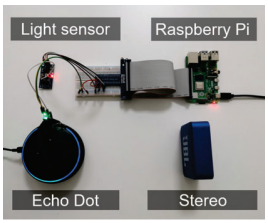


Figure 10: The experiment setup includes a laptop, a stereo, a Raspberry Pi, and a light sensor. The laptop runs the generation algorithm, and the stereo plays the generated fuzzy words over the air to the smart speaker. The Raspberry Pi equipped with the light sensor detects the activation of the smart speaker, which is fed back to the laptop to calculate the wake-up rate.

A.2 Wake-up Rate Distribution

We show the wake-up rate distribution of fuzzy words for different voice assistants in Figure 11, where we divide the wake-up rate into three ranges: low (0.1~0.3), medium (0.4~0.7), and high (0.8~1.0). More than 40% fuzzy words have high wake-up rate except for Baidu, which may be another benefit of word repetition.

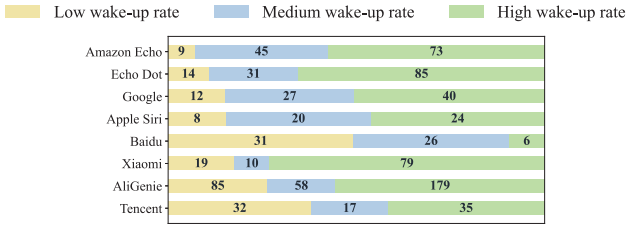


Figure 11: Distribution of fuzzy words for different voice assistants. Wake-up rate ranges: low (0-0.3), medium (0.4-0.7) and high (0.8-1.0). The numbers of fuzzy words are marked.

A.3 Impact of Speaker Gender

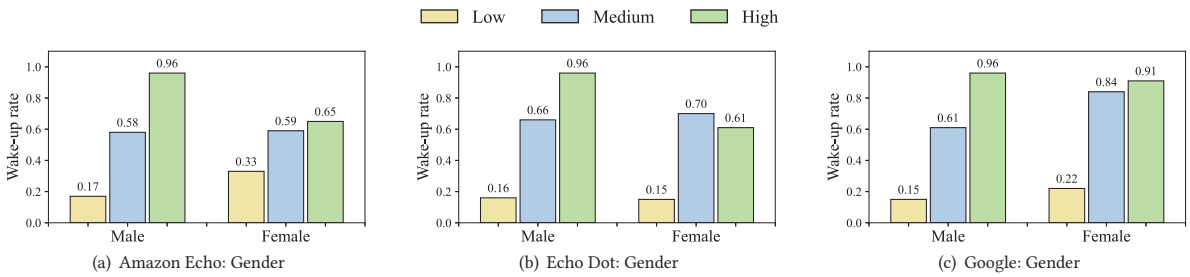


Figure 12: Wake-up rate of fuzzy words under different speaker genders for English voice assistants.

A.4 Environmental Impact on Fuzzy Words

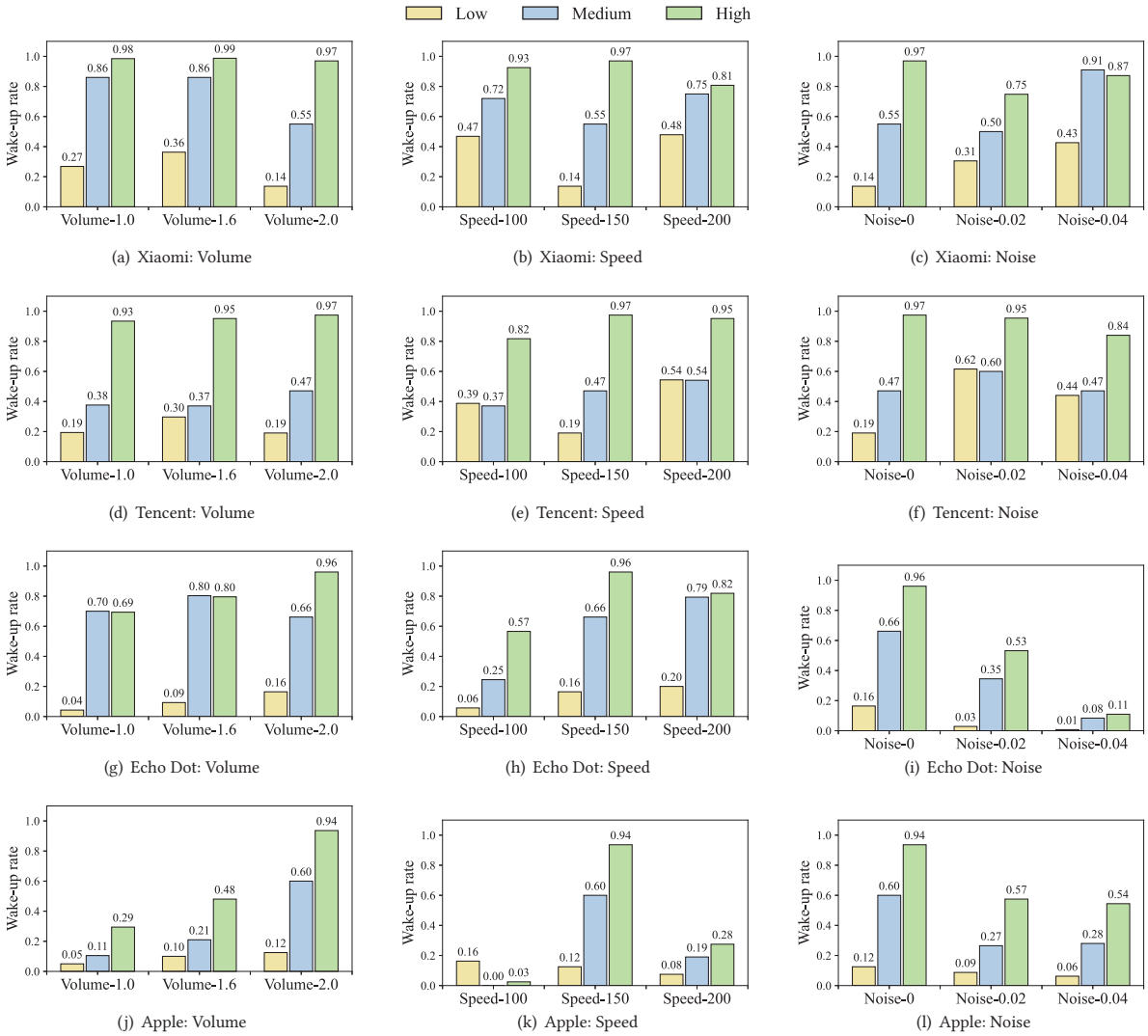
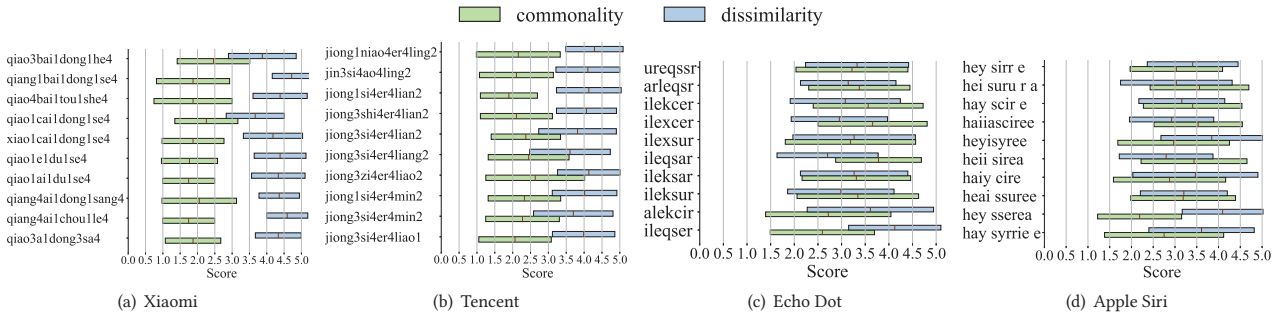


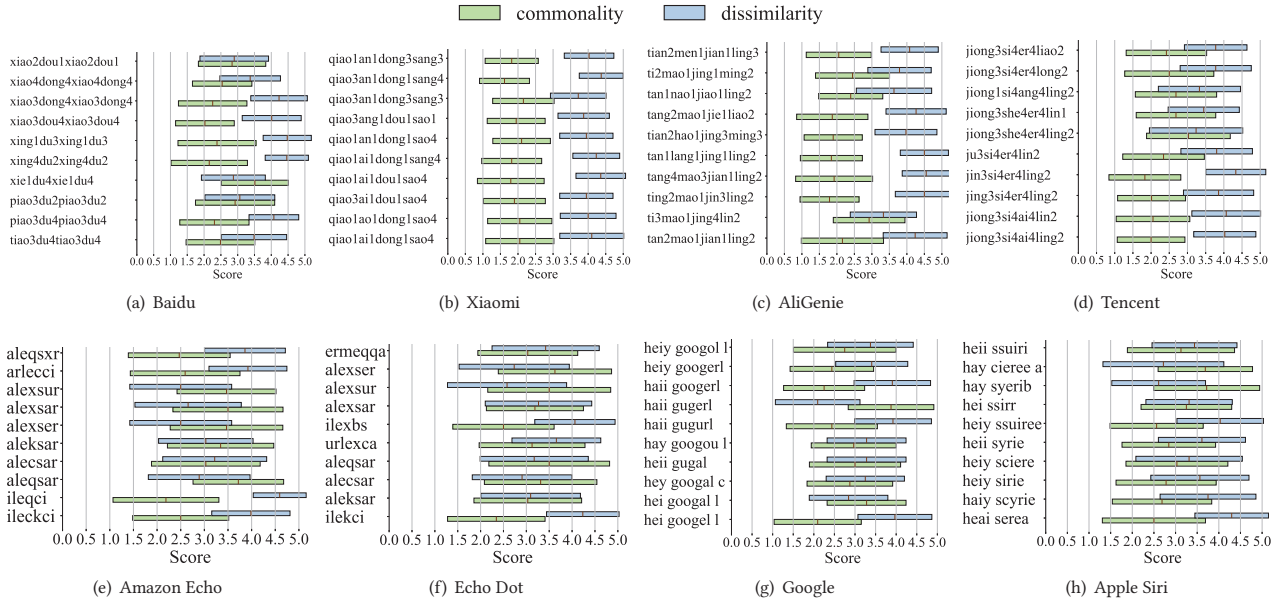
Figure 13: Wake-up rate of fuzzy words under different environments for Xiaomi, Tencent, Echo Dot, and Apple Siri voice assistants.

A.5 Subjective Tests



(a) Xiaomi (b) Tencent (c) Echo Dot (d) Apple Siri

Figure 14: The results of subjective tests for the top-10 fuzzy words for Xiaomi, Tencent, Echo Dot, and Apple Siri. The perceptual dissimilarity between fuzzy words and the real wake-up words ranges from 1 (very similar) to 5 (very dissimilar). The commonality of a fuzzy word in daily life ranges from 1 (not common) to 5 (very common).

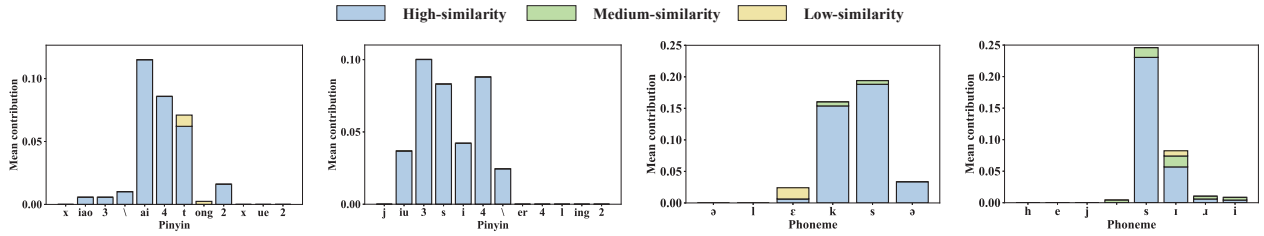


(a) Baidu (b) Xiaomi (c) AliGenie (d) Tencent

(e) Amazon Echo (f) Echo Dot (g) Google (h) Apple Siri

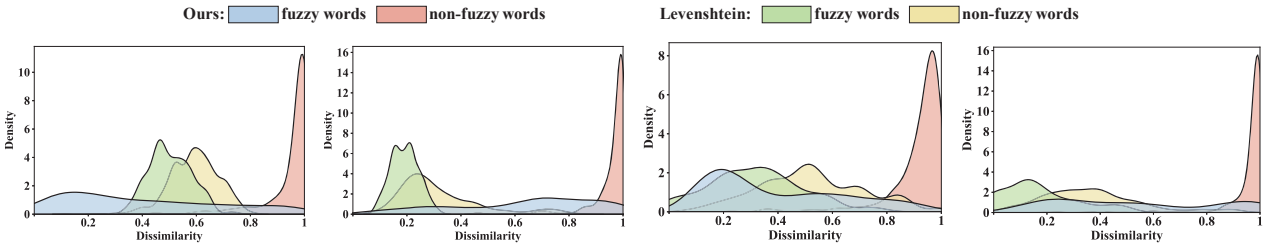
Figure 15: The results of subjective tests for the top-11~20 fuzzy words. The perceptual dissimilarity between fuzzy words and the real wake-up words ranges from 1 (very similar) to 5 (very dissimilar). The commonality of a fuzzy word in daily life ranges from 1 (not common) to 5 (very common).

A.6 Understanding Fuzzy Word



(a) Xiaomi (b) Tencent (c) Echo Dot (d) Apple Siri

Figure 16: Black-box explanation for private wake-up word detectors of commercial voice assitants. The contributions of decisive factors at different positions of the real wake-up word for Xiaomi, Tencent, Echo Dot, and Apple Siri. The legend *High-similarity* refers to decisive factors that are phonetically similar to their counterparts in the real wake-up word. *Medium-similarity* and *Low-similarity* are defined accordingly.



(a) Xiaomi (b) Tencent (c) Echo Dot (d) Apple Siri

Figure 17: The distribution of dissimilarity between fuzzy-words/non-fuzzy-words and real wake-up words for Xiaomi, Tencent, Echo Dot, and Apple Siri. Both fuzzy words and non-fuzzy words have similar distribution of Levenshtein distance, but have distinctively different distributions of our proposed dissimilarity score.

A.7 Real Human Audio of Fuzzy Words

Table 4: Wake-up rate of fuzzy words spoken by real humans for Chinese voice assistants. The column *w.rate* presents the mean wake-up rate of a fuzzy word across 30 volunteers. The highest and the lowest wake-up rates are highlighted in bold.

Baidu	w.rate	Xiaomi	w.rate	AliGenie	w.rate	Tencent	w. rate
xiāo dù xiāo dù	0.69	qiāo ān chōu sè	0.12	tián móu qīng líng	0.65	jiāo sì èr līng	0.67
xiào dù xiào dù	0.57	qiāo ài dōng qíng	0.13	tiān māo jīng lǐng	0.68	jiāo sì èr lín	**0.68**
xiǎo dū xiǎo dū	0.63	qiàng āi chōu lè	0.04	tián māo jǐng níng	0.66	jiāo sī ér líng	0.67
xiáo dū xiáo dū	0.57	qiào āi dōng sè	0.25	tiǎn māo qín lǐn	0.61	jiǒng sì èr líng	0.49
xiāo dú xiāo dú	0.48	piāo āi dōng rè	0.28	tián móu jìng líng	0.60	jiāo sī ér líng	0.57
xiōng dù xiōng dù	0.11	qiāo āi dōu sè	0.16	tiān mào jǐn lìn	0.54	jiāo sì èr lìn	0.54
shāo dù shāo dù	0.46	qiāo ān dōng sè	0.04	tián máo jǐng lǐng	0.66	jiǒng sì èr lín	0.53
xiáo dù xiáo dù	**0.71**	qiào ā dōng sà	0.06	diān máo jīng lǐng	**0.80**	jiǒng sì èr líng	0.55
shāo dú shāo dú	0.30	qiāng ān dōu sè	0.05	diǎn māo jìng níng	0.69	jiǒng sì èr lǐn	0.56
piāo dù piāo dù	0.56	qiāo ān dōng sī	0.04	tiān mào jìng lìn	0.58	jiǒng sī èr líng	0.49
xiōng dū xiōng dū	0.22	piāo ān chōng sè	0.09	tián māo qìng lǐn	0.71	jiǒng sī èr lìng	0.51
xiāo dū xiāo dū	0.44	qiào ān dōu shè	0.05	diān māo jīn lǐn	0.77	jiǒng sì ér líng	0.55
xiòng dǔ xiòng dǔ	**0.04**	qiào āi dōng gè	0.10	diǎn máo jīn lín	0.74	jiǒng sè èr líng	0.36
tiāo dù tiāo dù	0.56	qiào ān dōu lè	**0.03**	diǎn máo jīng lǐng	0.72	jiǒng sì èr liáo	0.35
xiāo dòng xiāo dòng	0.60	qiàng āi dōng sàng	0.08	tián mào jīng lǐng	0.58	jiǒng sì èr lǐng	0.52
xiáo dū xiáo dū	0.57	qiào āi dōu sè	0.35	diǎn māo qí níng	**0.23**	jiǒng sì èr lǐng	0.55
xiāo dú xiāo dú	0.54	qiǎng āi gōng lè	0.04	diǎn māo jǐng níng	0.62	jiǒng sì ěr líng	0.51
tiāo dōu tiāo dōu	0.25	qiāo ān gōng tè	0.04	tiān máo jìng lèng	0.52	jiǒng zì èr líng	0.08
xiāo dōu xiāo dōu	0.50	piāo āi dōu lè	0.08	tiàn móu jìn líng	0.52	jiǒng shì ěr líng	**0.07**
qiāo dōu qiāo dōu	0.27	qiāo āi dōng sè	**0.40**	tiān miāo jìng lǐng	0.46	jiǒng sì ěr lǐng	0.27

Table 5: Wake-up rate of fuzzy words spoken by real humans for English voice assistants.

Amazon Echo	w.rate	Echo Dot	w.rate	Google	w.rate	Apple Siri	w. rate
ilexsar	0.35	ileqsar	0.26	heii gugal	0.56	hey sirr e	0.17
ilexca	0.19	urlexca	0.10	hei goooar	0.51	hay sere e	0.20
ilekhsa	0.66	ileqci	0.10	heiy googourl	**0.34**	hey scirih	**0.30**
ileksca	0.06	erleksa	0.32	hei googala	0.78	hay syrrie e	0.10
allexsa	0.65	ileccsa	0.31	hey googal c	**0.81**	hay syrieqe	0.18
ilekssa	0.59	alexsua	0.11	heii googal	0.75	hay cieri	**0.00**
arleksa	0.57	aqlecsa	0.06	haii gugal	0.57	hai sirie	0.00
alebsa	**0.84**	alelksa	0.80	hey googau	0.64	hay sciria	0.23
alecqsa	0.78	ilekssa	0.48	haii googal	0.49	hey sori	0.00
ilexcer	0.33	blebssa	0.46	hey googav	0.61	hay surie	0.07
ilexsur	0.24	aleqpsa	**0.82**	heay gugal	0.43	hey sierie	0.26
urlecsa	0.47	hleqsaa	0.19	hei googal l	0.43	hay psyrria	0.16
ilexqa	**0.03**	alexci	**0.07**	heii gugull	0.35	hay scirih	0.29
alexsib	0.41	arleqsr	0.44	hei googal a	0.71	heiy suree	0.15
ilekcsa	0.59	ileksar	0.15	heiy gugourl	0.43	hei ssoree	0.25
ilexssa	0.11	alexser	0.21	he googal	0.65	hai sciri	0.00
ileqsaa	0.25	ilefksa	0.20	heiy googall	0.49	heai serea	0.00
dpecssa	0.34	urleqsa	0.34	hay gugal	0.74	hay ssirib	0.13
ilexsa	0.43	ilexca	0.43	hey gooogov a	0.46	hey ssire	0.00
alexssa	0.35	arlesca	0.47	hei googll a	0.61	hey sserea	0.07

A.8 Accuracy of Interpretable Model

Table 6: Prediction accuracy of the interpretable model in differentiating fuzzy words and non-fuzzy words. The columns *AUC*, *FPR* and *FNR* present the area under curve, false positive rate and false negative rate of the interpretable model.

voice assistant	accuracy	AUC	FPR	FNR	voice assistant	accuracy	AUC	FPR	FNR
Baidu	93.81%	92.62%	4.19%	22.22%	Amazon Echo	87.65%	89.03%	6.17%	38.89%
Xiaomi	91.27%	94.52%	4.17%	29.10%	Echo Dot	84.28%	87.39%	8.92%	44.31%
AliGenie	87.09%	92.74%	11.15%	15.66%	Google	92.35%	91.41%	3.49%	44.33%
Tencent	90.74%	92.31%	6.36%	39.82%	Apple Siri	77.18%	55.43%	20.63%	59.38%

A.9 Performance of Fuzzy Words in Conversations

Table 7: Wake-up rate of fuzzy words spoken in real conversations for Chinese voice assistants. The column *b.rate*, *m.rate*, and *e.rate* present the wake-up rates of the 10 fuzzy words when inserted at the beginning, the middle and the end of the conversations, respectively.

Baidu	b.rate	m.rate	e.rate	Xiaomi	b.rate	m.rate	e.rate	AliGenie	b.rate	m.rate	e.rate	Tencent	b.rate	m.rate	e.rate
xiāo dù xiāo dù	0.9	0.7	0.8	qiāo ān chōu sè	1.0	0.5	0.9	tián móu qīng líng	1.0	0.9	0.8	jiāo sì èr līng	0.9	0.4	0.6
xiào dù xiào dù	0.8	0.9	1.0	qiāo ài dōng qíng	1.0	0.6	0.8	tiǎn māo jīng líng	1.0	1.0	1.0	jiāo sì èr lín	1.0	0.8	1.0
xiāo dū xiāo dū	1.0	0.7	0.6	qiàng āi chōu lè	0.8	0.8	0.7	tián māo jīng níng	1.0	1.0	1.0	jiāo sī ér líng	1.0	0.6	0.6
xiáo dū xiáo dū	1.0	1.0	0.8	qiào āi dōng sè	0.6	0.5	0.7	tiǎn māo qín līn	1.0	0.8	0.8	jiǒng sī èr líng	0.9	1.0	1.0
xiāo dú xiāo dú	0.8	0.7	1.0	piāo āi dōng rè	0.7	0.8	1.0	tián móu jìng líng	0.9	0.6	1.0	jiāo sī ér líng	0.8	0.6	0.7
xiōng dù xiōng dù	1.0	0.9	0.6	qiāo āi dōu sè	0.9	0.7	0.6	tiǎn mào jīn lìn	1.0	1.0	1.0	jiāo sì èr lìn	1.0	1.0	0.8
shāo dù shāo dù	0.8	0.7	0.9	qiāo ān dōng sè	0.7	1.0	0.8	tián máo jǐng lǐng	1.0	1.0	0.9	jiǒng sì èr lín	0.8	0.7	1.0
xiáo dù xiáo dù	0.5	0.4	0.6	qiào ā dōng sà	0.7	0.6	0.9	diān máo jīng lǐng	1.0	0.6	0.7	jiǒng sì èr líng	1.0	0.6	0.8
shāo dú shāo dú	0.7	0.6	0.9	qiāng ān dōu sè	0.6	0.7	1.0	diān māo jìng níng	0.8	1.0	1.0	jiǒng sì èr līn	1.0	0.9	0.7
piāo dù piāo dù	0.6	0.7	0.6	qiāo ān dōng sī	0.7	0.4	0.3	tiǎn māo jìng lìn	0.8	0.7	1.0	jiǒng sī èr líng	1.0	1.0	1.0

Table 8: Wake-up rate of fuzzy words spoken in real conversations for English voice assistants.

Amazon Echo	b.rate	m.rate	e.rate	Echo Dot	b.rate	m.rate	e.rate	Google	b.rate	m.rate	e.rate	Apple Siri	b.rate	m.rate	e.rate
ilexsar	1.0	0.7	0.8	ileqsar	0.8	0.6	0.5	heii gugal	1.0	0.6	0.7	hey sirr e	0.7	0.8	0.5
ilexca	1.0	0.7	0.6	urlexca	0.7	0.5	1.0	hei goooar	1.0	1.0	1.0	hay sere e	1.0	0.9	0.7
ilekhsa	0.8	0.9	0.8	ileqci	0.8	0.8	0.6	heiy googourl	1.0	0.4	0.5	hey scirih	0.6	0.3	0.4
ileksca	0.8	0.5	1.0	erleksa	0.9	0.6	0.4	hei googala	1.0	1.0	0.8	hay syrrie e	0.7	0.4	0.8
allexsa	1.0	0.8	1.0	ileccsa	0.6	1.0	1.0	hey googal c	0.7	0.9	0.5	hay syrieqe	1.0	0.5	1.0
ilekssa	0.6	0.8	0.9	alexsua	0.8	0.5	0.4	heii googal	0.9	1.0	1.0	hay cieri	0.6	0.2	0.3
arleksa	1.0	0.9	0.6	aqlecsa	0.7	1.0	0.8	haii gugal	1.0	1.0	0.7	hai sirie	0.5	0.1	0.4
alebsa	0.8	1.0	0.5	alelksa	0.6	0.8	0.7	hey googau	1.0	0.9	1.0	hay sciria	1.0	0.5	1.0
alecqsa	1.0	0.6	0.8	ilekssa	1.0	0.9	0.6	haii googal	0.8	1.0	0.9	hey sori	0.6	0.7	1.0
ilexcer	0.8	0.8	0.7	blebssa	1.0	0.7	0.5	hey googav	0.6	0.3	0.5	hay surie	0.9	0.6	0.8

A.10 Decisive Factors

Table 9: Fraction of fuzzy words containing decisive factors. The column *top-n* presents the proportion of fuzzy words that comprise at least one of the top-n decisive factors with the highest contribution.

	top-1	top-2	top-3		top-1	top-2	top-3
Echo	85.61%	89.90%	90.71%	Baidu	68.75%	84.38%	100.00%
Echo Dot	83.74%	88.99%	88.93%	Xiaomi	65.74%	95.37%	95.37%
Google	90.72%	90.72%	100.00%	AliGenie	62.73%	97.52%	99.69%
Apple	92.19%	92.19%	96.88%	Tencent	72.62%	72.62%	97.62%

A.11 Mitigating Fuzzy Words

Table 10: Mitigation results of Computer and Athena

Wake-up word	Evaluation metric	Original model	Strengthened model	Wake-up word	Evaluation metric	Original model	Strengthened model
Computer	False Positive rate	0.00%	0.00%	Athena	False Positive rate	1.37%	0.88%
	False negative rate	5.00%	0.00%		False negative rate	1.04%	1.04%
	Detection accuracy	97.96%	100%		Detection accuracy	98.73%	99.07%
	Fuzzy rate	**10.04%**	**0.02%**		**Fuzzy rate**	**17.67%**	**1.18%**

A.12 Full List of Fuzzy Words

Table 11: Fuzzy words of Baidu. The column *dis.* presents the dissimilarity distance according to the generation algorithm, rounded up to the second decimal point. The column *w. rate* presents the wake-up rate. Note that dissimilarity distances smaller than 0.005 are rounded up to 0.00.

	dis.	w. rate		dis.	w. rate		dis.	w. rate		dis.	w. rate		dis.	w. rate
xiāo lóng xiāo lòng	0.30	0.2	piāo dòu piāo dòu	0.22	0.1	tiāo dōu tiāo dōu	0.19	0.6	tiāo dòng tiāo dòng	0.19	0.2	shāo dōu shāo dōu	0.16	0.2
jiāo dōu jiāo dōu	0.15	0.4	qiāo dōu qiāo dōu	0.15	0.3	qiāo dōu qiāo dōu	0.15	0.5	qiāo dòng qiāo dòng	0.15	0.2	xiāo dōu xiāo dōu	0.14	0.5
xiāo dōu xiāo dōu	0.14	0.5	xiāo dòng xiāo dòng	0.14	0.2	xiāo dòng xiāo dòng	0.14	0.7	xiāo dòu xiāo dòu	0.14	0.4	xīng dǔ xīng dǔ	0.13	0.3
xìng dú xìng dú	0.13	0.2	xiē dù xiē dù	0.08	0.1	piāo dú piāo dú	0.08	0.3	piāo dù piāo dù	0.08	0.7	tiāo dù tiāo dù	0.05	0.7
xiōng dù xiōng dù	0.04	0.7	xiōng dù xiōng dù	0.04	0.8	xiòng dù xiòng dù	0.04	0.7	qiāo bù qiāo bù	0.03	0.1	xiāo bù xiāo bù	0.03	0.3
xiāo lù xiāo lù	0.03	0.3	xiāo lù xiāo lù	0.03	0.4	xiāo nù xiāo nù	0.02	0.2	shāo dū shāo dū	0.01	0.4	shāo dú shāo dú	0.01	0.7
shāo dù shāo dù	0.01	0.7	xiá dù xiá dù	0.01	0.3	xiá dù xiá dù	0.01	0.2	xiá dù xiá dù	0.01	0.5	xiàn dú xiàn dú	0.01	0.1
xiàng dù xiàng dú	0.01	0.2	xiáng dǔ xiáng dǔ	0.01	0.1	xiáng dù xiáng dù	0.01	0.2	xiàn dù xiàn dù	0.01	0.3	xiàng dū xiàng dū	0.01	0.3
xiāng dù xiāng dù	0.01	0.5	jiāo dù jiāo dù	0.01	0.2	qiāo dú qiāo dú	0.01	0.1	qiāo dù qiāo dù	0.01	0.2	qiáo dǔ qiáo dǔ	0.01	0.1
qiáo dù qiáo dù	0.01	0.5	qiāo dū qiāo dū	0.01	0.3	qiāo dú qiāo dú	0.01	0.3	qiāo dù qiāo dù	0.01	0.4	xiāo dū xiāo dū	0.00	0.3
xiáo dū xiáo dū	0.00	0.8	xiāo dú xiāo dú	0.00	0.8	xiāo dū xiāo dū	0.00	0.8	xiāo dù xiāo dù	0.00	0.7	xiāo dú xiāo dú	0.00	0.5
xiáo dú xiáo dú	0.00	0.4	xiāo dù xiāo dù	0.00	0.9	xiāo dǔ xiāo dǔ	0.00	0.6	xiāo dú xiāo dú	0.00	0.6	xiāo dǔ xiāo dǔ	0.00	0.4
xiāo dù xiāo dù	0.00	0.9	xiáo dù xiáo dù	0.00	0.7	xiāo dū xiāo dū	0.00	0.3	-	-	-	-	-	-

Table 12: Fuzzy words of Xiaomi. The column *dis.* presents the dissimilarity distance according to the generation algorithm, rounded up to the second decimal point. The column *w. rate* presents the wake-up rate. Note that dissimilarity distances smaller than 0.005 are rounded up to 0.00.

	dis.	w. rate		dis.	w. rate		dis.	w. rate		dis.	w. rate		dis.	w. rate
qiāo bāi dōng hè	0.33	0.9	qiāng bāi dōng sè	0.29	1	qiāo bāi tōu shè	0.27	0.9	qiāo cāi dōng sè	0.27	1	xiāo cāi dōng sè	0.27	1
qiāo ē dū sè	0.19	1	qiāo āi dū sè	0.15	1	qiāng āi dōng sàng	0.11	1	qiāng āi chōu lè	0.11	1	qiāo ā dòng sà	0.11	1
qiāo ān dōng sàng	0.11	0.1	qiāo ān dōng sàng	0.11	1	qiāo ān dōng sàng	0.11	0.6	qiāo àng dōu sāo	0.10	0.9	qiāo ān dōng sào	0.10	0.9
qiāo āi dōng sàng	0.10	1	qiāo āi dōu sào	0.10	1	qiāo āi dōu sào	0.10	1	qiāo ào dōng sào	0.09	0.1	qiāo āi dōng sào	0.09	1
qiāng āi chōu sè	0.09	1	piāo ān chōng sè	0.08	1	piāo āi dōu lè	0.08	1	qiāo ān gōng tè	0.08	1	qiāo ān chōu sè	0.08	1
qiāo ān chōu sè	0.08	1	qiā āi duàn jiě	0.08	0.1	qiāo āi dōng gè	0.08	1	piāo āi dōng rè	0.08	1	qiāo āi dōu tè	0.07	0.8
qiāo āi gòng jìn	0.07	0.1	qiāo ān dōng sī	0.06	1	piāo ān dōng sè	0.06	1	qiāng āi gōng lè	0.06	1	piāo āi dōu sè	0.06	0.2
qiāo āi dòng jìn	0.06	0.1	qiáo āi dòng jìn	0.06	0.1	qiāo āi dōng jìn	0.06	0.9	qiāo āi tōu qìng	0.06	0.8	qiāo ēn dōng sè	0.06	1
qiāo āi gòu rè	0.06	1	piāo āi gòng sè	0.06	1	qiāo ài dōng qíng	0.06	1	qiāo ān dōu lè	0.06	1	qiāo ān dōu lè	0.06	0.8
qiāo ā dōu lè	0.06	1	qiāo ān gōng nè	0.06	0.1	piāo āi dōng shè	0.06	0.8	piāo āi dòng sè	0.05	1	piāo āi dòng sè	0.05	0.8
piāo āi dōng sè	0.05	1	qiāo ān dōng nè	0.05	0.7	qiāo ān dōng rè	0.05	0.7	qiāng ān dōu sè	0.05	1	qiāo ān dòng rè	0.05	0.6
qiāo ān chōng sè	0.05	0.4	qiāo āi gòng rè	0.05	0.9	qiāng ān dòng sè	0.04	0.6	qiāo āi dòng rè	0.04	1	tiāo ài dòng shè	0.04	0.1
qiāo āi chōng sè	0.04	1	qiāo āi chōng sè	0.04	1	xiāo āi dòng rè	0.04	0.3	xiāo āi cōng sè	0.04	1	qiāo ān dōu shè	0.04	1
qiāo ān dōu shè	0.04	0.9	qiāo ān dōu sè	0.04	0.1	qiāo ān dōu sè	0.04	0.9	qiāo ā dōu sè	0.04	1	qiāng āi dōng sè	0.04	0.9
xiāo āi cōng jiè	0.03	1	qiāo āi gòu jiè	0.03	0.1	qiāo ān dōng sè	0.03	1	qiāo ā dōng sè	0.03	0.9	qiāo ān dōng sè	0.03	0.5
qiāo ān dōng sè	0.03	1	qiāo āi dōu shè	0.03	1	qiāo ài dōu shè	0.03	0.8	qiāo āi dōu sè	0.03	1	qiāo āi dōu sè	0.03	0.9
qiāo āi dōu sè	0.03	1	xiāo ān dōng sè	0.03	1	xiāo ā dòng sè	0.03	1	qiāo āi gōng sè	0.02	1	qiāo āo dōng shè	0.02	1
qiāo āi dōu jiè	0.02	0.2	qiāo āi tōu sè	0.02	1	qiāo āi dōng shè	0.02	1	qiāo ài dōng shè	0.02	0.3	qiāo āi gōng jiè	0.02	0.9
qiāo āi dòng sè	0.02	0.5	qiāo āi gòng jiě	0.02	0.4	qiāo āi dòng sè	0.02	1	qiāo āi dōng sè	0.02	1	qiāo āi dōng sè	0.02	1
qiāo āi dòng sè	0.02	1	qiāo āi dòng shèng	0.02	1	xiāo āi gòng jiè	0.02	0.1	xiāo āi gòng jiè	0.02	0.1	xiāo āi dòng sè	0.02	1
xiāo āi dòng sè	0.02	1	qiāo āi dòng jiè	0.02	0.1	qiāo āi dòng jiè	0.02	0.5	qiāo āi dōng jiè	0.02	1	qiāo āi dòng jiè	0.02	0.1
xiāo āi dòng jiè	0.01	0.2	xiāo āi dòng jiè	0.01	1	xiāo āi dōng jiè	0.01	1	-	-	-	-	-	-

Table 13: Fuzzy words of Tencent. The column *dis.* presents the dissimilarity distance according to the generation algorithm, rounded up to the second decimal point. The column *w. rate* presents the wake-up rate.

	dis.	w. rate		dis.	w. rate		dis.	w. rate		dis.	w. rate		dis.	w. rate
jiōng niǎo èr líng	0.15	0.3	jīn sì ào líng	0.09	0.1	jiōng sì èr lián	0.08	0.1	jiōng shì èr lián	0.08	0.2	jiōng sì èr lián	0.08	1
jiōng sì èr liáng	0.08	0.3	jiōng zì èr liáo	0.07	0.9	jiōng sì èr mín	0.07	0.1	jiōng sì èr mín	0.07	0.3	jiōng sì èr liāo	0.07	0.3
jiōng sì èr liáo	0.07	1	jiōng sì èr lóng	0.05	0.2	jiōng sì àng líng	0.05	0.1	jiōng shè èr lín	0.05	0.8	jiōng shè èr líng	0.05	0.1
jù sì èr lín	0.05	0.1	jīn sì èr líng	0.05	0.2	jīng sì èr líng	0.05	0.3	jiōng sì ài lín	0.05	1	jiōng sì ài líng	0.05	1
jiōng sì èr míng	0.04	0.7	jiōng tì èr líng	0.03	0.1	jiōng sè èr lín	0.03	1	jiōng sè èr líng	0.03	0.5	jiōng sè èr lìng	0.03	1
jiōng sè èr líng	0.03	1	jiāng sì èr líng	0.02	0.5	jiē sì èr líng	0.02	0.4	jiāng sì èr líng	0.02	0.1	jiē sì èr líng	0.02	0.4
jiāo shī èr lín	0.02	0.4	jiāo sì èr níng	0.02	0.9	jiāo sī èr lín	0.02	0.4	jiāo sì èr lín	0.02	1	jiāo sī èr lín	0.02	0.1
jiāo sì èr lín	0.02	0.2	jiāo sì èr lín	0.02	1	jiāo sī ér líng	0.02	1	jiāo sì èr lìng	0.02	0.1	jiāo sī èr lìng	0.02	0.7
jiāo sī ér líng	0.02	1	jiāo sì èr lìng	0.02	0.2	jiāo sì èr lìng	0.02	0.9	jiāo sī èr lìng	0.02	0.8	jiāo sì èr lìng	0.02	1
jiāo sī èr líng	0.02	0.5	jiāo sì èr lìng	0.02	0.9	jiāo sī èr líng	0.02	0.2	jiōng zì èr líng	0.01	1	jiōng qì èr líng	0.01	0.1
jiōng sì èr lí	0.01	1	jiōng sì èn líng	0.01	0.3	jiōng sì èr níng	0.01	0.1	jiōng sī èr níng	0.01	0.2	jiōng sì è líng	0.01	0.4
jiōng sì èr níng	0.01	0.3	jiōng shì èr lín	0.01	0.5	jiōng sì èr lín	0.01	0.3	jiōng shì èr lín	0.01	0.2	jiōng sì èr lín	0.01	1
jiōng shī èr líng	0.01	0.5	jiōng sì ér lín	0.01	1	jiōng sì èr lín	0.01	0.9	jiōng sì èr lín	0.01	1	jiōng sī èr lìng	0.01	0.3
jiōng sì èr lìng	0.01	0.5	jiōng sì ér líng	0.01	0.4	jiōng sì èr lìng	0.01	0.4	jiōng sì èr lìng	0.01	0.3	jiōng sì èr líng	0.01	0.2
jiōng shì èr lìng	0.01	1	jiōng shì èr líng	0.01	1	jiōng shì èr lìng	0.01	0.4	jiōng shì èr lìng	0.01	0.1	jiōng shì èr lìng	0.01	0.4
jiōng sī èr líng	0.01	1	jiōng sī èr lìng	0.01	1	jiōng sì ér líng	0.01	1	jiōng sì èr lìng	0.01	1	jiōng sī èr líng	0.01	1
jiōng sì èr líng	0.01	1	jiōng sì èr lìng	0.01	1	jiōng sì èr líng	0.01	1	jiōng sì èr líng	0.01	1	-	-	-

Table 14: Fuzzy words of AliGenie. The column *dis.* presents the dissimilarity distance according to the generation algorithm, rounded up to the second decimal point. The column *w. rate* presents the wake-up rate. Note that dissimilarity distances smaller than 0.005 are rounded up to 0.00.

	dis.	w. rate		dis.	w. rate		dis.	w. rate		dis.	w. rate		dis.	w. rate
yān mēn jīng líng	0.29	0.5	yān māng jīng líng	0.26	1	wán māng jīng líng	0.26	0.4	wáng mào jǐng lǐn	0.25	0.1	yán māo jīng lǐng	0.25	0.9
yān māo jīng líng	0.25	1	wān māo jīng líng	0.25	0.9	yán māo jīng líng	0.25	0.9	wán māo jīng líng	0.25	0.2	wáng māo jīng líng	0.25	0.1
tián mēn jiān lǐng	0.11	0.7	tí māo jīng míng	0.10	1	tān nāo jiāo líng	0.10	0.4	táng māo jiē liáo	0.09	0.1	tián hāo jǐng mǐng	0.08	0.7
tān lāng jīng líng	0.07	0.3	tàng mǎo jiān líng	0.07	0.2	tíng māo jǐn líng	0.07	1	tǐ māo jìng lín	0.07	1	tán māo jiān líng	0.07	0.4
tān māo jīng lián	0.07	0.2	tān māo jīng liáng	0.07	0.4	tián māo jiǎng lìng	0.07	0.4	tián māo jīng liǎn	0.07	1	tí máo jīng líng	0.07	0.8
tí māo jīng líng	0.07	1	tīng māo jǐng lǐng	0.07	1	tiǎn mào jìng liáng	0.07	1	ti1 māo jīng líng	0.07	1	shāi māo jīng líng	0.07	0.6
tān māo jīng liáo	0.07	0.1	tiān mǎo jìng liào	0.06	0.9	tián māo jiāo lǐng	0.06	0.2	tián māo jīng liáo	0.06	1	tāng pāo jīng lìn	0.06	0.1
tián pǎo jìng níng	0.06	1	tán lǎo jiě líng	0.06	0.1	tān pāo jīng líng	0.06	0.1	shàng māo jǐng lǐng	0.05	0.1	tiē mā jīng líng	0.05	1
shān māo jǐng lǐng	0.05	0.2	shān māo jīng líng	0.05	1	sān māo jǐng líng	0.05	1	sān māo jīng lǐng	0.05	0.5	sān māo jīng líng	0.05	1
tán hāo jīng líng	0.05	0.1	tān hāo jīng líng	0.05	0.1	tāng máo jīng niè	0.05	0.1	tiān kāo jīng líng	0.05	1	teng1 māo jìng lǐng	0.05	1
tān māo jiōng líng	0.05	0.7	tián māo jiǒng līng	0.05	0.8	tián māo jiǒng líng	0.05	1	tián mào jiōng lǐng	0.05	1	tián māo jīng lóu	0.05	1
zān māo jīng líng	0.04	1	tiǎn mēn jǐng lǐng	0.04	0.5	tián mē jìng líng	0.04	1	tiān mē jīng líng	0.04	1	diān māo qí mìng	0.04	0.9
tián lāo līng líng	0.04	0.3	tiǎn mie1 jǐng līng	0.04	0.8	tān nāo jīng níng	0.04	0.5	tān lāo jīng níng	0.04	0.7	tán māo jīn míng	0.03	0.2
tá lāo jīng líng	0.03	0.3	tiǎn nào jìng līn	0.03	0.9	tán lāo jīng lǐng	0.03	0.1	tán māo jīng míng	0.03	0.4	tān nāo jǐng lǐng	0.03	0.2
tān nāo jìng líng	0.03	0.1	tān nāo jīng líng	0.03	0.2	tān lǎo jīng lǐng	0.03	0.3	tān lāo jīng líng	0.03	0.8	tān māo jīng míng	0.03	0.3
tiān lāo jǐng līn	0.03	1	tiǎn mǎo jìng mìng	0.03	1	tián nāo jǐng lǐng	0.03	1	tián nāo jīng lǐng	0.03	1	tián nāo jīng líng	0.03	1
tiān nāo jīng líng	0.03	1	tiǎn lāo jǐng lìng	0.03	1	tiǎn māo jǐng míng	0.03	1	tián māo jǐng míng	0.03	1	tián lāo jīng lǐng	0.03	1
tián lāo jīng líng	0.03	1	tiān māo jīng míng	0.03	1	tiān lāo jīng líng	0.03	1	rán māo jīng lǐng	0.03	0.6	rán māo jǐng líng	0.03	0.7
xiǎn miáo jǐng lǐng	0.03	0.1	tán māo jiē lín	0.03	0.4	tān māo jiě lǐng	0.03	0.4	táng mào jiē lǐng	0.03	0.2	tán māo jiē líng	0.03	0.5
tiǎn mào jǐn liè	0.03	1	tān māo jiē líng	0.02	0.7	tián mào jiē lǐn	0.02	1	tiǎn māo mǐng líng	0.02	0.2	tián mào jiě lǐng	0.02	1
tián mǎo jǐng liě	0.02	1	tián māo jǐng liě	0.02	1	tiǎn máo jìng lèng	0.02	1	tián māo jìng lèng	0.02	0.5	tiàn māo dǐng lǐng	0.02	1
tián móu qìng líng	0.02	1	tiǎn móu jìng níng	0.02	1	tiàn móu jìn líng	0.02	1	tián móu jìng líng	0.02	1	pian2 māo jǐng líng	0.01	1
diān miào n ìng líng	0.01	0.2	tài mào jìng lín	0.01	0.3	tián máo lǐng níng	0.01	0.3	tāi māo jīng lín	0.01	0.4	diǎn mǎo qí nǐng	0.01	0.4
diǎn mǎo qí níng	0.01	1	tāi māo jǐng lǐng	0.01	0.5	tāi māo jǐng líng	0.01	0.8	tāi māo jīng líng	0.01	0.1	tián māo lǐng lǐng	0.01	0.2
tiān māo qí nìng	0.01	0.9	diān máng jǐn líng	0.01	1	tiān miáo qín lǐn	0.01	0.2	tiān miǎo qín lín	0.01	0.1	tián māng jǐng nǐng	0.01	1
tān māng jǐng lǐng	0.01	0.2	diān māng jǐng líng	0.01	1	tān māng jǐng líng	0.01	0.4	tān māng jīng líng	0.01	0.6	tiao2 māo jīng líng	0.01	0.2
tān mān jīng líng	0.01	0.4	tāo māo jǐng lǐng	0.01	0.1	tāo māo jǐng líng	0.01	0.1	diǎn mào qín līn	0.01	0.7	tān mā jǐng lǐng	0.01	0.2
diǎn mào qín lín	0.01	0.7	tián māng jǐng lǐng	0.01	1	tián māng jīng líng	0.01	1	tiān māng jīng līng	0.01	1	tiān māng jīng líng	0.01	1
tiǎn mān jìng lìng	0.00	1	tián mān jìng lìng	0.00	0.9	tiǎn mān jīng lìng	0.00	1	tián mān jīng líng	0.00	1	tiǎn māo qǐn lǐn	0.00	1
tián mā jīng líng	0.00	1	tiǎn māo qín līn	0.00	1	diǎn miáo jǐn líng	0.00	0.1	diān miào jǐn líng	0.00	0.3	tiǎn máo qīn lín	0.00	1
tián mào qǐn lǐng	0.00	1	diān miào jìng lín	0.00	0.3	diān miào jǐng lín	0.00	0.3	tiǎn miǎo jīn níng	0.00	0.6	diān miāo jǐn lǐng	0.00	0.4
tián māo jìng nín	0.00	1	tǎng miáo jìng lǐng	0.00	0.1	tǎng miāo jǐng líng	0.00	0.1	diǎn māo qíng lín	0.00	1	diān miào jǐng líng	0.00	0.5
tiān miāo jǐn lìn	0.00	0.9	tǎn māo jìn lǐn	0.00	0.1	tán māo jīn níng	0.00	0.2	diān miāo jìng líng	0.00	0.8	tiān māo qǐng nìng	0.00	1
diǎn mào qíng líng	0.00	0.7	tiān miǎo jǐng lǐn	0.00	0.1	diǎn māo qíng líng	0.00	1	tiān miáo jìng līn	0.00	0.5	tāng māo jīn lǐn	0.00	0.6
diǎn māo jìng níng	0.00	1	diǎn māo jǐng níng	0.00	1	diǎn máo jīn lín	0.00	1	tàng māo jǐn líng	0.00	0.6	tāng māo jǐng níng	0.00	0.7
tān māo jǐng nǐng	0.00	0.5	tán māo jīng níng	0.00	0.2	diān māo jīn līn	0.00	1	tān māo jīng níng	0.00	0.6	tiān miāo jǐng lín	0.00	1
táng māo jǐn líng	0.00	0.3	tián māo qìng līn	0.00	1	tāng mǎo jǐng lìn	0.00	0.2	tiǎn miào jìng líng	0.00	0.1	tián māo jǐn nǐng	0.00	1
tán māo jǐn líng	0.00	0.5	táng māo jīn líng	0.00	0.3	tāng mǎo jīng lǐn	0.00	0.6	tāng māo jǐng lǐn	0.00	0.6	tián mào qíng līn	0.00	1
tán māo jīng lǐn	0.00	0.1	tiān māo qǐng lín	0.00	1	diān máo jǐn lǐng	0.00	1	tián māo jǐn níng	0.00	1	tān māo jǐn lǐng	0.00	0.6
tiǎn miáo jìng lǐng	0.00	0.1	tán māo jīn līng	0.00	0.3	tán māo jīng lín	0.00	0.2	diān máo jǐng lǐn	0.00	1	tān māo jǐng lǐn	0.00	0.3
tá māo jīng līng	0.00	0.4	tiǎn miāo jǐng līng	0.00	0.9	tián miāo jǐng lìng	0.00	0.4	tiǎn miāo jǐng lǐng	0.00	0.2	tiān miāo jìng lǐng	0.00	1
tián miāo jìng líng	0.00	0.2	tián miāo jǐng líng	0.00	0.4	tiǎn miāo jīng lǐng	0.00	0.1	tián miāo jīng líng	0.00	0.6	tān māo jīng lín	0.00	0.7
tān māo jǐn líng	0.00	0.4	tiān miāo jīng líng	0.00	1	tiǎn mào jǐn lìn	0.00	1	tiǎn mǎo jǐn lìn	0.00	1	tā māo jǐng líng	0.00	0.2
tā māo jīng líng	0.00	0.3	tǎng māo jìng lǐng	0.00	0.1	tǎng māo jìng líng	0.00	0.1	tǎng māo jīng líng	0.00	0.3	tiǎn mào jǐn lǐn	0.00	1
tián mào qìng līng	0.00	1	tiān mǎo qìng líng	0.00	1	táng mǎo jìng líng	0.00	0.2	táng māo jìng lǐng	0.00	0.1	táng máo jǐng lǐng	0.00	0.1
táng mǎo jīng lǐng	0.00	0.1	táng māo jīng lǐng	0.00	0.2	tiǎn mǎo jìng nìng	0.00	1	tián mǎo jìng nìng	0.00	1	tǎn mào jīng lìng	0.00	0.1
diǎn máo jǐng lǐng	0.00	1	tǎn māo jìng líng	0.00	0.2	tián māo qǐng líng	0.00	1	tǎn māo jīng líng	0.00	0.3	tāng mǎo jìng līng	0.00	0.6
tāng māo jìng lǐng	0.00	0.6	tāng māo jìng líng	0.00	0.7	tāng māo jǐng líng	0.00	0.2	tāng māo jīng líng	0.00	0.5	tán mào jǐng líng	0.00	0.1
tán māo jìng lǐng	0.00	0.2	tán mǎo jǐng líng	0.00	0.1	tán māo jǐng lǐng	0.00	0.3	tán māo jǐng líng	0.00	0.3	tán māo jīng lǐng	0.00	0.2
tiān mào jīn lǐn	0.00	1	tián máo jīn līn	0.00	1	diān máo jǐng lǐng	0.00	1	tián máo jǐng níng	0.00	1	tiǎn máo jīn lín	0.00	1
tián māo jǐng níng	0.00	1	tān māo jìng líng	0.00	0.7	tān māo jǐng lǐng	0.00	0.1	diān māo jǐng líng	0.00	1	tān māo jǐng líng	0.00	0.4
tián māo jīng níng	0.00	1	tān māo jīng lǐng	0.00	0.6	tān māo jīng līng	0.00	0.4	tiān māo jīng níng	0.00	1	tān māo jǐng líng	0.00	0.3
tiǎn mǎo jìng lìn	0.00	1	tián mào jǐng lìn	0.00	1	tiān máo jǐng lìn	0.00	1	tiǎn māo jǐn lǐng	0.00	1	tián māo jǐn lǐng	0.00	1
tiǎn māo jīng lìn	0.00	1	tián māo jǐn líng	0.00	1	tián mào jìng līn	0.00	1	tiǎn mǎo jǐng lǐn	0.00	1	tián mǎo jìng lǐn	0.00	1
tiǎn māo jìng lǐn	0.00	1	tián máo jǐng līn	0.00	1	tiān mǎo jǐng lǐn	0.00	1	tiǎn mào jǐng lín	0.00	1	tiān mào jǐng lín	0.00	1
tián máo jǐng lín	0.00	1	tiǎn máo jīn līng	0.00	1	tián māo jīn líng	0.00	1	tiān māo jīn líng	0.00	1	tiān māo jīng lín	0.00	1
tiǎn mǎo jìng lìng	0.00	1	tiǎn mǎo jìng lǐng	0.00	1	tiǎn mǎo jǐng lìng	0.00	1	tián mào jìng lǐng	0.00	1	tiǎn mǎo jìng līng	0.00	1
tián mào jǐng lǐng	0.00	1	tián mǎo jìng līng	0.00	1	tián mǎo jìng lǐng	0.00	1	tiàn māo jǐng lìng	0.00	1	tián māo jìng lìng	0.00	1
tiǎn māo jǐng lìng	0.00	1	tiǎn māo jìng lǐng	0.00	1	tián máo jìng līng	0.00	1	tiǎn māo jǐng lǐng	0.00	1	tián máo jǐng lǐng	0.00	1
tián mào jīng lǐng	0.00	1	tiǎn māo jǐng līng	0.00	1	tiān mǎo jǐng lǐng	0.00	1	tián māo jǐng lìng	0.00	1	tiān mào jǐng líng	0.00	1
tián māo jìng lǐng	0.00	1	tián mǎo jǐng líng	0.00	1	tiǎn māo jǐng lìng	0.00	1	tián māo jǐng līng	0.00	1	tiǎn māo jǐng líng	0.00	1
tián māo jǐng lǐng	0.00	1	tián mǎo jīng lǐng	0.00	1	tián māo jìng líng	0.00	1	tián māo jǐng líng	0.00	1	tiān māo jǐng lǐng	0.00	1
tián māo jīng lìng	0.00	1	tiǎn māo jīng lǐng	0.00	1	tiǎn māo jǐng lǐng	0.00	1	tián māo jīng lǐng	0.00	1	tián māo jīng līng	0.00	1
tiǎn māo jīng líng	0.00	1	tiān māo jǐng líng	0.00	1	tián máo jǐng líng	0.00	1	tián māo jīng líng	0.00	1	tiān māo jīng līng	0.00	1
tiān māo jīng lǐng	0.00	1	tiān māo jīng líng	0.00	1	-	-	-	-	-	-			

Table 15: Fuzzy words of Apple Siri. The column *dis.* presents the dissimilarity distance according to the generation algorithm, rounded up to the second decimal point. The column *w. rate* presents the wake-up rate.

	dis.	w. rate		dis.	w. rate		dis.	w. rate		dis.	w. rate
hey sirr e	0.37	1	hei suru r a	0.35	0.7	hay scir e	0.33	0.6	haiiasciree	0.28	0.6
heyisyree	0.27	0.2	heii sirea	0.25	0.7	haiy cire	0.22	0.8	heai ssuree	0.22	0.5
hey sserea	0.20	0.8	hay syrrie e	0.20	1	heii ssuiri	0.20	0.8	hay cieree a	0.20	0.7
hay syerib	0.19	0.1	hei ssirr	0.19	0.8	heiy ssuiree	0.18	0.5	heii syrie	0.16	0.2
heiy sciere	0.13	0.6	heiy sirie	0.13	0.1	haiy scyrie	0.13	0.4	heai serea	0.12	0.9
haii sciree	0.12	0.7	heai ceri	0.12	0.7	heai ssuiri	0.12	0.5	hey sierie	0.11	1
hay sere e	0.11	1	hay siri e	0.11	0.8	haiy cori	0.10	0.4	hay ssirib	0.10	0.9
heiy suree	0.09	0.9	hei syeii	0.08	0.1	hai sirie	0.07	1	hey ssire	0.07	0.9
hai scere	0.07	0.8	heiy cieri	0.07	0.8	hey ceree	0.07	0.1	heiy scirie	0.06	0.1
heiy syrie	0.06	0.1	hay syrieqe	0.06	1	hay sciria	0.06	1	hay psyrria	0.06	1
haiy syrie	0.06	0.7	bay scirie	0.03	0.8	hey sori	0.02	1	hay scirih	0.02	1
hey scirih	0.02	1	hei ssoree	0.02	0.9	hay surie	0.01	1	hay suri	0.01	0.6
hey suri	0.01	0.5	hay cieri	0.01	1	hai sciri	0.01	0.9	hay sceri	0.01	0.7

Table 16: Fuzzy words of Amazon Echo. The column *dis.* presents the dissimilarity distance according to the generation algorithm, rounded up to the second decimal point. The column *w. rate* presents the wake-up rate.

	dis.	w. rate		dis.	w. rate		dis.	w. rate		dis.	w. rate		dis.	w. rate		dis.	w. rate
alexoer	0.44	0.6	ilexcer	0.37	1	ileqcer	0.37	0.9	ilekcer	0.37	0.7	ileqsar	0.35	1	ilexsar	0.35	1
ilexsur	0.35	1	ileksur	0.34	0.8	ileqsur	0.34	0.6	ilebser	0.33	0.7	aleqsxr	0.32	0.7	arlecci	0.30	0.6
alexsur	0.30	1	alexsar	0.30	0.9	alexser	0.30	0.8	aleksar	0.28	0.7	alecsar	0.28	0.6	aleqsar	0.28	0.5
ileqci	0.27	1	ileckci	0.27	0.1	ilekci	0.27	0.1	alecsur	0.26	0.7	aleqsur	0.26	0.6	aleksur	0.26	0.6
irlegsa	0.25	0.8	alexcer	0.25	0.8	urlecsa	0.24	1	urlexsa	0.24	1	erlecsa	0.24	1	urleksa	0.24	1
ilexca	0.24	1	ilexqa	0.24	1	erleksa	0.24	0.9	erlexsa	0.24	0.1	alekcer	0.23	0.7	aleqcer	0.23	0.6
alicsur	0.23	0.5	ilexsca	0.22	1	ileckca	0.22	0.2	aleksr	0.22	0.5	lexac	0.22	0.4	alexsib	0.21	1
ilellksa	0.19	1	ileccsa	0.19	1	ilecksa	0.19	1	ilekhsa	0.19	1	ilekcsa	0.19	1	alexusi	0.19	0.3
nlecses	0.17	1	ileqsca	0.17	1	ilekSca	0.17	1	ilexssa	0.17	1	alexcpa	0.17	0.7	jleksib	0.17	0.9
lleksra	0.17	0.6	lecsca	0.17	0.4	lecsba	0.17	0.2	alektci	0.16	0.5	alexsca	0.16	0.7	vqleksa	0.16	0.9
lfkqsa	0.16	0.6	arlekca	0.15	0.9	aldcxa	0.15	0.8	ameqdsa	0.14	0.4	dlecdsa	0.13	1	aleqsrk	0.13	0.5
ileqsaa	0.12	1	alexuaa	0.12	0.7	alebqsa	0.12	1	gleksta	0.12	0.5	allexsa	0.12	1	alelksa	0.12	0.9
aleqtca	0.12	0.7	alebsab	0.11	0.8	hlexqa	0.11	0.6	alfbsa	0.11	0.6	dpecssa	0.11	1	dpeccsa	0.11	0.9
ileqsa	0.10	1	ileqssa	0.10	1	ilekssa	0.10	1	ilexsa	0.10	1	alexca	0.10	1	alexqa	0.10	1
alecsoi	0.10	0.4	aleckci	0.10	0.7	aleqci	0.10	0.5	alekci	0.10	0.5	arlexsa	0.10	1	arlecsa	0.10	1
arleksa	0.10	1	alexssa	0.10	1	alexsra	0.10	1	alexsia	0.10	0.9	alexsta	0.10	0.9	lecsa	0.09	0.7
aleksai	0.09	0.5	alecdsa	0.08	1	alecpsa	0.08	0.9	alecrsa	0.08	0.7	aleqsca	0.08	0.8	aleksua	0.08	0.8
alecsda	0.08	0.8	aleksca	0.08	0.6	aleksba	0.08	0.6	aleqsda	0.08	0.6	alecsca	0.08	0.5	aleqsra	0.08	0.5
cleksaa	0.07	0.9	aldqssa	0.07	1	alexci	0.05	0.1	aleckca	0.05	0.5	alekca	0.05	0.5	alecssj	0.05	0.1
alexsaa	0.04	1	alebsa	0.04	1	alepsa	0.03	1	alepsah	0.03	0.9	alexsa	0.02	1	anexa	0.02	0.5
alekssi	0.02	0.3	aleqsaa	0.02	0.9	aleccsa	0.01	1	aleqhsa	0.01	1	alekcsa	0.01	1	alecqsa	0.01	1
alechsa	0.01	0.9	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Table 17: Fuzzy words of Amazon Echo Dot. The column *dis.* presents the dissimilarity distance according to the generation algorithm, rounded up to the second decimal point. The column *w. rate* presents the wake-up rate.

	dis.	w. rate		dis.	w. rate		dis.	w. rate		dis.	w. rate		dis.	w. rate
ureqssr	0.44	0.1	arleqsr	0.40	1	ilekcer	0.37	1	ilexcer	0.37	0.1	ilexsur	0.35	1
ileqsar	0.35	1	ileksar	0.35	1	ileksur	0.34	1	alekcir	0.32	0.7	ileqser	0.30	1
ermeqqa	0.30	0.9	alexser	0.30	1	alexsur	0.30	1	alexsar	0.30	1	ilexbs	0.28	0.1
urlexca	0.28	1	aleqsar	0.28	0.9	alecsar	0.28	0.7	aleksar	0.28	0.7	ilekci	0.27	1
ileqci	0.27	1	ilefksa	0.26	1	alecsur	0.26	0.8	aleksur	0.26	0.8	aleqsur	0.26	0.7
alexcer	0.25	0.8	urleqsa	0.24	1	urleksa	0.24	1	erleqsa	0.24	1	urlecsa	0.24	1
erleksa	0.24	1	ilexca	0.24	1	erlexsa	0.24	0.7	allesab	0.24	0.1	aleqcer	0.23	0.7
alekcer	0.23	0.7	ilexsca	0.22	0.8	ileckca	0.22	1	ilekca	0.22	0.7	alekser	0.22	1
aleqser	0.22	0.7	alekqci	0.22	0.5	blexasi	0.21	0.4	uklefca	0.20	1	alexa c	0.20	0.7
aleqxda	0.19	0.9	arlesca	0.19	1	ilellksa	0.19	1	allexba	0.19	0.9	ilecksa	0.19	1
ileccsa	0.19	1	arlexca	0.18	1	arlcxsa	0.17	0.5	ileksda	0.17	1	ilexssa	0.17	0.9
ileksca	0.17	0.2	leqsuq	0.17	0.7	ikecssa	0.16	0.9	clebkca	0.16	0.2	alexsca	0.16	1
alexsua	0.16	1	alelkca	0.15	0.8	cleqsai	0.15	0.1	aleqcda	0.15	0.7	ajecksa	0.14	0.3
alexa a	0.14	0.7	akhdxsa	0.14	0.7	aleqciq	0.13	0.3	brleksa	0.13	1	alecsid	0.13	0.6
aqlecsa	0.13	1	clemcsa	0.13	1	cleqsra	0.13	0.7	ilexsaa	0.12	1	allexsa	0.12	1
alellksa	0.12	1	aleqtba	0.11	0.7	ileksa	0.10	1	ileqsa	0.10	1	ilexsa	0.10	1
ilekssa	0.10	1	alexca	0.10	1	ileqssa	0.10	1	aleckci	0.10	0.7	alekci	0.10	0.7
aleqci	0.10	0.6	lecta	0.10	0.3	arlecsa	0.10	1	arleksa	0.10	1	arleqsa	0.10	0.9
arlexsa	0.10	0.9	alexssa	0.10	1	alexsia	0.10	1	alexsda	0.10	0.9	anexsba	0.10	0.1
blebssa	0.09	1	alectda	0.09	0.8	ulzxsw	0.09	0.1	akekfa	0.09	0.1	alecpsa	0.08	1
aleqpsa	0.08	1	alaxssa	0.08	0.9	aleqtsa	0.08	1	aleqsca	0.08	0.8	aleksca	0.08	0.7
alecsra	0.08	0.7	alecsca	0.08	0.6	alfcssa	0.07	0.6	alfcsa	0.07	0.2	aldcsa	0.07	0.8
hleqsaa	0.06	1	dlechsa	0.06	0.9	fleqsia	0.06	0.9	dleccsa	0.06	0.9	alersa	0.06	0.7
alexci	0.05	1	blecssa	0.05	1	blecksa	0.05	1	kleqsia	0.05	0.9	alekca	0.05	0.9
aleckca	0.05	0.7	gleccsa	0.05	0.9	hleqsa	0.04	1	alebssa	0.04	1	alekfa	0.03	1
alepsia	0.03	0.7	aleqsha	0.02	0.8	alechsa	0.01	1	alecksa	0.01	1	aleqtta	0.01	0.6

Table 18: Fuzzy words of Google. The column *dis.* presents the dissimilarity distance according to the generation algorithm, rounded up to the second decimal point. The column *w. rate* presents the wake-up rate.

	dis.	w. rate		dis.	w. rate		dis.	w. rate		dis.	w. rate
hea gougll	0.33	0.5	heiigoogaa	0.30	0.3	heay gugal	0.30	1	hey gooogov a	0.30	1
heii googurl	0.27	0.7	heii gugurl	0.27	0.6	heii googerl	0.27	0.6	hei googll a	0.27	1
hei gooo r	0.23	0.2	heiy googow l	0.23	0.5	heiy googol l	0.23	0.1	heiy googerl	0.20	0.8
haii googerl	0.19	0.1	haii gugerl	0.19	0.1	haii gugurl	0.19	0.1	hay googou l	0.19	0.9
heii gugal	0.18	1	hey googal c	0.18	1	hei googal l	0.18	1	hei googel l	0.18	1
heii googourl	0.17	0.9	hay googgrl	0.16	0.6	heii gugull	0.16	1	hei googerl	0.15	0.7
hey googurl	0.15	0.7	hay googerl	0.15	0.6	hey googerl	0.15	0.6	hay gugerl	0.15	0.6
hei gugurl	0.15	0.6	hey gugerl	0.15	0.5	hay googurl	0.15	0.5	hei gugerl	0.15	0.5
he googarl	0.15	0.6	heii googal	0.14	1	hei googal a	0.14	1	hey goooarr	0.14	1
hei googar	0.13	0.7	hay googour	0.13	0.7	heiy gugourl	0.13	1	heiy googrln	0.13	1
heiy googarl	0.13	0.9	haii googarl	0.13	0.8	haii gugarl	0.13	0.1	hey googou	0.11	1
hei goooar	0.11	1	haii gugal	0.10	1	hey googout	0.10	0.5	haii googak	0.09	1
heiy googourl	0.08	1	hey gugarl	0.08	0.8	hei gugourl	0.08	0.8	hay googarl	0.08	0.7
hay gugarl	0.08	0.7	hey googarl	0.08	0.6	hei gugarl	0.08	0.6	hay gugorl	0.08	0.6
hey goegil	0.07	0.9	heyy gugil	0.07	0.1	he googal	0.07	1	hey googouraa	0.07	1
hei googala	0.06	1	hey googau	0.06	1	heiy googall	0.06	1	heih googal	0.06	1
hey googourm	0.06	0.7	haii googal	0.05	1	hay gugal	0.05	1	hei gugal	0.05	1
hey guugal	0.05	0.7	hey googav	0.04	1	hey ggugil	0.04	0.3	hay ggufal	0.03	1
hay googourl	0.03	0.8	hei googourl	0.03	0.8	hey googourl	0.03	0.7	hay gugil	0.02	0.2
hey gugil	0.02	0.1	hei gugil	0.02	0.1	jay googal	0.02	0.8	-	-	-

D

GhostTouch: Targeted Attacks on Touchscreens without Physical Touch (USENIX 2022, CORE: A*)

---

[175] Kai Wang, Richard Mitev, Chen Yan, Xiaoyu Ji, Ahmad-Reza Sadeghi, and Wenyuan Xu. Ghosttouch: Targeted attacks on touchscreens without physical touch. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 1543–1559, 2022. Originally published by USENIX: `https://www.usenix.org/conference/usenixsecurity22/presentation/wang-kai`.

GhostTouch: Targeted Attacks on Touchscreens without Physical Touch

Kai Wang, *Zhejiang University;* Richard Mitev, *Technical University of Darmstadt;*
Chen Yan and Xiaoyu Ji, *Zhejiang University;* Ahmad-Reza Sadeghi,
Technical University of Darmstadt; Wenyuan Xu, *Zhejiang University*

https://www.usenix.org/conference/usenixsecurity22/presentation/wang-kai

This paper is included in the Proceedings of the
31st USENIX Security Symposium.

August 10–12, 2022 • Boston, MA, USA

978-1-939133-31-1

Open access to the Proceedings of the
31st USENIX Security Symposium is
sponsored by USENIX.

GhostTouch: Targeted Attacks on Touchscreens without Physical Touch

Kai Wang*
Zhejiang University

Richard Mitev*
Technical University of Darmstadt

Chen Yan
Zhejiang University

Xiaoyu Ji†
Zhejiang University

Ahmad-Reza Sadeghi
Technical University of Darmstadt

Wenyuan Xu
Zhejiang University

Abstract

Capacitive touchscreens have become the primary human-machine interface for personal devices such as smartphones and tablets. In this paper, we present GhostTouch, the first active contactless attack against capacitive touchscreens. GhostTouch uses electromagnetic interference (EMI) to inject fake touch points into a touchscreen without the need to physically touch it. By tuning the parameters of the electromagnetic signal and adjusting the antenna, we can inject two types of basic touch events, taps and swipes, into targeted locations of the touchscreen and control them to manipulate the underlying device. We successfully launch the GhostTouch attacks on nine smartphone models. We can inject targeted taps continuously with a standard deviation of as low as 14.6×19.2 pixels from the target area, a delay of less than $0.5s$ and a distance of up to $40mm$. We show the real-world impact of the GhostTouch attacks in a few proof-of-concept scenarios, including answering an eavesdropping phone call, pressing the button, swiping up to unlock, and entering a password. Finally, we discuss potential hardware and software countermeasures to mitigate the attack.

1 Introduction

Touchscreens allow users to interact with computers using their fingers and have become a trending alternative to mouses and keyboards. In particular, *capacitive* touchscreens provide multi-touch capabilities, long service life, and cost-effectiveness, and therefore, have been widely used on personal devices such as smartphones, tablets and watches, and even on safety-critical devices such as medical equipment [26] and spacecraft [13].

Reliable and accurate touch sensing is a critical requirement for touchscreens on all devices. However, the ability to measure small electric fields also makes capacitive touchscreens sensitive to environmental impacts such as electromagnetic interference (EMI) [10] and charger noise [23], which can induce fake touches that may greatly impair user experience and even cause unintended device behavior. For instance, there are numerous reports about unresponsive, self-tapping and malfunctioning touchscreens caused by EMI emitted through fluorescent lights [6] and faulty chargers [28, 41, 43].[1]

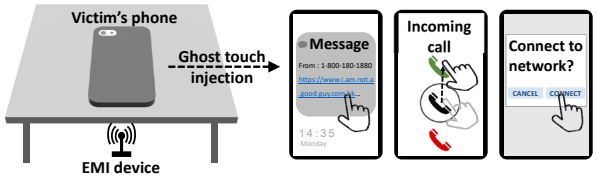


Figure 1: A GhostTouch attack scenario. The attacker uses an EMI device under a table to remotely attack the touchscreen of a smartphone face-down on the table. By injecting fake touches, the attacker can trick the smartphone to (1) click a malicious link containing malware, (2) connect a malicious network, and (3) answer an eavesdropping phone call.

At first glance, the impact of EMI on capacitive touchscreens seems to be largely unpredictable, and hence may not be exploited for targeted attack on the underlying device. Therefore, our motivation and focus in this paper is to address the research question of whether it is possible for an attacker to use EMI to inject controllable fake touches to a touchscreen without any physical contact and manipulate the underlying device in a predictable way. EMI attacks on devices have been studied before. However, as we elaborate on related work in Section 8, and to the best of our knowledge, there have been no published EMI attacks on capacitive touchscreens that can manipulate touch points on the touch screen without requiring any physical contact.

We propose GhostTouch, the first contactless EMI-attack against capacitive touchscreens. The core idea is to interfere with the capacitance measurement of touchscreens using electromagnetic signals, which are injected into the receiv-

*Kai and Richard are co-first authors.
†Xiaoyu Ji is the corresponding author.

[1] In one case a malfunctioning touchscreen even booked a thousand-dollar hotel room itself without the owner's awareness [33].

ing electrodes integrated into the touchscreen. As a result, an electromotive force is induced in the measuring circuit that affects the touch point detection.

To achieve our attack we had to overcome two technical challenges: 1) It is difficult to affect a touchscreen by EMI, since modern touchscreens and devices go through thorough electromagnetic compatibility (EMC) tests [31] and utilize anti-interference design such as shielding [35] and layout optimization [7] to avoid the influence of environmental interference. To address this challenge, we carefully design the transmitting antenna, signal frequency, and attack distance to improve the electromagnetic signal propagation gain, therefore, achieving an effective touch injection. 2) Even if we can inject touches, it is still difficult to create predictable touch events with the touchscreen specifics undisclosed and varying from device to device. We probe the screen to disclose the touchscreen specifics and adjust the parameters of the attack signal accordingly to inject predictable touch events, such as a tap, a swipe-up, or a swipe-down in targeted locations.

Overcomming these challenges, our attack can inject two types of basic touch events, *taps* and *swipes*, into a targeted location of the touchscreen without any physical contact. Most complex gestures can be achieved by the combination of these two basic interactions. By tuning the parameters of the electromagnetic signal and adjusting the antenna, we can control the location and pattern of the fake touches and achieve various touch behaviors including press and hold, swipe to select, slide to scroll, etc., depending on the device model.

We demonstrate the feasibility of this attack in the real world with the setup as shown in Figure 1. In places like a cafe, library, meeting room, or conference lobbies, people might place their smartphone face-down on the table[2]. An attacker may embed the attack equipment under the table and launch attacks remotely. For example, an attacker may impersonate the victim to answer a phone call which would eavesdrop the private conversation, or visit a malicious website.

Our main contributions are as follows:

- We propose `GhostTouch`, the first contactless attack against capacitive touchscreens that can inject taps and swipes into a targeted location of the touchscreen and control the fake touch behavior without any physical contact.
- We evaluate `GhostTouch` attacks successfully on touchscreens of 9 different smartphone models. Our attack can target any area on the touch screen. For example, on Nexus 5X we can inject taps continuously into an area as small as 36.3×175.8 pixels with an delay of less than 0.5 seconds and inject targeted swipes with a success rate of 62.5% and an average delay of 1.6 seconds.
- We demonstrate the real-world impact of `GhostTouch` with 4 practical attack scenarios: answering an eavesdropping phone call[3], pressing the button, swiping up to unlock, and entering a password.
- We suggest both hardware and software countermeasures to protect touchscreens from such attacks.

[2] A study [20] among 3246 participants shows that 54.37% of people would often or sometimes set their phones face-down on the table.

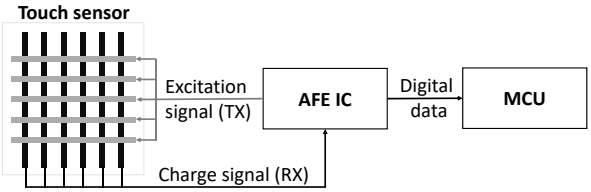


Figure 2: Typical system architecture of a capacitive touchscreen.

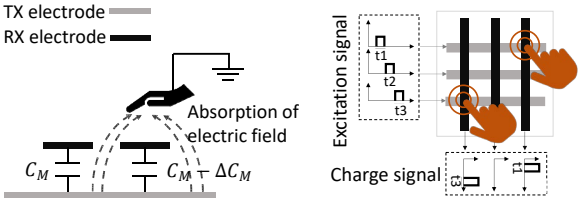
2 Background on Capacitive Touchscreens

A capacitive touchscreen is an input device normally layered on top of the display that detects human touches based on the capacitance variation. When a finger touches the screen, the capacitance at the touch point changes significantly as the charge stored in the screen gets drawn to the finger. By monitoring the capacitance variation at each point of the screen, a touchscreen can detect *touch points* and report *touch events*, e.g., tap, swipe, based on the timing and the locations of the detected touch points. For example, two consecutive touch points in aerial vicinity are recognized by the OS as a swipe, reported to the corresponding app as a swipe over two (touch) points. In the following, we introduce the most popular design of capacitive touchscreens used on smartphones, i.e., a system architecture supporting mutual capacitive sensing and scan driving method, which we consider in this paper.

System architecture: Figure 2 shows a typical system architecture of capacitive touchscreens [25], which includes a touch sensor, an analog front-end integrated circuit (AFE IC), and a micro controller unit (MCU). The touch sensor consists of a grid of transmitting (TX) and receiving (RX) electrodes made of transparent conductive materials, e.g., indium-tin-oxide (ITO). The AFE IC sends excitation signals to the TX electrodes and measures the charge signal from the RX electrodes. The charge signal is digitized and sent to the MCU, which processes the signal and detects touch events. This architecture is designed to support two efficient methods that enable multi-touch sensing, i.e., mutual capacitive sensing, which relates to how the capacitance variance at a single point is measured, and scan driving method, which is used in combination with mutual capacitive sensing to locate the touch points.

Mutual capacitive sensing: When an excitation signal is applied to a TX electrode, the electrode generates an electric

[3] Video demo: https://github.com/USSLab/GhostTouch



(a) Mutual capacitive sensing. (b) Scan driving method.

Figure 3: Illustration of mutual capacitive sensing and scan driving method.

(a) Electrostatic gun and antenna. (b) Generated pulse signal.

Figure 4: (a) The experiment setup of the feasibility study and (b) a pulse signal generated by the electrostatic gun. The pulse lasts for 186 ns.

field that creates a flow of electric charge to the air-gapped RX electrodes, which essentially forms a mutual capacitance C_M between the RX and TX electrodes at each intersection [27], as shown in Figure 3(a). When a finger touches the screen, it absorbs a part of the electric field and changes the mutual capacitance to $C_M - \Delta C_M$, where ΔC_M is caused by the charge drawn to the finger. In this case, the variance of capacitance changes the charge signal measured by the RX electrode, and a touch point at the intersection of RX and TX is detected. The excitation signal is normally a square wave with a frequency of 100 kHz to 500 kHz. Mutual capacitive sensing outperforms other methods such as self-capacitive as it can efficiently locate the touch point by exploiting TX-RX pairs.

Scan driving method: The scan driving method (SDM) [16, 17] is designed to locate the touch points on the screen by exciting all TX electrodes in turn, as shown in Figure 3(b). As only one TX is excited at a time, the touchscreen can locate a touch to a specific position by the row and column of the active TX-RX pair and can also support multi-touch detection. SDM involves several major parameters: the number of TX electrodes N, the time it takes to scan all TX electrodes T_p, and the time it takes to scan one TX electrode T_{p1}. Compared with other methods, the scan driving method has a simple structure, shorter sensing time, and lower signal-to-noise ratio (SNR) and has been adopted on most smartphones.

3 Feasibility of Injecting Touches with EMI

Capacitive touchscreens detect touches based on the charge signals on the RX electrodes and locate touches by scanning the TX electrodes. However, the RX and TX electrodes are essentially conductors that electromagnetic waves may couple on (i.e., convert to electrical signals) and interfere with the touch sensing. Therefore, we are motivated to explore the feasibility of injecting fake touch points into the touchscreens of various smartphones using electromagnetic interference (EMI) and analyze the distribution of the injected touch points.

Electromagnetic interference: Electromagnetic interference [21] appears when undesirable voltages or currents are present in the environment of a device. This can lead to malfunctioning or degradation of the performance. The voltages or currents may affect a device by conduction or radiation. In our case, we focus on electromagnetic interference by radiation, which can interfere with a device through electromagnetic coupling, a phenomenon that generates an electrical charge in the electrical wiring or circuits of a device. Maxwell's equation explains the principle of electromagnetic coupling: $\oint_{\partial\Sigma} \mathbf{E} \cdot d\mathbf{l} = -\int_{\Sigma} \frac{\partial \mathbf{B}}{\partial t} \cdot d\mathbf{A}$, where $\partial\Sigma$ is a closed contour, Σ is a surface bounded by $\partial\Sigma$, $\mathbf{E}$ is the electric field, $\mathbf{B}$ is the magnetic field, $d\mathbf{l}$ is an infinitesimal vector of $\partial\Sigma$, and $d\mathbf{A}$ is an infinitesimal vector of Σ. In a touchscreen, the measuring circuit can be considered as the closed contour $\partial\Sigma$. The changing magnetic field $\mathbf{B}$ of an electromagnetic radiation passing through the surface Σ of a touchscreen can induce an electromotive force as the left part of the equation, which may affect the capacitance measurement of touchscreens.

Experiment setup: The experiment setup is shown in Figure 4(a). We use an electrostatic gun [1] to generate a strong pulse signal, which is sent to an antenna we made using Dupont jumper wires and gets converted to strong electromagnetic interference. The electrostatic gun can generate short pulses with the waveform shown in Figure 4(b), and the pulse amplitude can vary from 1kV to 18kV. In this experiment, we set the amplitude to 10kV. We place a 5mm-thick acrylic board between the antenna and the phone's touchscreen, and inject EMI with the antenna at two types of positions: parallel to the vertical or horizontal edges of the phone. We experiment on 12 phone models and show the results in Table 1.

Results: Although the capacitive touchscreens of smartphones go through thorough electromagnetic compatibility tests and anti-interference design, 8 of the 12 tested smartphones are susceptible to EMI. We record and analyze the distribution of the injected points on the 8 phones. We observe two types of results regarding the density and distribution of the injected points. Among the 8 susceptible phones, two can only be injected with sparse fake points while six can be injected with dense fake points, indicating a greater susceptibility and a higher attack success rate. The injected points distribute along a horizontal line of the touchscreen on 3 phones and a vertical line on other three other phones, validating the possibility to inject fake points along both the

Table 1: Results of injecting fake touch points on 12 phones.

Phone model	Success	Injection Speed		Point Distribution	
		Sparse	Dense	Horizontal	Vertical
Nexus 5X	✓	×	✓	×	✓
Google Pixel 1	✓	×	✓	×	✓
OPPO K3	×	N/A	N/A	N/A	N/A
OPPO Reno	×	N/A	N/A	N/A	N/A
OPPO Reno 2	✓	×	✓	✓	×
OPPO Reno 3	×	N/A	N/A	N/A	N/A
OPPO Reno 3 Pro	✓	×	✓	✓	×
One Plus 8T	✓	✓	×	×	×
Huawei P10 Plus	✓	×	✓	✓	×
Huawei P40	✓	✓	×	×	×
Samsung S20 FE	✓	×	✓	×	✓
iPhone 7 Plus	×	N/A	N/A	N/A	N/A

horizontal and vertical direction of the touchscreen.

Observations of the touch point distribution: We show the distribution of the injected touch points with Google Pixel 1, Nexus 5X and Huawei P10 Plus as an example. The touch point data is recorded using the Android Debug Bridge (ADB). Figure 19 in Appendix shows a visual distribution of the injected points on the three phones, and Figure 20 in Appendix shows the cumulative distribution function (CDF) plots of the injected points along the horizontal (X-axis) and vertical (Y-axis) directions. The results show that more than half of the injected points are distributed nearly uniformly on a specific line of the touchscreen where the antenna is placed, either vertical (Google Pixel 1 and Nexus 5X) or horizontal (Huawei P10 Plus). The direction of fake point distribution is the same as the direction of the antenna. We believe the difference in distribution is due to the different touchscreen layouts, especially the RX electrodes. For example, the RX electrodes of Nexus 5X are vertical while the RX electrodes of Huawei P10 Plus are horizontal, which all correspond to the distribution of fake points on these phones.

Observations of the capacitance variation: We further explore the reason behind fake touch points based on the raw capacitance data of the touchscreen. Using ADB, we are able to record the capacitance data on the Huawei P10 Plus before and during the electromagnetic interference. We calculate the difference of the capacitance data to acquire the variation caused by the EMI and plot the result in Figure 5. The capacitance of a line in the middle of the screen decreases dramatically, which corresponds to the distribution of injected points on this phone in Figure 19(c) in Appendix.

Our feasibility study confirms that the touchscreens of various phone models are susceptible to EMI. Therefore, it is feasible for an attacker to inject fake touch points to the touchscreen of the victim's smartphone without any physical contact. However, as the next step, we will study methods to transform random electromagnetic interference to elaborate electromagnetic attacks that can inject controllable touch events and manipulate the smartphone in real-life scenarios.

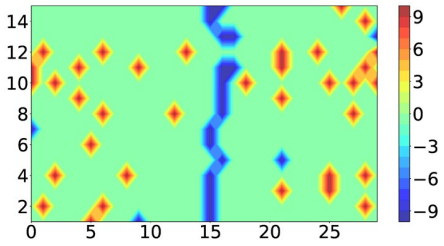


Figure 5: A visual illustration of the capacitance variations caused by EMI on the touchscreen of a Huawei P10 Plus. The plot corresponds to the screen in the landscape orientation. The capacitance variations in the middle (valuing between -3 to -9) conform to the results in Figure 19(c) in Appendix.

4 Threat Model

The attacker's goal is to manipulate the victim device by injecting fake touches to the touchscreen in a contactless manner. We make the following assumptions for the attack:

- **Victim device:** The victim device is equipped with a capacitive touchscreen, such as a smartphone or tablet. The device is unaltered before the attack and placed *face-down* on a surface (e.g., a table) during the attack.
- **Attacker's knowledge:** The adversary may know the victim's device model and acquire a device of the same model to study beforehand. Further, the adversary may acquire the victim's phone number via social engineering.
- **Attacker's capability:** The attacker can only attack the device by manipulating the touchscreen via electromagnetic signals. However, the adversary cannot physically touch the victim device or ask the user to perform any tasks.
- **Attack setup:** The attacker may hide the attack equipment under the table where the victim devices might be attached to a surface (e.g., under a table in a meeting room, or charging station). The attacker can control the attack equipment remotely.

5 Attack Design

In this section, we present `GhostTouch`, the first contactless attack against capacitive touchscreens of smartphones. Our goal is to inject controllable touch events, such as taps and swipes, into a targeted area of the touchscreen and use them to manipulate the device. To achieve this goal, we need to tackle the following technical challenges:

1) Effectively inject fake touch points: Although our study confirms that EMI injection is feasible, the fake points are injected using unpredictable signals after trial and error. To

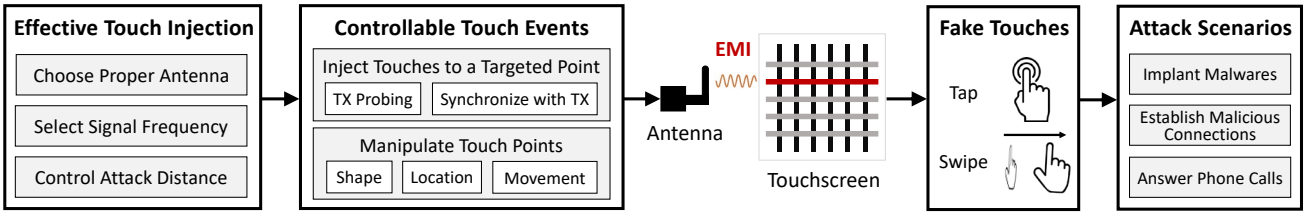


Figure 6: An illustration of the GhostTouch design. We first prepare the EMI signal for effective touch point injection and then design the signal for controllable touch events. We emit the crafted signal with an antenna to make the touchscreen mistakenly sense two types of basic touch events, i.e., taps and swipes, which can be used to construct more complex touch behaviors for various attack scenarios.

achieve a powerful attack, we need to understand the interference process and design electromagnetic signals for effective and efficient injection.

2) Create controllable touch events: The fake touch points in previous work and our feasibility study can only distribute randomly on the screen. To achieve controllable touch events such as taps and swipes, we need to constrict the fake touch points into a target area of the screen and adjust their positions as desired.

To address the first challenge, we study three main factors that affect the effectiveness and efficiency of touch point injection, i.e., the transmitting antenna, signal frequency, and attack distance. Our goal is to find the best options for these factors that in combination can achieve the maximum intensity of touch point injection in a reproducible and cost-effective way. To address the second challenge, we study methods to make the randomly distributed touch points repeatedly appear in a constricted area of the touchscreen. By adjusting the timing of the transmission, the transmitting period, and antenna positions, we can control the position of the injected touch points.

The design of GhostTouch is shown in Figure 6. In the first stage, we prepare the EMI signal for effective touch point injection by choosing a proper antenna, selecting signal frequency, and controlling the attack distance. In the second stage, we design the EMI signal for controllable touch events by probing the touchscreen, synchronization, and adjusting key signal parameters. After these stages, the crafted EMI signal is emitted by the antenna to attack the touchscreen of a smartphone. GhostTouch can induce two types of basic touch events, taps and swipes, which can be used to construct more complex touch behaviors for various attack scenarios. We introduce the details in the following.

5.1 Effective Touch Point Injection

In the first stage, we study the main factors that affect the effectiveness and efficiency of touch point injection, including the type and length of the transmitting antenna, the frequency of the EMI signal, and the distance between the transmitting antenna and touchscreen. We seek to find the optimal combination of these factors to increase the possibility of touch point injection. Though the transmitting power plays an important role for EMI, it is generally considered that the higher the power the stronger the interference, therefore we do not discuss the transmitting power in the attack design.

In our attack, the electromagnetic interference needs to satisfy two requirements: (1) the intensity of the induced electromotive force needs to be high enough to influence the touch sensing, and (2) the electromagnetic interference needs to affect only a part of the touchscreen so that the injected touch points can appear in a restricted area instead of all over the screen. To meet these requirements, we elaborate on our considerations in the transmitting antenna, signal frequency, and attack distance.

5.1.1 Transmitting Antenna

An electromagnetic field can be generated and received by an antenna. In our attack, the electrodes in a touchscreen essentially act as antennas that unintentionally pick up the electromagnetic interference. The RX electrodes are especially vulnerable because the induced electrical signals can directly affect the touch sensing. To maximize the efficiency of electromagnetic coupling, the antenna we use to emit the EMI needs to match the equivalent antennas in the touchscreen, including both the antenna type and length. There are many types of antenna, mainly including electric dipole (e.g., Hertzian antenna) antenna and magnetic dipole (e.g., small loop antenna) antenna [22]. The electrodes of a touchscreen can be regarded as electric dipole antennas, and the circuit formed by a TX electrode, an RX electrode and the AFE IC can be regarded as a magnetic dipole antenna. Thus, we can use both types of antennas to transmit the EMI. To make the antennas' length match on a similar magnitude, we measure the size of the touchscreen and make a rough estimation.

For example, the size of a Nexus 5X is 147×72.6 mm. We have verified that both our self-made electric dipole antenna using a Dupont jumper wire of 140mm and a 4mm-diameter tip antenna (with a total coil length of around 70mm) are effective in our attack as they match the equivalent antennas in the touchscreen.

5.1.2 Signal Frequency

The frequency of an electromagnetic signal determines the efficiency of it being transmitted or received by a given antenna. We can estimate the effective signal frequency based on the electrical length [36] of the targeted/selected antenna. Electrical length is defined as the ratio of the physical length of the antenna to the wavelength of the electromagnetic signal. Empirically, an antenna whose electrical length is less than 1/20 or 1/50 can be considered as electrically short, which means that it can barely emit EM signals of the desired wavelength (frequency). Higher antenna gain can generally be achieved with larger electrical length, e.g., when the ratio between the antenna's physical length and the signal's wavelength is 1/2 or 1/4. In our attack, assume the physical length of the antenna is l, then the signal frequency corresponding to a 1/50 electrical length is $c/50l$, where c is the speed of light. To have the signal being effectively transmitted and received, the signal frequency needs to be higher than $c/50l$ to ensure an electrical length higher than 1/50.

For example, consider the Nexus 5X we discussed earlier, to make the electromagnetic signal couple into the 147mm RX electrode, the signal's frequency needs to exceed 40.8MHz. To verify the estimated frequency, we conduct a frequency sweeping experiment starting from 4MHz using a Rigol DG5072 arbitrary waveform generator and the 140mm self-made antenna. The end frequency is 70MHz because it is the maximum frequency supported by the equipment. We set the signal amplitude to 20Vpp in the experiment. The results show that we can inject a significant amount of touches into the touchscreen of a Nexus 5X at a signal frequency of 46MHz, which conforms to our estimation. Here we need to note that we may be able to find other effective frequencies if we can scan above 70MHz with appropriate equipment.

5.1.3 Attack Distance

The energy of an EM signal is inversely proportional to the square of the transmitted distance [44]. In addition, the distance affects the spatial distribution of the electromagnetic field. There are mainly two types of electromagnetic fields, i.e., near field and far field, which are different in the energy distribution [36]. Near field stores the energy of the signal source and is mainly distributed near the source. Empirically, we can consider an electromagnetic field as a near field when the distance to the source is smaller than $2D^2/\lambda$, where D is the size of the antenna and λ is the wavelength. Since our attack requires the EM interference to affect a part of the screen and possess a high intensity, we keep the attack distance within the near field and as short as possible.

For example, with a 140mm antenna and a 46MHz signal frequency, a distance within 6mm to the antenna can be considered as the near field. We explore the impact of attack distance on a Google Pixel 1. The results in Figure 21 in Appendix show that as the attack distance increases from 5mm to 10mm, the injected touch points get less intense and less concentrated.

In this stage, we prepare the attack signal for effective touch point injection by studying the optimal combination of the transmitting antenna, signal frequency, and attack distance. Although we try to constrict the injected touch points into a restricted area by controlling the attack distance, the best effect we can achieve in this stage is to inject fake touches randomly along a targeted line as shown in Figure 21 in Appendix, which corresponds to the location of one or several neighboring RX electrodes close to the antenna in the touchscreen. This is because our EM signal is coupled into the RX electrodes when varying TX electrodes are driven. Note that the RX electrodes on different smartphones have different orientations, so a specific smartphone allows either vertical or horizontal excitations. In the next stage, we will study how to inject fake touch points into a smaller area, e.g., around a targeted point, and explore methods to create controllable touch events.

5.2 Creating Controllable Touch Events

Creating controllable touch events such as tapping on a specific button or swiping in a specific direction requires us to achieve a higher level of control over the injected touch points. Specifically, we need to inject touches to *a targeted point* on the screen instead of along a targeted line and be able to manipulate the injected touch points, such as their shape, location, and movement.

5.2.1 Injecting Touches to a Targeted Point

The core idea of injecting touches to a targeted point on the screen is to synchronize our interference signal with the TX scanning of touchscreen. Before elaborating on this, we probe the TX electrodes to understand how the excitation signals are sent.

Touchscreen probing: Not all smartphone manufacturer are publishing their devices touch sampling rate. Moreover, some devices dynamically change the sampling rate depending on the app displayed or other parameters (e.g., last touch point time). For these it is possible to find the sampling rate and the currently sensed position of the screen by probing the screen, as described in Appendix A.

Synchronization with TX scanning: We illustrate our idea in Figure 7. The excitation signal is sketched according to the probing results. We generate a short interfering signal with a duration T_d equal to or less than the time to scan one TX electrode T_{p1}. Suppose that the signal is coupled into an RX electrode at the time when the 2nd TX electrode is driven, then a fake touch point will appear at the intersection of the interfered RX electrode and the 2nd TX electrode. The touch point will not appear at other locations because there is no interference when other TX electrodes are driven. By setting

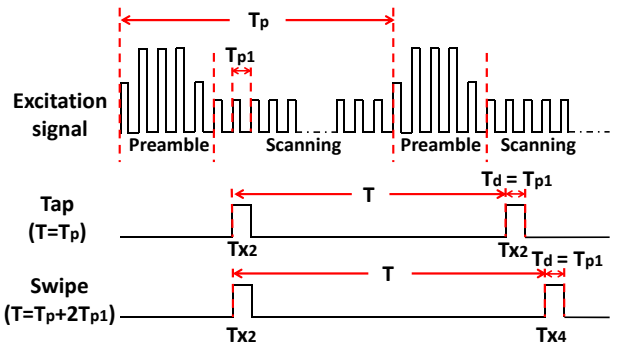


Figure 7: An illustration of synchronizing the interfering signal with the excitation signal of the touchscreen to achieve controllable taps and swipes.

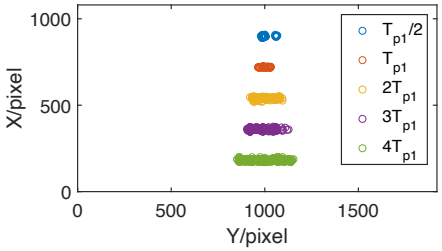


Figure 8: A visual distribution of the injected touch points on a Nexus 5X using five different transmitting durations. The touch points become more disperse as the duration increases.

the period of the interfering signal T equal to the scanning period T_p, i.e., synchronizing with the targeted TX electrode, we can make fake touch points repeatedly appear at the same point, which can be detected by the touchscreen as a single tap.

5.2.2 Manipulation of Touch Points

We can manipulate the touch points' shape, location, and movement by adjusting several waveform parameters such as the transmitting time, duration T_d, and period T.

Shape: We can control the shape by adjusting the duration T_d of the interfering signal, because as the duration increases, more TX electrodes are driven when the RX electrode is interfered, therefore making the touch points appear in a larger area. However, there is a trade-off when adjusting the duration. When the duration is too short, the intensity of the interference may be too small to affect the touchscreen. We demonstrate the ability to manipulate the shape with an experiment on Nexus 5X, where we set the duration T_d to $0.5T_{p1}$, T_{p1}, $2T_{p1}$, $3T_{p1}$, and $4T_{p1}$. Figure 8 shows the touch points' visual distribution on the screen. As we increase the duration, the injected touch points get more disperse in shape. By default, in `GhostTouch` we set the duration T_d to T_{p1}.

Location: The location of fake touch points can be modelled by the intersected RX-TX electrodes. We can inject fake touch points to any location on the screen by adjusting the timing of interference and the antenna's position. To interfere when a targeted TX electrode is driven, an attacker can either adjust the timing based on the feedback of existing locations or by prediction based on real-time touchscreen probing. To interfere with a targeted RX electrodes, the attacker can move the antenna near the RX or use an antenna array.

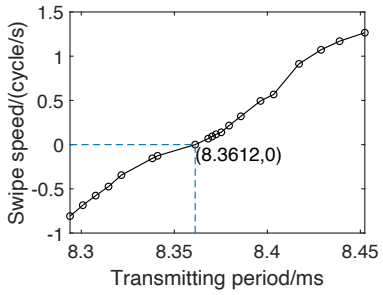
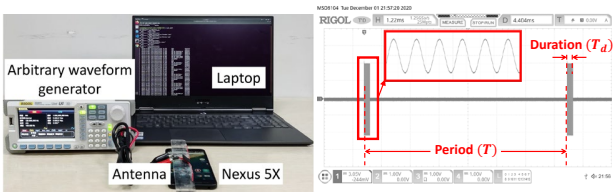


Figure 9: The speed of touch point movement changes while we adjust the transmitting period. It demonstrates the ability to move fake touches in arbitrary directions and with arbitrary speeds.

Movement: In some cases the attacker needs to move the injected touch points, e.g., to adjust the touch location or create swipes. We can easily achieve touch movement along both directions of the RX electrodes by setting the transmitting period T higher or lower than the scanning period T_p. The amount of deviation from T_p determines the speed of movement. To demonstrate the ability to move the injected points, we experiment on a Nexus 5X with varying transmitting period. We record the touch data using ADB and calculate the speed of movement. The results in Figure 9 show that we can move the fake touch points in arbitrary direction and speed.

5.2.3 Controllable Touch Events

With the above methods to generate and fine-tune the EMI signal, we are able to create controllable touch events. Specifically, in `GhostTouch` we focus on two types of basic touch events, i.e., taps and swipes. We validate injecting controllable tap, swipe-up, and swipe-down on a Nexus 5X using the experiment setup and interfering signal shown in Figure 10. The interfering signal is generated using a Rigol DG5072 arbitrary waveform generator [9] and a self-made antenna. For each type of touch event, we try 20 times and each trial lasts for 3 seconds. We are able to inject taps consistently into any area targeted on the touch screen even an area as small as 180×180 pixels in the middle of the screen with a success rate of 85%. Moreover, we can inject swipe-up and swipe-down into any part of a targeted line with success rates of both 90%. An attacker can use the injected taps and swipes to construct more complex touch behaviors for various attack scenarios, which we will show in the following.



(a) Experiment setup. (b) Interfering signal.

Figure 10: The experiment setup and the interfering signal used to validate GhostTouch attack on a Nexus 5X.

5.3 Attack Scenarios

We demonstrate the threat of GhostTouch in three practical attack scenarios that can be implemented by injecting taps and swipes.

(1) *Implant a malware.* Suppose the adversary knows the victim's phone number or messaging app account, and sends a message to the victim containing a malicious link. When the victim's phone displays a notification upon receiving the message, the adversary can use GhostTouch to tap the notification. After automatically opening the message app, the attacker then taps the malicious link to initiate a drive-by download of a malware.

(2) *Establish a malicious connection.* To establish a malicious connection, e.g., WiFi or Bluetooth, the adversary sends a request to the victim's phone or uses an NFC tag to trigger the connection, which will make the phone display a notification. The adversary can tap the notification to open the connection request window and then tap the "CONNECT" button to approve the request. After establishing the connection, the attacker can perform Man-in-the-Middle attacks or control the phone with a Bluetooth mouse.

(3) *Answer an eavesdropping phone call.* Suppose the adversary knows the victim's phone number, and calls the number and inject a swipe to answer the call on the victim's device. This enables the adversary to eavesdrop on the victim user. This attack will not raise the victim's attention when the phone is switched to silent mode, which many users would do when they are sleeping, or are at work [4] or a conference. The adversary may also prevent the phone from ringing by answering the call before the first ring.

6 GhostTouch System Evaluation

In this section, we discuss the implementation of the GhostTouch system and its evaluation.

6.1 Implementation

The GhostTouch system consists of two parts, a *touch injector* and a *phone locator*, as shown in Figure 11. The touch injector can inject touch events, e.g., a tap, a swipe, or multi-touch, into the touchscreen, and it includes a signal generator, an amplifier, an on/off switch, and a receiving antenna array. The on/off switch is used to select the correct antennas to emit the EMI signals such that it can inject touch events into the targeted RX lines. The phone locator can identify the position of the touchscreen. It consists of a sensing antenna array, a data acquisition device, and a location calculator.

Antenna Array. The antenna array is consisting of the transmitting and sensing antennas. As mentioned in Section 4, it will be placed in the appropriate location to facilitate signal emitting and sensing and attack the target device. Note that the transmitting antennas are dipole antennas because they are designed to inject touch points along the RX lines, and the sensing antennas are coil antennas because they are designed to receive signals radiating from the touchscreen.

Touch Injector. In our experiments, we implemented two types of touch injectors with different capabilities: a powerful full-fledged injector and a smaller portable one. As shown in Figure 12, the powerful injector utilizes an arbitrary waveform generator (Rigol DG5072), an amplifier (Mini-Circuits ZHL-100W-GAN+ [50]) and an on/off switch Time Relay.

The portable injector consists of a signal oscilloscope and a ChipSHOUTER [5] that integrates the amplifier and an antenna. The signal oscilloscope generates a square wave to drive the ChipSHOUTER to emit pulses of a broadband signal that covers the frequency bands proven to be effective in interfering with touchscreens. Due to the hardware limits of the ChipSHOUTER, the pulse width is limited to 80ns and 960ns, and an example pulse output of ChipSHOUTER is shown in Figure 22 in Appendix. Since the powerful injector has the flexibility of emitting different EMI signals, we utilize the powerful version for feasibility evaluation. The ChipSHOUTER injector is used to validate the attacks in real-world scenarios due to its small size and the fact that it represents a lower bound for our experiments.

Phone Locator. As mentioned above, the antenna array is part of the phone locator, and can be used in combination with both touch injectors. The sensing antennas use a data acquisition device of NI MyDAQ [32]. This device allows for measuring and analysing the radiated signals of the touchscreen and inferring the phone position relative to the sensing antenna. Note that we can reuse the hardware of the phone locator to assist attack synchronization, as described in Appendix A.

In the rest of this section, we will evaluate the single touch injection, multi-touch injection, touch injection scenarios in real world, and phone locator.

6.2 Single Touch Injection

We evaluate the performance of single touch injection, including two basic touch events, tap and swipe.

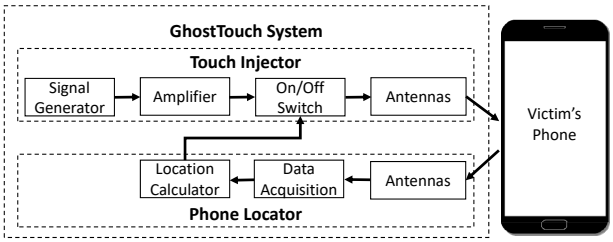
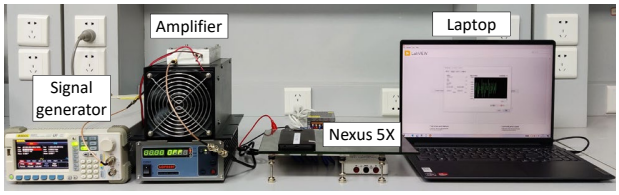
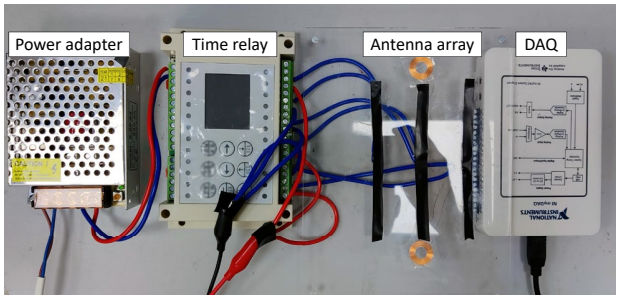


Figure 11: An illustration of the GhostTouch system. We build a touch injector to inject touch events into the touchscreen, and a phone locator to locate the victim's phone.



(a) System setup.



(b) Antenna array.

Figure 12: The high-power experiment setup of the GhostTouch system including the antenna array.

6.2.1 Experiment Methodology

Experiment setup: In this part, we use the powerful setup, as shown in Figure 12, and set the transmitting duration to be $3T_{p1}$, the transmitting period to be the same as the scanning period T_p (120Hz), and the signal frequency of the EMI to be 46MHz. The default distance between the antenna array and the screen is 5mm. The targeted smartphones (Nexus 5X by default) are connected to a laptop for touch injection recording.

Data collection: To quantify the attack results, we need to obtain the touch event data, e.g., the timestamps and locations of the injected points. We can acquire the data either from an Android application or by using Android Debug Bridge (ADB), a command-line tool that can communicate with Android devices.

Metrics: We evaluate the performance of GhostTouch from two perspectives: the similarity to real human touches and the attack capability. The injected touch points and real human touch points may vary mainly in two aspects, the shape and concentration of touch points, which can be quantified by the following metrics:

1. *Range*, which is the difference between the maximum and minimum of the injected points' X/Y coordinates. It describes the shape of the injected points.

2. *Standard deviation of the X/Y coordinates*, which describes the concentration of the injected points.

Metrics to evaluate the attack capability includes the injection speed, attack delay and success rate:

1. *Injection speed*, which is the number of injected points in unit time.

2. *Attack delay*, which is the time it takes to inject the first successful touch event since the attack starts.

3. *Success rate*, which is the proportion of attacks that successfully inject the targeted touch event.

6.2.2 Tap

We use the powerful experiment setup, to evaluate the performance of injecting targeted taps. We set the transmitting period to T_p and repeat the experiments 100 times with each time lasting for 3 seconds. Since the T_p is drifting over time, we measure it constantly and adjust the transmitting period. After recording the data of the injected points by ADB, we calculate the range of x and y coordinates, the standard deviation of the coordinates, and the injection speed. We show the samples of 'taps' in Figure 23(a) in Appendix.

Similarity between injected points and real touch events: The metrics of the injected points are shown in Figure 13. The mean of $range_x$ and $range_y$ are 36.3 pixels and 175.8 pixels, respectively, and the mean of std_x and std_y are 5.4 pixels and 44.0 pixels, respectively.

To compare our GhostTouch with real touch events, we recruited 30 volunteers of three professions (students, professors, administrative staff), including 8 females and 22 males aged between 20 and 50 to tap the 'Home' button on the Nexus 5X, using the thumb and forefinger 30 times each. Then we randomly select 1800 samples from the volunteers' taps and the taps injected by the attacker, respectively. The comparison between the GhostTouch and the real touch events are shown in Figure 14. Compared with human taps, the injected taps are distributed in a smaller range on the x-axis, and distributed in a larger range on the y-axis. However, the difference between the injected taps and user's taps is very small and hence not distinguishable from the source.

Attack capability: (1) Injection speed: According to Figure 13, the mean of the injecting speed is 45.38 point/s. Considering the maximum human touch speed is about 7 points/s,

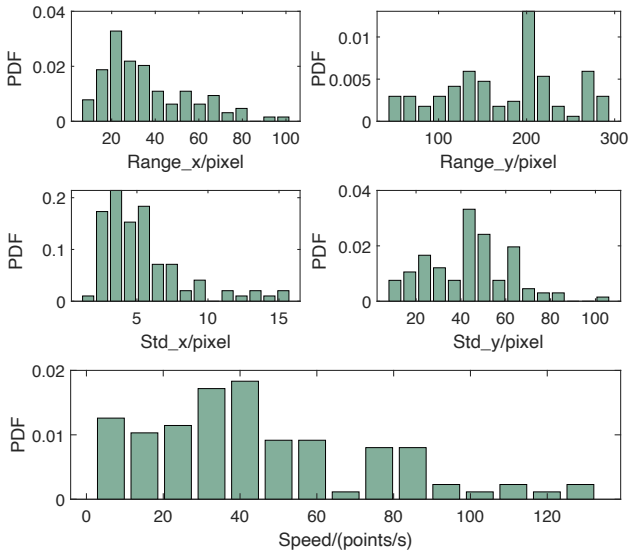


Figure 13: The performance of injecting taps in 100 trials, including the range of the x and y coordinates (top), the standard deviation of the coordinates (middle), and the injection speed (bottom).

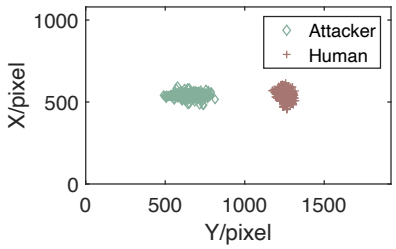


Figure 14: A comparison of the similarity between the taps from 30 volunteers and the ones injected by the attacker. Compared with real touch events, the taps injected by the attacker are distributed in a smaller range on the x-axis, and distributed in a larger range on the y-axis.

this injection speed can satisfy the attacker's requirements. (2) Consistency means how long the injected points will stay in one position, which is important because an adversary may need to tap the same point repeatedly for a few seconds. For example, we inject 'taps' into Nexus 5X's touchscreen for 15 seconds and the injected points stay within a small area for 15 seconds. Detailed results are reported in Appendix (Figure 24). (3) The attack delay is less than 0.5 seconds.

6.2.3 Swipe

We use the experiment setup as shown in Figure 12(a) for evaluation. By slightly changing the transmitting period T around T_p, we can 'swipe up' or 'swipe down'. For a sample

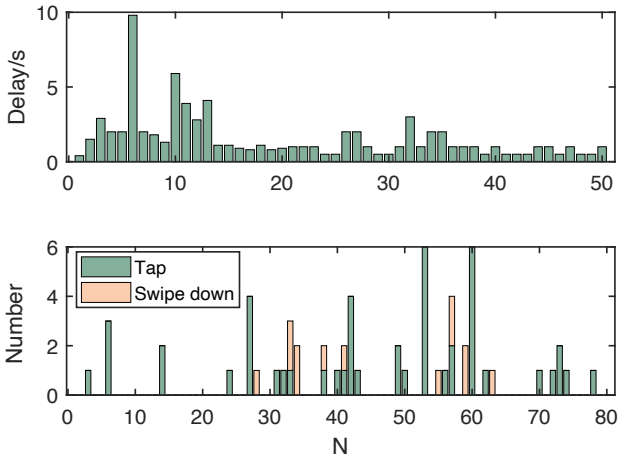


Figure 15: The performance of injecting a swipe-up in 80 trials, including the time delay until a successful swipe-up is injected (top) and the number of unintended touch events, e.g., taps or swipe-downs (bottom).

of injected swipes refer to Figure 23(b) in Appendix. Similar to the results shown in Figure 9, the direction and speed of the swipe are related to the difference between T and T_p.

We evaluate the attack delay and success rate of injecting swipes by setting the transmitting period T to $7.7ms$ and repeatedly trying to swipe up. It is counted as a success if there are no taps or swipes down until the first swipe up on Nexus 5X as our showcase. (1) To calculate the success rate, we measure the number of false events before the first successful swipe up, e.g, taps and swipes-down. The number of false events are 30 out of 80 times, which leads to the success rate of 62.5% (50/80). (2) Attack delay: It takes 1.6 seconds on average to inject a successful swipe up. Detailed results are shown in Figure 15.

6.3 Multi-touch Injection

Multi-touch gestures have become a popular input operation, they can either be multiple simultaneous touches or multiple swipes at different locations of the screen. The most robust way of realizing multi-touch injections is to inject touches or swipes into different RX lines at the same time. To inject multiple touches at the touchscreen, we utilize an antenna array, as shown in Figure 12(b). The on/off switch will choose multiple antennas over the right locations to emit the EMI signal, with each chosen antenna being coupled with the corresponding RX lines. Thus, the attacker can successfully inject taps along the targeted RX lines simultaneously and achieve multi-touch injection. We validate the feasibility of injecting three touch points using three antennas, and the performance results of multi-touch injection is similar to the attack capabilities and properties using one antenna (c.f. Section 6.2).

6.4 Touch Injecting Scenarios

Experiment Setup: To demonstrate the threat of injecting taps and swipes, we illustrate three attack scenarios conducted with the ChipSHOUTER touch injector setup and one scenario using the powerful setup with an antenna array. We use a ChipSHOUTER device to generate pulses by charging to $500V$ and discharging its capacitors. Although the shape of the pulses is fixed, we can adjust how often to emit a pulse (i.e., pulse periods) by adjusting the frequency of the square wave that drives the ChipSHOUTER. In our experiments, we set the square wave signal with 20% duty cycle when the transmitting period equals the scanning period T_p (120Hz), and pulse width is set to 350ns.

Answering the phone call: We inject swiping actions in the middle line on the Nexus 5X to answer the phone call. The tip of the ChipSHOUTER was positioned 5mm over the middle of the screen. When the phone is called, we transmit the EMI signal with a transmitting period of 130Hz to swipe up. As a result, we answer the phone call successfully in all the 10 tests and it takes about 4.1 seconds on average, with a max of 6 seconds and a min of 2 seconds.

Pressing the button: We inject a tap into a certain button on the screen to press this button. We implemented an Android app. It displays a button oriented on the middle of the X-axis and with $77dp$ distance to the right side of the screen, where normally "OK" or "Accept" buttons would be displayed. The button is sized at $36dp$ height and $80dp$ width. The app collects all taps not on the button and stops when the button is pressed for the first time. We evaluated this attack on the Nexus 5X using the ChipSHOUTER, and the tip was hovering 5mm over the bottom of the screen. With an injection frequency of 120Hz, it took 7.5 seconds for the injected touch points to press the button. 11.3 taps were injected wrongly until the next tap hit the button.

Swiping up to unlock: We inject swiping actions in the middle line on the Nexus 5X. For the Nexus 5X device, its lock screen has a "swipe up to unlock" mechanism. The tip of the ChipSHOUTER was positioned 5mm over the middle of the screen. After a proper pulse frequency for injecting swipes was found (130Hz), injecting a swipe to unlock 20 times took 8.5 seconds on average with a minimum time of 1s and a maximum of 20s.

Entering a password: Once the attacker acquires the information of the password either by shoulder surfing or social engineering, she can utilize the touch injector to unlock the phone. We first select the correct antennas that are close to the target areas and adjust the timing to emit EMI signals such that we can 'press' the desired numbers in the virtual keyboard. As test cases we picked two PIN codes 3699 and 9999 to test. We were able to enter the PIN codes successfully in about 20s and 1s, respectively. It is possible to enter any PIN or password with `GhostTouch`, yet complex passwords will require extra time to accomplish successful injection.

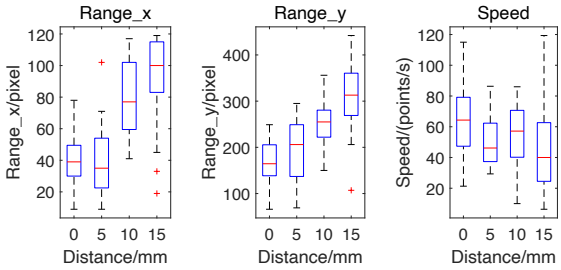


Figure 16: An illustration of the impact of the attack distance. The injection speed tends to decline as the distance increases.

6.5 Factors Affecting Touch Injection

We evaluate the factors that may affect the performance of touch injector, including attack distance, the phone model, ChipSHOUTER coil buzzing, and wireless charging.

Attack distance. We evaluate the impact of the distance between the transmitting antenna and the touchscreen using the setup shown in Figure 12(a), which can output a high-intensity EMI. The devices are attacked 40 times at a range of 0 to 15mm lasting 3 seconds each. Subsequently, the range of the injected points' x/y coordinate and the injecting speed is calculated. The results are presented using a box plot from Matlab, as shown in Figure 16. The central mark of each box indicates the median, and the bottom and top edges respectively indicate the 25th and 75th percentiles. The outliers are plotted individually using the '+' symbol, and the whiskers extend to the most extreme data points except for outliers.

According to the results, the injecting speed tends to decline as the distance increases, which is according to the attenuation of the EMI. We were capable to achieve a distance of up to 40mm.

Phone models. We evaluate the `GhostTouch` attacks on 11 phone models, using the portable setup with the ChipSHOUTER. We set the attack distance to 6mm for all phones for comparison. Broadly distributed touch points can be injected into 9 of these smartphones. We therefore explore whether we can realize the GhostTouch attack on these phones, and once successful we evaluate the performance by injecting taps for 4 seconds each. We calculate the injection speed and the standard deviation of the injected points' x/y coordinates. We record the direction of the swipe for each phone. According to the results, as shown in Table 2, we can inject touch points at chosen positions for 6 out of 11 phone models with GhostTouch attacks and therefore they are vulnerable to the attack scenarios described in Section 6.4. Since the injecting speed for Huawei Honor View 10 is low, we extend the experiment duration of injecting taps to 40 seconds and calculate the average result. For Galaxy S20 FE 5G and iPhone SE (2020), our approach can inject touch points successfully and perform malicious operations, but not always with high precision. Such a vulnerability is still dangerous,

Table 2: Attack performance on 11 phone models using the ChipSHOUTER. Nine phones are vulnerable to the attack, on six we can inject touch points precisely.

Phone model	Success	Inje. Speed	Direction	Std/pixel		Std/mm	
				X	Y	X	Y
Galaxy A10s	✓	3.5	Vertical	158.9	111.9	14.9	10.5
Huawei P30 Lite	✓	2.0	Vertical	182.0	189.0	11.1	11.6
Honor View 10	✓	0.3	Vertical	41.4	4.9	1.3	0.6
Huawei Mate 40 Pro	×	N/A	N/A	N/A	N/A	N/A	N/A
Galaxy S20 FE 5G	(✓)	2.8	Vertical	N/A	N/A	N/A	N/A
iPhone 12	×	N/A	N/A	N/A	N/A	N/A	N/A
iPhone SE (2020)	(✓)	1.0	Vertical	N/A	N/A	N/A	N/A
Nexus 5X	✓	8.2	Vertical	14.6	19.2	0.9	1.1
Redmi Note 9S	✓	2.5	Horizontal	73.3	210.2	4.7	13.5
Nokia 7.2	✓	8.7	Vertical	36.5	156.3	2.3	9.9
Redmi 8	(✓)	1.5	Horizontal	N/A	N/A	N/A	N/A

to illustrate, we managed to establish a malicious Bluetooth connection on iPhone SE (2020), with an average delay of 7.1 seconds.

ChipSHOUTER coil buzzing. The ChipSHOUTER while generating a wide range signal and due to its small form factor is emitting a high-pitched audible coil buzzing noise. We measure this buzzing noise using the Benetech GM1357 [12]. It is 44dB right next to the ChipSHOUTER, 42dB 20cm above the table under which the ChipSHOUTER is placed, and 38dB 30cm above the table. Note that the buzz of a refrigerator is about 40dB. While this noise is audible when in close proximity in a silent room, it is too faint to hear in a crowded place (e.g., conference hall, cafe).

Wireless charging. Our attack can be successfully launched in the same manner while the device is being charged. The device's touchscreen is still fully functional during wireless charging, as the smartphone components shield the touchscreen from the magnetic field. Further, wireless chargers usually operate at $130-175kHz$, insufficient to couple to an RX electrode.

6.6 Phone Locator

To inject touch points precisely, the attacker needs to know how the victim's phone is placed. We implement the phone locator as shown in Figure 12(b). It consists of sensing antennas and a NI MyDAQ. In practice we can place a matrix of sensing antennas to locate the phone positions with the following observation: a sensing antenna over the touchscreen can detect the radiated signals (e.g., the leaked signals of the TX excitation signal of the touchscreen at a frequency of 120Hz), while the ones away from the touchscreen will detect a weaker signal or none at all due to attenuation. Thus, the sensing antennas are used to receive the leaked signals, which are processed by the NI MyDAQ to deduce the phone location and orientation, with respect to the sensing antennas. After obtaining the phone position, we can infer the positions of the buttons based on the operating system and target application, e.g., Bluetooth connection accept dialog.

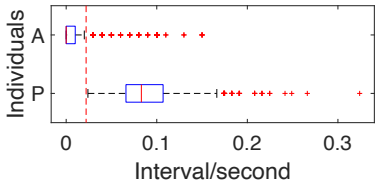


Figure 17: Touch interval of 30 volunteers (P) and a `GhostTouch` attacker (A). The touch events from a human and an attacker can be separated by setting a threshold, represented by the dotted line.

7 Potential Countermeasures

Our results showed that certain smartphones are less vulnerable to the `GhostTouch` attack, which could be due to better electromagnetic shielding or effective validation. Inspired by this, we propose three categories of countermeasures to mitigate the threat of `GhostTouch` attack:

Reinforcement: Manufacturers may reinforce the touchscreen to protect it from the threat of `GhostTouch` attack. First, adding electromagnetic shielding is effective to block EMI. Second, increasing the voltage of the excitation signal may increase the SNR, which will mitigate the influence of EMI. Third, a driving method based on a more sophisticated excitation signal waveform may be used to filter out injected current by EMI. Although these countermeasures could reduce the impact of EMI on the touch screen, they require modifications to the hardware of the touchscreen or lead to higher energy consumption. Thus, these countermeasures are not suitable for existing touchscreens.

Detection algorithm: The manufacturer can improve the detection algorithm of the touchscreen. For example, the `GhostTouch` attack can be detected utilizing the touch interval between pressing and lifting the finger. To explore the effectiveness of this method, we calculate the touch interval of a `GhostTouch` attacker and 30 volunteers as in Section 6.2.2. We randomly choose 2000 samples from the attacker and volunteers each and analyze the distribution of the touch interval using a box plot from Matlab. As shown in Figure 17, the upper adjacent of the attacker is lower than the lower adjacent of the volunteers. We can set a threshold, e.g., the red line in Figure 17, to identify whether the touch point belongs to a user or an attacker.

Moreover, the capacitance distribution shows a certain pattern when the touchscreen is under our `GhostTouch` attack opposed to being used by a human. Based on these facts, the manufacturer could detect abnormal touch points, reject them and warn the user.

Identity verification: Application permissions may be re-

stricted and identity verification needs to be conducted when executing high-risk actions. For example, conduct identity verification before connecting to a Bluetooth device or an unknown WiFi. Identity verification can be realized by requesting the user to verify his fingerprint, face or provide his PIN or password.

8 Related Work

In the following, we provide a summary of the existing attacks on touchscreens as well as attacks utilizing electromagnetic interference (EMI).

EMI attack: There have been several studies on EMI attacks over the last years. EMI attacks are used for Denial of Service (DoS), injection of false information into sensing circuits, or to glitch computations. Sabath et al. [37] launched a jamming attack using high-power EMI, which causes degradation or loss of the main function of critical electronic systems. Hayashi et al. [14] showed that electrical devices with cryptographic modules are vulnerable to electromagnetic interference. Schmidt et al. [38] showed that it is possible to induce faults into cryptographic systems using EMI and therefore breaking RSA. Dehbaoui et al. [8] showed that it is possible to inject faults into hardware and software implementations of AES using EMI. O'Flynn et al. [34] used EMI to force a hardware keystore to leak sensitive data by precisely manipulating packets sent over the USB stack. Kune et al. [24] investigated the susceptibility of analog sensors to EMI attacks and implemented the EMI attack against implantable cardiac electronic devices and consumer electronic devices containing microphones. Selvaraj et al. [39] presented an EMI attack which can cause bit flips or inject false actuation signals in embedded systems. Giechaskiel et al. [11] demonstrated the threat of EMI attack by injecting malicious commands into a smartphone. There have been other signal injection attacks on sensors [19, 42, 45, 48, 49] and defense mechanisms [46, 47]. Our attack `GhostTouch` takes another approach and utilizes the observation that EMI can induce current flow into a sensing circuit. We focus on capacitive touchscreen and their sensing mechanism to inject wrongly recognized touches into the touch panel. Capacitive touchscreens have more complex structures and mechanisms. Furthermore, capacitive touchscreens of smartphones have been tested for electromagnetic compatibility, and therefore, it is significantly more challenging to launch a fake touch injection attack on them.

Attacks on touchscreen: Research on the security of touchscreens can be divided into two categories, passive eavesdropping and active spoofing attacks. In passive eavesdropping, the adversary infers the input of a touchscreen using side-channel information. In prior work, Maggi et al. [29] took the image from surveillance as side channel information to get the keystrokes of a victim. Aviv et al. [3] leveraged smudge to get the graphical password. Hayashi et al. [15] leveraged the electromagnetic signal emanations of a tablet display to reconstruct the displayed screen image.

Active spoofing attacks may modify software or hardware like the work proposed by Schwartz et al. [40], modifying the touch display driver to inject false touch points. Attacks like the one described by Maruyama et al. [30] use EMI to attack the touchscreen controller. However, the attack needs the victim to touch the panel while the attack is active and thus could be easily perceived by the victim. Our attack does not require the victim to touch the panel. Moreover, they could not control the position of the injected touch points. Approximately, the injected points are uniformly scattered along a line which is the RX electrode touched by a finger.

In contrast our attack `GhostTouch` takes a Dupont jumper wire or a 4mm tip as the antenna. By changing the position of the antenna, the adversary could control which line the touch points are injected into. By shaping the EMI signal, the adversary could control which segment of this line the points are injected into.

9 Conclusion

In this paper, we introduced a novel attack coined `GhostTouch`, which targets the capacitive touchscreen used on many mobile devices such as smartphones or tablets. `GhostTouch` controls and shapes the near-field electromagnetic signal, and injects touch events into the targeted area on the touchscreen, without the need for physical touch or access to the victim's device. Consequently, the adversary can stealthily manipulate the victim's smartphone. Through the extensive experiments and evaluation, we demonstrate that our `GhostTouch` attack works for most widely-used smartphones. Moreover, we discuss possible countermeasures against the `GhostTouch` attack.

Acknowledgments

We thank the anonymous reviewers for their valuable comments. This work is supported by China NSFC Grant 61925109, 61941120, and 62071428, as well as, the German Research Foundation (DFG) within CRC 1119 CROSSING (S2 and P3), the Intel Private AI Center and Huawei Open Lab for Sustainable Security and Safety (openS3Lab).

References

[1] 3ctest EDS 20H. http://www.3ctest.cn/product/show/178.

[2] Adey64. Nexus 5 has a 120hz touch controller. https://forum.xda-developers.com/t/nexus-5-has-a-120hz-touch-controller.2559505/, 2013.

[3] Adam J Aviv, Katherine L Gibson, Evan Mossop, Matt Blaze, and Jonathan M Smith. Smudge attacks on smartphone touch screens. *Woot*, 10:1–7, 2010.

[4] Yung-Ju Chang and John C Tang. Investigating mobile users' ringer mode usage and attentiveness and responsiveness to communication. In *Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services*, pages 6–15, 2015.

[5] ChipSHOUTER. https://www.newae.com/chipshouter.

[6] Dankgus. Touchscreen interference issue, confirmed–weird! https://www.surfaceforums.net/threads/touchscreen-interference-issue-confirmed-weird.14810/, 2015.

[7] Chris Daskalou. How conducted emi influences touch sensors. https://fieldscale.com/blog/emi-noise-touch-sensors/, 2020.

[8] Amine Dehbaoui, Jean-Max Dutertre, Bruno Robisson, and Assia Tria. Electromagnetic transient faults injection on a hardware and a software implementations of AES. In *Proceedings of 2012 Workshop on Fault Diagnosis and Tolerance in Cryptography*, pages 7–15, 2012.

[9] Rigol DG5072. https://www.batronix.com/shop/waveformgenerator/Rigol-DG5072.html.

[10] Embedded. Understanding electromagnetic interference sources in touchscreens. https://www.embedded.com/understanding-electromagnetic-interference-sources-in-touchscreens/, 2011.

[11] Ilias Giechaskiel, Youqian Zhang, and Kasper B Rasmussen. A framework for evaluating security in the presence of signal injection attacks. In *Proceedings of European Symposium on Research in Computer Security*, pages 512–532, 2019.

[12] Benetech GM1357. https://www.i-tech.com.au/benetech-gm1357-digital-sound-level-meter-gm-1357-55795.html.

[13] Chelsea Gohd. The touchscreen controls of spacex's crew dragon give astronauts a sci-fi way to fly in space. https://www.space.com/spacex-crew-dragon-touchscreen-astronaut-thoughts.html, 2020.

[14] Yu-ichi Hayashi, Naofumi Homma, Takeshi Sugawara, Takaaki Mizuki, Takafumi Aoki, and Hideaki Sone. Non-invasive EMI-based fault injection attack against cryptographic modules. In *Proceedings of 2011 IEEE International Symposium on Electromagnetic Compatibility*, pages 763–767, 2011.

[15] Yuichi Hayashi, Naofumi Homma, Mamoru Miura, Takafumi Aoki, and Hideaki Sone. A Threat for Tablet PCs in Public Space: Remote Visualization of Screen Images Using EM Emanation. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 954–965, 2014.

[16] Steve Hotelling, Joshua A Strickon, and Brian Q Huppi. Multipoint touchscreen, February 16 2010. US Patent 7,663,607.

[17] Steven P Hotelling, Christoph H Krah, and Brian Quentin Huppi. Multipoint touch surface controller, October 2 2012. US Patent 8,279,180.

[18] IDC. Smartphone market share. https://www.idc.com/promo/smartphone-market-share/vendor, 2021.

[19] Xiaoyu Ji, Yushi Cheng, Yuepeng Zhang, Kai Wang, Chen Yan, Wenyuan Xu, and Kevin Fu. Poltergeist: Acoustic adversarial machine learning against cameras and computer vision. In *Proceedings of 2021 IEEE Symposium on Security and Privacy (SP)*, pages 160–175, 2021.

[20] Paul K. Do you ever set your phone face-down on the table? https://www.phonearena.com/news/Poll-Do-you-ever-set-your-phone-face-down-on-the-table_id88055, 2016.

[21] Bernhard E Keiser. *Principles of electromagnetic compatibility*. Artech House, 1987.

[22] Abdul Qadir Khan, Muhammad Riaz, and Anas Bilal. Various types of antenna with respect to their applications: a review. *International Journal of Multidisciplinary sciences and Engineering*, 7(3):1–8, 2016.

[23] Hans W Klein. Noise immunity of touchscreen devices. *Cypress Semiconductor Corporation, White Paper*, 2013.

[24] Denis Foo Kune, John Backes, Shane S Clark, Daniel Kramer, Matthew Reynolds, Kevin Fu, Yongdae Kim, and Wenyuan Xu. Ghost talk: Mitigating emi signal injection attacks against analog sensors. In *Proceedings of 2013 IEEE Symposium on Security and Privacy*, pages 145–159, 2013.

[25] O. Kwon, J. An, and S. Hong. Capacitive touch systems with styli for touch sensors: A review. *IEEE Sensors Journal*, 18(12):4832–4846, 2018.

[26] FOCUS LCDs. Touch screens for use in medical instrument displays. https://focuslcds.com/journals/touch-screens-for-use-in-medical-instrument-displays/, 2019.

[27] Jeffrey Lee, Matthew T Cole, Jackson Chi Sun Lai, and Arokia Nathan. An analysis of electrode patterns in capacitive touch screen panels. *Journal of display technology*, 10(5):362–366, 2014.

[28] Leelauer. Why does my touch screen go crazy while charging? https://forums.androidcentral.com/google-nexus-7-tablet-2012/497397-why-does-my-touch-screen-go-crazy-while-charging.html, 2019.

[29] Federico Maggi, Alberto Volpatto, Simone Gasparini, Giacomo Boracchi, and Stefano Zanero. A fast eavesdropping attack against touchscreens. In *Proceedings of 2011 7th International Conference on Information Assurance and Security (IAS)*, pages 320–325, 2011.

[30] Seita Maruyama, Satohiro Wakabayashi, and Tatsuya Mori. Tap'n Ghost: A Compilation of Novel Attack Techniques against Smartphone Touchscreens. In *Proceedings of 2019 IEEE Symposium on Security and Privacy*, pages 620–637, 2019.

[31] Manisha Mathur, JK Rai, and N Sridhar. Electromagnetic compatibility analysis of projected capacitive touch technology based panel computer for military application. *Journal of ElEctromagnEtic WavEs and applications*, 30(13):1689–1701, 2016.

[32] NI MyDAQ. https://www.ni.com/pdf/manuals/373060g.pdf.

[33] NBD. The cell phone being charged automatically booked a ten thousand yuan presidential suite and checked the chat history. http://www.nbd.com.cn/articles/2018-10-08/1260630.html, 2018.

[34] Colin O'Flynn. MIN () imum Failure: EMFI Attacks against USB Stacks. In *Proceedings of 13th USENIX Workshop on Offensive Technologies*, 2019.

[35] Electronic Products. Shielding touchscreens from emi. https://www.electronicproducts.com/shielding-touchscreens-from-emi/#, 2008.

[36] Jean-Michel Redouté and Michiel Steyaert. *EMC of analog integrated circuits*. Springer Science & Business Media, 2009.

[37] Frank Sabath. What can be learned from documented Intentional Electromagnetic Interference (IEMI) attacks? In *Proceedings of 2011 URSI General Assembly and Scientific Symposium*, pages 1–4, 2011.

[38] Jörn-Marc Schmidt and Michael Hutter. *Optical and EM Fault-Attacks on CRT-based RSA: Concrete Results*. Verlag der Technischen Universität Graz, 2007.

[39] Jayaprakash Selvaraj, Gökçen Yılmaz Dayanıklı, Neelam Prabhu Gaunkar, David Ware, Ryan M Gerdes, and Mani Mina. Electromagnetic induction attacks against embedded systems. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, pages 499–510, 2018.

[40] Omer Shwartz, Amir Cohen, Asaf Shabtai, and Yossi Oren. Shattered trust: When replacement smartphone components attack. In *Proceedings of 11th USENIX Workshop on Offensive Technologies (WOOT 17)*, 2017.

[41] Slane35. Touchscreen problems while charging. https://forum.xda-developers.com/showthread.php?t=1784773, 2012.

[42] Timothy Trippel, Ofir Weisse, Wenyuan Xu, Peter Honeyman, and Kevin Fu. Walnut: Waging doubt on the integrity of mems accelerometers with acoustic injection attacks. In *Proceedings of 2017 IEEE European Symposium on Security and Privacy (EuroS P)*, pages 3–18, 2017.

[43] User1950278. Glitchy touchscreen caused by charger [closed]. https://electronics.stackexchange.com/questions/77631/glitchy-touchscreen-caused-by-charger, 2013.

[44] Martin Weik. *Computer science and communications dictionary*. Springer Science & Business Media, 2000.

[45] Wenyuan Xu, Chen Yan, Weibin Jia, Xiaoyu Ji, and Jianhao Liu. Analyzing and enhancing the security of ultrasonic sensors for autonomous vehicles. *IEEE Internet of Things Journal*, 5(6):5015–5029, 2018.

[46] Zhijian Xu, Guoming Zhang, Xiaoyu Ji, and Wenyuan Xu. Evaluation and defense of light commands attacks against voice controllable systems in smart cars. *Noise & Vibration Worldwide*, 52(4-5):113–123, 2021.

[47] Chen Yan, Hocheol Shin, Connor Bolton, Wenyuan Xu, Yongdae Kim, and Kevin Fu. Sok: A minimalist approach to formalizing analog sensor security. In *Proceedings of 2020 IEEE Symposium on Security and Privacy (SP)*, pages 233–248, 2020.

[48] Chen Yan, Guoming Zhang, Xiaoyu Ji, Tianchen Zhang, Taimin Zhang, and Wenyuan Xu. The feasibility of injecting inaudible voice commands to voice assistants. *IEEE Transactions on Dependable and Secure Computing*, 18(3):1108–1124, 2021.

[49] Guoming Zhang, Chen Yan, Xiaoyu Ji, Tianchen Zhang, Taimin Zhang, and Wenyuan Xu. Dolphinattack: Inaudible voice commands. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 103–117, 2017.

[50] Mini-Circuits ZHL-100W-GAN+. https://www.minicircuits.com/pdfs/ZHL-100W-GAN+.pdf.

Appendix

A Screen Refresh Rate Probing

Probing the excitation signals directly on the TX electrodes is difficult because the pins are insulated and hidden inside the touchscreen. To overcome this problem, we infer the excitation signals by measuring their electromagnetic emissions. From the probed signal, we can acquire the number of TX electrodes N, the time it takes to scan all TX electrodes T_p, and the time it takes to scan one TX electrode T_{p1}. For example, Figure 18 shows a waveform of the excitation signal we measured on a Nexus 5X. It shows that it takes approximately $T_p = 8.6$ms to finish scanning all TX electrodes, which consists of a preamble and a scanning segment. Within the scanning segment, there are 27 pulses, which corresponds to the $N = 27$ TX electrodes being scanned in turn, and it takes $T_{p1} = 0.27$ms to scan one TX electrode. The scanning period T_p is the reciprocal of the touch sampling rate of the touchscreen. Our measurement therefore suggests the touch sampling rate of the Nexus 5X to be around 120Hz [2].

By counting the pulses, it is possible to synchronize the emission of the electromagnetic signal to the RX electrode which is currently sensed. For that we used a MyDAQ for signal filtering and preamble extracting, it then outputs a signal to trigger the emission of our EMI signal. Hence, it is also possible to use the same antenna to receive passively before the attack.

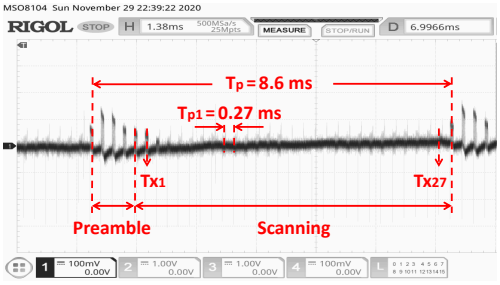
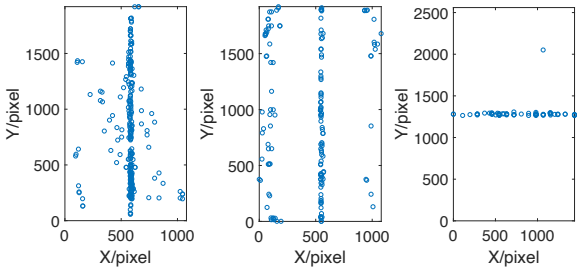


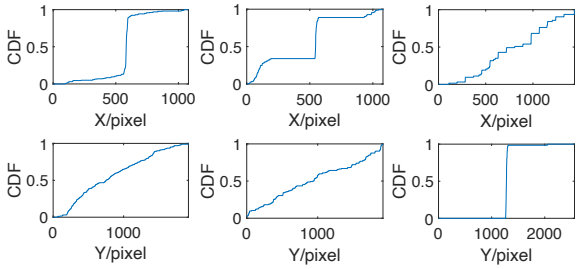
Figure 18: A waveform of the TX excitation signals on a Nexus 5X recorded by an oscilloscope. It shows that it takes approximately 8.6ms to scan all 27 TX electrodes and 0.27ms to scan one TX electrode.

B Feasibility



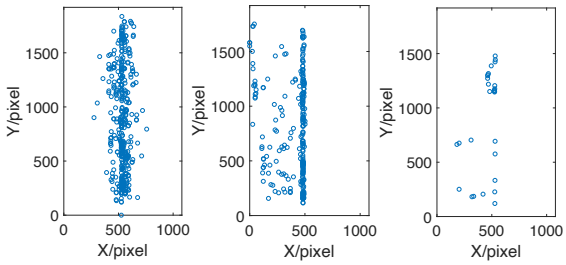
(a) Google Pixel 1. (b) Nexus 5X. (c) Huawei P10 Plus.

Figure 19: Visual distribution of the injected touch points on a Google Pixel 1, Nexus 5X and Huawei P10 Plus in the portrait orientation. The X and Y axes refer to the horizontal and vertical edges of the screen.



(a) Google Pixel 1. (b) Nexus 5X. (c) Huawei P10 Plus.

Figure 20: Cumulative distribution of the injected points along the X (1st row) and Y (2nd row) axes of the screen on Google Pixel 1, Nexus 5X, and Huawei P10 Plus. More than half of the injected points distribute on a specific line of the screen, either vertical or horizontal depending on the phone model.



(a) 5mm. (b) 7mm. (c) 10mm.

Figure 21: Visual distribution of the injected touch points on Google Pixel 1 at three attack distances. The injected touch points become less intense and less concentrated as the attack distance increases.

C ChipSHOUTER's Pulse

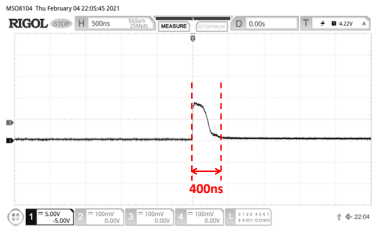
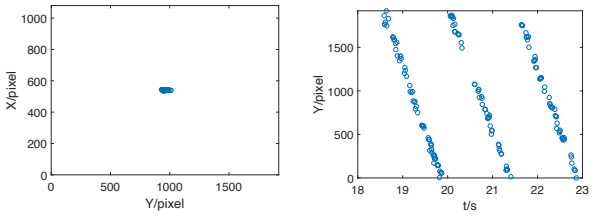


Figure 22: A waveform of the pulse generated by the ChipSHOUTER.

D Taps and Swipes



(a) Tap. (b) Swipe.

Figure 23: Illustrations of the injected taps on the screen and an injected swipe by drifting the touch points over time.

E Consistency of Taps

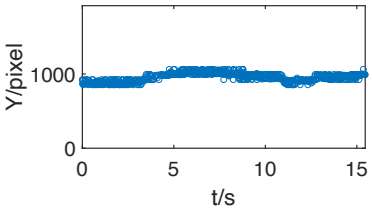
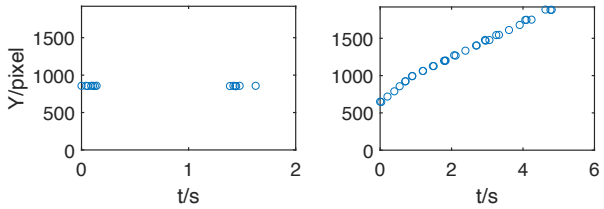


Figure 24: An illustration of the consistency of tap locations. The injected points stay in a small area for 15 seconds.

F Samples with an 8mm-thick Table



(a) Tap (b) Swipe

Figure 25: Tap and swipe injected beneath an 8mm-thick table.

G Accidental Touch Protection

Many smartphones have a "Mistouch Prevention Mode" also called "Pocket Mode", which turns off the touchscreen, prevents it from turning on or disables input in order to prevent accidental touches when the proximity sensor detects that the screen is blocked. This function may affect the practicality of the `GhostTouch` attack: 1) For phone models like, e.g., Google Pixel 1, Nexus 5X, MIX2, this function is only supported in calls. 2) For iPhones, the function turns off the screen during calls or prevents waking the screen on a notification. 3) For devices like Huawei P40, it would prevent touch detection when the proximity sensor is covered. Case 1) will not affect the `GhostTouch` attack. For case 2), it can still answer an eavesdropping call. For case 3), for Huawei phones this mode is turned off by default, which results in not many devices having this mode enabled. In addition, it is still possible to answer the phone call e.g., on Samsung Galaxy Note10, and the touchscreen can be activated by completing the check shown on the screen, e.g., by swiping twice or dragging the 'lock' icon. Considering the global market shares of Huawei and Apple smartphones are respectively 14.6% and 11.8% in the third quarter of 2020 [18], a significant proportion of smartphones are exposed to the threat of the `GhostTouch` attack.

E

Don't Shoot the Messenger: Localization Prevention of Satellite Internet Users (IEEE S&P'24, CORE: A*)

IEEE's copyright prohibits the publication of *"the final published version"*. Hence, attached is the *"accepted version"*.

[77] David Koisser, Richard Mitev, Marco Chilese, and Ahmad-Reza Sadeghi. Don't shoot the messenger: Localization prevention of satellite internet users. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 426–444. IEEE Computer Society, 2023. https://www.computer.org/csdl/proceedings-article/sp/2024/313000a066/1RjEaCUhdxm.

Don't Shoot the Messenger: Localization Prevention of Satellite Internet Users

David Koisser
Technical University of Darmstadt
david.koisser@trust.tu-darmstadt.de

Richard Mitev
Technical University of Darmstadt
richard.mitev@trust.tu-darmstadt.de

Marco Chilese
Technical University of Darmstadt
marco.chilese@trust.tu-darmstadt.de

Ahmad-Reza Sadeghi
Technical University of Darmstadt
ahmad.sadeghi@trust.tu-darmstadt.de

***Abstract*—Satellite Internet plays an increasingly important role in geopolitical conflicts. This notion was affirmed in the Ukrainian conflict escalating at the beginning of 2022, with the large-scale deployment of the Starlink satellite Internet service which consequently demonstrated the strategic importance of a free flow of information. Aside from military use, many citizens publish sensitive information on social media platforms to influence the public narrative. However, the use of satellite communication has proven to be dangerous, as the signals can be monitored by other satellites and used to triangulate the source on the ground. Unfortunately, the targeted killings of journalists have shown this threat to be effective. While the increasing deployment of satellite Internet systems gives citizens an unprecedented mouthpiece in conflicts, protecting them against localization is an unaddressed problem.**

To address this threat, we present AnonSat, a novel scheme to protect satellite Internet users from triangulation. AnonSat works with cheap off-the-shelf devices, leveraging long-range wireless communication to span a local network among satellite base stations. This allows rerouting users' communication to other satellite base stations, some distance away from each user, thus, preventing their localization. AnonSat is designed for easy deployment and usability, which we demonstrate with a prototype implementation. Our large-scale network simulations using real-world data sets show the effectiveness of AnonSat in various practical settings.

1. Introduction

The Internet has fundamentally changed the way conflicts unfold and are perceived on the global stage. A crucial aspect is the growing role of social media with civilians sharing, publishing, and forwarding information, including sensitive strategic data. The term *hybrid warfare* [1] alludes to the increasing focus on information warfare, such as disinformation campaigns. For example, allegedly, journalists were specifically and lethally targeted in both the second Chechen war and the Syrian civil war, to control the public narrative [2]. Today, individual citizens can leverage social media to reach a global audience, further escalating the dynamics of (dis-)information.

This new paradigm is particularly visible in the Ukrainian conflict that escalated in February 2022. United Nations appointed independent rights experts warned that journalists in Ukraine are targeted and in danger [3], with 15 journalists confirmed to have been killed in Ukraine in 2022 [4]. Another perspective on the conflict claims Ukraine and its citizens got the upper hand in the so-called *social media war* [5]. Indeed, the U.S. government recognized the importance of a free flow of information among Ukrainians. After spending millions of dollars to fund the widespread deployment of Starlink satellite Internet terminals and service in Ukraine [6], the service quickly reached over 150 000 users shortly after deployment [7]. While a large share of Starlink's usage in Ukraine seems to be military, the Starlink smartphone app was downloaded over 806 000 from Ukraine, making it the most downloaded app at the time [8]. As a response, Russia has started a barrage of cyber and jamming attacks on Starlink since [9], yet they were repelled [10]. While the precedent of Starlink's satellite Internet in a conflict is still ongoing at the time of writing, the E.U. already announced a deal to deploy its own satellite Internet system [11].

Clearly, satellite-based Internet has a significant impact, as citizens and journalists gain the ability to freely share, e.g., sensitive information, evidence of war crimes, or timely warnings. However, openly sharing information also carries great risks, and thus, many social media companies have added additional security measures to protect Ukrainian users, along with some guidelines to minimize risks [12]. Moreover, there is a specific danger when using satellite-based communication, especially when used to publish sensitive information. As satellite signals can be monitored by virtually anyone in the sky and space above, satellite uplink communications can also be used to geolocate their users on the ground by triangulating their signals. One example is the killings of two American journalists [13] and another a missile strike on the leader of the Chechen republic [14]. In both cases it is assumed that the attacks were possible by

tracing satellite phones. While official confirmation of such state-backed attacks is rare, it was shown that techniques to geolocate transmitters by satellites (target tracking) are practical [15]. After a security researcher gained traction with a tweet warning Ukrainian Starlink users potentially being geolocated and becoming targets [16], the CEO of Starlink's company issued a public warning to Ukrainian Starlink users [17].

Considering the scale at which a satellite-based Internet can be monitored, and worse, individual users triangulated to get their physical position, it is an important aspect to protect citizens against this threat while preserving this novel and free flow of information during conflicts. However, there are no works to properly address this issue in a practical manner. On the one hand, using typical Internet encryption (i.e., TLS) on the communication channel is insufficient. Eavesdropping on satellite communication targets the actual physical medium, whereas TLS is a high-level protocol not designed to provide anonymity. The Tor network aims to fix this problem for the traditional Internet. However, our case has a crucial difference, as satellite communications can be monitored by any satellite and, worse, triangulated to geolocate the user. For example, numerous attacks on Tor assume an adversary can do *entry point* monitoring [18], [19]. While typically an ambitious position for the adversary, with a satellite-based Internet, it becomes quite straightforward, as the connection is first sent to the satellite before reaching the Tor network. Other attacks on Tor assume the adversary can monitor both the *entry* and *exit point* [20], [21], [22]. However, if the adversary aims to prevent a user from publicly sharing information, it may anticipate and monitor popular social media sites, such as Twitter, for the exit point.

Prior works on *location privacy* in mesh and wireless sensor networks have different shortcomings [23], [24], [25], [26], [27], [28], [29], [30], [31], [32], [33], [34], [35], [36], [37], [38], [39], [40], [41], [42]. For example, some works have impractical assumptions for our purposes and others induce significant overheads. There are also works that aim to establish a reliable network in case the existing infrastructure fails, called *emergency networks* [43], [44], [45], [46], [47], [48]. These approaches are typically based on specialized hardware, such as vehicles equipped with bulky communication equipment and even flying vehicles, and custom network protocols to facilitate basic communication, making them quite impractical for our purposes. There are also emergency networks working with satellites [49], [50], [51]; yet, they do not consider protection against triangulation. We will discuss the related work more thoroughly in Section 8.

In summary, the remote monitoring and the possibility to triangulate satellite Internet users is a global threat to the new-found free flow of information by citizens. To the best of our knowledge, there is no existing system that prevents geolocating satellite Internet users. In this paper, we present *AnonSat* to close this gap.

Goals & Contributions: Our primary goal is to hide the geographic position of satellite Internet users in case their connection is being triangulated. AnonSat works by leveraging long-range wireless communication to span a simple network among satellite base stations. Our system is agnostic with respect to the used wireless communication technology. Leveraging this local network, a client's WAN connection is routed to another randomly selected satellite base station, which acts as a delegate to do the actual connection uplink to the satellite. Further, the targeted satellite base station is regularly changed to avoid tracing back a long-lasting connection.

Our secondary goal is to focus on the accessibility of our system. Therefore, AnonSat is designed to work with cheap and simple devices, and thus, it can be deployed with widely available hardware and does not require impractical extensions on the user's device, such as a specific app or radio device. Further, we aim at the usage of popular Internet services, like Twitter or WhatsApp, refraining from custom network protocols. AnonSat effectively protects users from being geolocated and becoming targeted.

Our main contributions include:

- AnonSat is the first scheme to address the triangulation of satellite Internet users by rerouting connections to more distant satellite base stations. We introduce two security parameters that adjust the selection of routes to avoid geolocating a user over time.
- We derive requirements for wireless communication technologies and give an overview of possible candidates that can be used to enable AnonSat. Similarly, due to our aim to design a practical system, we discuss numerous approaches, such that AnonSat can access typical Internet services without needing to install custom software or hardware on the user's device.
- We implemented a proof of concept demonstrating the feasibility of AnonSat, leveraging a cheap Raspberry Pi equipped with a LoRa shield for the local network.
- We further developed a large-scale simulation using real-world data sets to evaluate key aspects of AnonSat in different environments, such as effective distances from the user or the use of more powerful wireless technologies.

2. System Model

Our system model consists of the following entities: A **network** is a collection of *gateways*, *clients*, and satellites providing internet access. Each **gateway** is equipped with a base station, i.e., means of providing access to the Internet via satellite communication, a radio transmitter to connect to the *local network*, and two WiFi access points to provide access to the *WiFi Network*. A **local network** is spanned among the *gateways* via their radio transmitter capable of communicating with each other. A **WiFi network** is a direct connection between *clients* and *gateways* separated into two WiFi access points. One is a connection to the gateway's WAN, directly uplinking to the satellite internet. The other

provides a secure WiFi connection via our AnonSat system. **Clients** are simple end-user devices, such as smartphones, which aim to establish a secured WAN connection to send sensitive data. For this, *clients* simply connect to the secure *WiFi Network* provided by a close-by *gateway*.

Note, for simplicity, we assume all *gateways* have a base station providing Internet access, even though in a real-world scenario simple relay nodes for the *local network* may also be deployed. We further assume that standard means of Internet access are impaired or even unavailable entirely, and thus clients need to rely on satellite Internet.

Furthermore, *gateways* are equipped with certificates to identify each other and to establish secret key pairs to enable symmetric encryption between any two *gateways*. We assume the *gateways* can trust each other's certificates. For example, in the real world, this could be realized via exchanging certificates via direct contacts in combination with a Web-of-Trust approach.

2.1. Adversary Model and Assumptions

The adversary $\mathcal{A}$ has the goal of geolocating a specific client. To do this, $\mathcal{A}$ has a range of satellites deployed, which can eavesdrop on the Internet communication between base stations and the receiving satellite. We assume that $\mathcal{A}$ is able to correlate the communication data to identify a specific client. Further, $\mathcal{A}$ is capable of triangulating the sending base station via the mentioned satellites, which was shown to be practical [15]. Thus, as the client has to use the closest base station via a close-range WiFi connection, $\mathcal{A}$ can infer the client's approximate geographic position. However, as our system aims to reroute the client's connection to another gateway, simply triangulating the base station is not enough to geolocate the client.

We assume $\mathcal{A}$ is not capable of establishing a holistic view of the local network. To achieve this, $\mathcal{A}$ would need to monitor a significant number of local network connections, which also requires prolonged physical proximity in a multitude of locations. This contradicts the scenario we are targeting, i.e., an active conflict zone, as widespread deployment of eavesdropping devices is infeasible. However, $\mathcal{A}$ is able to intercept individual messages sent between gateways. This assumption is comparable to the Tor network, which can also be broken by an adversary with a global view of the network; yet, in practice, this is hard to achieve. We further argue due to the nature of the limited range of each node (discussed in Section 4.1), the local network cannot be surveilled by satellites to attain a global view.

We will thoroughly discuss several possible local attacks in Section 7, including jamming attacks that deal with similar assumptions. Nevertheless, our system focuses on preventing remote and globally applicable triangulation via satellites.

Further, we assume the WiFi connection between the client and gateway cannot be intercepted by $\mathcal{A}$ due to its close-range nature. We also assume $\mathcal{A}$ aims to minimize any collateral damage, as this might have grave political repercussions [52], [53]. Finally, we further assume $\mathcal{A}$ cannot forge digital signatures or break symmetric encryption.

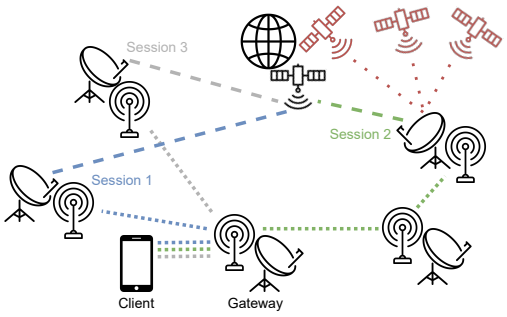


Figure 1. An example setup of AnonSat with five gateways and one client.

2.2. Requirements

To formalize the setting outlined in the Introduction, we aim to design a secure satellite Internet scheme with the following requirements:

R.1 *Prevent geolocating base station:* The scheme shall prevent $\mathcal{A}$ from geolocating the client. More specifically, the gateway uplinking traffic to the WAN shall not indicate the geographic position of the client. For example, as a client has to use the closest gateway, if this gateway uplinks the client's traffic to the WAN, $\mathcal{A}$ may assume the client is very close.

R.2 *Prevent local geolocation leakage*: In addition to requirement R.1, $\mathcal{A}$ may intercept individual messages in the local network traffic and trace back the actually used gateway by the client. Thus, the scheme shall further prevent geolocation leakage in terms of the local network.

R.3 *Internet compatibility:* The use of, e.g., simplified network protocols may greatly increase the performance of the scheme. However, this implies that most common Internet services are not accessible, and thus, the scheme shall be compatible with most Internet services.

R.4 *Out-of-the-box for clients:* From the client's point of view, the scheme shall impose minimal requirements on the client. For example, a client may need to unexpectedly and urgently send some sensitive data using a smartphone. In such a case, requiring the client to install an additional app or even an additional hardware device is impractical.

3. AnonSat Design

In this section, we focus on the general design of AnonSat. However, as our focus is designing a practical system (cf. Section 2.2), we will discuss essential technical aspects in Section 4. Figure 1 illustrates a simplified setup of AnonSat with five gateways, each with a base station to connect to the Internet and a local radio transmitter to communicate with other gateways. The client uses a

smartphone to upload sensitive data to a gateway in close proximity, called the *origin*. For example, the client may try to publish an incriminating picture on Twitter. Instead of directly forwarding the data over the satellite link, the gateway randomly selects an output gateway. In *Session 1* in the figure, the origin will then transmit the client's data over the local radio connection to the output gateway, which in turn will do the actual satellite transmission. Thus, if the output gateway is identified and triangulated by $\mathcal{A}$, the actual geographic position of the client stays hidden. After a chosen amount of time, the origin gateway will select a new output gateway for this client's connection in *Session 2*. This new output gateway is only reachable via an intermediate gateway, and thus, establishes a 2-hop connection. As seen in *Session 3*, the origin will continue to change the client's output gateway regularly while the connection lasts.

Changing Gateways. While we assume $\mathcal{A}$ cannot eavesdrop on the entire local gateway network (cf. Section 2.1), $\mathcal{A}$ may be able to intercept individual messages. As a first step, all communication between the gateways is per-hop encrypted. Thus, a forwarding gateway will receive an encrypted message, decrypt it, encrypt it with the key established with the next-hop gateway, and forward it. However, if $\mathcal{A}$ intercepts enough messages in relation to a specific client, $\mathcal{A}$ may be able to trace the origin gateway, and thus, geolocate the client eventually. Therefore, we introduce the security parameter $gateway\_timeout$, which defines a timeframe. When a client starts a connection and the origin gateway selects a random output gateway, a timer is started. After $gateway\_timeout$ time, the origin will select a new output gateway. Choosing a proper value for $gateway\_timeout$ depends on the bandwidth and delay of the local gateway network. Note that $gateway\_timeout$ may also define a message count instead of a timeframe. If $gateway\_timeout$ is set too high, $\mathcal{A}$ has an increasing chance to trace back the origin. Setting $gateway\_timeout$ too low may lead to technical problems with the client's connection. We will discuss this further in Section 4.4. Note, AnonSat's goal is not to make all traffic indistinguishable from unobjectionable traffic, as current approaches, such as Dummy Data Sources create non-negligible overhead (we discuss this in Section 8).

Distance to Origin. The selection of the output gateways has some crucial implications as well. If the origin only selects very close gateways, then the clients are not adequately protected. If the output gateways are too far away, this would require many hops between the origin and the output gateways, negatively affecting the performance of AnonSat. Thus, we introduce the security parameter max_hops, which defines how many hops away the output gateway shall be from the origin. Increasing this parameter increases the average distance from the client (i.e., the origin gateway) to the output gateway, and thus, increasing the protection of the client. However, increasing max_hops will negatively affect the performance of the client's connection.

Output Selection Bias. While simply routing the client's traffic to output gateways max_hops hops away is fine for individual short sessions, we need to consider a crucial aspect for longer sessions. Suppose we have a uniformly spread network of gateways and a client that needs a long-lived connection to, e.g., upload many pictures. In this case, many gateway changes happen over time and the selected gateways will eventually be selected in all directions from the origin. Simply put, $\mathcal{A}$ can observe these changes and will be able to draw a circle containing all output gateways. The gateway closest to the centroid of this circle is most likely the origin gateway; thus, endangering the client. To counteract this effect, we additionally introduce a selection bias for the output gateway. For each client, the origin gateway will generate a random *direction* and *weight* bias. When selecting a new output gateway, the origin will prefer random output gateways in the given direction and with the assigned weight. In a practical context, the direction can simply be a bias for selecting the next neighbor gateway via an index without the need to consider the gateways' geolocations. This way, the centroid of the mentioned circle shifts to a random direction, and thus, cannot be used to trace the origin. The selected biases will be preserved for each client.

Additionally, this allows us to change the role of the max_hops security parameter. Instead of always selecting an output gateway exactly max_hops hops away, we can select a range of hops between 0 and max_hops. This does not weaken the previously discussed selection bias. However, the range improves the client's network performance on average, as some output gateways may be only one hop away and fewer hops mean better performance for the client. Note that it is important for the origin to select itself as the output gateway. Otherwise, $\mathcal{A}$ can simply identify the origin by checking which gateway never sends.

4. Technical Considerations

After we described the theoretical design in Section 3, it is crucial to consider the technical challenges of AnonSat to satisfy requirements R.3 and R.4. Therefore, this section will discuss key practical aspects to implement our theoretical design. Note, we focus on available techniques or techniques currently in deployment, i.e., we will not address recent research approaches, as these might take many more years until ready for deployment. Our focus is on techniques usable in a practical deployment now or in the near future.

4.1. Wireless Communication Technology

As AnonSat relies on a local network between gateways, this section discusses possible options for wireless communication technologies. Fortunately, due to the prevalence of the Internet of Things (IoT), there have been many proposals and advancements in recent years to enable remote IoT devices to connect directly to the Internet or via a mesh network. We will leverage these advancements for AnonSat.

Table 1 shows an overview of the technologies we considered. Note, this table is not comprehensive of all available technologies, as we carefully selected technologies we deem

TABLE 1. OVERVIEW OF LONG-RANGE RADIO TRANSMISSION TECHNOLOGIES APPLICABLE TO ANONSAT.

Name	Maximum Data Rate	Range (urban)	Range (rural)	Unlicensed Frequency Bands
LoRa Sub-GHz [54]	27 kbps 50 kbps (FSK)	5 km	15 km	✓
LoRa 2.4 GHz	250 kbps 1 Mbps (FLRC)	1 km	n/a	✓
LTE-M Cat-M1 [55]	1 Mbps 4 Mbps (Cat-M2)	1 km	10 km	✗
NB-IoT Cat-NB2 [56]	200 kbps	1 km	10 km	✗
DASH7 [54]	166 kbps	5 km	n/a	✓
Weightless-W [54]	10 Mbps	5 km	n/a	✗

applicable to AnonSat. For example, while Sigfox is a mature wireless technology already deployed in many regions around the world, it only supports 100 bps data rates [56], which is too slow to be meaningfully used to connect to traditional Internet services (violating requirement R.3).

One may also consider repurposing existing infrastructure, such as the cellular network for mobile Internet. However, in a conflict, existing infrastructure (e.g., cell towers) is unreliable or may not be functional at all, making satellite Internet necessary in the first place. In addition, repurposing this infrastructure for a disaster network is not feasible as cellular networks do not have mesh network capabilities and access to the hardware is limited.

For the *Maximum Data Rate*, we also considered alternatives, like the Fast Long Range Communication (FLRC) for Lora 2.4 GHz, which uses demodulation and error correction techniques for a significantly improved data rate [57]. Further, the last column of Table 1 shows if the respective technology uses unlicensed frequency bands. For example, Lora supports a variety of different sub-GHz bands, which are publicly usable in the respective region, e.g., North America allocates different bands than Europe [56]. While the other technologies use licensed bands, we expect that legal restrictions play a lesser role in the settings applicable to AnonSat or may even be officially lifted.

4.2. MTU Size Mismatch

One technical hurdle with a significant practical impact is the Maximum Transmission Unit (MTU), which defines the maximum amount of bytes sent in a single network layer transaction. Considering our scenario targeting the Internet, typically the MTU of Ethernet is used (1500 bytes). The hurdle arises when using a wireless communication technology, which only supports a smaller MTU. For example, LoRa restricts the MTU to 255 bytes, which leads to problems with Internet servers expecting a large MTU, such as extra negotiation steps for smaller messages and spurious retransmissions. For example, in our preliminary tests using a small MTU, we saw significant delays with TLS handshakes, even to a point in which some servers simply declined the session entirely. Another downside of a small MTU is the overhead of the headers, such as the IP and TCP protocols. With a small MTU, the headers consume a significant share of the bandwidth.

Therefore, a mechanism is required to split Ethernet MTUs received by the client to fit them into, e.g., multiple LoRa frames for forwarding them over the local network. These kinds of operations are usually implemented by the Operative System's (OS) kernel. An example of LPWAN is IEEE 802.15.4 (*Low-rate Wireless Personal Area Network*) [58] over IPv6, documented as 6LoWPAN in RFC 8930 [59], which specifies how to forward 6LoWPAN fragments over a multi-hop network. The implementation is already available in the Linux Kernel [60] for IEEE 802.15.4 compliant devices. This could benefit AnonSat greatly, as it was demonstrated that a proper fragmentation strategy can lead to significant performance improvements [61]. Yet, not all wireless technologies fit this standard. For example, there are discrete definitions of header compression and segmentation for LoRa in RFC 9011 [62]; yet, there are no implementations so far.

4.3. Slow TCP Connections

The Transmission Control Protocol (TCP) and how servers handle TCP sessions are usually optimized for fast and reliable connections nowadays, as this is the most common scenario. However, for AnonSat this is not the case. Indeed, in our case, the connections are slow and potentially lossy, leading to problems with many TCP deployments. The most relevant one is the way retransmissions are handled, as typical deployments are optimized to maximize data rates for fast connections. One aspect is the Retransmit Timeout (RTO), which is typically set quite low, such that the server can react quickly to network congestion, which is also identified with timeouts. If the acknowledgement for a packet is not received by the server in time, to optimize data rates for typical connections, the server will quickly do a retransmission. This results in many duplicate, and thus, unnecessary retransmissions with slow connections called *spurious* retransmissions, quickly saturating the connection. Another problem to consider is the loss of packets, especially when using one of the wireless communication technologies (cf. Section 4.1). For example, today's TCP deployments bundle the transmission of multiple resources over a single connection to achieve a form of concurrency. However, when a packet is lost for one resource, this delays all of the resources, resulting in even more retransmissions.

Unfortunately, the most effective TCP settings to avoid these issues are controlled by the endpoints, such as retransmission timeouts or the used congestion control algorithm. Thus, in AnonSat, we cannot change these settings, as we can only influence the gateways. An exception to this is the `txqueuelen` property for each network interface in Linux. Settings this to very low values (e.g., 1) prevents the client to send many packets in a short time, which will exacerbate the mentioned retransmission problems. Another setting to consider is setting TCP's window size, influencing how much data is bundled for each acknowledgment. Setting

this to a low value further helps to avoid retransmission problems.

Another approach to counteract these problems is optimizations for the used wireless communication technologies in a multi-hop setup. A local retransmission scheme may be used among the gateways, essentially dealing with packet losses on the local gateway network level. There are different strategies for such local retransmissions [63]. Handling retransmissions locally would circumvent many of the issues with TCP packet losses.

Furthermore, a more sustainable solution is being deployed at the time of writing and will likely be widely available in the near future. QUIC [64] is a network protocol introduced by Google as a more performant and flexible alternative to TCP. Among other functionalities, it leverages UDP with merged handshakes, custom congestion controls, loss detection, and retransmission algorithms [65]. This allows QUIC to handle slow connections with high latency with much better performance than TCP. For example, Google published a large-scale performance study on QUIC, which showed that QUIC can improve latency by over 30% compared to TCP with large round-trip-times in a common Internet scenario [66]. Thus, QUIC would likely have a significant performance impact for our purposes.

Practically speaking, QUIC is already a reality, as many companies are already adopting it. For example, the company Meta already deployed it for most of its applications [67].

4.4. Changing Routing Paths

As described in Section 3, AnonSat needs to reroute packets through different nodes. Typically, TCP sessions stay alive until the end of the communication and it is not possible to dynamically change the stream's endpoints, i.e., the source or destination IP. However, changing the output gateway implies a change in the endpoint IP; thus, we need to adopt a strategy for establishing a new endpoint.

A possible way for achieving this goal is using the Reset (RST) flag in the TCP header. When this flag is set, the receiver will close the TCP session immediately. The use of the RST flag is also used for malicious applications, such as the so-called "TCP Reset Attack" [68]. Here an attacker forges TCP packets with a set RST flag in order to maliciously interrupt or disturb the Internet connection. However, for our purposes, the origin gateway can leverage the RST flag to stop a TCP session and force the endpoints (i.e., the client and the Internet server) to start a new session with a different endpoint, i.e., another output gateway selected by the origin. Indeed, for AnonSat, this comes with the additional overhead of establishing a new TCP session regularly. Thus, when using this approach the $gateway\_timeout$ parameter should not be set too low.

However, this is exactly what the Multipath TCP (MPTCP) protocol sets out to do in a more elegant way. The protocol was originally documented in 2013 by RFC 6824 [69] and later updated with RFC 8684 [70]. This protocol allows a single TCP session to take different routing paths and use different endpoints without interruption. As widespread support is relevant, MPTCP is already implemented in the Linux Kernel [71] and, e.g., Apple's iOS uses it to be able to quickly switch between a WiFi and cellular connection [72].

Another elegant way is provided by the aforementioned QUIC protocol. While there is also a multipath extension for it in the making [73], similar to MPTCP, the original QUIC protocol already supports *Connection Migration* [64]. Here, each session is assigned a connection ID, which allows the connection to survive endpoint address changes.

Furthermore, depending on the used underlying protocols for the local network, e.g., IPv4 or LoRaWAN, the client needs additional protection. Namely, the output gateway should execute a Network Address Translation (NAT) on the connection, translating its own address to the origin. Otherwise, the address of the origin gateway may leak, and thus, potentially reveal the client's geographic position.

4.5. Node Discovery

The local gateway network acts as a mesh network, and thus, needs protocols for discovering nodes and establishing routing tables for later network message routing. For our purposes, a link state routing protocol works well. While there are many different approaches to these types of protocols, we expect the local gateway network to only observe a limited degree of dynamics, as opposed to, e.g., a mobile ad-hoc network. The latter typically requires more advanced techniques for routing and node discovery. Thus, the Optimized Link State Routing Protocol, defined in RFC 3626 [74], or its successor [75] is sufficient for our approach. Here, neighbors exchange information about their neighbors to build simple routing tables, which show the shortest path to any node in the network.

Further, there are also protocols to dynamically optimize the radio settings between nodes. For example, there is the *Adaptive Data Rate* optimization between LoRa nodes [76]. These protocols may be used to further optimize the data rate for the local gateway network.

4.6. Leakage in Presence of Malicious Nodes

In the mesh and wireless sensor network research area for location privacy (we give an overview of them in Section 8), there are approaches additionally considering defenses against compromised nodes [77]. Many approaches like *Network Coding* and *In network location anonymization* are designed to protect against an attacker with a *global* view, which is infeasible in our discussed scenario (cf. Section 2.1). These approaches introduce many new requirements and overheads, making them inapplicable for our scenario. However, approaches considering a *local* adversary are not applicable due to their assumptions. One type of approach assumes a hierarchical structure among the nodes in the network, in which certain nodes are trusted and cannot be compromised [30], [78], [36]. Due to the nature of our targeted scenario, assuming only some gateways to

be trusted is not practical in AnonSat. A different approach is to use end-to-end encryption between source and destination, such that a potentially compromised intermediary node cannot know the endpoints of a message [79]. However, this requires that all possible endpoint pairs have pre-shared keys deployed, which is infeasible in our scenario. Another approach can only hide the destination of the connection, not the source [27]. Yet, protecting the source, i.e., the origin gateway, is the main goal of AnonSat. A different approach protects against compromised nodes, yet relies on the assumption that intermediary nodes cannot be too close to the source [80]. We deem this assumption too strict to be practical for AnonSat.

The main concern for AnonSat regarding compromises is malicious intermediary gateways, which can extract the origin gateway and thus target the client. Onion routing [81] is an effective method to hide the origin gateway from compromised intermediaries. No intermediary hop can know its position in the path, as a packet's amount of onion shells is unknown. This is due to AnonSat randomly choosing a path length between 0 and max_hops. Additionally, onion routing can also be used in wireless networks [82], where, when mapped to our approach, no node (except for the first and last) is able to learn the source of a message.

5. Prototype Implementation

To demonstrate AnonSat, we describe the implementation of our Proof of Concept in this section. While we have shown advanced techniques in Section 4, many of them lack either proper wide-ranging support or applicable implementations. At this point in time, we consider the integration of these techniques as a significant engineering effort and out of scope for this research work. Note, however, there is potential to significantly improve the practical performance of AnonSat. We will give justifications for the choice of the respective techniques in the following.

Scenario. For our prototype, we deployed five off-the-shelf consumer devices. For the Internet connection, we deployed a Starlink dish [83] on the roof of our building. We used three gateways deployed as Raspberry Pi 3 Model B+ [84] (called *RaspberryPi1*, *RaspberryPi2* and *RaspberryPi3*), equipped with a sub-GHz Lora SX1262 868M shield [85]. Finally, we used a simple Android phone as the client. *RaspberryPi3* acts as a proper gateway, as it is connected to the Starlink router, which is connected to the dish. All Raspberry Pis are interconnected via Lora and provide a WiFi access point for the client to connect. Figure 2 shows the client connected to the origin's secure WiFi access point as well as the output gateway of our test setup.

We have chosen to use sub-GHz Lora due to its use of unlicensed frequency bands. Another aspect was the availability of development kits that include proper integration for both hardware (e.g., connection to our Raspberries) and software (e.g., driver). Note, we have also evaluated using the Lora 2.4 GHz shield SX1280Z3DSFGW1 [86]; yet, we found that the provided implementation by the manufacturer[1] is impractical, as it has limitations that severely limit the effective data rates, even below the Lora sub-GHz shield mentioned above. For example, the implementation demands a 50 ms sleep after each packet is sent, meaning a 1500 bytes MTU split into 7 Lora packets already implies a 350 ms delay before considering the actual transmission time.

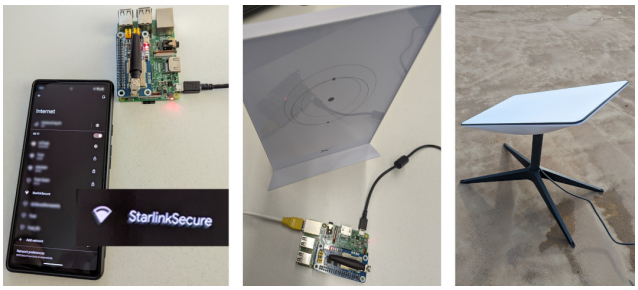


Figure 2. Our test setup. The left picture shows the client with the origin gateway, the center picture shows the output gateway connected to the Starlink router, and the right picture shows the Starlink dish that is connected to the Starlink router.

Implementation. To send IP packets over LoRa, we utilize the open source software `tncattach`[2] which creates and sets up network interfaces to translate IP to LoRa and back. Unfortunately, the implementation has no support for packet fragmentation and reassembly, and therefore, only utilizes the maximum LoRa packet size (236 bytes excluding header overhead), leading to the problem discussed in Section 4.2. To speed up the translation (and avoid package retransmissions) between Ethernet (typical MTU of 1500 bytes) and LoRa, we added simple fragmentation support to `tncattach`. The more advanced techniques for fragmentation and header compression implemented in the Linux kernel [60] are not applicable to LoRa, and the LoRa-specific specification [62] has no implementation.

In terms of routing, we pre-deploy our routing tables for simplicity, as opposed to implementing an advanced protocol, as described in Section 4.4. For example, *RaspberryPi1* sets up a Wi-Fi Access Point with its own subnet and forwards all packages to the LoRa interface using another subnet utilizing NAT. *RaspberryPi2* receives these packets and forwards them to *RaspberryPi3*. *RaspberryPi3* is then forwarding them to the Starlink router connected via Ethernet, utilizing NAT again. By providing every device its own static IP address, the operating system takes over tasks such as device discovery and routing.

As the origin gateway, *RaspberryPi1* is in charge of setting up routes for the client's traffic to keep track of the duration of each link. As described in Section 4.4, we chose to break TCP connections using a Reset Attack. We utilized `scapy`[3] to listen to TCP packets traveling on *RaspberryPi1's* interfaces. As the WiFi Access Point utilizes DHCP, a list of connected devices is known, including the connec-

1. https://github.com/Lora-net/gateway_2g4_hal
2. https://github.com/markqvist/tncattach
3. https://scapy.net/

TABLE 2. AVERAGE RTT AND PACKET LOSS FOR PINGING 8.8.8.8 WITH 100 64 BYTES ICMP.

LoRa Hops	RTT	Packet Loss
0	49.111 ms	0%
1	157.938 ms	3%
2	211.786 ms	4%

TABLE 3. AVERAGE TIME FOR UPLOADING AND DOWNLOADING IMAGES OF DIFFERENT SIZES USING 2 LORA HOPS.

Image Size	Upload	Download
50 kB	65.84 s	60.40 s
100 kB	137.84 s	117.80 s
150 kB	171.92 s	223.20 s
200 kB	269.31 s	276.80 s

tion time; thus, we can listen to established TCP connections of these devices using a packet filter. If a connection is kept alive for too long (i.e., longer than $gateway\_timeout$), the origin gateway creates a TCP packet with a set RST flag and injects it into the packet flow in both directions, killing the connection. Afterwards, the origin can establish a new route for this client.

Further, to counteract the TCP retransmission problems outlined in Section 4.3, we set the `txqueuelen` to 1 and TCP's TX buffer to exactly one packet. This mitigated many spurious retransmissions created because of delayed ACKs.

6. Evaluation

In this section, we evaluate AnonSat. On the one hand, we show real-world results of our prototype setup. On the other hand, we show the large-scale performance in different settings based on a network simulator run on real-world data sets.

6.1. Prototype

Recall Section 2.1, the goal of our approach is to hinder $\mathcal{A}$ to localize a client, e.g., publishing incriminating information. Usually, such information is published in the form of text or images. Therefore, to measure the performance of our prototype, we considered three use cases for the client in our setup. One is simply sending messages via the popular WhatsApp messenger. Another was to send out tweets via the Twitter Lite app. The third is to publish an image. For accurate measurement numbers, we used pings as well as downloaded and uploaded small images via the client.

Round-Trip-Time (RTT). We measured the RTT using a standard ping via zero, one and two LoRa hops to a server on the internet via the Starlink connection. Note that using Starlink alone already adds around 50 ms of delay. We send 100 pings and averaged the results. The results are depicted in Table 2.

Upload & Download. We implemented our own API service to upload images using REST over TLS using HTTP/2. We used a POST form request to upload a picture of size between 50 kB and 200 kB. These numbers correspond to the lower and upper bounds of typical mobile applications image compression (e.g., WhatsApp), which we measured with common photographs. Similarly, we downloaded the files using `wget`. We repeated this experiment 10 times and averaged the results. The results are shown in Table 3. Note, while we employed both fragmentation for the LoRa packets and optimized the TCP settings, as discussed in Section 5, we still observed a significant amount of spurious retransmissions.

Naturally, AnonSat incurs an overhead. Securely uploading a high-resolution photo using WhatsApp (i.e., 150 kB) takes 171.92 s. However, we argue it is reasonable considering the alternative of being either localized or not publishing at all.

6.2. Simulation

To measure the large-scale performance of AnonSat, we implemented a network simulation. In the following, we describe our evaluation setup by first describing how we simulated the local wireless network, presenting the used real-world data sets, and how the network simulation works. Afterwards, we present our results showing how the security parameter max_hops affects the distance from the position of a client, the delays to establish a TLS session in different settings, and finally, the practical data rates.

6.2.1. Local Wireless Network. While we were restricted to Lora with the sub-GHz frequencies, the simulator allows us to simulate more powerful wireless technologies. Informed by the data shown in Table 1, we selected the following combinations of assumed ranges and data rates between nodes:

1) 5 km @50 kbps (Lora Sub-GHz)
2) 5 km @166 kbps (DASH7)
3) 1 km @1 Mbps (Lora 2.4GHz & LTE-M Cat-M1)
4) 1 km @4 Mbps (LTE-M Cat-M2)

We excluded NB-IoT due to its low performance compared to the other technologies on our list, which is mostly due to its low-power requirement. We further excluded Weightless-W, even though its impressive data rates, as we could not establish the readiness of the technology. For example, unlike the other technologies, we could not find any purchasable devices equipped with Weightless-W components or any practical demonstrators.

Thus, for our simulation we assume, e.g., a range of 5 km between nodes with a maximum data rate of 50 kbps, simulating Sub-GHz Lora. However, the maximum data rate is practically not achievable, especially at longer ranges. While there are some practical measurements regarding this phenomenon, we found the used evaluation setups (e.g., obstructions or radio settings) and results vary significantly[4].

4. For example, one work measured around 10 kbps [87] while another measured double the data rate [88] for similar settings.

TABLE 4. URBAN WIFI HOTSPOT DATASETS USED FOR SIMULATION. *Close* REFERS TO THE NUMBER OF RECORDS AFTER FILTERING OUT TOO CLOSE RECORDS. *CC* REFERS TO THE NUMBER OF RECORDS CONTAINED IN THE LARGEST CONNECTED COMPONENT WHEN CONNECTING THE GRAPH WITH THE GIVEN RANGE.

Data Set	Total	Close	CC 5 km	CC 1 km
Hong Kong [89]	5441	874	866	332
New York City [90]	3319	765	753	602
Rhein-Neckar [91]	1338	551	530	30
Brisbane [92]	347	97	95	38
Paris [93]	277	182	181	178
Adelaide [94]	272	51	51	51
Leeds [95]	236	169	163	83
Linz [96]	124	35	34	29

For our simulation, we approximate the *log-distance path loss model* as a simple logarithmic function over the distance between nodes $r = e^{-2d}$. d is the distance between two nodes as a relative distance $[0, 1]$ with respect to the maximum range. The resulting data rate r is also relative $[0, 1]$ to the maximum data rate. Thus, the maximum data rate is only achievable if two nodes are right next to each other, while two nodes that are far away, e.g., close to the maximum range, have a severely reduced data rate with only a small fraction of the maximum data rate.

6.2.2. Data Sets. As far as we are aware, there are no representative data for gateway positions for the settings we target. Nevertheless, we found public data sets on various cities' WiFi hotspots to be a good approximation for an urban environment. Table 4 lists all of the data sets we used, each containing the geographic positions of WiFi hotspots for cities of different sizes. Figure 3 shows renderings of the Hong Kong, New York City, and Rhein-Neckar data sets.

However, we had to filter these data sets to fit our needs for the simulation, due to the following two problems. For one, these data sets contain points that are very close together, which is to be expected, e.g., in the city center, there will be a large number of shops or similar with a high density of hotspots. As such a dense concentration of gateways is not representative in our settings, we filtered out nodes that are closer than 200 m from each other. In Table 4, the number of nodes left after this filtering step is shown as *Close*. The second problem is that our assumed maximum range leads to a disconnected graph, as some sub-graphs may not be in range for another sub-graph. Thus, we constructed a graph of nodes from the data sets with the respective maximum range and found the set of connected components in the overall graph. Finally, we chose the largest connected component for each data set as the actual set of nodes for our simulation. In Table 4, this is shown as *CC 5 km* and *CC 1 km* for a range of 5 km and 1 km respectively. Note, for some data sets the 1 km range limitation creates a very small network, such as Rhein-Neckar, which covers a large area, but many of the nodes are further than 1 km away from each other. Therefore, this results in a significant reduction of the size, if the used connected network, e.g., Rhein-Neckar *CC 1 km* only has ∼5.4% nodes left relative to *Close*. However, note that Rhein-Neckar is an exceptional outlier, due to the data set spanning relatively few nodes over almost 100 km. The other data sets with higher reductions comprise of dense clusters that are far from one another, e.g., see Figure 3 for Hong Kong. In such a scenario, AnonSat could be applied individually for each cluster. Generally, a restriction true for all physical wireless networks is that a more densely packed network results in a holistically better connectivity among nodes [97]. Therefore, we stress that, if there are no alternatives to using satellite Internet (cf. Section 2), even a suboptimal network setup benefits from our approach.

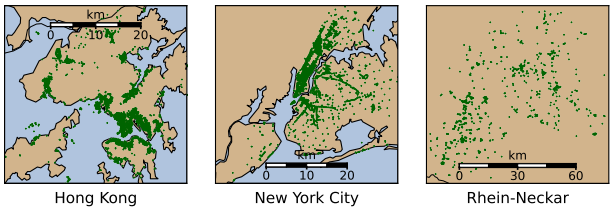


Figure 3. Renderings of the three largest data sets shown in Table 4. Each green dot is one geographic position.

6.2.3. Network Simulator. To implement the network simulation, we used the OMNeT++ 6.0 network simulator [98]. We load the processed data sets, as described in Section 6.2.2, as the individual nodes into the simulation. These nodes are then connected, if they are in range of each other, with a data rate calculated at initialization, as described in section 6.2.1. We further use the provided routing component in OMNeT++ to route individual messages as well as to ensure max_hops is satisfied when randomly selecting gateways by the clients.

For the actual simulation, we assign each client to a random gateway as its origin. Each client will then proceed to execute the following steps:

1) The client's gateway will randomly select an output gateway less than max_hops hops away.
2) The client will send a `TCP SYN` message out to simulate establishing a connection. This message is routed to the output gateway.
3) After the output gateway received each message, we simulate a 100 ms delay for the WAN server to answer[5].
4) The output gateway will send a TCP `SYN-ACK` message back to the client's gateway.
5) The client will answer this with a TCP `ACK` and TLS `ClientHello` combined message, as is a common optimization practice of the Internet.
6) The simulated server will answer this with a TLS `ServerHello` message; thus, establishing the TLS connection when the client receives it.
7) The client will then start sending a 200 kB data package (the upper bound for WhatsApp image compression)

5. We based this number on the upper average of 50 ms delay with Starlink and some additional processing time.

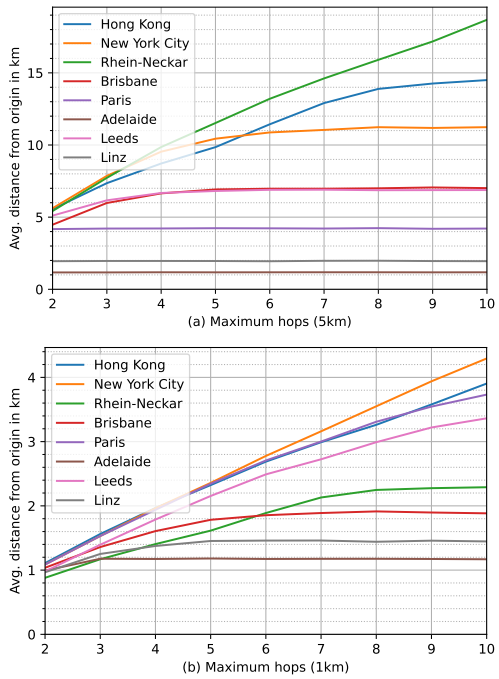


Figure 4. Graphs showing the average distance from a chosen output gateway to the origin for different max_hops over all data sets with (a) 5 km and (b) 1 km range.

divided into multiple messages, i.e., according to the MTU sizes.

The client will repeat these steps, selecting a new output gateway each time, until the simulation ends after a simulated hour. Put simply, each client has a fixed origin and will constantly send images with changing output gateways. For each parameter combination and data set, we execute 30 runs with different random seeds. While quite a simple setup, this allows us to approximately measure TLS session delays, the effective data rates in the network, and the interactions of both, e.g., TLS session delays of clients over a gateway currently busy sending many data messages.

6.2.4. Distance to Origin. To evaluate the key security parameter max_hops, we measured the average distance from the client's gateway (origin) to the output gateway. For this, we employed a simplified simulation over our data sets (cf. Table 4) to get a large sample size of 10 000. For each sample, we select a random origin with a random output gateway less than max_hops away and measured the actual distance from the origin via their geographic position. Further, we executed this simulation with differently set max_hops. The results of this simulation are shown in Figure 4. Wide-ranging data sets in terms of the overall covered area by the nodes, like Hong Kong or Rhein-Neckar, show a nearly linear increase in distance with an increase of max_hops. We discuss this more thoroughly in Section A.

6.2.5. TLS Session Delay. To measure the performance of AnonSat, we focus on two aspects: delays and data rate. However, as the primary goal of AnonSat is security, we focus on practical applications. Namely, for delay we measure the average delay it takes to establish a TLS session. As most Internet services today are based on these sessions, this is a more practical number than measuring simple round-trip delays, especially as we want to see the effects of concurrent data transfers and session establishments in the network.

Figure 5 shows our measurements. Noticeable between the slower networks (a) & (b) and the faster networks (c) & (d) is the effect of max_hops with many clients. With max_hops = 5 congestion of the gateways affects the delay significantly. With a range of 5 km (a) & (b), the two data sets Adelaide and Linz show a much better performance, due to all nodes being packed closely together in a small area, which also results in much better data rates on average. Note that there is an implicit tradeoff, as the low average distances between nodes naturally affects the *distance to origin*, as shown in Section 6.2.4. A similar effect can be observed for the 1 km range measurements (c) & (d). Data sets with closely packed nodes in general, like New York City, show much better performance than data sets with spread-out nodes, like Brisbane.

6.2.6. Practical Data Rates. In a practical scenario, the data rates may depend on multiple TLS sessions first, which would heavily reduce effective the data rates when simply calculating overall sent bytes divided by time. Thus, we decided to measure the time it takes to upload a 200 kB sized image as a practical and isolated example.

Figure 6 shows our measurements. Generally, the observation made in Section 6.2.5 regarding the effect of the density of nodes on data rates applies here as well. However, while the delays imply a logarithmic trend with a growing number of clients, we can clearly see the effects of many clients sending data and the resulting congestion in the network. Particularly noticeable are the measurements for Paris with a 5 km range. The increase in the number of clients has an especially detrimental effect on the transmission speed. We believe this is due to the unique and dense spread of nodes in the data set, creating some *bottleneck* nodes that serve multiple connections simultaneously. This effect disappears with a 1 km range, as fewer nodes can connect to these bottleneck nodes.

7. Anonymity & Security

In this section, we first analyze the practical anonymity guarantees of AnonSat and the security of AnonSat.

7.1. Anonymity

We quantify and evaluate the anonymity provided by AnonSat on the data sets described in Section 6.2.2. Table 5 summarizes our analysis for each data set for both the 5 km and 1 km cases, with max_hops = 3 and max_hops = 5,

respectively. In the following, we will explain our metrics and evaluate our results.

The first metric we evaluated is the traditional *Anonymity Set* size as defined by Chaum [99]. The idea is that one's anonymity can be quantified by the set of parties from which one is not distinguishable. Thus, $\mathcal{A}$ ideally does not know, which of the parties in the set is the actual target. In our case, we counted all gateways reachable by an output gateway, i.e., less than max_hops away, as these are the potential alternatives among the origin gateway. We considered all gateways in our data sets for both the average and the gateway with the lowest number of reachable gateways in Table 5. While a dense data set like New York City has a high average, the less dense Rhein-Neckar set has a comparably low average. When looking at the minima, i.e., the least connected gateways, some data sets contain remote nodes with few reachable nodes.

Nevertheless, just counting the possible alternatives is not a complete metric, as the probabilities to be the origin among the set of reachable gateways may not be uniform. A way to model this is to consider the entropy of the anonymity set based on implicit information on the network and its nodes leveraged by $\mathcal{A}$ [100], e.g., the reachability of each node throughout the graph. In our case, similar to our Anonymity Set metric, we look at all reachable gateways from an output gateway. In addition, we consider the number of paths between each reachable gateway and the output gateway, as a gateway with more paths to the output gateway is more likely to be the origin than gateways with fewer. With this entropy, we are able to calculate the *Effective Set* size for the anonymity set (see Table 5). Concretely, we calculate per reachable gateway g the number of paths to the output gateway divided by all possible paths from any reachable node to the output gateway as p_g (c.f. [100]). With this, we can calculate the effective anonymity set for all g in the reachable gateway set for an output gateway:

$$-\sum p_g \log_2(p_g)$$

We considered all gateways in our data sets to get both the average and the minimum effective anonymity set. Compared to the uniform anonymity set, data sets that are more clustered in terms of gateway distribution have a significantly lower effective set size (e.g., Rhein-Neckar 5 km with a $\sim 39\%$ reduction) than data sets that are well-connected (e.g., Paris 5 km with a $\sim 4\%$ reduction). These numbers demonstrate that different data sets with different degrees of connectivity among the gateways result in varying degrees of anonymity, which needs to be considered in practical deployment. However, the average effective anonymity set sizes show the efficacy of AnonSat.

Finally, we measured the average number of paths between any two gateways as *Node2node Paths* within max_hops distance. This gives a hint at the general connectivity among the nodes. To further refine this metric, we also counted all *Unique Paths*, i.e., the number of paths that do not share any common gateways in their route. This indicates how reliable AnonSat is when individual gateways fail, i.e., the number of alternative paths. In Table 5, we can examine that in more dense networks with many connections, like Paris 5 km, there are many alternative paths on average. Contrarily, if there are few connections between nodes, like Paris 1 km, then there are few alternative paths on average. Generally, a higher number shows a more resilient network against gateway failures.

7.2. Security

The adversary $\mathcal{A}$ aims to geolocate a specific client by identifying the origin gateway. $\mathcal{A}$ may use the following strategies to accomplish this: (1) $\mathcal{A}$ assumes the used output gateway is close enough and target it instead, (2) $\mathcal{A}$ uses individual intercepted messages from the local network to trace the origin, and (3) $\mathcal{A}$ monitors the gateways for extended periods to collect data pointing to the origin.

Strategy (1) is prevented in our system by rerouting communication away from the origin gateway used by the client. In Section 6.2.4, we analyze the effective distances achieved on real-world data sets. Nevertheless, due to our *Changing Gateway* approach (cf. Section 3), eventually, the actual origin gateway is used for the satellite uplink. Thus, $\mathcal{A}$ may simply target each gateway and eventually be successful. However, this would effectively result in a large-scale attack, e.g., in an urban context, targeting the entire city. As stated in our adversary model (Section 2.1), we deem this undesirable for $\mathcal{A}$. Note that this also applies to $\mathcal{A}$ taking kinetic measures against gateways in subregions of the network. As described in Table 5 (column *Unique Paths*), every network has at least two unique paths between any two nodes, i.e., alternative paths that do not share any nodes. This demonstrates the network's resilience against forceful disconnection, even in the presence of a costly, wide-ranging attack on many gateways instead of targeting individual ones. We deem this undesirable, as $\mathcal{A}$ tries to limit its attacks as much as possible. Further, in case $\mathcal{A}$ captures extensive territory, resulting in few gateways in the region, we assume that clients will not remain in the area and transmit sensitive data.

For the second strategy (2), $\mathcal{A}$ is prevented from learning any information about the route with the end-to-end encryption employed by the gateways. Yet, $\mathcal{A}$ may intercept numerous local gateway messages over time and eventually be able to correlate the route from the output gateway back to the origin. Our system prevents this by regularly changing the output gateway (cf. Section 3). The effectiveness of this approach is dependent on the gateway distribution, which we evaluate in Section 6.2.4 on our real-world data sets.

$\mathcal{A}$ may monitor all used output gateways by a client to infer the origin. The success of this strategy (3) is prevented by the *Selection Bias*, as described in Section 3, which shifts the centroid of all used gateways over time away from the origin.

In the following, we discuss additional local attacks, motivating our adversary model.

Total Local Network Monitoring. In Section 2.1, we assume $\mathcal{A}$ is unable to monitor the entire gateway network

TABLE 5. ANONYMITY METRICS FOR ALL DATA SETS FOR BOTH THE 5 KM AND 1 KM CASES.

	5 km Average Anonymity Set	5 km Minimum Anonymity Set	5 km Average Effective Set	5 km Minimum Effective Set	5 km Average Node2node Paths	5 km Average Unique Paths	1 km Average Anonymity Set	1 km Minimum Anonymity Set	1 km Average Effective Set	1 km Minimum Effective Set	1 km Average Node2node Paths	1 km Average Unique Paths
Hong Kong	417.3	12	278.0	9.1	6 080.3	54.2	111.8	26	60.1	21.8	5 147.9	5.7
New York City	523.7	13	330.0	12.6	6 415.1	59.9	92.7	8	47.7	8.0	3 734.4	4.2
Rhein-Neckar	98.5	11	60.4	7.5	193.1	8.9	18.3	13	12.2	9.1	116.2	2.3
Brisbane	83.7	17	61.9	14.9	862.9	20.2	35.3	30	26.7	20.8	116.5	4.3
Paris	180.0	179	172.7	162.4	10 096.9	93.8	78.4	12	43.0	9.4	332.2	2.7
Adelaide	50.0	50	50.0	50.0	2 402.0	50.0	50.0	50	44.7	43.7	2 045.5	13.4
Leeds	149.7	49	121.4	39.5	1 880.1	35.6	37.9	10	22.5	8.4	61.2	2.1
Linz	33.0	33	33.0	33.0	959.8	31.6	37.9	26	20.7	19.2	834.3	4.6

to establish a holistic view of the local network. $\mathcal{A}$ would aim to trace back routes from the output gateway back to the origin. $\mathcal{A}$ would need to get an extensive coverage of the gateway network. Practically speaking, to achieve this, $\mathcal{A}$ needs to deploy numerous devices, which must be widely spread in close proximity to the gateways and potentially deployed over long periods, as it is unknown where and when the client may become active. Considering the potential scale of an active conflict, we deem this strategy infeasible.

Jamming. $\mathcal{A}$ may try to use jamming to interrupt the client's ability to communicate. There are three types of jamming to consider. One type is targeting the satellites, e.g., with a high-power ground-based jamming signal directed at individual satellites. However, with over 3 000 Starlink satellites deployed [101] at the time of writing, this strategy does not scale well. Further, according to reports, Starlink used specialized firmware updates to withstand numerous jamming attacks [10].

An additional type of jamming is adversarial satellites jamming ground stations. Theoretically, a signal can be considered jammed if the Signal to Noise Ratio (SNR) at the receiver is 1. To achieve this the jamming signal must be received with at least the same power as the benign signal [102]. The power of electromagnetic signals decays quadratically with distance. Thus, a satellite targeting a ground station would need a jamming signal powerful enough to cover the vast distances in space. As satellites are significantly limited in terms of power, we deem this strategy infeasible.

Another strategy is local jamming ground-to-ground, as modern jammers may cover a wide area. However, depending on the scale of the gateway network, $\mathcal{A}$ would need to either deploy many jammers or target the client's general area, which might be unknown. In case $\mathcal{A}$ *is* able to jam the client's gateway, this is difficult to circumvent; however, in this case, $\mathcal{A}$ is not able to geolocate the client.

Malicious Gateways. $\mathcal{A}$ may try to deploy malicious gateways. However, we deem this strategy to be unviable. Similar to the holistic monitoring of the gateway network, $\mathcal{A}$ would need to deploy or compromise a plethora of devices spread throughout the gateway network to reliably trace clients. Individual gateways may reveal the client's route if the client actually routes over them. However, unlike *overlay* networks, in which an adversary can remotely create new nodes, $\mathcal{A}$ must control *physical* gateways in advantageous geographic locations to reliably target clients. Thus, such an elaborate strategy requires physical access, resulting in low probabilities of success. To protect against leakage of the origin gateway by intermediary malicious nodes, approaches such as onion routing can be deployed to AnonSat, as described in Section 4.6.

8. Related Work

To the best of our knowledge, our approach is the first work to address the triangulation of satellite Internet users in critical circumstances. Thus, we could not find directly related work for our purposes. However, one related research topic is providing *location privacy* in mesh and wireless sensor networks. Another related research topic is *emergency communication networks*, which, similarly to AnonSat, often employ mesh-like network topologies and are designed to work under exceptional situations.

Location Privacy in Mesh and Wireless Sensor Networks. In these networks, due to their physical properties, communication is vulnerable to tracking and monitoring. If vulnerable participants utilize such a network, keeping the (geographic) position undisclosed becomes essential. Approaches for location privacy can be grouped into eleven categories [77].

The high-level idea of *Random Walk*-based approaches is to direct packets to traverse a network through a random path to a sink (e.g., base station). This makes the path of a packet unpredictable to an adversary attempting local traffic analysis [23], [24]. *Geographic Routing* is similar to Random Walk, yet utilizes the physical location information of nodes for more efficient routing of packets towards the sink. For location privacy, these approaches additionally leverage pseudonyms, reputation, and a fixed set of intermediary nodes creating a mix subnetwork [25], [26]. Both these types of approaches assume a *backtrack* or *hunter* adversary model, in which the adversary starts close to the sink, traces each sent-out message back to the next hop, and this is repeated until the adversary eventually arrives at the source. However, this is incompatible with our assumptions, as all

nodes in AnonSat are sinks, we regularly switch the sink, and a backtracking adversary is unlikely in a conflict zone.

In *Delay*-based solutions, nodes store incoming packets and transmit them after a random period of time, disrupting the chronological order of the packets. This also modifies the traffic pattern, rendering it difficult for a local adversary to trace the origin of the traffic [27], [28]. *Limiting node detectability* temporarily throttles or disables the transmission power of nodes, making it harder for an adversary to receive packets [29], [30]. *Network Coding* uses homomorphic encryption at intermediary nodes to hide traffic flows. After receiving the aggregated data, the sink is then able to reverse the encryption process [31], [32]. However, these three classes of solutions add significant delays to the network that adversely affect low-latency networks, such as AnonSat, especially when the aim is to be compatible with the Internet (cf., Section 4.3).

Dummy Data Sources generate authentic-looking dummy traffic to obscure the authentic traffic. The objective is to prevent an adversary from differentiating between genuine and fabricated traffic [33], [34]. *Cyclic Entrapment* confuses potential adversaries by routing the traffic between nodes with cyclical patterns [35], [36]. *Separate Path Routing* splits data into multiple packets, which will be sent over multiple, non-intersecting paths to the sink. Therefore, the local adversary is only able to capture part of the data [37]. Similarly to introducing delays, these three approaches induce large traffic overheads to the network (e.g., dummy traffic or additional retransmissions). Thus, they are incompatible with the goals of AnonSat (cf., Section 2.2).

Cross Layer Routing utilizes multiple OSI layers to hide information from adversaries [38]. Yet, this is based on the assumption that the adversary may not see certain OSI layers, which we deem impractical. Approaches focusing on *Wireless Mesh Networks* specifically, usually assume a hierarchical network with base stations, mesh routers, and mesh clients. Numerous security features, including location privacy, rely on pseudonyms in the form of public key certificates, either directly distributed by an authority [39] or are self-generated with a domain authority backing it [40]. *In network location anonymization* utilizes hierarchical (e.g., clusters of nodes) pseudonyms or aggregation of traffic to hide the source of the traffic [41], [42]. Both the Wireless Mesh Networks and In network location anonymization approaches assume a hierarchical trust structure in the network, which is not feasible in AnonSat's local network as, e.g., all gateways are set up by citizens and trusted equally.

Emergency Communication Networks. These networks are designed to provide reliable communication during an emergency when other communication infrastructures fail. The research community has since proposed many approaches for establishing a network in case of natural disasters, conflicts, or any adverse situation that prevents typical access to the Internet. In particular, Portmann *et al*. [43] defined the fundamental characteristics an emergency network must have in its design: *privacy, data integrity, authentication, and access control*. The most common emergency networks are considering the use of Locally Deployed Resource Units (LDRU) (e.g., base stations of cellular networks), satellites, ad-hoc networks, or a combination of them. Usually, portable devices span a network, while only a subset of them are actually capable of external means of communication (e.g., satellites) [103], [104].

To establish emergency networks, many recent works focus on using Unmanned Aerial Vehicles (UAVs) [105]. Numerous works in this area leverage *drones* to establish the emergency network. One proposal is to directly leverage the drones as base stations [44]. Other works use drones to span a mesh network back to a static base station. Proposed wireless communication technologies for the mesh range from leveraging WiFi [45], LTE [46], 5G [106], and LoRa [47]. Besides the use of drones, the application of aerostatic balloons found space in the context of emergency networks. One approach is to build an ad-hoc network based on IEEE 802.11j between the ballons [107]. Another approach is to span a multihop WiFi backbone from one area to a satellite-based base station via zeppelin-like balloons [108], while another extends this approach to a mesh network [109]. Besides all the possible proposed designs, multiple recent works are proposing several optimizations, including load balancing between the units [48], [110].

A practical concern is that drones exhibit limited fly times, and thus, coverage. Further, the individual hardware required (i.e., the drones and balloons) is expensive; yet, hundreds or even thousands of units are necessary to operate in an emergency. Moreover, drones and aerostatic balloons are subject to weather conditions (e.g., cannot easily fly in a storm), which is heavily limiting their operative scenario.

Other works focus on deploying an ad-hoc mesh network between base stations that are then put into communication with satellites. Zhou *et al*. [49] develop such a network based on WiFi 2.4 GHz for the network backbone and 5.8 GHz for data transmission. However, due to the limited range of WiFi, this approach requires the devices to be quite close to each other and does not scale well to cover a wide area. Instead, Iapichino *et al*. [50] propose a hybrid system where equipped vehicles (Vehicle Communication Gateways) establish a connection with satellites and users can connect to these mobile gateways. Similarly, Patricelli *et al*. [51] are proposing a MOBSAT access point (that has to be carried by car or helicopter), which provides high-speed data connection to the users through WLAN and WiMAX through GEO satellites. Nevertheless, these approaches assume that numerous, specially equipped vehicles are prepared and ready for use.

In contrast, the target scenario of AnonSat is novel in the context of emergency networks. The continuous expansion of satellite Internet services allows for the deployment of comparably cheap access points. This enables to provide widespread access to many deployed satellite base stations. Therefore, the mentioned works above are not taking the unique problems of this scenario into account. As outlined in the Introduction, AnonSat specifically aims to protect its users from triangulation. Furthermore, our focus is on the accessibility of the design system. Thus, AnonSat does not

require any additional application or hardware installed on the user's device and the gateways are easy to deploy.

9. Conclusion

In this work, we presented AnonSat, the first scheme to address the triangulation of satellite Internet users, and thus, protect them from being targeted. To achieve this, AnonSat leverages a local wireless communication technology to span a network between the gateways, rerouting a client's connection away from the origin gateway. Additionally, AnonSat regularly changes the output gateway for each client to avoid detection of long-lasting connections. We thoroughly discussed different technical aspects, meeting our defined requirements to make AnonSat usable with both existing Internet protocols and widely available hardware. We implemented a prototype demonstrating AnonSat's feasibility and evaluated its performance via network simulation on various real-world data sets.

Acknowledgment

This work was supported by the European Space Operations Centre with the Networking/Partnering Initiative.

References

[1] F. G. Hoffman, *Conflict in the 21st century: The rise of hybrid wars*. Potomac Institute for Policy Studies Arlington, 2007.

[2] Foreign Policy Magazine, "Kill the Messenger," https://foreignpolicy.com/2012/03/03/kill-the-messenger/, 2012.

[3] United Nations News, "Ukraine: Journalists targeted and in danger, warn top rights experts," https://news.un.org/en/story/2022/05/1117462, 2022.

[4] Committee to Protect Journalists, "Journalists and Media Workers Killed in Ukraine," https://cpj.org/data/killed/europe/ukraine/, 2022.

[5] British Broadcasting Corporation, "How Ukraine is winning the social media war," https://www.bbc.com/news/world-europe-63272202, 2022.

[6] The Washington Post, "U.S. quietly paying millions to send Starlink terminals to Ukraine, contrary to SpaceX claims," https://www.washingtonpost.com/politics/2022/04/08/us-quietly-paying-millions-send-starlink-terminals-ukraine-contrary-spacexs-claims/, 2022.

[7] Daily Mail, "SpaceX Starlink internet service has 150,000 daily users in Ukraine as citizens of war-torn parts of the country fight to stay connected, government official reveals," https://www.dailymail.co.uk/sciencetech/article-10781461/SpaceX-Starlink-150-000-daily-users-Ukraine-just-five-weeks-activated.html, 2022.

[8] TeslaRati, "Starlink broke top 100 most downloaded iPhone apps on Wednesday," https://www.teslarati.com/starlink-top-100-iphone-apps-wednesday/, 2022.

[9] Space.com, "Elon Musk says Russia is ramping up cyberattacks on SpaceX's Starlink systems in Ukraine," https://www.space.com/starlink-russian-cyberattacks-ramp-up-efforts-elon-musk, 2022.

[10] Breaking Defense, "SpaceX beating Russian jamming attack was 'eyewatering': DoD official," https://breakingdefense.com/2022/04/spacex-beating-russian-jamming-attack-was-eyewatering-dod-official/, 2022.

[11] Reuters, "EU secures deal on satellite internet system," https://www.reuters.com/business/aerospace-defense/eu-secures-deal-satellite-internet-system-2022-11-17/, 2022.

[12] The Washington Post, "Social media companies push Ukrainian users to add safeguards," https://www.washingtonpost.com/technology/2022/02/26/protecting-identity-socialmedia/, 2022.

[13] Electronic Frontier Foundation, "Satphones, Syria, and Surveillance," https://www.eff.org/deeplinks/2012/02/satphones-syria-and-surveillance, 2012.

[14] Radio Free Europe/Radio Liberty, "10th Anniversary Of Chechen Leader's Death Noted," https://www.rferl.org/a/1067831.html, 2006.

[15] A. Elgamoudi, H. Benzerrouk, G. A. Elango, and R. Landry Jr, "A survey for recent techniques and algorithms of geolocation and target tracking in wireless and satellite systems," *Applied Sciences*, vol. 11, no. 13, p. 6079, 2021.

[16] Wayback Machine / Twitter, "Archived Tweet of John Scott-Railton," https://web.archive.org/web/20220301105339/twitter.com/jsrailton/status/1497745011932286979, 2022.

[17] New York Post, "Elon Musk warns Starlink users in Ukraine could be Russian targets," https://nypost.com/2022/03/04/elon-musk-warns-starlink-users-in-could-be-russian-targets/, 2022.

[18] S. J. Murdoch and P. Zieliński, "Sampled traffic analysis by internet-exchange-level adversaries," in *International workshop on privacy enhancing technologies*. Springer, 2007, pp. 167–183.

[19] M. Yang, X. Gu, Z. Ling, C. Yin, and J. Luo, "An active de-anonymizing attack against tor web traffic," *Tsinghua Science and Technology*, vol. 22, no. 6, pp. 702–713, 2017.

[20] K. Bauer, D. Grunwald, and D. Sicker, "Predicting tor path compromise by exit port," in *2009 IEEE 28th International Performance Computing and Communications Conference*. IEEE, 2009, pp. 384–387.

[21] S. Le Blond, P. Manils, A. Chaabane, M. A. Kaafar, C. Castelluccia, A. Legout, and W. Dabbous, "One bad apple spoils the bunch: Exploiting P2P applications to trace and profile tor users," in *4th USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET 11)*, 2011.

[22] F. Palmieri, "A distributed flow correlation attack to anonymizing overlay networks based on wavelet multi-resolution analysis," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 5, pp. 2271–2284, 2019.

[23] P. Kamat, Y. Zhang, W. Trappe, and C. Ozturk, "Enhancing source-location privacy in sensor network routing," in *25th IEEE international conference on distributed computing systems (ICDCS'05)*. IEEE, 2005, pp. 599–608.

[24] Y. Xi, L. Schwiebert, and W. Shi, "Preserving source location privacy in monitoring-based wireless sensor networks," in *Proceedings 20th IEEE International Parallel & Distributed Processing Symposium*. IEEE, 2006, pp. 8–pp.

[25] R. A. Shaikh, H. Jameel, B. J. d'Auriol, H. Lee, S. Lee, and Y.-J. Song, "Achieving network level privacy in wireless sensor networks," *Sensors*, vol. 10, no. 3, pp. 1447–1472, 2010.

[26] Y. Li and J. Ren, "Preserving source-location privacy in wireless sensor networks," in *2009 6th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*. IEEE, 2009, pp. 1–9.

[27] X. Hong, P. Wang, J. Kong, Q. Zheng *et al.*, "Effective probabilistic approach protecting sensor traffic," in *MILCOM 2005-2005 IEEE Military Communications Conference*. IEEE, 2005, pp. 169–175.

[28] P. Kamat, W. Xu, W. Trappe, and Y. Zhang, "Temporal privacy in wireless sensor networks: Theory and practice," *ACM Transactions on Sensor Networks (TOSN)*, vol. 5, no. 4, pp. 1–24, 2009.

[29] R. Rios and J. Lopez, "Exploiting context-awareness to enhance source-location privacy in wireless sensor networks," *The Computer Journal*, vol. 54, no. 10, pp. 1603–1615, 2011.

[30] R. El-Badry, A. Sultan, and M. Youssef, "Hyberloc: providing physical layer location privacy in hybrid sensor networks," in *2010 IEEE International Conference on Communications*. IEEE, 2010, pp. 1–5.

[31] Y. Fan, Y. Jiang, H. Zhu, and X. Shen, "An efficient privacy-preserving scheme against traffic analysis attacks in network coding," in *IEEE INFOCOM 2009*. IEEE, 2009, pp. 2213–2221.

[32] Y. Fan, J. Chen, X. Lin, and X. Shen, "Preventing traffic explosion and achieving source unobservability in multi-hop wireless networks using network coding," in *2010 IEEE Global Telecommunications Conference GLOBECOM 2010*. IEEE, 2010, pp. 1–5.

[33] Y. Yang, M. Shao, S. Zhu, and G. Cao, "Towards statistically strong source anonymity for sensor networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 9, no. 3, pp. 1–23, 2013.

[34] K. Mehta, D. Liu, and M. Wright, "Location privacy in sensor networks against a global eavesdropper," in *2007 IEEE International Conference on Network Protocols*. IEEE, 2007, pp. 314–323.

[35] Y. Ouyang, X. Le, G. Chen, J. Ford, and F. Makedon, "Entrapping adversaries for source protection in sensor networks," in *2006 International symposium on a world of wireless, mobile and multimedia networks (WoWMoM'06)*. IEEE, 2006, pp. 10–pp.

[36] L. Kazatzopoulos, C. Delakouridis, G. F. Marias, and P. Georgiadis, "ihide: Hiding sources of information in wsns," in *Second International Workshop on Security, Privacy and Trust in Pervasive and Ubiquitous Computing (SecPerU'06)*. IEEE, 2006, pp. 8–pp.

[37] H. Wang, B. Sheng, and Q. Li, "Privacy-aware routing in sensor networks," *Computer Networks*, vol. 53, no. 9, pp. 1512–1529, 2009.

[38] M. Shao, W. Hu, S. Zhu, G. Cao, S. Krishnamurth, and T. La Porta, "Cross-layer enhanced source location privacy in sensor networks," in *2009 6th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*. IEEE, 2009, pp. 1–9.

[39] Y. Zhang and Y. Fang, "Arsa: An attack-resilient security architecture for multihop wireless mesh networks," *IEEE Journal on Selected areas in communications*, vol. 24, no. 10, pp. 1916–1928, 2006.

[40] J. Sun, C. Zhang, Y. Zhang, and Y. Fang, "Sat: A security architecture achieving anonymity and traceability in wireless mesh networks," *IEEE Transactions on Dependable and Secure Computing*, vol. 8, no. 2, pp. 295–307, 2010.

[41] S. Misra and G. Xue, "Efficient anonymity schemes for clustered wireless sensor networks," *International Journal of Sensor Networks*, vol. 1, no. 1-2, pp. 50–63, 2006.

[42] X. Luo, X. Ji, and M.-S. Park, "Location privacy against traffic analysis attacks in wireless sensor networks," in *2010 International Conference on Information Science and Applications*. IEEE, 2010, pp. 1–6.

[43] M. Portmann and A. A. Pirzada, "Wireless mesh networks for public safety and crisis management applications," *IEEE Internet computing*, vol. 12, no. 1, pp. 18–25, 2008.

[44] N. Zhao, W. Lu, M. Sheng, Y. Chen, J. Tang, F. R. Yu, and K.-K. Wong, "Uav-assisted emergency networks in disasters," *IEEE Wireless Communications*, vol. 26, no. 1, pp. 45–51, 2019.

[45] K. G. Panda, S. Das, D. Sen, and W. Arif, "Design and deployment of uav-aided post-disaster emergency network," *IEEE Access*, vol. 7, pp. 102 985–102 999, 2019.

[46] M. Deruyck, J. Wyckmans, W. Joseph, and L. Martens, "Designing uav-aided emergency networks for large-scale disaster scenarios," *EURASIP Journal on Wireless Communications and Networking*, vol. 2018, no. 1, pp. 1–12, 2018.

[47] M. Pan, C. Chen, X. Yin, and Z. Huang, "Uav-aided emergency environmental monitoring in infrastructure-less areas: Lora mesh networking approach," *IEEE Internet of Things Journal*, vol. 9, no. 4, pp. 2918–2932, 2021.

[48] N. Lin, Y. Liu, L. Zhao, D. O. Wu, and Y. Wang, "An adaptive uav deployment scheme for emergency networking," *IEEE Transactions on Wireless Communications*, vol. 21, no. 4, pp. 2383–2398, 2021.

[49] J. Zhou, C. Zhou, Y. Kang, and S. Tu, "Integrated satellite-ground post-disaster emergency communication networking technology," *Natural Hazards Research*, vol. 1, no. 1, pp. 4–10, 2021.

[50] G. Iapichino, C. Bonnet, O. del Rio Herrero, C. Baudoin, and I. Buret, "Advanced hybrid satellite and terrestrial system architecture for emergency mobile communications," in *26th international communications satellite systems conference (ICSSC)*, 2008.

[51] F. Patricelli, J. E. Beakley, A. Carnevale, M. Tarabochia, and D. K. Von Lubitz, "Disaster management and mitigation: the telecommunications infrastructure," *Disasters*, vol. 33, no. 1, pp. 23–37, 2009.

[52] AP News, "US hits Russia with 'war crimes' sanctions, Europe following," https://apnews.com/article/russia-ukraine-kyiv-business-european-commission-united-kingdom-acb86730120a1230b9eb95c3ebdded77, 2012.

[53] Reuters, "ICC judges issue arrest warrant for Putin over war crimes in Ukraine," https://www.reuters.com/world/europe/icc-judges-issue-arrest-warrant-against-putin-over-alleged-war-crimes-2023-03-17/, 2023.

[54] B. Foubert and N. Mitton, "Long-range wireless radio technologies: A survey," *Future internet*, vol. 12, no. 1, p. 13, 2020.

[55] Amazon AWS, "LTE-M," https://docs.aws.amazon.com/whitepapers/latest/implementing-lpwan-solutions-with-aws/lte-m.html, 2022.

[56] K. Mekki, E. Bajic, F. Chaxel, and F. Meyer, "A comparative study of lpwan technologies for large-scale iot deployment," *ICT express*, vol. 5, no. 1, pp. 1–7, 2019.

[57] IMST GmbH, "High range with LoRa® on worldwide 2.4 GHz band," https://wireless-solutions.de/blog/2020/07/24/im282a-high-range-with-lora-on-worldwide-2-4-ghz-band/, 2020.

[58] IEEE Computer Society, "Ieee standard for low-rate wireless networks," 5 2020, IEEE 802.15.4.

[59] B. Watteyne, Thubert, "On forwarding 6lowpan fragments over a multi-hop ipv6 network," 11 2020, RFC 8930.

[60] The Linux kernel development community, "IEEE 802.15.4 Developer's Guide," https://www.kernel.org/doc/html/latest/networking/ieee802154.html, 2022.

[61] A. Bruniaux, R.-A. Koutsiamanis, G. Z. Papadopoulos, and N. Montavont, "Defragmenting the 6lowpan fragmentation landscape: A performance evaluation," *Sensors*, vol. 21, no. 5, p. 1711, 2021.

[62] P. Gimenez, "Static context header compression and fragmentation (schc) over lorawan," 4 2021, RFC 9011.

[63] H. She, Z. Lu, A. Jantsch, D. Zhou, and L.-R. Zheng, "Analytical evaluation of retransmission schemes in wireless sensor networks," in *VTC Spring 2009-IEEE 69th Vehicular Technology Conference*. IEEE, 2009, pp. 1–5.

[64] J. Iyengar and M. Thomson, "Quic: A udp-based multiplexed and secure transport," Internet Requests for Comments, RFC Editor, RFC 9000, May 2021.

[65] J. Iyengar and I. Swett, "Quic loss detection and congestion control," Internet Requests for Comments, RFC Editor, RFC 9002, May 2021.

[66] A. Langley, A. Riddoch, A. Wilk, A. Vicente, C. Krasic, D. Zhang, F. Yang, F. Kouranov, I. Swett, J. Iyengar *et al.*, "The quic transport protocol: Design and internet-scale deployment," in *Proceedings of the conference of the ACM special interest group on data communication*, 2017, pp. 183–196.

[67] Engineering at Meta, "How Facebook is bringing QUIC to billions," https://engineering.fb.com/2020/10/21/networking-traffic/how-facebook-is-bringing-quic-to-billions/, 2022.

[68] P. Watson, "Slipping in the window: Tcp reset attacks," *Presentation at*, 2004.

[69] Ford, Raiciu, Handley, Bonaventure, "Tcp extensions for multipath operation with multiple addresses," 1 2013, RFC 6824.

[70] Ford, Raiciu, Handley, Bonaventure, Paasch, "Tcp extensions for multipath operation with multiple addresses," 3 2020, RFC 8684.

[71] Christoph Paasch, Fabien Duchêne and Gregory Detal, "Multi-Path TCP - Linux Kernel implementation," www.multipath-tcp.org/, 2022.

[72] Apple Support, "Use Multipath TCP to create backup connections for iOS," https://support.apple.com/en-us/HT201373, 2022.

[73] Y. Liu, Y. Ma, Q. D. Coninck, O. Bonaventure, C. Huitema, and M. Kühlewind, "Multipath Extension for QUIC," Internet Engineering Task Force, Internet-Draft draft-ietf-quic-multipath-03, 2022.

[74] Clausen, Jacquet, "Optimized link state routing protocol (OLSR)," 10 2003, RFC 3626.

[75] Clausen, Dearlove, Jacquet, Herberg, "The optimized link state routing protocol version 2," 4 2014, RFC 7181.

[76] R. Kufakunesu, G. P. Hancke, and A. M. Abu-Mahfouz, "A survey on adaptive data rate optimization in lorawan: Recent solutions and major challenges," *Sensors*, vol. 20, no. 18, p. 5044, 2020.

[77] M. Conti, J. Willemsen, and B. Crispo, "Providing source location privacy in wireless sensor networks: A survey," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 3, pp. 1238–1280, 2013.

[78] R. El-Badry, M. Youssef, and M. Eltoweissy, "Hidden anchor: Providing physical layer location privacy in hybrid wireless sensor networks," in *2009 3rd International Conference on New Technologies, Mobility and Security*. IEEE, 2009, pp. 1–5.

[79] J.-P. Sheu, J.-R. Jiang, and C. Tu, "Anonymous path routing in wireless sensor networks," in *2008 IEEE International Conference on Communications*. IEEE, 2008, pp. 2728–2734.

[80] L. Lightfoot, Y. Li, and J. Ren, "Preserving source-location privacy in wireless sensor network using star routing," in *2010 IEEE Global Telecommunications Conference GLOBECOM 2010*. IEEE, 2010, pp. 1–5.

[81] M. G. Reed, P. F. Syverson, and D. M. Goldschlag, "Anonymous connections and onion routing," *IEEE Journal on Selected areas in Communications*, vol. 16, no. 4, pp. 482–494, 1998.

[82] A. El Mougy and S. Sameh, "Preserving privacy in wireless sensor networks using onion routing," in *2018 International Symposium on Networks, Computers and Communications (ISNCC)*. IEEE, 2018, pp. 1–6.

[83] Starlink, "Starlink Specifications," https://www.starlink.com/specifications, 2022.

[84] Raspberry, "Raspberry Pi 3 Model B+," https://www.raspberrypi.com/products/raspberry-pi-3-model-b-plus/, 2022.

[85] Waveshare, "SX1262 868M LoRa HAT," https://www.waveshare.com/wiki/SX1262_868M_LoRa_HAT, 2022.

[86] Semtech, "LoRa® Reference Design for 2.4GHz," https://www.semtech.com/products/wireless-rf/lora-core/sx1280zxxxxgw1, 2022.

[87] J. Petäjäjärvi, K. Mikhaylov, M. Pettissalo, J. Janhunen, and J. Iinatti, "Performance of a low-power wide-area network based on lora technology: Doppler robustness, scalability, and coverage," *International Journal of Distributed Sensor Networks*, vol. 13, no. 3, p. 1550147717699412, 2017.

[88] M. Swain, D. Zimon, R. Singh, M. F. Hashmi, M. Rashid, and S. Hakak, "Lora-lbo: an experimental analysis of lora link budget optimization in custom build iot test bed for agriculture 4.0," *Agronomy*, vol. 11, no. 5, p. 820, 2021.

[89] data.goc.hk, "Information on Wi-Fi.HK locations," https://data.gov.hk/en-data/dataset/hk-ogcio-ogcio_hp-wi-fi-hk-locations, 2022.

[90] NYC OpenData, "NYC Wi-Fi Hotspot Locations," https://data.cityofnewyork.us/City-Government/NYC-Wi-Fi-Hotspot-Locations/yjub-udmw, 2022.

[91] GovData, "Freifunk Rhein-Neckar," https://www.govdata.de/web/guest/daten/-/details/freifunk-rhein-neckar, 2022.

[92] data.gov.au, "Wireless hotspot locations - Libraries, Parks and Public spaces," https://data.gov.au/dataset/ds-brisbane-17fb3724-ecfc-4802-8f16-62839fb73fc0/details, 2022.

[93] Paris Data, "Paris Wi-Fi - Sites disposant du service," https://parisdata.opendatasoft.com/explore/dataset/sites-disposant-du-service-paris-wi-fi/information/?disjunctive.cp&disjunctive.etat2, 2022.

[94] data.gov.au, "AdelaideFree Wi-Fi Access Point Locations," https://data.gov.au/dataset/ds-sa-8b8040ac-c71c-4c9b-b361-be84b4cd3dc6/details, 2022.

[95] data.europe.eu, "Public access free WiFi," https://data.europa.eu/data/datasets/public-access-free-wifi/?locale=de, 2020.

[96] data.gv.at, "Hotspot - Standorte (Linz)," https://www.data.gv.at/katalog/dataset/stadt-linz_hotspotstandorte, 2022.

[97] T. ANDREW S and W. DAVID J, "Computer networks fifth edition," 2011.

[98] OMNeT++, "OMNeT++ Discrete Event Simulator," https://hub.packtpub.com/iot-forensics-security-connected-world/, 2018.

[99] D. Chaum, "The dining cryptographers problem," *Journal of Cryptology*, vol. 1, pp. 65–75, 1988.

[100] A. Serjantov and G. Danezis, "Towards an information theoretic metric for anonymity," in *Privacy Enhancing Technologies: Second International Workshop, PET 2002 San Francisco, CA, USA, April 14–15, 2002 Revised Papers 2*. Springer, 2003, pp. 41–53.

[101] Jonathan's Space Pages, "Starlink Launch Statistics," https://planet4589.org/space/con/star/stats.html, 2022.

[102] A. Mpitziopoulos and D. Gavalas, "An effective defensive node against jamming attacks in sensor networks," *Security and Communication Networks*, vol. 2, no. 2, pp. 145–163, 2009.

[103] P. D. Pradeep, B. A. Kumar *et al.*, "A survey of emergency communication network architectures," *International Journal of u-and e-Service, Science and Technology*, vol. 8, no. 4, pp. 61–68, 2015.

[104] V. Y. Kishorbhai and N. N. Vasantbhai, "Aon: a survey on emergency communication systems during a catastrophic disaster," *Procedia computer science*, vol. 115, pp. 838–845, 2017.

[105] S. Debnath, W. Arif, S. Roy, S. Baishya, and D. Sen, "A comprehensive survey of emergency communication network and management," *Wireless Personal Communications*, pp. 1–47, 2021.

[106] Y. Gao, J. Cao, P. Wang, J. Yin, M. He, M. Zhao, M. Peng, S. Hu, Y. Sun, J. Wang *et al.*, "Intelligent uav based flexible 5g emergency networks: Field trial and system level results," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2020, pp. 138–143.

[107] Y. Shibata, Y. Sato, N. Ogasawara, and G. Chiba, "A disaster information system by ballooned wireless adhoc network," in *2009 international conference on complex, intelligent and software intensive systems*. IEEE, 2009, pp. 299–304.

[108] H. Suzuki, Y. Kaneko, K. Mase, S. Yamazaki, and H. Makino, "An ad hoc network in the sky, skymesh, for large-scale disaster recovery," in *IEEE vehicular technology conference*. IEEE, 2006, pp. 1–5.

[109] H. Okada, H. Oka, and K. Mase, "Network construction management for emergency communication system skymesh in large scale disaster," in *2012 IEEE Globecom Workshops*. IEEE, 2012, pp. 875–880.

[110] H. Niu, X. Zhao, and J. Li, "3d location and resource allocation optimization for uav-enabled emergency networks under statistical qos constraint," *IEEE Access*, vol. 9, pp. 41 566–41 576, 2021.

Appendix A.
Distance To Origin Evaluation

The data sets with an assumed 5 km maximum range between nodes in Figure 4 (a) shows the limitations of the different data sets. In contrast, very densely populated sets that do not cover a lot of area, like Adelaide or Paris, show no increase in distance. This effect is primarily dependent on the set's covered area. For example, if we roughly draw a circle around the nodes provided by each data set, we get a diameter of ∼5 km for Adelaide and ∼15 km for Paris, while Hong Kong has ∼50 km and Rhein-Neckar even ∼100 km. Data sets, like New York City (∼30 km) or Brisbane (∼20 km) that cover a medium-sized area, show a diminishing return in terms of an increased max_hops. We also found that some sets have more uniformly spread nodes over the covered area, while others have a decreasing density from the center to the borders of the covered area. For example, the more uniformly dense Paris reaches the limit sooner than the unevenly dense Brisbane. With the Adelaide data set, we cannot get past 5 km and when considering the random selection of nodes with a more dense center, our average distance from the origin is naturally quite low.

Figure 4 (b) shows similar measurements for the 1 km range case. We can see the effects analogously to the 5 km results, just with a significantly reduced distance from the origin. Note that some data sets, especially Rhein-Neckar, have severely reduced node count when considering a connected network with a 1 km range (cf. Section 6.2.2). The effect of this can be seen in this graph. Overall, there is an inherent trade-off for setting the max_hops parameter, as setting it higher leads to an increased distance; yet, more hops for the communication will lead to more delays and overhead in the network. Our measurements show that it is crucial to take the underlying spread and density of nodes into account when choosing max_hops. For the following simulation results, we assumed max_hops = 3 for the 5 km range networks and max_hops = 5 for the 1 km range networks.

Appendix B.
Result Graphs

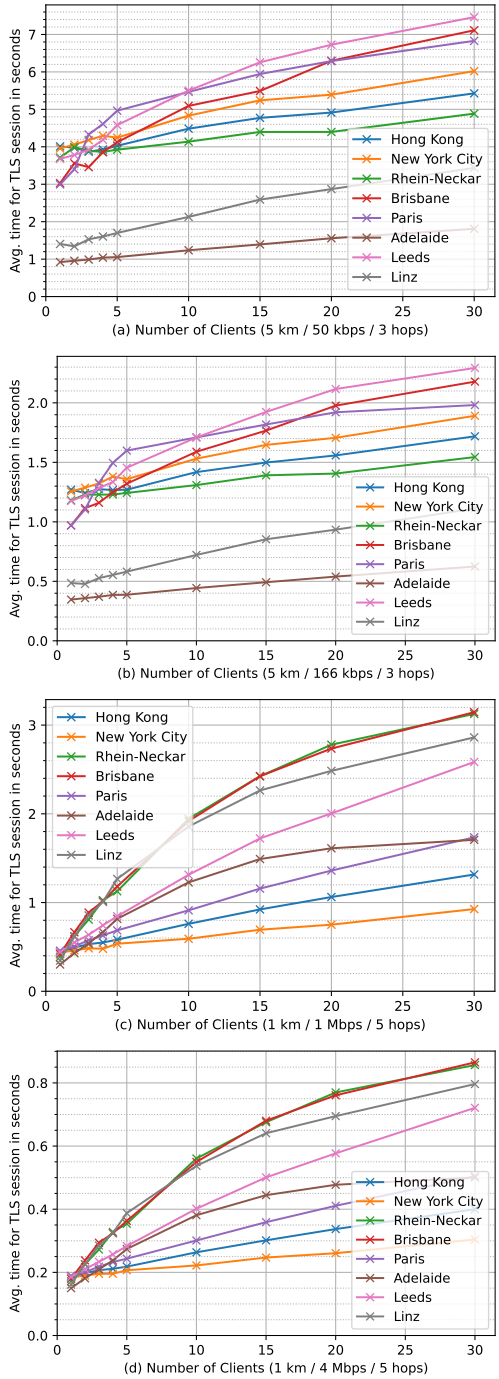


Figure 5. Graphs showing the measurements of TLS session delay in different network types for all data sets.

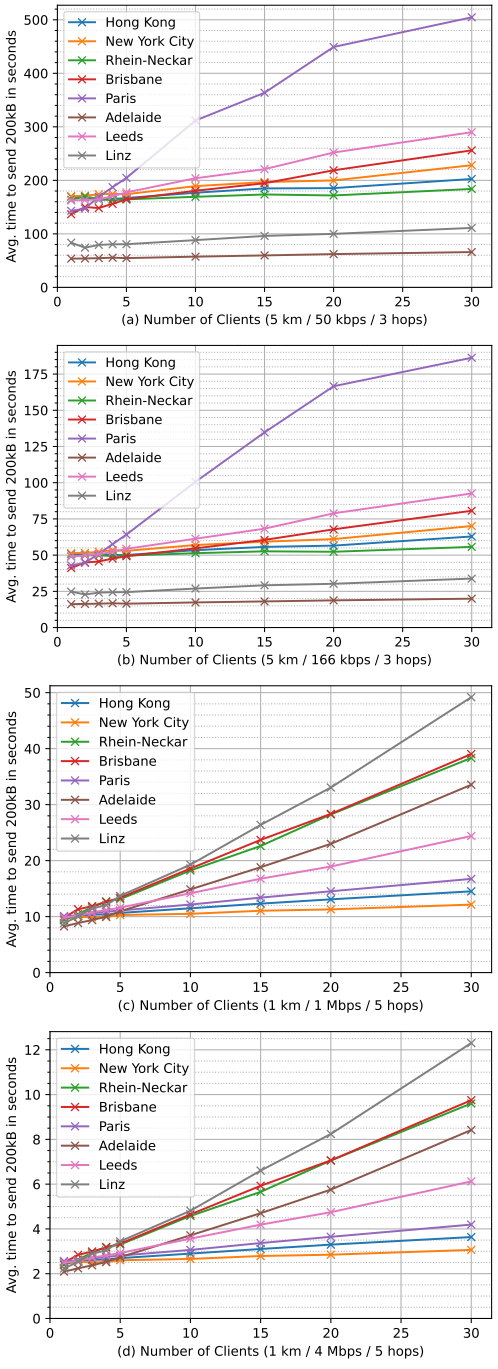


Figure 6. Graphs showing the measurements of length to upload 200 kB image in different network types for all data sets.