University of Padua

Doctorate Degree in Brain, Mind and Computer Science

Curriculum in Computer Science

Series XXXIV

Authentication methods:

Novel attacks and defenses

Coordinator
Prof.ssa Anna Spagnolli
Supervisor
Prof. Mauro Conti
Co-supervisor
Prof. Giuseppe Sartori

Ph.D. Candidate
Stefano Cecconello

University of Padova, Italy

Acknowledgments

I want to express my sincere gratitude to my advisor Prof. Mauro Conti for his continuous support, motivation, opportunities, and guidance in my research career. It is an honor working with him.

I sincerely thank my Ph.D. co-advisor, Prof. Giuseppe Sartori, who gave me the opportunity to understand how the notions I learned during my doctorate are interdisciplinary and widely applicable.

Many thanks to my parents for their continuous and irreplaceable support. A special thanks to my fiance Tiziana that, every day of my doctorate, helped me find my way.

I want to thank all my friends, colleagues, and officemates at the University of Padua. A particular thanks go to Ankit Gangwal, Matteo Cardaioli, Luca Pasa, and Daniele Lain, who helped me during these three years thanks to constant discussions and valuable advice. Moreover, a sincere thanks to the other members of the SPRITZ Research Group.

Stefano Cecconello
Padova, February 25, 2022

Abstract

Nowadays, authentication systems are widespread in our devices. They protect the security of our systems, guaranteeing that only authorized people have access to reserved services and data. Thanks to this role, authentication systems made their first appearance in the '60s, with the diffusion of the first computers in universities. Over the years, these systems have evolved. If the first authentication systems were based only on passwords, they are considerably advanced today. In particular, with the widespread diffusion of smartphones, authentication systems became commonly used, focusing on user-friendly systems such as biometrics. This evolution generated a market that is growing strongly, and it is expected to increase by 15% again in the next few years, with revenues of hundreds of billions of dollars.

Usability is not the only factor influencing the evolution of authentication systems. Their safety also determines their evolution over time. If an authentication method guarantees the security of a system, it is also true that it is the first to suffer from cyber-attacks. After the appearance of new authentication technologies on the market, it is frequent to notice the spread of new methods to bypass the security of the novel technology. Research in these areas becomes fundamental: on one side, to discover new authentication systems that can improve the usability of our devices, and on the other, to anticipate possible vulnerabilities and make these systems more secure.

This thesis investigates the security of authentication methods, and it is composed of two logical parts that focus on: (i) the development of novel attacks against existing authentication methods, (ii) the development of novel authentication methods. In the first part of this thesis, we focus on attacks against authentication methods. In particular, we show the effectiveness of three attacks against the security of password and PIN authentication methods. The first work shows how an attacker can use the audio recorded

during a VoIP call to infer the keys pressed by a victim. We showed how this attack could be used to infer passwords successfully. The other two works of the first part consist of two distinct methods to steal secret codes from ATM PIN pads. For all these attacks, we propose effective countermeasures showing how important it is to actively participate in research in this field to improve the security of authentication systems. In the second part of this thesis, we explored authentication systems from the perspective of both users and devices authentication. In particular, we investigated a novel biometrics method based on recognizing the user's chewing movement and a new authentication method to ensure the security of legacy cyber-physical systems. We present our experimental results for the former, showing how our method can guarantee user security by keeping a user-friendly environment. For the latter, we present our authentication method showing how it can improve the security of legacy infrastructures, keeping costs down.

Contents

Chapter 1

Introduction

In computer science, authentication is the process by which the identity of a computer, software, or user is verified and authenticated to be used for one or more services. The first authentication system was born in the '60s with the spread of computers in universities as a computing tool shared between several students and professors. Initially, these computers did not have any access control system, and users could access all files on these machines. In 1961, this precarious condition led *Fernando J. Corbató* to develop the first password login system to secure access to the files of the Compatible Time-Sharing System (the operating system used by MIT). This first rudimentary system did not provide any security for passwords (stored in clear text on disk), but formed the conceptual basis for the followed systems. In the late '60s and early '70s, programmers became aware of the storing passwords in cleartext problem. In the early '70s, the cryptographer *Robert Morris* developed the first password storage system based on hash functions. A few years later, this password encryption scheme was included in the 6th version of UNIX, building the first password storage system as we know it today. Even though the introduction of cryptography significantly improved the security of computer systems, new threats continued to appear, and authentication systems continued their development. In the early '80s, to overcome the problem of password reuse, One-Time Password (OTP) systems were introduced, which generated random passwords to be used mainly as a second authentication factor. However, the real game-changer was the introduction of asymmetric cryptography, which significantly changed the management of secure communications. The first Public Key Infrastructure (PKI) was

created thanks to the asymmetric algorithms, allowing two users to authenticate and identify each other.

Over the years, the use of authentication systems has changed significantly. If in the '60s they were only used by professionals, they have become part of everyday life over time. This phenomenon has reached its peak in the last decade, with the widespread diffusion of smartphones. Thanks to these devices, there has been a widespread diffusion of biometric authentication systems (particularly fingerprint and face recognition), with the consequent generation of a large expanding market.

In conclusion, in about 60 years, we have seen the rapid development of digital authentication. Started as simple passwords saved in plain text, it evolved into advanced systems widespread in our everyday lives and equipped with sophisticated biometrics.

1.1 Research Motivation and Contribution

As described in the previous section, authentication systems have seen progressive development over the decades. A particular case concerns the authentication of people, which is based on the assumption that a person to be authenticated knows a secret: the authentication system will recognize as legitimate a user who can provide this secret. There are three different kinds of secrets to be provided to a digital system: (i) "A thing you know", typically a password or a PIN; (ii) "A thing you have", such as a smartphone, a card with a magnetic stripe or microchip (e.g., a credit card), a physical object such as a token, or a certified application; (iii) "A thing you are", which includes all the unique characteristics of the human body that can define the identity of a person. This last category of authentication methods is called biometrics and includes technologies like fingerprint, voice stamp, retina, and/or iris. Thanks to their transparency for the user, investments in biometrics have seen a significant increase in recent years. The user no longer has to enter a card or password actively but only has to place his finger on a reader or, even better, a camera independently scans his eye and gives him access.

In this thesis, we investigate the security and privacy of authentication methods. Our research work includes the following two parts:

- *Attack against existing authentication methods* proposes practically feasible attacks that can be used to violate the users' security and privacy, but also possible countermeasures against such attacks.

- *Novel authentication methods* presents two novel authentication systems developed for smartphones (a novel biometric) and cyber-physical systems, respectively.

In what follows, we briefly introduce the parts mentioned above and highlight our contributions. In this thesis, some passages have been quoted verbatim, and some figures have been reused from the works [23, 35, 37, 39], all coauthored by the author of this thesis.

1.1.1 Attack Against Existing Authentication Methods

As we have discussed so far, it is a constant challenge between authentication methods and new attacks. Since the first authentication methods were developed, cyber-attackers began to imagine techniques to illicit access to protected systems. In particular, attacks against authentication methods have multiplied in recent decades. This proliferation is due to the great variety of authentication methods available to date (e.g., passwords, unlock signs, pins, biometrics), for which there are specific attacks that exploit the weaknesses of a given system. Therefore, if initially only brute force or hash table attacks were used to steal passwords, then attacks became more specialized, exploiting specific hardware vulnerabilities. An example consists in using a spray to freeze a computer's RAM to read it later and extract the password (at this point saved in plain text in the volatile memory). Still, to carry out attacks against systems that use an unlock sign, the residual grease on the screen has been exploited to infer the secret symbol [13]. Other attacks have been developed to recover the fingerprints that a victim leaves on solid objects, but the same can be reconstructed from photos of the victim's hands recovered on the internet using only open-source intelligence techniques.

In the first part of this thesis, we developed new methods to attack existing authentication methods. In particular, we focused on attacks against passwords and PINs (specifically, ATM PINs).

1.1.1.1 Skype & Type: Keyboard Eavesdropping in Voice-over-IP

Over the years, several keyboard eavesdropping techniques have been developed, which allow stealing sensitive information such as passwords, PINs, or simply private texts. These techniques exploit the emanations of the keyboards as the electromagnetic ones, acoustic, and the vibrations emitted during the typing. Some other techniques exploit the residual thermal radi-

ation that a user leaves on a keyboard or analyze the inter-keystroke timing to leak information about the pressed keys. Among all these techniques, one of the most powerful is undoubtedly the one developed by Asonov [10], which allows discriminating between the keys pressed by the victim based on the characteristic sound that each of these keys emits. However, even if this attack is effective, one of its main limitations is the need for physical proximity. Indeed, to perform the attack, a microphone must be placed in the vicinity of the target keyboard, limiting the feasibility in a real context.

Contributions: In Chapter 2, we present a novel technique for acoustic eavesdropping that exploits VoIP calls. We propose a method that removes the physical limitation of previous attacks by exploiting the victim microphone. During the VoIP call, the attacker records the call audio and uses this audio to infer the keys pressed by the victim. Thanks to this side-channel, the attacker can infer sensitive information typed by the victim during the VoIP call (e.g., sensible text or password). Our paper also demonstrates how to use our attack by exploiting alternative microphones, such as headphones connected to the victim's computer or a smartphone on the same table as the keyboard. We also show how it is possible to use this attack to perform word recognition and password reconstruction. Thanks to our attack, the computation times for a brute force attack can be drastically reduced to the point of making it feasible to use it in a real context. Finally, we conclude by proposing a potential countermeasure to our attack that can improve the security of VoIP without worsening the call quality.

1.1.1.2 Hand Me Your PIN! Inferring ATM PINs of Users Typing with a Covered Hand

One of the most commonly used attacks against PIN-based authentication systems is placing a hidden camera to record the PINs entered by victims. This technique was mainly used to attack ATMs, where it is relatively easy to hide a camera. However, these attacks became rapidly popular among ATM users, who adopted countermeasures. One of the most popular hides the entered PIN by covering the PIN pad using the non-typing hand. This technique, although simple, is highly effective in stopping the hidden camera attack.

Contributions: In Chapter 3, we present a novel kind of PIN security attack on ATMs. Our method is based on the assumption that the user covers the PIN pad while typing. Thanks to the testbed we built, we collected videos of 58 people entering PINs on an ATM PIN pad. The results we obtained show that it is possible to reconstruct the PIN entered by the victim

even though the view on the PIN pad is totally or almost totally covered. To demonstrate that this is not a trivial task, we also conducted a test to compare human performance with our algorithm performance. These results showed that our algorithm outperforms humans' capabilities, demonstrating how this task is difficult for an attacker without proper support. We also conducted a study to show the most effective covering strategies that a user might adopt, and we proposed possible countermeasures to our attack. Moreover, we made available publicly the dataset we collected, hoping that this will be useful for future research in this area and the security of ATMs.

1.1.1.3 We Can Hear Your PIN Drop: An Acoustic Side-Channel Attack on ATM PIN Pads

Attacks that exploit video recordings of PIN pads have become more widespread over time due to their great simplicity of use. However, this kind of attack requires placing a camera directly on-site, limiting its applicability in many real-world contexts. This limitation led to the development of several attacks specialized on PIN pads. Indeed, this restricted category of keyboards has a more straightforward layout than a standard computer keyboard. Besides favoring the performance of the attacks, this simplified layout allows the use of a series of techniques otherwise not feasible on large keyboards. To date, attacks against PIN pad exploit heat signatures, inter-keystroke timing, and even a combination of these.

Contributions: In Chapter 4, we propose a novel attack to infer keystrokes in an ATM PIN pad. This attack exploits a microphone hidden in the vicinity of the ATM to record the keys pressed by the victim. The main advantages of our attack are the high effectiveness and the significant distance between the hidden microphone and the target PIN pad. These two aspects combined make this attack a severe threat for ATMs. In the experiments we conducted, we also investigated the performance of our attack in realistic conditions by simulating its performance in environments with different levels of noise. Results showed how our attack is resilient even in realistic non-optimal cases. Finally, we proposed several countermeasures, thus improving the security of existing ATMs against this threat.

1.1.2 Novel Authentication Methods

Thanks to the development of new sensors, our smartphones gradually integrated new authentication methods in the last decade. Among the most popular, we find fingerprint recognition even if, in recent years, the use of facial recognition and retinal scanning systems had a substantial increase.

One of the research directions investigated in this thesis concerns the study of biometrics authentication methods. A particular emphasis is placed on behavioral biometrics, which authenticates a user based on the unique behavioral pattern when interacting with the system. These authentication methods can consider a wide range of parameters, which include how a user: interact with the keyboard (keystroke dynamics), move the mouse, answers a call [44], or moves his eyes in certain situations [53]. The second part of this thesis focused on implementing new authentication methods and improving existing methods. We investigated authentication systems from the perspective of both user and device authentication. In our first work, we developed a new user-friendly authentication method falling into the biometrics category, while in the second, we developed an authentication method for securing an unsafe legacy industrial protocol that is still widely used today.

1.1.2.1 Eathentication: A Chewing-based Authentication Method

In the last years, biometric authentications have been discovering a solid development thanks to their high transparency to the user and good reliability. Transparency is given by the intrinsic nature of these methods, which only require the user to interact with the device as they always do. Security can be given by two factors: (i) the ability of these methods to uniquely recognize a person based on physical characteristics that are difficult to violate or reproduce, (ii) the intrinsic possibility of some of these methods to be used as behavioral biometrics to authenticate a user on an ongoing basis. In particular, in the case of behavioral biometrics, this implies that an impostor should be able to correctly impersonate the victim for the entire period he uses the device and not only be able to authenticate at login time.

Contributions: In Chapter 5, we present a novel behavioral biometrics authentication method that relies on authenticating a person based on the moments of their external ear channel during chewing. To ensure this method to be user-friendly, we investigated its performance with both users chewing food and users simulating an empty mouth chewing. Smartphones could use this system as a secure unlocking system integrated into the earphones. Indeed, intelligent earphones already provide unlocking features, but their security level is shallow since they do not authenticate the user. Our method aims to fill this gap by offering a simple to use and secure system. In this regard, we initiated a patent application to protect the in-

tellectual property of this method (Patent n. 102020000001192, Publication number WO2021148932A1).

1.1.2.2 TAMBUS: A Novel Authentication Method Through Covert Channels for Securing Industrial Networks

In recent decades, information systems have progressively merged with industrial systems, giving rise to what is now called Cyber-Physical Systems (CPS). This integration became more and more widespread over the years, thus producing a security gap in the industrial world. Indeed, to integrate novel features for legacy CPS devices, often, these have been integrated into more modern networks. This innovation process has not always considered the risks that these integrations could lead to, thus exposing critical systems to threats not foreseen at the time of their design. In particular, two crucial points are the use of legacy communication protocols and the presence of outdated devices: the first because they may lack security features, and the second because confidential information is stored in unsecured hardware. All these vulnerabilities can be a risk for the security of CPSs, exposing them to possible cyber-attacks. Nowadays, these problems are pervasive, as updating industrial plants, although simple from a strictly IT point of view, is problematic for the high costs. What drives up costs is often the need to interrupt the continuous production of facilities to replace and test new hardware: these operations must be planned years in advance and require enormous resources.

Contributions: In Chapter 6, we present our new authentication method for MODBUS/TCP. MODBUS is an industrial protocol created at the end of the '70s that, thanks to its widespread use in CPS it became a de-facto standard over time. This protocol is so widely integrated into manufactories that updating the environment is economically disadvantageous for many companies. For this reason, its version over TCP is still widely used in industrial systems, even though there are new protocols more secure and performing to date. MODBUS is, in fact, a protocol in the clear, which does not provide cryptography features to send messages over the TCP network. Therefore, anyone with access to the network can modify the messages sent with potentially catastrophic consequences for the system. Since legacy devices only support MODBUS, our approach introduces a new protocol that improves system security and is fully backward compatible with the legacy version. Our proposal exploited two covert channels to transfer security information over the MODBUS protocol transparently. To the best of our knowledge, this is the first approach that proposes a solution that is at the

same time backward compatible with MODBUS and not implemented in a secure by obscurity design.

1.2 Publications

This section reports the list of publications produced from the works presented in this thesis. These manuscripts have been produced during my Ph.D. and published in peer-reviewed journals, conferences, and workshops. A complete list of published and currently submitted articles is listed in chronological order as follows: Section 1.2.1 lists journal articles, Section 1.2.2 lists conference and workshop papers, and Section 1.2.3 lists applied patents.

1.2.1 Journal Publications

J1 Stefano Cecconello, Alberto Compagno, Mauro Conti, Daniele Lain, & Gene Tsudik. Skype & Type: Keyboard Eavesdropping in Voice-over-IP. *ACM Transactions on Privacy and Security (TOPS)*, 22 columns (4), 1-34, 2019. DOI:10.1145/3365366. **(JCR IF 2018: 2.103; IT-ANVUR Class: 1)**

J2 Giuseppe Bernieri, Stefano Cecconello, Mauro Conti, & Gianluca Lain. TAMBUS: A novel authentication method through covert channels for securing industrial networks. *Elsevier Computer Networks*, 183, 2020. DOI:10.1016/j.comnet.2020.107583. **(JCR IF 2020: 4.200; IT-ANVUR Class: 1)**

1.2.2 Conference and Workshop Publications

C1 Mattia Carlucci, Stefano Cecconello, Mauro Conti, & Piero Romare. Eathentication: A Chewing-based Authentication Method. *2020 IEEE Conference on Communications and Network Security (CNS)*, Avignon, France, 2020, pp. 1-9. DOI: 10.1109/CNS48642.2020.9162343. **(MA: C; Acceptance rate: 27.8%)**

C2 Matteo Cardaioli, Stefano Cecconello, Luca Pajola, & Federico Turrin. Fake news spreaders profiling through behavioural analysis. *In CLEF Labs and Workshops, Notebook Papers. 2020.*

C3 Matteo Cardaioli, Stefano Cecconello, Merylin Monaro, Giuseppe Sartori, Mauro Conti, & Graziella Orrù. Malingering Scraper: A novel

framework to reconstruct honest profiles from malingerer psychopathological tests. *In Proceedings of the 28th International Conference on Neural Information Processing (ICONIP), Virtual, December 8-12, 2021. DOI: 10.1007/978-3-030-92310-5_50.* **(GGS: 3/B; CORE:B; MA:B)**

C4 Kiran Balagani, Matteo Cardaioli, Stefano Cecconello, Mauro Conti, & Gene Tsudik. We Can Hear Your PIN Drop: An Acoustic Side-Channel Attack on ATM PIN Pads. Submitted at *European Symposium on Research in Computer Security.*

C5 Matteo Cardaioli, Stefano Cecconello, Mauro Conti, Stjepan Picek, Simone Milani, & Eugen Saraci. Hand Me Your PIN! Inferring ATM PINs of Users Typing with a Covered Hand. *2021 In Proceedings of the 31th USENIX Security Symposium (USENIX Security 2022), Boston, USA.* **(GGS: 1/A++; CORE: A++; LiveSHINE: A++; MA: A++)**

1.2.3 Patents

P1 Mattia Carlucci, Stefano Cecconello, Mauro Conti, & Piero Romare. Method of authenticating a user through analysis of changes in the external ear canal. *IT Patent n. 102020000001192, 2020. Publication number WO2021148932A1 (29/07/2021)*

Part I

Attack Against Existing Authentication Methods

Chapter 2

Skype & Type: Keyboard Eavesdropping in Voice-over-IP

Circumvention of cryptographic-based data protection techniques usually requires compromising one of the end-hosts, to be able to capture plaintext before it is encrypted. A more convenient way of capturing plaintext before encryption, thus avoiding the needs of compromising a system, is eavesdropping on physical emanations: observing unintentional leakage of emanations which happen during the regular devices' operations. First discovered in 1943 [62], this mechanism has been proven to be applicable to several types of emanations: electromagnetic [134], visual [16], tactile [123], and acoustic [10, 63].

Convenient targets for physical eavesdropping attacks are I/O peripherals (e.g., keyboards, mice, touch-screens and printers), as their emanations directly leak information on the unencrypted input or output text. In particular, exploitation of keyboard acoustic emanations proved effective in reconstructing the typed input. In this specific class of eavesdropping attacks, an adversary learn what a victim is typing by analyzing the sound produced by the keystrokes. Typically, sounds are recorded either directly, using microphones [10, 21, 67, 69, 95, 103, 135, 147, 149], or by exploiting various sensors (e.g., accelerometers [102, 138]) to re-construct the same acoustic information. Once collected, the audio stream is typically analyzed using techniques, such as supervised [10, 67, 69, 103] and unsupervised [21, 149] machine learning, or triangulation [95, 135, 147]. The final result is a full or partial reconstruction of the victim's input.

In the past years, all proposed keyboard acoustic eavesdropping attacks required a compromised (i.e., controlled by the adversary) microphone near the victim's keyboard [10, 21, 67, 69, 95, 103, 135, 147]. However, requiring physical access strongly limits applicability of such attacks, thus reducing their real-world feasibility. For this reason, recent proposals [8, 43] relaxed the physical proximity requirement, exploiting Voice-over-IP (VoIP) applications to move the adversary in a remote-setting scenario. Premise to this attack, called *Skype & Type attack* (or *S&T attack* for short), is the observation that people involved in VoIP calls often engage in secondary activities, such as: writing email, contributing their "wisdom" to social networks, reading news, watching videos, and even writing research papers. Many of these activities involve using the keyboard (e.g., entering a password). VoIP software automatically acquires all acoustic emanations, including those of the keyboard, and transmits them to all other parties involved in the call. If one of these parties is malicious, it can determine what the user typed based on keystroke sounds. Such an adversary is realistic: it is not always the case that two parties engaged in a VoIP call have mutual trust. For example, such a situation might happen between lawyers on opposite sides of a legal case, or negotiators for different parties. Moreover, the pervasiveness of VoIP software provides to an attacker a huge attack surface, hard to achieve with previous approaches. Considering Microsoft Skype alone, the number of active monthly user is about 300 million.

Contributions. In this chapter, we discuss the following contributions:

- We demonstrate *S&T attack* based on remote keyboard acoustic eavesdropping over VoIP software, with the goal of recovering text typed by the user during a VoIP call with the attacker. *S&T attack* can also recover random text, such as randomly generated passwords or PINs. We take advantage of spectral features of keystroke sounds and analyze them using supervised machine learning algorithms.

- We evaluate *S&T attack* over a very popular VoIP software: Skype. We designed a set of attack scenarios that we consider to be more realistic than those used in prior results on keyboard acoustic eavesdropping. We show that *S&T attack* is highly accurate with minimal profiling of the victim's typing style and keyboard. It remains quite accurate even if neither profiling is available to the adversary. Our results show that *S&T attack* is very feasible, and applicable to real-world settings under realistic assumptions. *S&T* allows to greatly speed up brute-force cracking of random passwords. If the target text is not random,

S&T exploits contextual information (such as the typing language) to recognize words with very high accuracy. Moreover, experiments with Google Hangouts indicate that it is likely susceptible to *S&T attack* as well.

- We show, via extensive experiments, that *S&T attack* works well with different common and inexpensive recording devices, on a great variety of typing styles and speed, and is robust to VoIP-related issues, such as limited available bandwidth that degrades call quality, as well as human speech over keystroke sounds.
- Based on the insights from the design and evaluation phases of this work, we propose a countermeasure to *S&T* and similar attacks that exploit spectral properties of keystroke sounds. Our proposed countermeasure is transparent, does not impact severely the quality of the voice during the call, and is able to disrupt spectral features — making previous data collected by an adversary useless.

Difference with Preliminary Version. The novel contributions of this work, compared to the preliminary version in [43], lie in a greatly extended experimental evaluation, and in improvements to the performance of *S&T* and of our proposed countermeasure. Regarding the experimental evaluation, our preliminary work considered only laptop keyboards and laptop microphones. We now also consider (i) different victim device setups, such as external membrane and mechanical keyboards; and (ii) different recording devices, such as victims that use a smartphone for their VoIP calls, or use a headset microphone, a typical setup in enterprise. Additionally, we shed light on the impact of typing speed, showing that even for the fastest typist *S&T* is still effective, and greatly expand the number of users in our experiments. Regarding the performance of *S&T*, we improve the recognition of non-random text by exploiting contextual information such as the typing language. Finally, we refine our countermeasure and are now able to successfully prevent *S&T attack* in a transparent way, without impacting the quality of the call.

Organization. Section 2.1 overviews related literature and state-of-the-art on keyboard eavesdropping. Next, Section 2.2 describes the system model for our attack and various attack scenarios. Section 2.3, presents *S&T attack*. Section 2.4 describes how we collected the data that we then use to evaluate *S&T attack* in Section 2.5. In Section 2.6 we show some practical

applications of *S&T attack*. Finally, Section 2.7 proposes some potential countermeasures, and Section 2.8 summarizes the chapter and overviews future work.

2.1 Related Work

Eavesdropping on keyboard input is an active and popular area of research. This section begins by overviewing attacks that rely strictly on acoustic emanations to recover the victim's typed text and then summarizes results that study eavesdropping on other physical emanations, such as the WiFi signal, and surface vibrations. For a complete treatment of keyboard side channel attacks, we refer to the recent survey in [107].

Attacks Using Sound Emanations. Research on keyboard acoustic eavesdropping started with the seminal paper of Asonov and Agrawal [10] who showed that, by training a neural network on a specific keyboard, good performance can be achieved in eavesdropping on the input to the same keyboard, or keyboards of the same model. This work also investigated the reasons for this attack and discovered that the plate beneath the keyboard (where the keys hit the sensors) has a drum-like behavior. This causes the sound produced by different keys to be slightly distinct. Subsequent efforts can be divided based on whether they use statistical properties of the sound spectrum or timing information.

Approaches that use statistical properties of the spectrum typically apply machine learning, both supervised [10, 67, 69, 103] and unsupervised [21, 149] versions.

Supervised learning techniques require many labeled samples and are highly dependent on: (1) the specific keyboard used for training [10], and (2) the typing style [67, 69]. Such techniques use Fast Fourier Transform (FFT) coefficients and neural networks to recover text that can also be random. Overall, supervised learning approaches yield very high accuracy. However, this comes at the price of strong assumptions on how the data is collected: obtaining labeled samples of the acoustic emanations of the victim on his keyboard can be difficult or unrealistic.

Unsupervised learning approaches can cluster together keys from sounds, or generate sets of constraints between different key-presses. It is feasible to cluster key sounds and assign labels to the clusters by using relative letter frequency of the input language [149]. It is also possible to generate sets of constraints from recorded sounds and select words from a dictionary that match these constraints [21]. Unsupervised learning techniques have

the advantage that they do not require ground truth. However, they make strong assumptions on user input, such as obtaining many samples, i.e., emanations corresponding to a long text [149], or requiring the targets to be dictionary words [21]. They are less effective when keyboard input is random.

An alternative approach involves analyzing timing information. One convenient way to exploit timing information is using multiple microphones, such as the ones on mobile phones [95, 135, 147], and analyze the Time Difference of Arrival (TDoA) information to triangulate the position of the pressed key. Such techniques differ mostly in whether they require a training phase [135], and rely on one [95] or more [147] mobile phones.

All the previously cited approaches assume physical proximity with the adversary — this requirement can be relaxed by using VoIP software to acquire keyboard sounds, as demonstrated in [8, 43]. The main difference between our work and [8] lies in a much more extensive experimental evaluation on more realistic scenarios, and in a less obtrusive countermeasure, that we validate through more complete testing.

Attacks Using Other Emanations. Another body of work focused on keyboard eavesdropping via non-acoustic side-channels.

Typing on a keyboard causes its electrical components to emit electromagnetic waves, and it is possible to collect such waves, to recover the original keystrokes [134]. Furthermore, typing causes vibrations of the surface under the keyboard. These vibrations can be collected by an accelerometer (e.g., of a smartphone) [102], or by a camera that captures small movements of the surface [81], and analyzed to determine the pressed keys.

Inter-keystroke timing also leaks information about the pressed keys. The idea is that users require slightly different time to type different sequences of characters, allowing keystroke recovery. This was successfully exploited by observing inter-keystroke timing over SSH connections [125], and from videos of users typing on masked password fields [15].

Analyzing movements of the user's hands and fingers on a keyboard represents another way of recovering input. This is possible by video-recording a typing user [16] or by using WiFi signal fluctuation on the user's laptop [7]. In general, changes in channel state information of wireless signal can also be leveraged to recover typed text in a known language without the need for training [57]. On touchscreen keyboards (e.g., smartphones), the typist's eyes follow the fingers' movement — therefore, tracking eye movements can leak information on the typed text [40].

Finally, heat radiation from hands to the device in use can be captured by thermal cameras, and used to infer keystrokes on PIN-entry devices [140], and standard keyboards [83].

2.2 System and Threat models

To identify precise attack scenarios, we begin by defining the system model that serves as the base for *S&T*. Section 2.2.1 describes our assumptions about the victim and the attacker, and then carefully defines the problem of remote keyboard acoustic eavesdropping. Section 2.2.2 then presents some realistic attack scenarios and discusses them in relation to the state-of-the-art.

2.2.1 System Model

The system model we consider in this work is depicted in Figure 2.1. We assume that the victim has a desktop or a laptop computer with a built-in or attached keyboard, i.e., **not** a smartphone or a tablet-like device. Hereafter, it is referred to as target-device. The victim also owns a device with a microphone, hereafter referred to as recording-device. A genuine copy of some VoIP software is assumed to be installed on recording-device; this software is not compromised in any way. Also, recording-device is connected to the Internet and engaged in a VoIP call with at least one party who plays the role of the attacker. We underline that target-device and recording-device might not correspond — as long as target-device is close enough for recording-device microphone to pick up its keystroke sounds.

The attacker is a malicious user who aims to learn some private information about the victim. The attacker owns and fully controls a computer that we refer to as attack-device, which has a genuine (unmodified) version of the same VoIP software as recording-device. The attacker uses attack-device to receive and record the victim's acoustic emanations using VoIP software. We assume that the attacker relies solely on information provided by VoIP software. In other words, *during the attack*, the attacker receives no additional acoustic information from the victim, besides what VoIP software transmits to attack-device.

2.2.2 Threat Model

S&T attack transpires as follows: during a VoIP call between the victim and the attacker, the former types something on target-device, for example a text

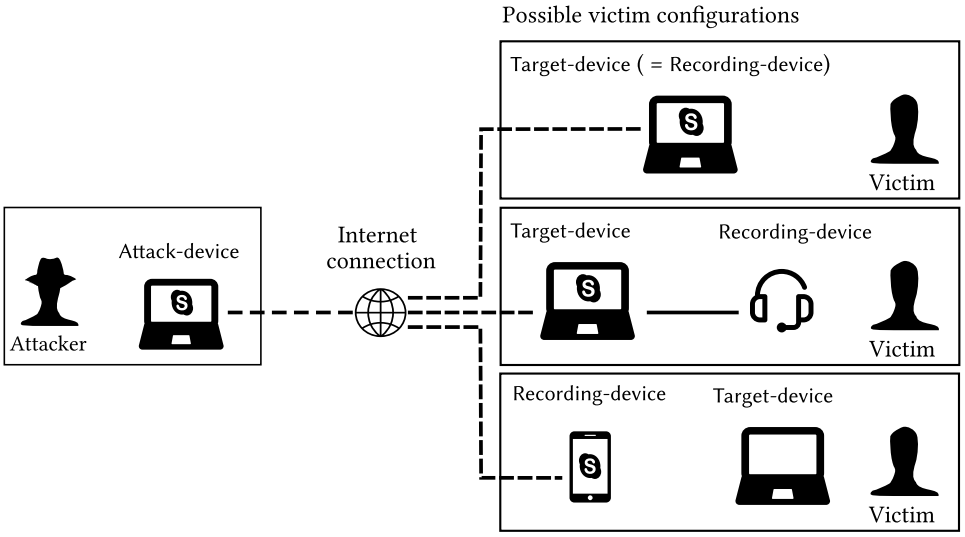


Figure 2.1: System model. There are different possible configuration of target-device and recording-device. Victim's target-device needs not to correspond to recording-device — as long as there is physical proximity.

of an email message or a password. We refer to this typed information as target-text. Typing target-text causes acoustic emanations from target-device's keyboard, which are picked up by the recording-device's microphone and faithfully transmitted to attack-device by VoIP. The goal of the attacker is to learn target-text by taking advantage of these emanations.

We make the following assumptions:

- As mentioned above, the attacker has no real-time audio-related information beyond that provided by VoIP software. Acoustic information can be degraded by VoIP software by downsampling and mixing. In particular, without loss of generality, we assume that audio is converted into a single (mono) signal, as is actually the case with some VoIP software, such as Skype and Google Hangouts.

- If the victim discloses some keyboard acoustic emanations **together** with the corresponding plaintext — the actual pressed keys (called *ground truth*) — the volume of this information is small, on the order of a chat message or a short e-mail. We expect it to be no more than a few hundred characters.

- target-text is very short (e.g., $\approx$ 10 characters) and random, corresponding to an ideal password. This keeps *S&T attack* as general as

possible, since dictionary words are a "special" case of random words, where optimization may be possible.

We now consider some realistic *S&T attack* scenarios. We describe them starting with the more generous setting where the attacker knows the victim's typing style and keyboard model, proceeding to the more challenging one where the attacker has neither type of information.

1) Complete Profiling: In this scenario, the attacker knows some of the victim's keyboard acoustic emanations on target-device, along with the ground truth for these emanations. This might happen if the victim unwittingly provides some text samples to the attacker during the VoIP call, e.g., sends chat messages, edits a shared document, or sends an email message[1]. We refer to such disclosed emanations as *"labeled data"*. To be realistic, the amount of labeled data should be limited to a few samples for each character.

We refer to this as *Complete Profiling* scenario, since the attacker has maximum information about the victim. It corresponds to attack scenarios used in prior supervised learning approaches [10, 67, 69, 103], with the difference that we collect acoustic emanations using VoIP software, while others collect emanations directly from microphones that are physically near target-device.

2) User Profiling: In this scenario, we assume that the attacker does not have any labeled data from the victim on target-device. However, the attacker can collect training data of the victim while the victim is using the same type of device (including the keyboard) as target-device[2]. This can be achieved via social engineering techniques or with the help of an accomplice. We refer to this as *User Profiling* scenario, since, unable to profile target-device, the attacker profiles the victim's typing style on the same device type.

3) Model Profiling: This is the most challenging, though the most realistic, scenario. The attacker has absolutely no training data for the victim. The attacker and the victim are engaged in a VoIP call and information that the attacker obtains is limited to the (unknown) victim keyboard's acoustic emanations.

[1] Ground truth could also be collected offline, if the attacker happened to be near the victim, at some point before or after the actual attack. Note that this still does not require physical proximity between the attacker and the victim in *real time*.

[2] In case the target-device is a desktop, knowing the model of the desktop does not necessarily mean knowing the type of the keyboard. However, in mixed video/audio call the keyboard model might be visually determined, when the keyboard is placed in the visual range of the camera.

The attacker's initial goal is to determine what laptop the victim is using. To do so, we assume that the attacker maintains a database of sounds from previous attacks. If the attacker already profiled the model of the current victim's target-device, it can use this information to mount the attack. We refer to this as *Model Profiling* scenario, since although the attacker can not profile the current victim, it can still profile a device of the same model as target-device. We observe that this scenario could also apply in the case where the attacker is not directly in the VoIP call, but compromised the VoIP software of the victim to act as a remote microphone.

2.3 Skype & Type attack

S&T consists of two main activities: *training* attack models tailored on specific victims or keyboard models, and *classification* of unknown keystroke sounds — the actual attack. We divide *S&T* in 4 different phases, depicted in Figure 2.2 for the first three, and Figure 2.3 for the last phase. We first overview the four phases. Then, we detail all the steps of the four phases.

A. **Data Collection -** The attacker receives keystroke sounds involuntarily leaked by the victim during a VoIP call. The attacker might know the keys corresponding to the sounds (and later use such sounds to train a machine learning model).

B. **Feature Extraction -** The attacker now pre-processes the keystroke sounds. He runs a pipeline that divides the recorded waveform, corresponding to all received keystrokes, into shorter waveforms, each containing the sound of a single keystroke. He then extracts meaningful features from the waveforms, obtaining one feature vector per keystroke.

After Phase B, the attacker can continue either with Phase C or Phase D. If the attacker knows the keys corresponding to each sound he received in Phase A, he continues with Phase C to train a tailored machine learning model:

C. **Model Training -** Using the labeled data, the attacker removes the least important features from the feature vectors (known as *feature selection*). Then, he trains a supervised machine learning model to classify keystroke sounds into different keys — and obtains a trained model that can be later used to perform *S&T attack*.

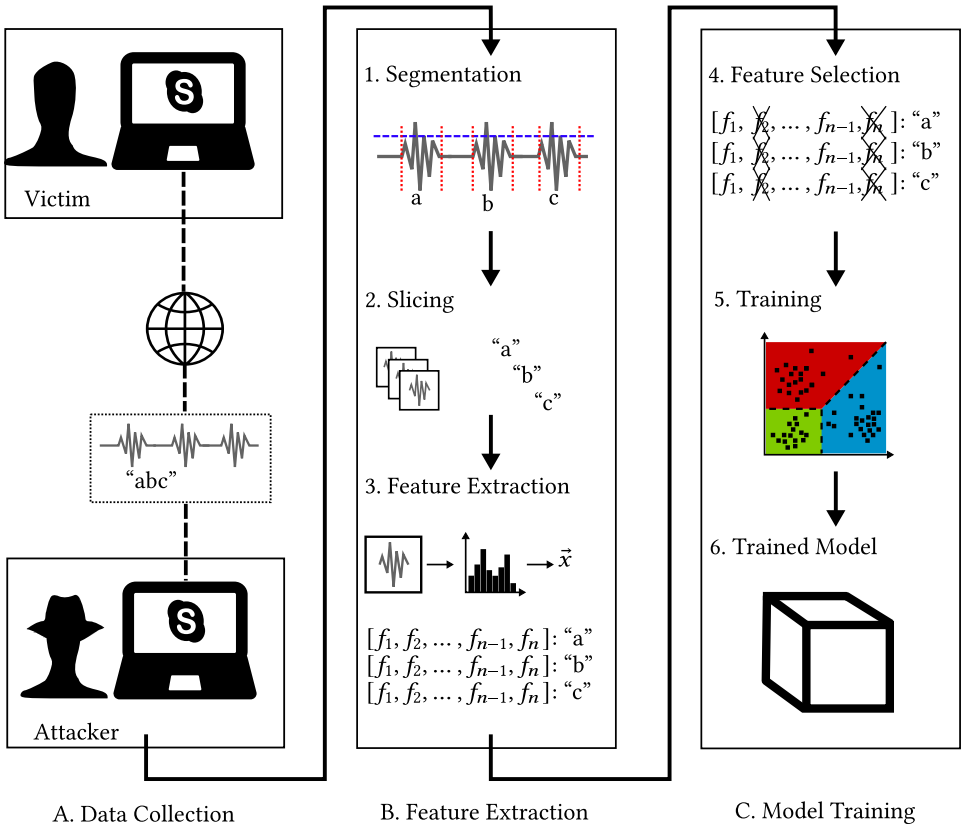


Figure 2.2: *S&T*, pre-processing steps: collection of victim's typing sounds (Step A), and machine learning pipeline to process them (Step B) and to train an attack model (Step C) if the keys corresponding to sounds are known.

If the attacker during Phase A received non-labeled keystroke sounds, his goal is to infer the typed keys. Therefore, he continues to Phase D, where he uses a trained machine learning model to obtain the predictions:

D. **Attack Phase -** The exact steps depend on the specific threat model scenario. If the attacker already knows the attack model to use (i.e., the Complete Profiling and User Profiling scenarios), he loads such model and uses it to predict the most likely keys for the keystroke sounds. Otherwise, in the challenging Model Profiling scenario, the attacker first performs Device-classification (to discover the likely keyboard in use by the victim), then loads the indicated model and proceeds to obtain predictions.

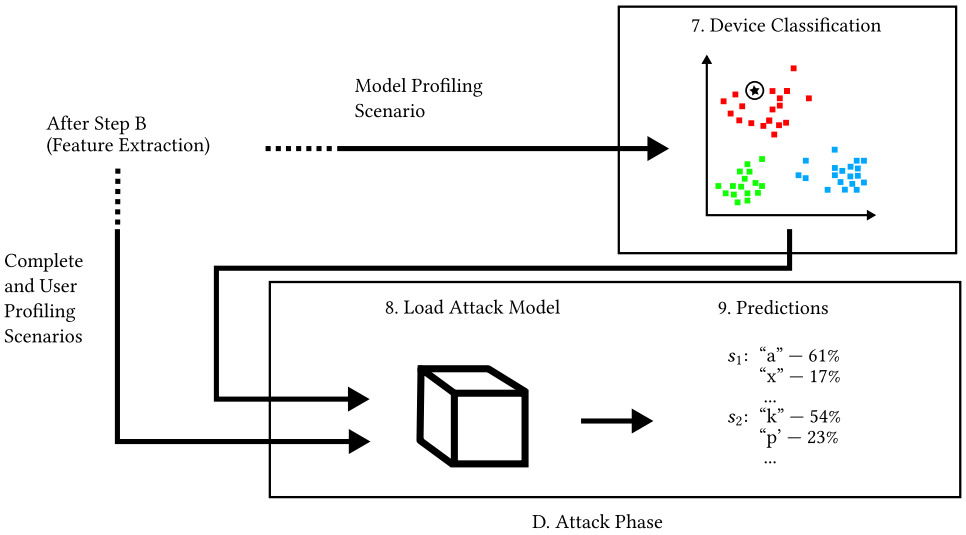


Figure 2.3: *S&T*, attack steps: the attacker determines which attack model to load (depending on the scenario), and then predicts the keys of the given sounds (Step D).

We now describe in more details all four phases and their different steps. Throughout the rest of the section, we refer to the phases and steps presented in Figure 2.2 and Figure 2.3.

2.3.1 Phase A: Data Collection

All envisaged scenarios involve the attacker engaged in a VoIP call with the victim. During the call, the victim types something on target-device's keyboard. As expected, the VoIP software records and transmits the keystroke sounds to the attacker. The attacker then records such sounds — for example by channeling VoIP output to some local recording software. If the attacker knows the keys corresponding to the keystroke sounds (for example, as discussed in Section 2.2.2, thanks to a common chat, shared document, or email), he keeps these annotations to generate *labeled data*. The attack then proceeds to the next phase.

2.3.2 Phase B: Feature Extraction

The main goal in this phase is to extract meaningful features from acoustic information. The first step is segmentation, needed to isolate distinct keystroke sounds within the recording. The attacker then slices the waveform into sound samples, containing one keystroke sound each. Subsequently, using these sound samples, we build derived values (called features)

that represent properties of the keystroke sounds. This step is referred to as feature extraction.

The details of the steps are the following:

1. **Segmentation.** We perform data segmentation according to the following observation: the waveform of a keystroke sound presents two distinct peaks, shown in Figure 2.4. These two peaks correspond to the events of: (1) the finger pressing the key — *press* peak, and (2) the finger releasing the key — *release* peak. Similar to [10], we only use the press peak to segment the data and ignore the release peak. This is because the former is generally louder than the latter and is thus easier to isolate, even in very noisy scenarios.

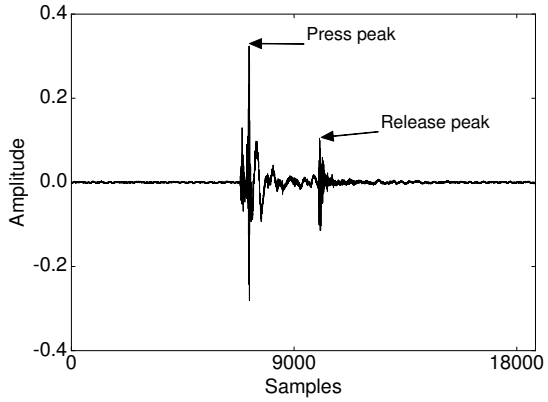


Figure 2.4: Waveform of the "A" key, recorded on an Apple Macbook Pro 13" laptop.

To perform automatic isolation of keystrokes, we set up a detection mechanism as follows: we first normalize the amplitude of the signal to have root mean square of 1. We then sum up the FFT coefficients over small windows of 10ms, to obtain the energy of each window. We detect a press event when the energy of a window is above a certain threshold. The threshold is a parameter that can be tuned. We observed that background noise is usually significantly quieter than keystroke sounds — therefore, all points in time when the intensity of the waveform is above the 90th percentile of all intensities are most likely "events" where, e.g., a keystroke happened. In case of more background noise, or external loud noises, the threshold might need manual tuning by the attacker, or manual data cleaning to remove the spurious high-intensity sounds (e.g., a door slamming). The same

holds if there is overlap of voice and keystroke sounds — the attacker might need to manually increase the threshold in order to only capture the high-intensity keystroke sounds.

2. **Slicing.** For each event detected by the data segmentation step, we then extract the subsequent 100ms [21, 149] of waveform as the keystroke event waveform. If keystroke sounds are very close to each other (i.e., less than 100ms), it is possible to extract a shorter waveform with minimal loss in accuracy (see Section 2.4.1).

3. **Feature Extraction.** As features, we extract the mel-frequency cepstral coefficients (MFCC) [98]. These features capture statistical properties of the sound spectrum, which is the only information that we can use. Indeed, due to the mono acoustic information, it is impossible to set up an attack that requires stereo audio and uses TDoA, such as [95, 135, 147]. Among possible statistical properties of the sound spectrum — including: MFCC, FFT coefficients, and cepstral coefficients — we chose MFCC which yielded the best results. To select the most suitable property as a feature, we ran the following experiment:

   > Using a Logistic Regression classifier we classified a dataset with 10 samples for each of the 26 keys corresponding to the letters of the English alphabet, in a 10-fold cross-validation scheme. We then evaluated the accuracy of the classifier with various spectral features: FFT coefficients, cepstral coefficients, and MFCC.

   We repeated this experiment with data from five users on a Macbook Pro laptop. Accuracy results were as follows: 90.61% (± 3.55%) for MFCC, 86.30% (± 6.34%) for FFT coefficients, and 51% (± 18.15%) for cepstral coefficients. This shows that MFCC offers the best features. For MFCC experiments we used parameters similar to those in [149]: a sliding window of 10ms with a step size of 2.5ms, 32 filters in the mel scale filterbank, and used the first 32 MFCC.

2.3.3 Phase C: Model Training

This phase happens when the attacker received keystroke sounds as labeled data, i.e., together with the corresponding keys. With labeled data, the attacker can train a supervised learning classification model.

We argue that the attacker needs to have labeled data, and use a supervised classification technique: as discussed in Section 2.2.2, approaches that

require lots of data to cluster, such as [21], are incompatible with our assumptions, because we might have only a small amount of both training and testing data. Moreover, potential randomness of target-text makes it impossible to realize constraint-based approaches, which would require target-text to be a meaningful word, as in [149].

The detailed steps of this phase are the following:

4. **Feature Selection. The attacker determines the most relevant features, among all features produced by the Feature Extraction step. To do so, *S&T* uses a Recursive Feature Elimination algorithm [66], that tries to find the best performing subset of features among the full set. This step helps eliminating features that are irrelevant and only capture background noise or non-discriminating parts of the sound spectrum.**

5. **Training.** The attacker now needs to train a machine learning model to discriminate between different keys based on their keystroke sounds. We consider key classification to be a multiclass classification problem, where different classes correspond to different keyboard keys. Therefore, the input to the classifier for training is a set of feature vectors, representing properties of keystroke sounds — each associated to the specific key that generated the sound. Ideally, the attacker should have more than one sample for each key. The classifier then learns to discriminate between different keys of target-device by sound properties.

   To perform key classification, we use a Logistic Regression (LR) classifier, since it outperformed other tested classifiers, including: Linear Discriminant Analysis (LDA), Support Vector Machines (SVM), Random Forest (RF), and k-nearest neighbors. We tested this in a preliminary experiment that uses each candidate to classify a dataset of 10 samples, for each of the 26 keys corresponding to the letters of the English alphabet, in a 10-fold cross-validation scenario. We use MFCC as features, and, for each classifier, we optimized its hyper-parameters with an extensive grid search: we repeated training and testing with all possible combinations of hyper-parameters, and selected the best performing combination.

   To evaluate the classifiers' quality in this experiment, we used *accuracy* and *top-n accuracy* measures. Given true values of k, accuracy is defined in the multiclass classification case as the fraction of correctly classified samples over all samples. Top-n accuracy is defined similarly. The sample is correctly classified if it is present among the

top n guesses of the classifier. Results of this preliminary experiment are shown in Figure 2.5 which demonstrates that the best performing classifiers are LR and SVM. This is especially the case if the classifier is allowed to make a small number of predictions (between 1 and 5), which is more realistic in an eavesdropping setting. In particular, both LR and SVM exhibit around 90% top-1 accuracy, and over 98.9% top-5 accuracy. However, LR slightly outperforms SVM until top-4 — for this reason, we selected LR for *S&T*.

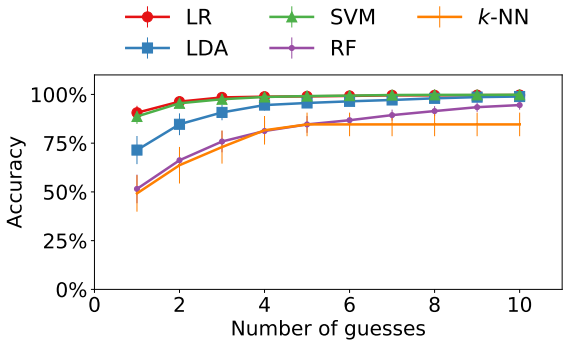


Figure 2.5: Average top-n accuracy of single key classification, as a function of the number of guesses, for each of the tested classifiers.

6. **Trained Model.** Once the attacker trained his LR model with the labeled data, he stores such model and any important auxiliary information (e.g., the features to eliminate in future samples). This model is tailored to the specific scenario the attacker trained his model for.

2.3.4 Phase D: Attack Phase

This phase happens when the attacker receives some keystroke sounds without knowing their corresponding keys. For example, the victim might have typed his password on some website during the VoIP call. In this case, the attacker wants to predict the keys corresponding to the received sounds. The exact steps depend on the threat scenario, as shown in Figure 2.3: in Complete and User Profiling scenarios, the attacker knows which attack model to load, and can directly proceed to Step 8. Instead, in Model Profiling scenarios, the attacker does not know the target-device used by the victim — he therefore needs to first understand this information by performing Step 7, Target-device classification.

7. **Device-classification.** In *Model Profiling* scenario, since the attacker has no knowledge of the victim's typing style or target-device, it begins

by trying to identify target-device by classifying its keyboard sounds. We consider the task of target-device classification as a multiclass classification problem, where different classes correspond to different target-device models known to the attacker. In particular, the attacker maintains a database of keystroke sounds corresponding to many different keyboard models, and wishes to predict which keyboard model generated some other unknown keystroke sounds.

To perform this classification task, we use a k-nearest neighbors (k-NN) classifier with $k = 10$ neighbors, that outperformed other classifiers such as Random Forest and Logistic Regression in our preliminary experiments. The k-NN classifier, when given an unknown sample (i.e., the feature vector of a keystroke sound) finds the k closest samples in terms of sound features, and outputs their corresponding target-device. The prediction for the target-device that generated such sound is the majority of these outputs.

The output of this phase is the keyboard used by the victim (i.e., his target-device), or that the target-device is unknown. In this latter case, the attacker can not perform *S&T*.

8. **Load Attack Model. Once the attacker knows the** target-device **used by the victim, and depending on the specific scenario, he loads the specific model previously trained on the appropriate data. This model, given a feature vector corresponding to a keystroke sound, retains only the appropriate features that were determined by Step 4 in its previous training, and then generates predictions for the keystroke.**

9. **Predictions.** Given a feature vector of a waveform of a keystroke, attack models can return a list of predictions of probable keys corresponding to the pressed key that generated the sound. Such predictions are ordered by probability, such that, e.g., the first 5 predictions returned by the model correspond to the 5 most likely keys according to the classifier. These ordered predictions help build further use-cases for *S&T*, for example prediction of typed words (Section 2.6.1) or of passwords (Section 2.6.2).

2.4 Experimental Setup

To assess the feasibility of *S&T attack* on all considered scenarios and combinations of target-devices and recording-devices described in Section 2.2, we

collected a large dataset of keystroke sounds on a variety of settings. In this section, we first describe the two data collection campaigns that we carried out, and motivate them by illustrating the different hypotheses we planned to test on each campaign in Section 2.4.1. We then discuss the demographics of our study participants, the implications and generality of our study, and detail the experiment protocol in Section 2.4.2.

2.4.1 Data Collection

We performed two separate data collection campaigns:

- **First Campaign.** We collected data to test our different attack scenarios (Complete, User, and Model Profiling, introduced in Section 2.3) — that requires to have multiple target-devices of the same model — and the difference between typing styles, i.e., *Hunt and Peck* and *Touch* typing (to simulate typists of different skill levels).
- **Second Campaign.** Besides expanding our collection of target-devices, we collected data to test the impact of different recording-devices on *S&T*. Moreover, we tested the impact of typing speed by requiring participants to type a popular sentence in English — a natural activity, that allowed participants to type at their own pace.

We summarize our broad choice of devices and setups in Table 2.1. In total, we collected data from 12 unique participants and 14 keyboards of 11 different models. Considering the use of different recording-devices, we collected 156 unique datasets.

First Campaign. We collected data from 5 distinct users. For each user, the task was to press the keys corresponding to the English alphabet, sequentially from "A" to "Z", and to repeat the sequence ten times, first by only using the right index finger (this is known as *Hunt and Peck* typing, referred to as *HP* from here on), and then by using all fingers of both hands (*Touch* typing) [69]. We believe that typing letters in the order of the English alphabet rather than, for example, typing English words, did not introduce bias. Typing the English alphabet in order is similar to typing random text, that *S&T attack* targets. Moreover, a very fast touch typist usually takes around 80ms to type consecutive letters [34]. We show that this is the case with our data in Section 2.5.4; moreover, we argue that *S&T attack* works without any accuracy loss with samples shorter than this interval. In order to test correctness of this assumption, we ran a preliminary experiment as follows:

Table 2.1: Configuration details of our data collection campaigns.

	First Setup	**Second Setup**
Users	5	8
Target Devices	6 *(2 laptops of 3 different models)*	4 laptops 3 USB membrane keyboard 1 USB mechanical keyboard
Typing Style	Hunt&Peck Touch typing	Touch typing
Recording Devices	Laptop microphone	Laptop microphone Smartphone Headset microphone
Typed Text	Ten times the entire English alphabet (a-z), in order	Ten times the sentence "The quick brown fox jumped over the lazy dog"
Total Datasets	60 *(5 users · 6 considered keyboards · 2 typing styles)*	96 *(4 users · 4 considered keyboards · 3 recording devices) — repeated twice*

> We recorded keystroke audio of a single user on a Macbook Pro laptop typing the English alphabet sequentially from "A" to "Z" via Touch typing. We then performed Step B (Feature Extraction) and Step C (Model Training) as described in Section 2.3. However, instead of extracting 100ms of the waveform, we extracted 3ms (as in [10]), and then from 10ms to 100ms at intervals of 10ms for each step. We then tested *S&T attack* in a 10-fold cross-validation scheme. Figure 2.6 shows top-5 accuracy of this preliminary experiment, for different lengths of the sound sample that we extracted.

We observe that, even with very short 20ms samples, *S&T attack* suffers minimal accuracy loss. Therefore, we believe that adjacent letters do not influence each other, since sound overlapping is very unlikely to occur.

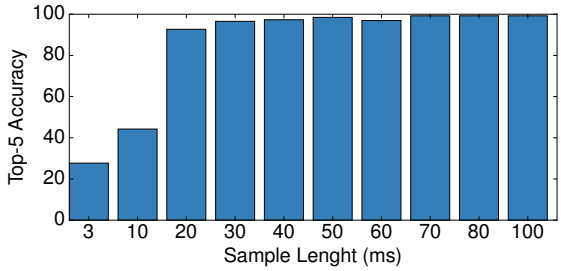


Figure 2.6: Top-5 accuracy of single key classification for different sample lengths.

Note that collecting only the sounds corresponding to letter keys, instead of those for the entire keyboard, does not affect our experiment. The "acoustic fingerprint" of every key is related to its position on the keyboard plate [10]. Therefore, all keys behave, and are detectable, in the same way [10]. Due to this property, we believe that considering only letters is sufficient to prove our point. Moreover, because of this property, it would be trivial to extend our approach to various keyboard layouts, by associating the keystroke sound with the position of the key, rather than the symbol of the key, and then mapping the positions to different keyboard layouts.

Every user performed the data collection task on six laptops: (1) two Apple Macbook Pro 13" 2014, (2) two Lenovo Thinkpad E540, and (3) two Toshiba Tecra M2. We selected these as being representative of many common modern laptop models: Macbook Pro is a very popular aluminium-case high-end laptop, Lenovo Thinkpad E540 is a 15" mid-priced laptop, and Toshiba Tecra M2 is an older laptop model, manufactured in 2004. We

needed pairs of laptops of the same model to test the User Profiling and Model Profiling scenarios. All acoustic emanations of the laptop keyboards were recorded by the microphone of the laptop in use, with Audacity software v2.0.0. We recorded all data with a sampling frequency of 44.1kHz, and then saved it in WAV format, 32-bit PCM signed.

Second Campaign. We collected data from 8 distinct users — 7 of which did not participate in the first data collection campaign. The task for each user was to type a famous pangram (i.e., a sentence with all the letters of English alphabet): *"The quick brown fox jumps over the lazy dog"*, as it is short and simple. All users typed the sentence ten times for each considered keyboard, using all fingers of both hands (Touch typing).

We recorded these users on 8 different keyboards: 4 laptop keyboards, and 4 external USB keyboards. We selected the devices to be representative of many common user setups. Regarding laptops, we tested an HP 250 G5 (a low-priced domestic laptop), a Dell Inspiron 13 5000 (mid-range business laptop), and a Dell XPS 13 9333 and an Asus Zenbook UX303UB (high-end slim laptops with aluminium bodies). Regarding external keyboards, we tested 3 popular office membrane keyboards — HP 9009, Fujitsu KB900, HP KU-0316 — and a mechanical keyboard, a AUKEY KM-G3 equipped with "OTEMU Blue" mechanical switches, with a distinctive "clicking" sound typical of mechanical switches.

We concurrently recorded keystroke sounds with three different recording devices:

- The **laptop** in use — external USB keyboards were recorded using either the Dell XPS 9333 laptop microphone (HP 9009, AUKEY KM-G3) or the Dell Inspiron 5000 laptop microphone (Fujitsu KB900, HP KU-0316);
- A **smartphone** placed in proximity of the laptop/keyboard in use — either a Samsung Galaxy S6 (HP 9009, AUKEY KM-G3) or a Huawei P10 lite (Fujitsu KB900, HP KU-0316);
- A **headset** with a microphone, worn by the study participant — we used a generic Samsung inexpensive headset, connected to a Huawei P9 Lite.

All recordings were done with a sampling frequency of 44.1kHz, and then saved in WAV format, 32-bit PCM signed.

Resulting Datasets. All recorded data was then processed through a VoIP

software — we chose Skype as representative, as (i) it is one of the most popular VoIP tools; (ii) its codecs are used in Opus, an IETF standard [133][3]; (iii) it reflects our general assumption about mono audio.

We filtered the sound samples by routing the recorded sounds through the Skype software, and recording the received emanations on a different computer (i.e., on the attacker's side). As we saved raw data from the microphones at their native quality, there is no difference between immediately recording Skype-filtered data, or filtering at a later moment, as we did: Skype receives the same data as if it were receiving it from the microphone in the first place. To perform this filtering, we used two machines running Linux, with Skype v4.3.0.3 for the first setup, and Skype v8.11.0.4 and 8.34.0.78 for the second setup[4], connected via a high-speed network. During the calls, there was no sensible data loss. We analyze bandwidth requirements needed for data loss to occur, and the impact of bandwidth reduction, in Section 2.5.4.

At the end of data collection and processing phases we obtained datasets for the first and second setups. For the first set of experiments, each dataset consists of 260 samples, 10 for each of the 26 letters of the English alphabet. For the second set of experiments, each dataset consists of 10 repetition of the aforementioned pangram.

In total, we recorded 12 unique participants and 14 keyboards of 11 different models. As mentioned, considering the use of different recording-devices, we collected 156 unique datasets. The number of users and of keyboards we considered greatly exceeds related works on keyboard acoustic eavesdropping [10, 67, 69, 103]: such works evaluated 1–2 users on 1–3 keyboards, testing on a quantity of data comparable to only 10 of our datasets. We further discuss generality in the next section.

2.4.2 Study Participants

We recruited a total of 12 unique participants for our study. Participants were randomly selected among (primarily) college and graduate students who agreed to be part of the experiment. Five participated in the first data collection campaign, and eight in the second campaign — one participant joined both the first and the second campaign. Out of 12, 4 were female and 8 were male. The youngest was 20 years old at the time of the experiment, while the oldest was 31; the mean age was 25.8 ($\pm$3.07). Nine participants

[3]Opus is employed in many other VoIP applications, such as Google Hangouts and Teamspeak.

[4]Between the first and second data collection campaigns, Skype v4 was discontinued by Microsoft.

had a Computer Science/Engineering background, two had humanities backgrounds, one had a high school diploma. All participants self-reported to be confident with typing on a computer keyboard, and experienced in the use of computers. All of them already knew how to type with both hands (Touch typing), with different degrees of confidence.

The participants' typing speed was diverse. We recorded the inter-key time of the 8 participants to the second data collection campaign (where the task was to type an English sentence), and show them in Figure 2.7.

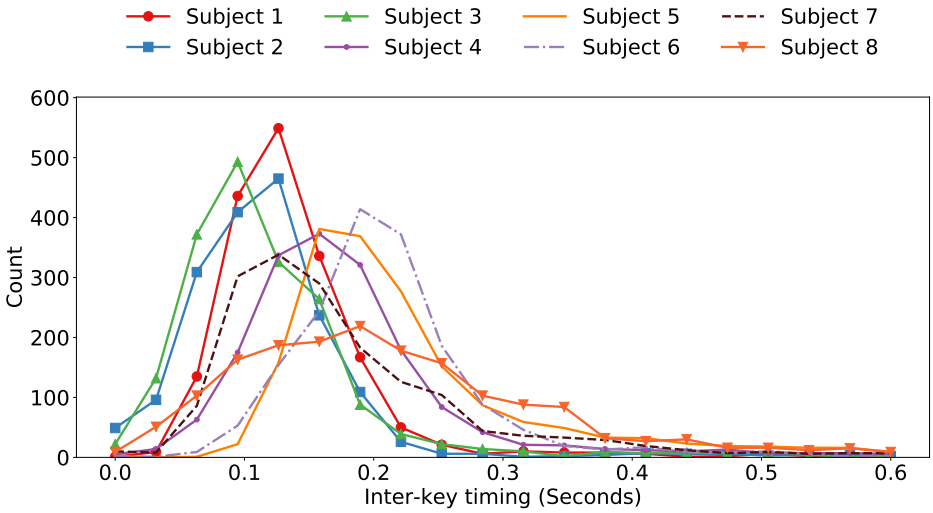


Figure 2.7: Distribution of inter-key timing of experiment participants.

We can see that three participants are consistently fast typists (subjects 1, 2, and 3), three are somewhat average-speed typists (subjects 4, 5, 6), one is a slow typist (Subject 7), and one is noticeably non-uniform in typing speed (Subject 8). There is no apparent correlation between demographics and average speed: one of the fast typists has a background in humanities, while the slowest is in Computer Science.

We believe that typing speed is a good proxy for typing experience, therefore our diverse population can be representative of skilled and less skilled typists. Moreover, less skilled typists are simulated by our HP typing style. Furthermore, intuitively skilled typists represent the worst case for *S&T*, because of the increased probability of overlapping keystroke sounds. However, our population exhibits an age bias, skewed towards young people, and is thus unclear how it generalizes to greatly different age groups.

Experimental Protocol. Approval from an Institutional Review Board

was not mandatory to collect and handle the recorded data in the institution where the tests were carried out. However, we followed well-established good practices in data collection and handling, that we now report in detail.

Participants were asked to sit on a chair in front of a computer with a keyboard. Where applicable, they were asked to wear a headset microphone. They got time to relax and get acquainted with the task, whether it was about typing the English alphabet or the pangram sentence. Participants were only required to type in a specific way when testing the HP typing style hypothesis, otherwise they were encouraged to type in their usual way, with as many fingers as desired. Participants (who signed an informed consent) were informed that they were free to interrupt the experiment at any time and request deletion of their data if they desired. There was no incentive for participants to complete the task as fast as possible, or to meet any performance goal (e.g., participants were not penalized for making mistakes, and did not have to re-type a sentence in case of mistakes) — therefore, we believe there was no bias in our data collection procedure.

Data was handled confidentially, and is stored encrypted in our laboratory. The data is anonymized together with the identities of the participants, and was only used for the specified purpose of running this study. The mapping between anonymous identifiers and demographic information is securely stored in a different location.

2.5 S&T Attack Evaluation

We evaluated *S&T attack* in all scenarios described in Section 2.2.2:

- We report *Complete Profiling* scenario in full detail (Section 2.5.1), by analyzing performance of *S&T attack* separately on all considered keyboards, typing styles, and VoIP filtered and unfiltered data. These results serve as a reference for the accuracy of *S&T attack* on other scenarios.

- We then analyze *User Profiling* (Section 2.5.2) and *Model Profiling* (Section 2.5.3) scenarios only on realistic settings, namely Touch typing, VoIP filtered data. We compare these results with results of *Complete Profiling*, to understand the impact on accuracy of less powerful adversaries.

- We thoroughly evaluate the impact of different conditions on the baseline *Complete Profiling* scenario (Section 2.5.4). We explore the impact on *S&T attack* accuracy of different typists, different inexpensive

recording devices that are commonly used in VoIP calls, small training sets, and VoIP-specific issues such as degradation of call quality due to low bandwidth, and voice on top of keystroke sounds.

- We conclude by discussing our results and their significance (Section 2.5.5).

Evaluation Technique. Unless otherwise specified, we evaluate *S&T* accuracy on a given dataset as follows. We consider the given dataset in a stratified 10-fold cross-validation scheme[5]. For every fold, we proceed with Step C and Step D of *S&T* (see Section 2.3): we perform feature selection on training data using Recursive Feature Elimination [66], and then train a machine learning model. We then calculated the accuracy of the classifier over each fold, and then computed the mean and standard deviation of accuracy values.

We underline that data collected in our first campaign is already balanced (i.e., there are 10 samples for each letter), while data collected in our second campaign is not (i.e., we have 10 repetitions of the pangram, where letters occur with different frequencies). Therefore, every time we use data from the second campaign, we perform *undersampling*: we randomly select a subset of the data such that every letter appears 10 times — and thus the resulting dataset is balanced. To mitigate randomness in the undersampling process, we repeat it for 5 times, and then average the results.

Baseline. We evaluated the accuracy of *S&T attack* in recognizing single characters, according to the top-n accuracy, defined in [30]. As a baseline, we considered a random guess with accuracy n/l, where n is the number of guesses, and l is the size of the alphabet. Therefore, in our experimental setup, accuracy of the random guess is $n/26, n \in [1 \dots 26]$, since we considered 26 letters of the English alphabet. Because of the need to eavesdrop on random text, we can not use "smarter" random guesses that, for example, take into account letter frequencies in a given language. Moreover, these results serve as a baseline to more complex attacks that consider additional information (e.g., the language of target-text).

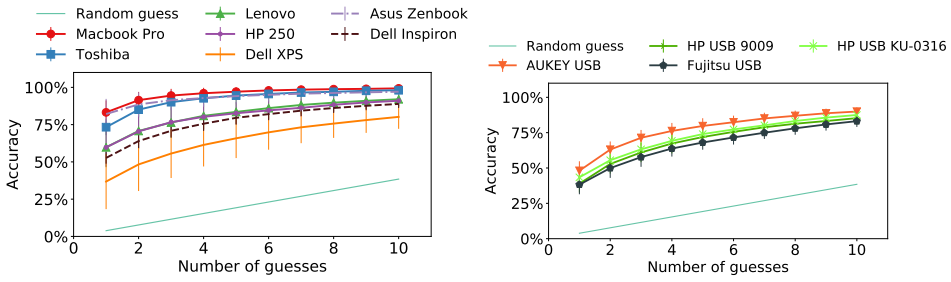
[5] In a stratified k-fold cross-validation scheme, the dataset is split in k sub-samples of equal size, each having the same percentage of samples for every class as the complete dataset. One sub-sample is used as testing data, and the other $(k-1)$ — as training data. The process is repeated k times, using each of the sub-samples as testing data.

2.5.1 Complete Profiling Scenario

To evaluate the scenario where the victim disclosed some labeled data to the attacker, we considered all datasets registered with laptop internal microphones (i.e., all datasets of the first setup, and a subset of the datasets of the second), one at a time, following our evaluation technique.

Figure 2.8 depicts results of the experiment on the realistic Touch typing, Skype-filtered data combination. We observe that *S&T attack* achieves its lowest performance on the Dell XPS laptop and on the Fujitsu USB keyboard (recorded through the Dell Inspiron 5000 internal microphone), with low top-1 accuracies of 36.81% and 38.92%, but satisfactory top-5 accuracies of 65.81% and 68.02%, respectively. Performance on the other USB keyboards are similar to the Fujitsu, with the mechanical AUKEY keyboard being the best performing at a top-1 accuracy of 48.08% and top-5 of 79.72%. Interestingly, the second worst performing laptop is the Dell Inspiron 5000 with top-1 and top-5 accuracies of 52.67% and 79.56% — given that the four USB keyboards were recorded using the two Dell laptops' integrated microphones, these results hint to these microphones leading to inaccurate recordings.

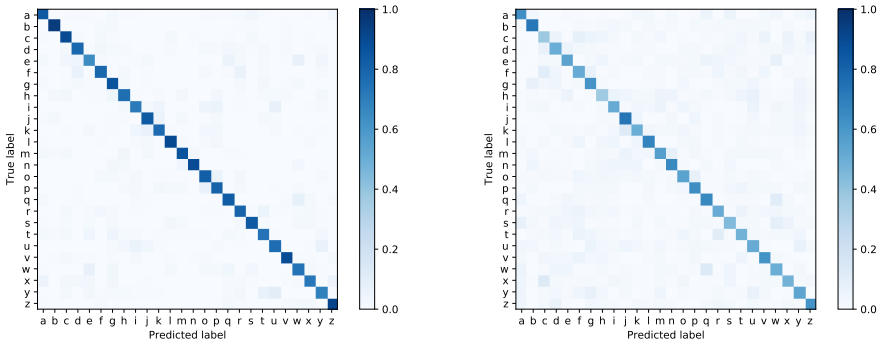
Lenovo laptops have intermediate performance, with top-1 accuracy of 59.8%, and a top-5 accuracy 83.5%. The similar HP laptop exhibits similar performance. On the Macbook Pro, Asus Zenbook, and Toshiba, we obtained very high top-1 accuracy between 73.3% and 83.23%, and top-5 accuracy between 94.5% and 97.1%. We believe that these differences are due to variable quality of manufacturing, e.g., the bodies and keyboards of our particular Lenovo and HP laptops are made of cheap plastic materials. Another possible reason are different microphones: we show in Section 2.5.4 that different recording devices can have an impact on accuracy up to 10%.



(a) Laptop Keyboards. (b) USB Keyboards.

Figure 2.8: *S&T attack* performance — *Complete Profiling* scenario, Touch typing, Skype-filtered data, average accuracy.

Confusion Matrices. We extend our investigation to classification *mistakes*: we analyzed the confusion matrices of our classifiers, to see if misclassifications give us some insights. We report the confusion matrices for two sample keyboards, the Asus Zenbook and Dell Inspiron 5000 laptops, in Figure 2.9 (confusion matrices for other keyboards look similar and exhibit similar patterns).



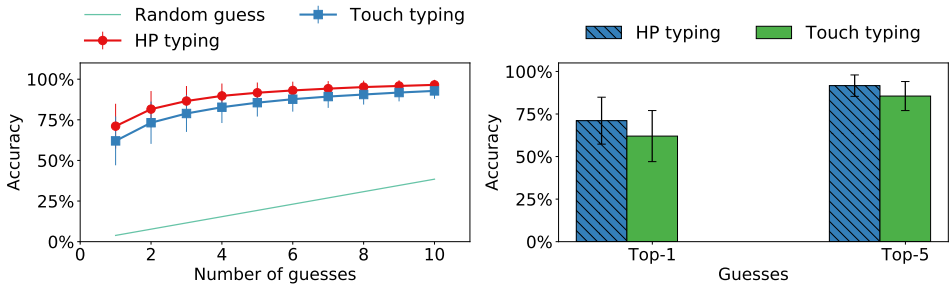
(a) Asus Zenbook. (b) Dell Inspiron.

Figure 2.9: *S&T attack* performance — confusion matrices.

We observe that many misclassifications involve *neighboring* keys on the keyboard: some examples are y–u and t–y on the Asus Zenbook, x–c, and s–w on the Dell Inspiron, and c–f, q–w, t–g, and w–e on both keyboards. These results further confirm the observations of Asonov and Agrawal [10] regarding the influence of position on keyboards on keys' sound — and suggest that it is possible to further increase the accuracy of *S&T* by biasing the classifiers' predictions to exploit locality.

Impact of Typing Style and VoIP Software. Interestingly, we found that there is little difference between this data combination (*Touch* typing, Skype-filtered data, that we consider the most unfavorable) and the others. In particular, we compared average accuracy of *S&T attack* on HP and Touch typing data, and found that the average difference in accuracy is 0.80%. Such results are reported in more detail in Figure 2.10. Moreover, we compared the results of unfiltered data with Skype filtered data, and found that the average difference in accuracy is a surprising 0.33%. This clearly shows that Skype does not reduce accuracy of *S&T attack*.

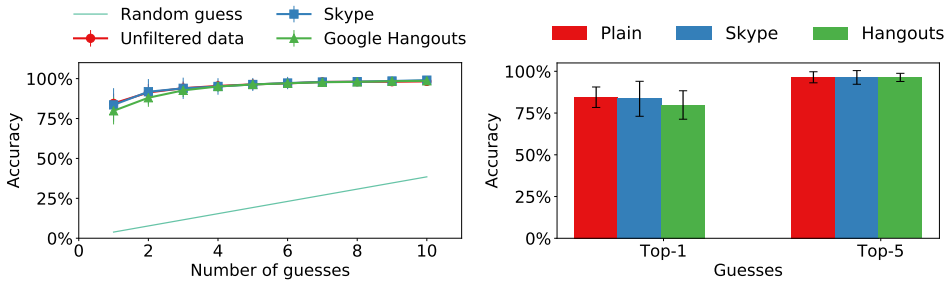
Generality of VoIP Software. We also ran a smaller set of these ex-



(a) Average accuracy. (b) Top-1 and top-5 accuracy.

Figure 2.10: *S&T attack* performance — accuracy of HP and Touch typing data.

periments over Google Hangouts and observed the same tendency, that we show in Figure 2.11. This means that the keyboard acoustic eavesdropping attack is applicable to other VoIP software, not only Skype. It also makes this attack more credible as a real threat.



(a) Average accuracy. (b) Top-1 and top-5 accuracy.

Figure 2.11: *S&T attack* performance — accuracy of unfiltered, Skype-filtered and Google Hangouts-filtered data.

From now on, we only focus on the most realistic combination — Touch typing and Skype filtered data. We consider this combination to be the most realistic, because *S&T attack* is conducted over Skype, and it is more common for users to type with the Touch typing style, rather than the HP typing style. We limit ourselves to this combination to further understand real-world performance of *S&T*.

2.5.2 User Profiling Scenario

In this case, the attacker profiles the victim on a laptop of the same model of target-device. We selected the dataset of a particular user on one of the

six laptops, and used it as our training set. Recall that it includes 260 samples, 10 for every letter. This training set modeled data that the attacker acquired, e.g., via social engineering techniques. We used the dataset of the same user on the other laptop of the same type, to model target-device. We conducted this experiment for all six laptops.

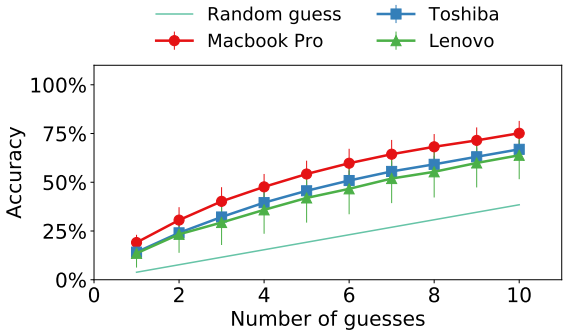


Figure 2.12: *S&T attack* performance — *User Profiling* scenario, average accuracy.

Results reflected in Figure 2.12 show that top-1 accuracy decreases to as low as 14% on Toshiba and Lenovo laptops, and to 19% on Macbook Pro. However, top-5 accuracy grows to 41.9%, 54%, and 45.6% on Lenovo, Macbook Pro, and Toshiba, respectively. This shows the utility of social engineering techniques used to obtain labeled data of the victim, even on a different laptop.

2.5.3 Model Profiling Scenario

We now evaluate the most unfavorable and the most realistic scenario where the attacker does not know anything about the victim. Conducting *S&T attack* in this scenario requires: (i) target-device classification, followed by (ii) key classification.

Target-device classification. The first step for the attacker is to determine whether target-device is a known model. We assume that the attacker collected a database of acoustic emanations from many keyboards.

When acoustic emanations from target-device are received, if the model of target-device is present in the database, the attacker can use this data to train the classifier. To evaluate this scenario, we completely excluded all records of one user and of one specific laptop of the original dataset. We did this to create a training set where both the victim's typing style and the victim's target-device are unknown to the attacker. We also added to the training set

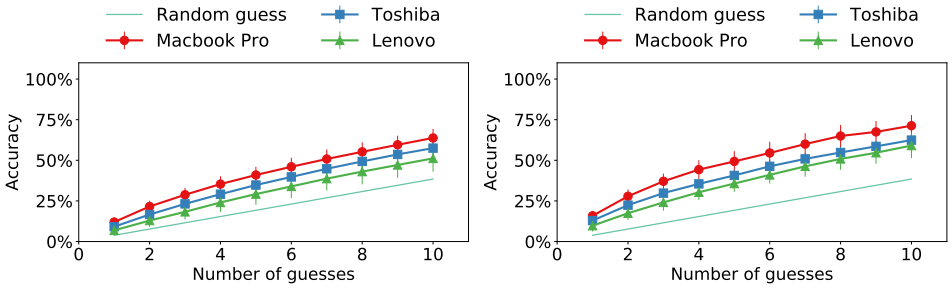
several devices, including 7 USB keyboards (Apple Pro, Logitech Internet, Logitech Y, and the four USB keyboards we recorded for our experiments), as well as 6 laptops (Acer E15, Sony Vaio Pro 2013, and the four laptops we recorded in the second data collection campaign: HP 250, Dell XPS, Asus Zenbook, and Dell Inspiron). We did this to show that a laptop is recognizable from its keyboard acoustic emanations among many different models: our experiments try to recognize one target-device model among 16 different candidate devices.

We evaluated the accuracy of k-NN classifier in identifying the correct laptop model, on the Touch typing and Skype filtered data combination. Results show very high accuracy of 96%, while a random guess would only have a probability of succeeding of $1/16 = 6.25\%$. This experiment confirms that an attacker can determine the victim's device, by using acoustic emanations.

We now consider the case when the model of target-device is not in the database. The attacker must first determine that this is indeed so. This can be done using the confidence of the classifier. If target-device is in the database, most samples are classified correctly, i.e., they "vote"correctly. However, when target-device is not in the database, predicted labels for the samples are spread among known models. One way to assess whether this is the case is to calculate the difference between the mean and the most-voted labels. We observed that trying to classify an unknown laptop consistently leads to a lower value of this metric: 0.22 *vs* 0.49. The attacker can use such observations, and then attempt to obtain further information via social engineering techniques, e.g., laptop [89], microphone [46] or webcam [100] fingerprinting.

Key classification. Once the attacker learns target-device, it proceeds to determine keyboard input. However, it does not have any extra information about the victim that can be used to train the classifier. Nonetheless, the attacker can use, as a training set, data obtained from another user on a laptop of the same model as target-device.

Results of *S&T attack* in this scenario are shown in Figure 2.13a. As expected, accuracy decreases with respect to previous scenarios. However, especially with Macbook Pro and Toshiba datasets, we still have an appreciable advantage from a random guess baseline. In particular, top-1 accuracy goes from a 178% improvement from the baseline random guess on Lenovo datasets, to a 312% improvement on Macbook Pro datasets. Top-5 accuracy goes from a 152% on Lenovo to a 213% on Macbook Pro.



(a) *Model Profiling* scenario

(b) *Model Profiling* scenario, "crowd" training data

Figure 2.13: Average attack accuracy

To further improve these results, the attacker can use an alternative strategy to build the training set. Suppose that the attacker recorded multiple users on a laptop of the same model of the target-device and then combines them to form a "crowd" training set. We evaluated this scenario as follows. We selected the dataset of one user on a given laptop, as a test set. We then created the training set by combining the data of other users of the same laptop model. We repeated this experiment, selecting every combination of user and laptop as a test set, and the corresponding other users and laptop as a training set. Results reported in Figure 2.13b show that overall accuracy grows by 6-10%, meaning that this technique further improves classifier's detection rate. In particular, this increase in accuracy, from 185% to 412% (with respect to a baseline random guess) yields a greater improvement than the approach with a single user on the training set.

Results show that *S&T attack* is still quite viable in a realistic VoIP scenario, with a target text which is both short and random. Moreover, this is possible with little to none specific training data of the victim, i.e., the attacker might even have *no prior knowledge* of the victim.

2.5.4 Variations

We thoroughly investigated the impact of different conditions on the baseline *Complete Profiling* scenario. In the following, we report the impact on *S&T attack* accuracy of different typists, different inexpensive recording devices that are commonly used in VoIP calls, small training sets, and of VoIP-specific issues such as degradation of call quality due to low bandwidth, and voice on top of keystroke sounds.

Impact of Typist.

We report here a detailed breakdown of results by different typists on the *Complete Profiling* scenario in Figure 2.14. Without loss of generality we only consider keystrokes recorded through internal microphones of laptops, to ease presentation of the results. We observed the same tendencies that we report over all the different recording devices.

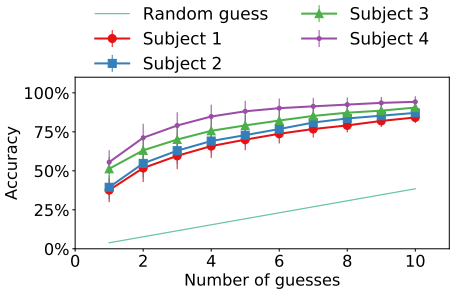
On these considered datasets, all subjects typed with all their fingers (*Touch typing*), however, as observed in Section 2.4.2, their typing speed is different. We observe that *S&T attack* often performs better on the slowest typist: accuracy is from 10 % to 35% better on Subject 4, who is the second slowest participant, and around 10% better on Subject 8, who had a non-uniform typing speed. However, Subject 7 who is the slowest is more susceptible only on 2 out of 4 keyboards, with the notable exception of the Fujitsu USB keyboard, where he is the worst performing. Another notable exception is with HP 9009, where the fastest subject is the best performing. Overall, we observe that the impact of typing speed is variable and, in most cases, accuracy is bounded within a few percentage points.

Impact of Recording-device.

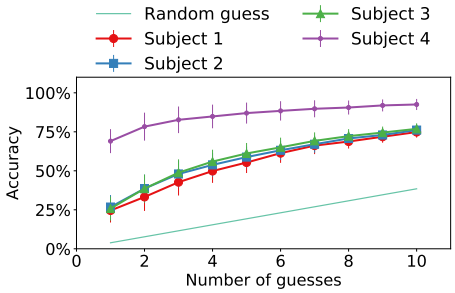
We report a detailed breakdown of the impact of different recording-devices, divided by target keyboard, in Figure 2.15.

Performance of *S&T attack* are consistent across different recording-devices. In particular, accuracy on some considered keyboards (HP 9009, HP 250 G5, Dell Inspiron 5000, Fujitsu KB900) is very similar across recording-devices— for example, the HP 9009 only has 3.5% difference in top-5 accuracy on average. Our results do not show significant difference between external and integrated keyboards, showing how *S&T attack* is feasible if the keyboard is not physically part of the same hardware of the microphone. The best performance is usually achieved by using a headset or the laptop itself as recording-device— with the notable exception of the HP KU-0316 USB keyboard, where the smartphone is slightly outperforming the other recording-devices.

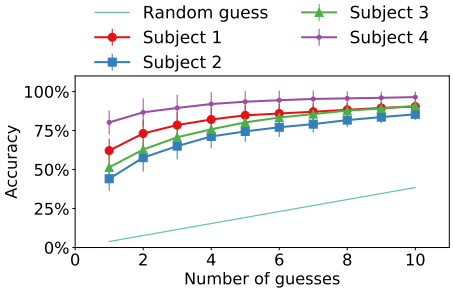
These results further prove that it is not necessary for *S&T attack* to have strong physical proximity between recording-device and target-device (or to sit in the same enclosing, as it is the case for laptops). Another important outcome of these experiments is the extremely good performance of *S&T attack* when using an inexpensive headset microphone as recording-device: indeed, its results are comparable, or even better, than the laptop's integrated microphone, and to the smartphone.



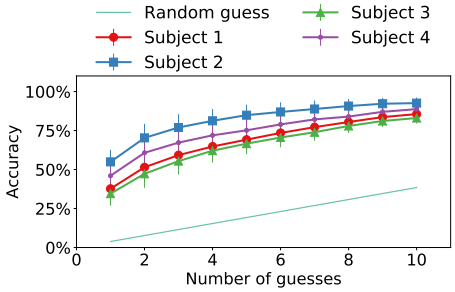
(a) AUKEY USB mechanical keyboard.



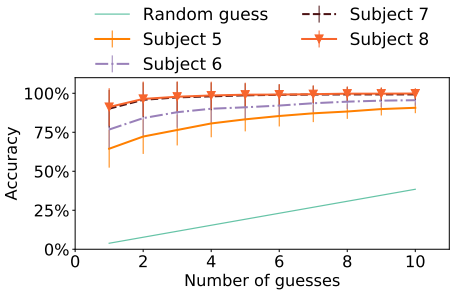
(b) Dell XPS 9333.



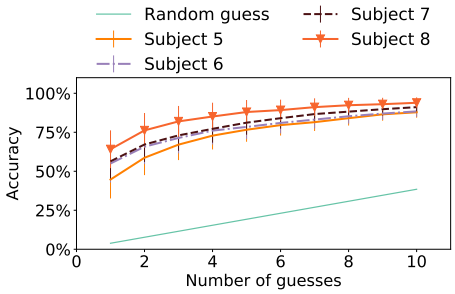
(c) HP HP 250 G5.



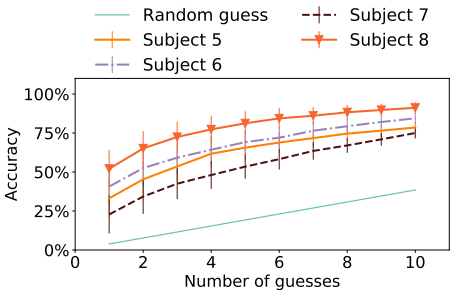
(d) HP 9009 USB keyboard.



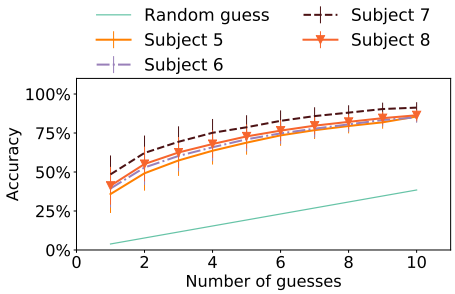
(e) Asus Zenbook UX303UB.



(f) Dell Inspiron 13 5000 series.

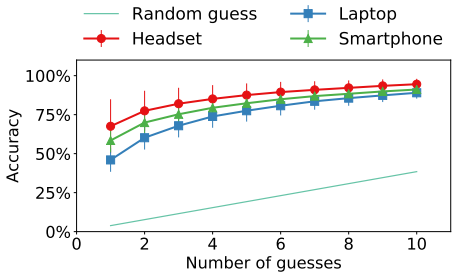


(g) Fujitsu USB keyboard.

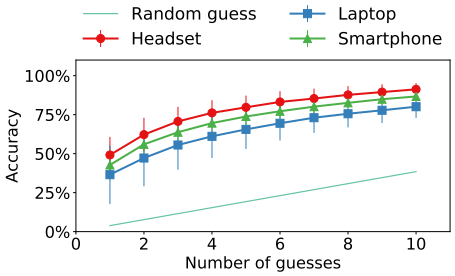


(h) HP USB keyboard.

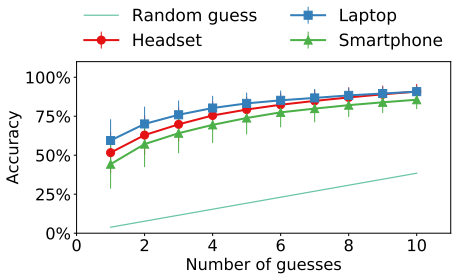
Figure 2.14: *S&T attack* performance — *Complete Profiling* scenario, Touch typing, Skype-filtered data; average accuracy for different typists.



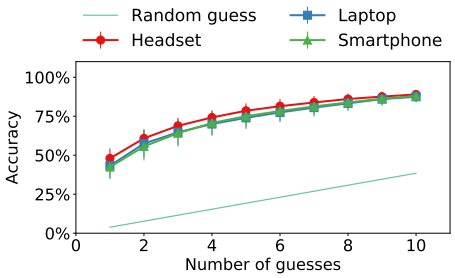
(a) AUKEY USB mechanical keyboard.



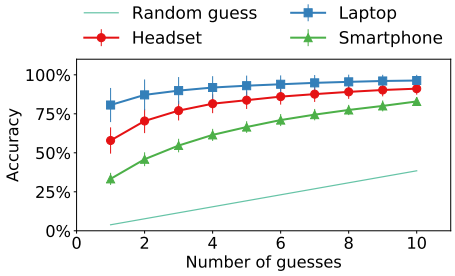
(b) Dell XPS 9333.



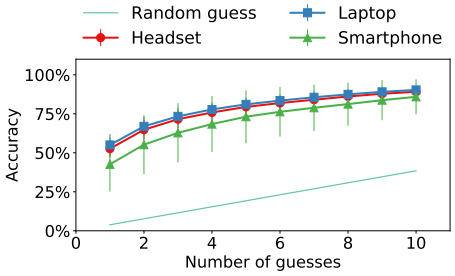
(c) HP 250 G5.



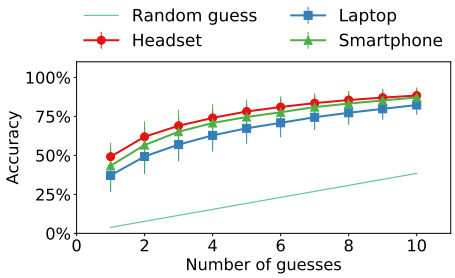
(d) HP 9009 USB keyboard.



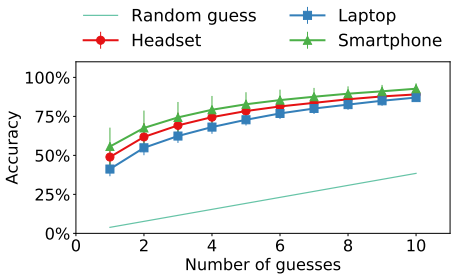
(e) Asus Zenbook UX303UB.



(f) Dell Inspiron 13 5000 series.



(g) Fujitsu KB900 USB keyboard.



(h) HP KU-0316 USB keyboard.

Figure 2.15: *S&T attack* performance — *Complete Profiling* scenario, Touch typing, Skype-filtered data; average accuracy for different recording devices.

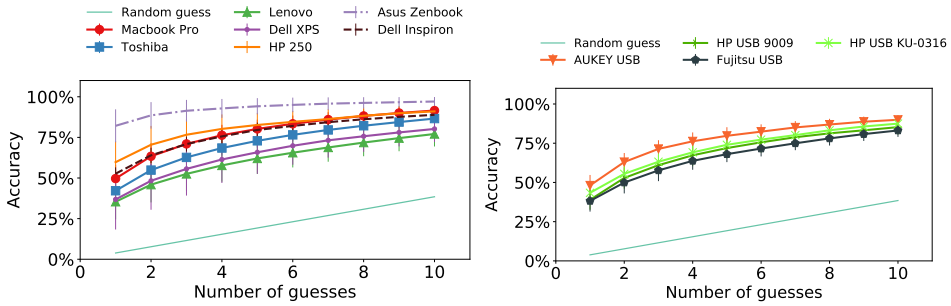
We believe that these results further prove the dangerousness of *S&T attack*. First, one could feel more inclined to type on his keyboard and multi-task during a Skype call if he's using a smartphone or a headset to do it — as the two devices are not the same. However, our results prove that smartphones and headset microphones are as good as the laptop's microphone in leaking keystroke sounds. Second, the effectiveness of eavesdropping through microphones shows that it could be possible to attack a **third party's** keyboard: someone who is only close to the user having the call, who could easily think to be immune to eavesdropping.

Impact of Small Training Set.

As discussed in Section 2.2.2, one way to mount *S&T attack* in the *Complete Profiling* scenario is by exploiting data accidentally disclosed by the victim, e.g., via Skype instant-messaging with the attacker during the call. However, our different training datasets have an important shortcoming: they do not respect the letter frequency distribution of any language, as models are trained with a balanced amount of letters — 10 samples for each letter. Ideally, the attacker should be able to mount *S&T* even if the training data is limited to a handful of chat messages. Therefore, to understand the impact of realistic letter frequency distribution, and of the attacker being able to collect just a few sentences together with their ground truth from the victim, we operated as follows:

- **"A-Z" dataset:** to account for realistic letter frequency distribution, we retained 10 samples of the most frequent letters according to the Oxford Dictionary [1]. Then, we randomly excluded samples of less frequent letters until only one sample for the least frequent letters was available. Ultimately, the subset contained 105 samples, that might correspond to a typical short chat message or a brief email. We then evaluated performance of the classifier trained with this subset, on a 10-fold cross-validation scheme. This random exclusion scheme was repeated 20 times for every fold.
- **"Quick Brown Fox" dataset:** As this dataset is composed by English sentences, we don't need to alter its letter frequency distribution. However, to reduce its size, we randomly selected 4 sentences out of the 10 available for each subject and device: 3 sentences (135 characters) form the training set — one, the test set. We then evaluated performance of the classifier. This random exclusion scheme was repeated 20 times for every laptop and user combination.

We show results on Touch typing, Skype filtered data in Figure 2.16.



(a) Laptop Keyboards. (b) USB Keyboards.

Figure 2.16: *S&T attack* performance — average accuracy when training on a small subset of samples that respects the letter frequency of the English language.

We observe that, with both datasets, there is an acceptable top-5 accuracy loss of 10%-25%, depending on the specific model and user. This mainly happens because now the training sets are very small (105 and 135 characters, respectively), and the less frequent letters for which we have only a few samples in the training set are harder to classify. However, with an impressively small training set, *S&T attack* still provides very good performance. There is an evident outlier: the Asus Zenbook laptop performs almost as good as with the full training set, despite the reduced size of the training set. These results further motivate the *Complete Profiling* scenario: the attacker can exploit even a few acoustic emanations that the victim discloses via a short message during a Skype call.

Impact of Fluctuating VoIP Bandwidth.

A prominent issue that stems from using VoIP to perform *S&T attack*, is that the SILK codec [133] degrades performance of *S&T attack*. For example, this codec reduces audible bandwidth whenever available Internet bandwidth is low, in order to use less bandwidth. This useful technical feature can, however, harm accuracy of *S&T attack*, because it degrades the sound spectrum.

In the experimental setup, both VoIP end-points were connected to a high-speed network. However, a realistic call might go over slower or more error-prone network links. Therefore, we performed a number of sample Skype calls between the two end-points while monitoring network load of the transmitter (i.e., the one producing emanations).

We experimented as follows: we filtered all data recorded on one Macbook Pro laptop by all the users with the HP typing style using Skype, together with a five minutes sample of the *Harvard Sentences*, commonly used to evaluate the quality of VoIP applications [119]. We initially let the Skype software use the full bandwidth available, and we measured that the software used an average of 70 Kbit/s without any noticeable packet loss. We subsequently limited the bandwidth of the transmitting machine at 60 Kbit/s, 50 Kbit/s, 40 Kbit/s, 30 Kbit/s, respectively, 20 Kbit/s. We observed that, with values below 20 Kbit/s, the quality of the call is compromised, because of frequent disconnections. *S&T attack* with such a small bandwidth is therefore not possible, and we argue that real users suffering this degradation of service would anyway not be willing neither able to continue the Skype call. Therefore, we believe the bandwidths we selected are representative of all the conditions on which we find the Skype software is able to operate. We then evaluated both the accuracy of *S&T attack*, and the quality of the call by using the voice recognition software CMU Sphinx v5 [91] on the Harvard Sentences. We show the results in Figure 2.17.

From Figure 2.17, we can see that, while there is no change to the accuracy of the voice recognition software until the 20 Kbit/s threshold, the classifier suffers a noticeable loss at and under 40 Kbit/s. This analysis shows that aggressive downsampling, and communication errors, can greatly hinder the accuracy of the attacker on the eavesdropping task, and that a loss of the order of 20% is to be expected if the connection speed is very low. We also observe that, at 20 Kbit/s, even if the Skype call is working, many samples of both the speech and keyboard sounds are lost or irreparably damaged due to the small bandwidth, and the final quality of the call might be undesirable for the user. However, it is realistic to assume Skype

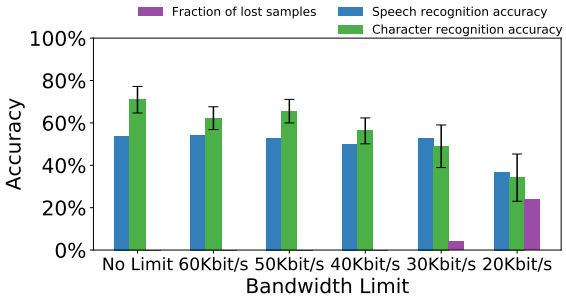


Figure 2.17: Voice recognition and *S&T attack* accuracy, on data acquired through Skype with different connection bandwidths.

to be always working at the best possible quality or almost at the best possible quality, since 70-50 Kbit/s are bandwidths that are small enough to be almost guaranteed.

Impact Of Voice.

In the experiments we described so far, we did not consider that the victim can possibly be talking while he types target-text. However, in a VoIP call, this can happen frequently, as it is probable that the victim is talking while he types something on the keyboard of his target-device. We evaluated the impact of this scenario as follows: we considered all the data of one user on the Macbook Pro laptop, consisting of 260 samples, 10 for every class, in a 10-fold cross-validation scheme. As usual, we used 9 folds to train a model as described in Step C (see Section 2.3). We then overlapped the sounds of the last fold (i.e., the test data) with random portions of a recording of a male voice spelling some Harvard Sentences [119]. The overlap was done by summing the amplitude of the waveforms, and taking care that it was not exceeding the maximum intensity allowed by its discrete representation. We underline that the algebraic sum of the amplitudes of two waveforms leads to a very good approximation of recording the two sounds on the same microphone at the same time. This physical phenomenon is known as superposition of waves (also defined as wave interference), that happens when two waves are incident on the same point (in our case, the microphone) — we refer the interested reader to the definition of acoustic interference in [19]. To account for the random overlap, we repeated the process 10 times, to have the keystroke sound overlap different random phonemes. We then evaluated the mean and standard deviation of the accuracy of the classifier.

We repeated the described experiment with different relative intensities of the voice against the intensity of the sound of the keystrokes. We started at -20dB, meaning that the keystrokes are 20dB louder than the voice of the speaker, and evaluated progressive steps of 5dB, until we had the voice of the speaker 20dB louder than the keystrokes. We performed this scheme on the data for all users on the Macbook Pro laptop, with Touch typing and data filtered with Skype. We show the results in Figure 2.18.

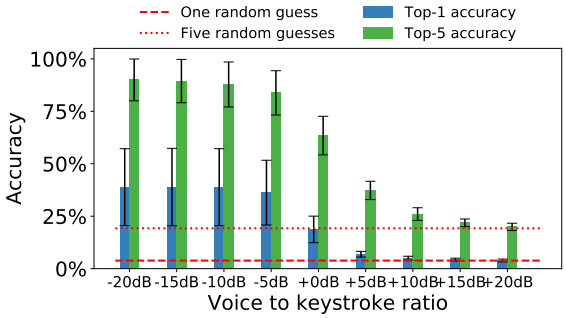


Figure 2.18: *S&T attack* performance — average accuracy, overlap of keystroke sounds and voice, at different relative intensity.

We observe that, from −20dB until 0dB, *S&T attack* does not suffer almost any performance loss, and then the accuracy rapidly decreases, until it reaches the random guess baseline at +20dB. We explain both the positive and the negative results with the phenomenon of auditory masking [137], where only the most powerful tone among all the tones at a given frequency is audible. This phenomenon is a direct consequence of wave interference. In our case, the greater the difference between the intensity of the sound of the keystroke and of the voice, the more only the frequencies of the louder sound will be audible.

We believe that the takeaway of our experiment is the following: keystrokes that are still audible can be predicted by S&T, while keystrokes that are covered by voice cannot, as too much information is lost because of wave interference. Given that the keystrokes are very loud when recorded from a laptop microphone, or in close proximity of the keyboard, it is likely that most keystrokes can be recovered despite the presence of voice.

2.5.5 Discussion and Limitations

To corroborate our claims, we evaluate the statistical significance of our obtained results. To do so, we applied a statistical test that measures how likely the results could have been generated by chance: the *binomial test* [42].

This statistical test allows to model scenarios where the observations are divided into two categories. In our case, we consider the two categories to be the results of *S&T* on our baseline scenario (Complete Profiling), and the baseline random guess. The null hypothesis is that our results are generated by chance — and thus the two categories are similarly likely. We first modeled the baseline random guess observation as a binomial distribution $B(n, p)$ with the following parameters: n (the number of trials) corresponds to the number of collected sample for a single letter — 320; while p (the probability of success on a single trial) corresponds to the top-10 accuracy of the baseline — 10/26.

We then perform the binomial test to compare our observation (the accuracy of *S&T*) to this binomial distribution modeling the baseline: we evaluate the probability $P(X > x)$, where X is the distribution B(n,p) and x is the minimum achieved score obtained by our attack. The minimum achieved score in our attack is 0.5. We then calculate $P(X > 0.5) = 1.09709 \cdot 10^{-5}$, allowing us to refuse the null hypothesis for $p < 0.001$ — thus showing statistical significance of our results.

The calculation of the binomial test, combined with all our results on different settings, target-devices, and recording-devices, shows that *S&T attack* is a provable threat for a large population of keyboard users. However, as previously discussed in Section 2.4.2, our study has some limitations. While we believe that good typists are somehow the *worst* case for *S&T attack*, as observed by highest accuracy obtained by the slowest typists, our experimental population is composed of mostly young people that have a certain degree of confidence with keyboards and computers. There are hints that *S&T* is applicable to less expert typists, as our HP typing style experiments tried to simulate the typing patterns of less accustomed users; however, we cannot affirm with certainty if our results generalize to greatly different populations, e.g., older users with very low proficiency with a keyboard.

2.6 S&T Practical Applications

We now consider two practical applications of the results of *S&T attack*: understanding words, and cracking random passwords. In particular, if the victim is typing English words, we analyze how *S&T* can help understanding such words. If the victim is typing a random password, we show how *S&T attack* can greatly reduce the average number of trials required in order to crack it, via a brute force attack. In the following, we report the results of these practical applications on the *Complete Profiling* scenario, and on the *Model Profiling* scenario.

2.6.1 Word Recognition

To evaluate how *S&T* helps understanding the words that the victim typed, we proceeded as follows. We drew a number of random words from an English dictionary; we call such words *actual words*. For each *actual word*, we reconstructed its typing sound combining the sound samples of each letter in the actual word. We used the sound sample of the letters we collected in Section 2.4.1. We then performed *S&T attack*, to obtain the top-10 predictions for each letter of the actual word. We combined these predictions with an English dictionary, by sorting it according to each word's probability given the predictions from *S&T*, and retain the first 20 words: the most probable *guessed words* for the actual word. We then count how many times the *actual word* is contained in the 20 most probable guessed words. Furthermore, we calculated the number of correctly guessed characters as the minimum of the Hamming distances between the actual word and the guessed words.

We report in the following the results of our experiments on 1000 random English words, on the *Complete Profiling* scenario. We show in Figure 2.19 how many times the actual word was in the set of guessed words. In particular, for 4-characters long words, we were able to recover 71% of them on the Lenovo laptop, up to 98% of the words on the MacBook Pro laptop. The accuracy increases for longer words, because the impact of wrong guesses is mitigated by the presence of more characters. From 10-characters long words, on MacBook Pro and Toshiba, we were able to recover 100% of the words. For Lenovo, from 15-character words.

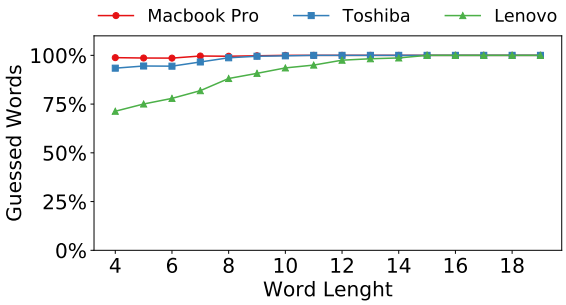


Figure 2.19: *S&T attack* performance — fraction of entirely correct recovered words of different lengths, by combining *S&T attack* guesses with a dictionary.

Figure 2.20 shows the average amount of correctly guessed characters. We observe that on the Toshiba laptops the average number of correctly recovered letters of the word is around 50% for words of length 4-7, and then

rapidly increases. Average correctness of the recovered word for Lenovo is between 80% and 90%, and rapidly approaches 100% from words of length 10, as discussed in Figure 2.19. With Macbook Pro laptops, *S&T* almost always recovers all the letters correctly, consistently with its very high results in the *Complete Profiling* scenario.

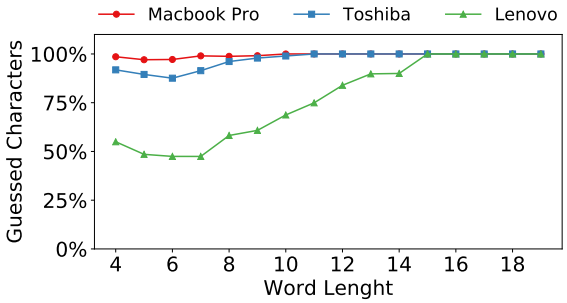


Figure 2.20: *S&T attack* performance — fraction of correctly guessed characters for not completely correct recovered words, by combining *S&T attack* guesses with a dictionary.

We believe these results highlight the threat posed by *S&T attack*. Indeed, when recovering text in a known language (e.g., English), additional information on words amplifies the results of *S&T*, allowing almost perfect word recovery in many settings. Furthermore, as shown by the confusion matrices in Figure 2.9, misclassifications often happen among neighboring keys — a language model could further be expanded to weight more predictions corresponding to neighbors of the first few guesses.

2.6.2 Password Recognition

Secure passwords that prevent dictionary attacks are random combinations of alphanumeric characters. In this section, we shed some light on how *S&T* can help cracking random password with an improved brute-force scheme that takes advantage of our results. We distinguish between two cases: characters that are typed with a single keystroke (i.e., lowercase letters, numbers, some symbols, the caps-lock key, and uppercase letters during caps-lock), and characters that are typed with two keystrokes, Shift + a key (i.e., symbols, and uppercase letters with Shift). We first detail a scheme that works ok single-keystroke characters, as it reflects our experimental results, and calculate the speedup of the improved brute-force scheme for such keys. We then outline how a more complete scheme could consider the Shift key.

The improved brute-force scheme for single-keystroke characters is as follows: given the x guesses of *S&T* for each of the n characters of the target password, we first consider all the x^n combinations of such characters. We then assume that the set of x guesses of the first character was wrong, and subsequently consider all the other characters. When we finish considering that one set of guesses was wrong, we consider all the combinations of two wrong guesses (i.e., first and second sets of guesses were wrong, first and third sets were wrong, up to the seventh and eighth sets). We repeat this scheme until we finally try the combinations where the classifier was always wrong. This brute-force scheme leverages the probability of success of *S&T* to minimize, on average, the required time to crack a password. If we consider a target password of 10 lowercase characters of the English alphabet, a regular brute-force scheme requires requires $\frac{(26)^{10}}{2} = 8.39 \cdot 10^{13}$ guesses to have 50% probability. On the *Complete Profiling* scenario, that we recall has an average top-5 accuracy of more than 90%, we only need $9.76 \cdot 10^6$ tries to have 50% probability. This corresponds to a very high average speedup of 10^7, and an entropy reduction of more than 50%. On the *Model Profiling* scenario, where we have a top-5 accuracy around 40%, we need $7.79 \cdot 10^{12}$ tries to reach 50% probability of cracking the password, which is still one order of magnitude better than plain brute-force attacks, on average. There is similar tendency if the attack guesses ten characters for every character of the password. These results are calculated on letters only (our collected dataset) — but other single-keystroke keys are similarly detectable by S&T, given training data.

To consider characters that are typed with two keystrokes, an improved scheme would need to precisely pinpoint the pressure and release of the Shift key. We leave the development of a complete acoustic-enhanced password cracker as future work, and here hint on the following fact. As discussed in Section 2.3.2, keystroke sounds have a *press* and a *release* peak. Users keep the Shift key pressed down to type special characters — therefore, the attacker could detect the Shift key by finding keystrokes that have press and release peak separated by other keystrokes. This consideration can be used to deduct that properly-detected subsequent single key presses are instead the alternate character of such key. Another possibility is to train a dedicated attacker model to detect the Shift key only. We leave this as future work.

2.7 Countermeasures

In this section, we present and discuss some potential countermeasures and analyze their efficacy in preventing *S&T* and other attacks that use statistical properties of the sound spectrum. While we discussed some countermeasures in our preliminary version of this work [43], here we provide a complete countermeasure, able to completely prevent attacks based on spectral properties of keystroke sounds.

First, we argue that designing good countermeasures to keyboard acoustic eavesdropping attacks, such as *S&T*, is difficult. This is because the underlying goal for the user is to perform high-quality, undisturbed VoIP calls. However, many countermeasures impact VoIP call quality. For example, one simple countermeasure is applying a short "ducking" effect, a technique that drastically lowers microphone volume and overlaps it with a different sound, whenever a keystroke is detected. However, this approach would greatly degrade voice call quality. Another possible countermeasure, proposed in [8], is to introduce fake keystroke sounds during the call, when the user is typing. However, we note that this countermeasure requires training with keystroke sounds of the specific laptop in use, and users might find it annoying, as the amount of noise increases greatly. We believe that ideally, an effective countermeasure should be minimally intrusive and affect only keystroke sounds.

Moving from these observations, we design a less intrusive countermeasure against all techniques that use sound spectrum information — performing short random transformations to the sound whenever a keystroke is detected. One intuitive way to do this is to apply a random multi-band equalizer over a number of small frequency bands of the spectrum. This allows us to modify the intensity of specific frequency ranges, called "bands". Each band should be selected at random and its intensity should be modified by a small random amount, thus effectively changing the sound spectrum. All these random decisions should change every few seconds, to prevent the adversary from learning the modifications. An alternative approach, to reduce intrusiveness, can be to trigger the countermeasure only when the OS detects a keystroke.

To show the efficacy of this countermeasure, we ran the following experiment: we considered, as usual, our datasets in a 10-fold cross-validation scheme. As detailed in Section 2.3, we trained the classifier (Step C) with 9 folds. However, before evaluating the accuracy of the attacker, we transformed the test data as follows: we applied a multiband equalizer with 50 bands to the test data only, where each band has a random center between

100 Hz and 12,000 Hz, a very high resonance factor Q of 20 (meaning that the gain of each band has impact only on neighboring frequencies), and a random gain between -5dB and +5dB. We then measured the accuracy of the classifiers on the test data processed by our countermeasure.

We report that our countermeasure proves extremely effective: top-1 accuracy of the test data filtered by the countermeasure is 3.75%–6.18% (random guess baseline is 3.84%) and top-5 accuracy is 18.70%–24.70% (baseline is 19.23%). Even top-10 accuracy is greatly impacted: 37.38%–44.94% (baseline is 38.46%). This shows that *S&T* has no sensible advantage over random guesses after applying our countermeasure to keystroke sounds.

Our approach also allows the speaker's voice to remain intelligible. To show this, we operated as follows: we selected a corpus of recordings of 30 spoken sentences, and used the IBM Watson speech-to-text engine to extract the content of the sentences. We then applied our countermeasure on the sentences, and used again IBM Watson to obtain their transcripts. To account for the randomness in the countermeasure, we repeat the process 5 times and average the results. We report a very high similarity between the transcripts before and after the countermeasure: 93.69%(±11.89%). This is further confirmed by considering the average Pearson correlation between recordings before and after our countermeasure: 86.35%. As a final validation, we listened to the resulting files and confirm the acceptable quality — we uploaded an archive with the original files and their transformations by the countermeasure on the project webpage[6].

As a final experiment, we tested whether this countermeasure is effective against different spectral features: we repeated our experiment extracting both MFCC and FFT features, applying the countermeasure on test data. Results in Figure 2.21 show *S&T* accuracy, with and without the countermeasure, for MFCC and FFT features.

We observe that our proposed countermeasure also successfully disrupts FFT features (used, e.g., in [10, 67, 69, 103], besides MFCC features (used in this work, and in [8, 149]). Indeed, the accuracy of *S&T* goes down to the baseline random guess for both sound features.

A more simplistic approach is to use software or emulated keyboards, i.e., those that appear on the screen and are operated by the mouse. Similarly trivial ideas include: (1) activating a mute button before typing, or (2) not to type at all whenever engaged in a VoIP call.

[6] https://spritz.math.unipd.it/projects/dst

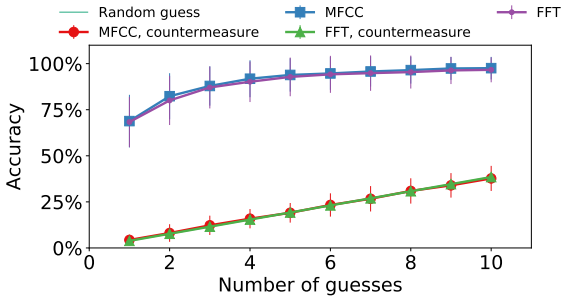


Figure 2.21: Average accuracy of single key classification against a random equalization countermeasure. Accuracy of both FFT and MFCC features after countermeasure lies exactly on the random guess baseline.

2.8 Summary

We demonstrated a highly accurate VoIP-based remote keyboard acoustic eavesdropping attack. We first described a number of practical attack scenarios, using VoIP as a novel means to acquire acoustic information under realistic assumptions: random target text and very small training sets, in Section 2.2. Then, in Section 2.3 we demonstrated an attack with these assumptions in mind and carefully selected the tools to maximize its accuracy. In Section 2.4 we detail the data we collected to then thoroughly evaluate *S&T attack* in several scenarios in Section 2.5. In Section 2.6 we showed that *S&T attack* can be used to recover both random and non-random text, and finally we discussed some potential countermeasures to *S&T* and other attacks that leverage spectral features of keyboard sounds, in Section 2.7.

We believe that this work, due to its real-world applicability, advances the state-of-the-art in acoustic eavesdropping attacks. *S&T attack* was shown to be both feasible and accurate over Skype, in all considered attack scenarios, with none or minimal profiling of the victim's typing style and keyboard. In particular, it is accurate in the *Model Profiling* scenario, where the attacker profiles a laptop of the same model as the victim's laptop, without any additional information about the victim. This allows the attacker to learn private information, such as sensitive text or passwords. We also took into account VoIP-specific issues – such as the impact of audible bandwidth reduction, and effects of human voice mixed with keystroke audio – and showed that *S&T* is robust with respect to both. Finally, we discussed some countermeasures and concluded that *S&T* is hard to mitigate.

Future Work. The high accuracy of *S&T* highlights the importance of

robust countermeasures to such sound-based side channels. Our proposed countermeasure is able to completely stop *S&T* (and other attacks that leverage spectral properties of keystroke sounds). However, as it might impact the call quality, more thorough user studies are needed to validate our preliminary claim that voice is not affected by the countermeasure. Another interesting direction might be, instead of random equalization in hopes of disrupting important features, leveraging results on adversarial machine learning and transferability to generate the minimal perturbations that cause attacks such as *S&T* to misclassify keystroke sounds.

Chapter 3

Hand Me Your PIN! Inferring ATM PINs of Users Typing with a Covered Hand

The wide deployment of various Cyber-Physical Systems (CPS) has a significant impact on our daily lives. Unfortunately, the increased use of CPS also brings more threats to users. This is especially pronounced considering new attack vectors that use machine learning approaches. As such, threats become a global issue, and the need to design secure and robust systems increases. One common security mechanism in devices like Automated Teller Machines (ATMs) and Point of Sale (PoS) depends on the security provided by the Personal Identification Numbers (PINs). While ATMs and PoS devices are widely used [1], many people do not consider security risks and defenses beyond those commonly mentioned [2]: i) hide the PIN while typing, and i) make sure no one watches the screen (shoulder-surfing attack). In the context of financial services, ISO 9564-1 [77] specifies the basic security principles for PINs and PIN entry devices (e.g., PIN pads). For example, to mitigate the shoulder surfing attacks [27, 54], the standard indicates that i) PIN digits must not be displayed on a screen, and ii) the duration and type of feedback sound emitted must be the same for each key. Consequently, as a compromise between security and usability, PIN entry systems display a fixed symbol (e.g., a dot) to represent a digit being pressed and provide the same audio feedback (i.e., same tone, same duration) for all

[1] https://sdw.ecb.europa.eu/reports.do?node=1000001407
[2] https://www.hsbc.com.hk/help/cybersecurity-and-fraud/atm-scams/

keys. Thus, the combination of security mechanisms enforced by standards and the common precaution measures taken by users should provide sufficient protection. Unfortunately, the attackers also improve their approaches over time and consider more sophisticated attacks.

The security of ATM and PoS devices is of great concern as millions of such devices are used [48]. Resourceful attackers that succeed in attacking even a small percentage of those devices can cause significant damage considering costs and public perception. This problem is especially pronounced as last years brought significant developments in the attack techniques [14, 36, 97]. At the same time, attacking ATM or PoS devices is not easy, especially if considering realistic settings. Most of the state-of-the-art attacks can be defeated by a careful user covering the PIN that is entered. Recent results that consider thermal cameras are also difficult to succeed, depending on the keypad type and the time users spend operating the device. The attacker can also use timing or acoustic attacks to infer information about the entered digits, but they are not as effective as the state-of-the-art attacks since they require additional information such as thermal residues [36], making it challenging to apply realistically such attacks.

This work proposes a novel attack aiming to reconstruct PINs entered by victims that cover the typing hand by the other hand. More precisely, we leverage the advances in the deep learning domain to develop an attack predicting what PIN is entered based on the position of the user's hand and the movements while pressing the keys. Our attack gives high accuracy rates even in the cases when the user perfectly covers the typing hand. What is more, our attack reaches higher accuracy values than previous works that needed to consider several sources of the information at the same time (timing, sound, and thermal signatures) [36].

Our attack considers a profiling setting where the attacker has access to a PIN pad that is identical (or at least similar) to the one used by the victim. Then, we build a profiling model that can predict what digit is entered on the target device. This is the first attack on PIN mechanisms that works even when the PIN is covered while being entered to the best of our knowledge. Our attack demonstrates that the ATM and PoS security mechanisms are insufficient, and we must provide novel defenses to mitigate attackers. We made our code and datasets publicly available at https://spritz.math.unipd.it/projects/HandMeYourPIN.

Main contributions

- We propose a novel attack to infer PINs from videos of users covering the typing hand with their non-typing hand.

- We demonstrate that our attack can reconstruct 30% of 5-digit PINs and 41% of 4-digit PINs within three attempts, showing that hiding the PIN while typing is insufficient to ensure proper protection.
- We evaluate our attack via extensive experiments, collecting videos of 5 800 5-digit PINs entered in a simulated ATM by 58 participants. We conduct a study to assess humans' accuracy in inferring covered PINs from videos. We show that our attack outperforms humans, achieving a four-fold improvement on reconstructing 5-digits PINs within three attempts.
- We pre-process our dataset, and we make it publicly available to the research community. We hope this is beneficial to understand the problem better and propose possible solutions.
- We discuss several countermeasures that would make the attack more difficult to conduct. We perform an analysis on the attack performance when covering the PIN pad (coverage 25%, 50%, 75%, and 100%) and show that attacks are possible even when using this countermeasure.

3.1 Threat Model

The attack is performed when a victim interacts with a generic ATM keypad and types the PIN. The ATM is equipped with a PIN pad that emits a feedback sound when a key is pressed. The feedback sound is the same for all the keys of the PIN pad. The ATM is equipped with a monitor where obfuscated symbols appear when users enter a PIN to mask the entered digits. We do not assume that the ATM or its PIN pad have been compromised during the attack. Our approach can be considered an alternative to card-skimmer attacks since we consider a different source of information to retrieve the PIN. Usually, card-skimming attacks rely on fake PIN pads that directly record the entered digits [121], while our approach infers the PINs from a video.

3.1.1 Attacker

The attacker is a malicious user aiming to steal the victim's secret PIN. The attacker can place a hidden camera near the ATM to record the PIN pad. We make no assumptions about the type of camera used by the attacker except that it records in the visible spectrum [3]. We assume that the camera can easily be hidden close to the ATM while keeping a direct view of the

[3] We will use cheap and easily-concealable video sensing equipment, where standard RGB cameras fit such requirements.

PIN pad (i.e., a pinhole camera if the attacker has access to the ATM [4] or any standard camera placed outside the ATM chassis). We also do not assume any specific position for the camera, but we discuss various camera placements' advantages. We primarily consider the scenario where the attacker uses only one camera, but we also discuss the attack performance when using multiple cameras.

The attack may take place together with different card stealing approaches: i) card skimming both on chip [29] or magnetic stripe [121] (currently, the two payment-enabling technologies work together [45]), ii) exploiting a relay attack on a contactless card [65], and iii) physically stealing the victim's card.

We assume a profiling side-channel attack where the side-channel information comes from the video of the victim's hand while entering the PIN. More precisely, side-channel information is the position of the victim's hand and the hand movements (both moving the hand/fingers to reach different keypads or movements observable due to muscle movements while a certain keypad is pressed). The attacker can record a number of PINs entered on a copy of the ATM device and train a profiling model to predict what key is pressed. The attacker can retrieve the timestamps when the victim has typed the single keys on the keypad and can do so by listening to the audio of the video recording. There are two different types of sound clues that the attacker can exploit: the first one is the feedback sound made by the keypad when a key is pressed [36], the second one is the sound of the physical button of the keypad that is pressed. External noise does not prevent the attacker from extracting the keypresses, as the camera is close enough to the keypad. As such, the sound can still be identified in the audio track. If, for any reason, the attacker has no way to retrieve the timestamps from the recorded audio (or if there is no audio at all), it is possible to place the camera to record both the keypad and the screen of the ATM [14]. This allows the attacker to extract the keypresses' timing by looking at the PIN masking symbols appearing on the screen. Common masking symbols are usually dots and asterisks. The attacker can use any method to build a profiling model to predict what keys are pressed. We consider the top three predictions as a measure of success since most ATMs will allow entering the PIN three times before blocking the card. Finally, we do not assume that the PIN has any specific structure (pattern) that could be used to improve the attack performance further.

[4] https://www.sperrywest.com/cameras/

3.1.2 Victim

We assume that the victim adopts basic countermeasures against card-skimming attacks, such as covering the hand while entering the PIN. The attacker does not need to be there when the victim types the PIN, as the attacker can freely access the camera's recorded video, either remotely or at a different time.

3.2 Attack Approach

Our attack assumes that the attacker has access to a training device and controls the PIN selection. Additionally, the attacker knows the layout of a target device and will select the training device to be similar. The attacker does not know the specific person to be attacked or the PIN for the attacked device.

3.2.1 Attack Phases

We can divide the attack into three phases: Phase A – Training, Phase B – Video Recording, and Phase C – PIN Inference. Figure 3.1 shows the required steps for the attack.

Phase A – Training

The attacker selects an ATM as the target of the attack. Next, the attacker sets up a replica of the target ATM. This replica does not have to be a faithful copy of the original, as our model takes in as input a crop around the keypad of the ATM. Therefore, the attacker must use a keypad similar to the one on the target ATM. The best situation is when the attacker can retrieve the same PIN pad model. Alternatively, the attacker can also use PIN pads that differ slightly (e.g., the key spacing can vary by a few millimeters). Note that the layout of ATM PIN pads has to follow the ISO 9564 standard [77]. The attacker uses the ATM replica to build the training set, simulating the victim's behavior while entering the PIN (i.e., covering the typing hand). The attacker must enter sequences of PINs on the replica PIN pad, including all ten digits (i.e., all the digits must be included in the training set). Without losing generality, the attacker can use a USB PIN pad that logs the keys pressed and the corresponding timestamps. The attacker uses this information to segment the videos and labels them. Leveraging the logs, the attacker builds a training set containing, for each key pressed, a sequence of frames and the corresponding

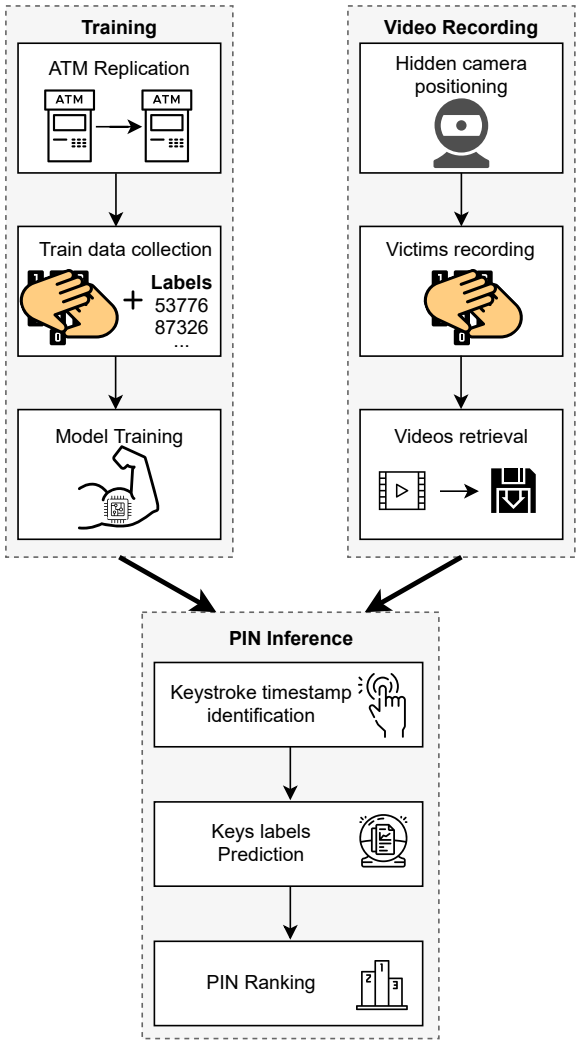


Figure 3.1: The attack step-by-step. The data collection process does not necessarily need to happen before the attacker steals the victim's PIN. Still, it is a required step of the attack.

label (digits). Finally, the attacker trains the predictive model on the collected training set. For a detailed discussion on the implemented model, we refer readers to Section 3.3.5.

Phase B – Video Recording

The attacker hides a camera near the target ATM to record the PIN pad. There are multiple places where the camera can be placed, and depending on this, the attack can be easier or more challenging to succeed. The camera records the victim while entering the PIN and covering the

PIN pad with the non-typing hand. The attacker retrieves the recorded video from the remote camera.

Phase C – PIN Inference

The attacker's goal is to infer the victim's PIN based on the video recorded during the PIN entering. First, the attacker retrieves the timestamps from the recorded video. The attacker can use both the pressed keys' feedback sound or the masking symbols appearing on the screen while the victim enters the PIN to perform this task. Leveraging the timestamps, the attacker performs the same procedure as in Phase A to generate an attack set. Differing from the training set, the attack set contains a sequence of frames for each victim key pressed but no information about the related label. The adversary detects in the attack set the frames corresponding to a PIN entry, and splits the video into N sub-sequences where N represents the number of digits composing the PIN. For each sub-sequence, the adversary applies the model trained in Phase A. The model provides the probability of each class (i.e., the ten possible digits) to be the one corresponding to the input sub-sequence. Exploiting the N sub-sequences predictions, the attacker builds a rank of PINs in the descending order of their probabilities. In particular, the probability of a PIN corresponds to the product of the predicted probabilities of its digits.

3.2.2 Attack Settings

We consider three realistic attack scenarios:

1. **Single PIN pad** scenario: the attacker knows the model of the target PIN pad and obtains a copy of it to carry out the training phase. While this scenario may seem unrealistic, we note it is not difficult to obtain a specific keypad copy. Indeed, the attacker can easily obtain information about the ATM to be attacked and then buy the keypad with the same layout. Naturally, there can be certain differences concerning how sensitive the keypad is (for instance, due to usage, pads can become somewhat more difficult to press), but our experiments indicate such differences are not substantial enough to pose issues for deep learning models.
2. **PIN pad independent** scenario: this is the most challenging scenario. The attacker does not know or cannot retrieve the model of the target PIN pad. The training phase is performed on a PIN pad with similar characteristics to the target (e.g., shape, distance between keys, keys layout, and the sensitivity of keys).

3. **Mixed** scenario: as for the *Single PIN* scenario, the attacker knows the target PIN pad model. In this case, the training is performed on two PIN pads: a copy of the target and at least one PIN pad with similar characteristics. Considering several keypads in the training set makes sense when 1) the attacker is not certain about the keypad model, 2) the attacker assumes that the keypad will behave differently due to environmental conditions, 3) the attacker aims to attack multiple types of keypads (ATMs) with the same machine learning model, and 4) for any reason, the attacker did not manage to obtain enough training examples with a single keypad. We also note that using more keypads in the training set makes the training process more difficult and reduces the chances to overfit (i.e., we can consider different keypads as one keypad with noise, having the regularization effect [28]).

3.2.3 Camera Positions

Since our threat model allows the arbitrary position of the camera, we discuss several representative scenarios. We consider positions at the top of the ATM preferable for the attacker as lower positions of the camera result in no visibility of the hand pressing the keys if the other hand is covering it. We also consider settings at the front side of the chassis as they give better visibility for the attacker and are significantly more difficult for the victim to notice the camera.

Then, without loss of generality, we can discuss three main positions for the camera to provide good results. The camera can be positioned in the top left, center, or right corner. If the camera is positioned in the right corner and the person entering the PIN is right-handed, it will be easier to observe the entered digits. The same happens for the camera in the left corner and the left-handed person. However, if the camera is in the center position, it does not favor any specific setting, making it the most general setting, but it also makes it somewhat more challenging to conduct the attack than the left/right position and left/right-handed persons. We will concentrate on the top center position of the camera mounted on the chassis's front side.

3.3 Experimental Setting

To assess the feasibility of our attack on all the scenarios described in Section 3.2, we collected two datasets containing videos of people covering their typing hands while entering PINs. This section first illustrates the differences between the considered PIN pads and then describes our data collec-

tion procedure. Finally, the adopted video pre-processing, the setup used to run the experiments, and the implemented deep learning models are presented.

3.3.1 Devices under Test

We performed two separated data collection campaigns on two different real-world ATM metal PIN pads: DAVO LIN Model *D-8201 F* [5] (Figure 3.2a) and Model *D-8203 B* [6] (Figure 3.2b). In particular, we report the following differences between the two PIN pads:

- Model *D-8201 F* has a dimension of 100 mm x 100 mm, while Model *D-8203 B* has a metal surface of 92 mm x 88 mm and is contoured by rubber protection.
- The horizontal key spacing is 1 mm larger between each key in Model *D-8203 B*.
- The keys of Model *D-8203 B* are harder to press and slightly taller than Model *D-8201 F*.
- For usability reasons, both the PIN pads emit a specific feedback sound (the same for all keys) when a key is pressed. The frequencies of the feedback sounds are 2 900 Hz for Model *D-8201 F* and 2 500 Hz for Model *D-8203 B*.

For the data collection, we embedded the PIN pad into a simulated ATM (see Figure 3.3). We chose the simulated ATM's size based on a real-world ATM [74]. In particular, the simulated ATM has a width of 60 cm, a height of 64 cm, and a depth of 40 cm. At 15 cm of height from the frame's base, we inserted a shelf to position the PIN pad and the monitor. The height of the PIN pad from the ground is 110 cm. We used three *Logitech HD C922 Pro* webcams anchored on the ATM's chassis to perform the video recording. A central webcam was placed 30 cm above the PIN pad, while the other two webcams were placed on the two top corners of the chassis 42 cm away from the PIN pad. The camera's maximum resolution is 1 080p with an acquisition rate of 30 fps. We recorded the videos with a resolution of 720p and an acquisition rate of 30 fps.

[5] https://www.davochina.com/4x4-ip65-waterproof-industrial-metal-keypad-stainless-steel-keyboard-for-access-control-atm-terminal-vending-machine-p00103p1.html

[6] https://www.davochina.com/4x4-ip65-stainless-steel-numeric-metal-keypad-with-waterproof-silicone-cover-p00126p1.html

(a) *DAVO LIN Model D-8201 F* (b) *DAVO LIN Model D-8203 B*

Figure 3.2: The PIN pads used in the data collection.

3.3.2 Data Collection

The first data collection involved 40 participants (age 38.23 ± 11.43, 24 male and 16 female). The second data collection involved 18 participants (age 29.50 ± 5.74, ten male and eight female). Both collections include right-hand participants only. All the participants gave their approval to collect and use the data by signing informed consent. All the data have been anonymized and used by the authors of this paper for research purposes only. Participants were asked to stand in front of the test ATM and cover the typing hand while entering the PIN during the experiment. The participants were left free to type as they pleased. The goal is to emulate an ATM user that is hiding the PIN, preventing possible shoulder-surfing attacks. Each participant typed 100 5-digits PINs randomly generated, divided into four sequences of 25 PINs. This split into four sequences has been performed to include short breaks in the experiments and prevent the participants from getting tired. The PINs were showed one at a time on the ATM screen: once a PIN has been entered on the PIN pad, the user had to press the enter button to move to the next PIN. We recorded a total of 5 800 random 5-digit PINs, resulting in a balanced dataset per digit. Since our study aims to reconstruct the PIN from the video sequence, regardless of the user's typing behavior and familiarity with the PIN or the PIN pad, we decided to randomize PINs rather than asking users to enter the same PIN multiple times. This approach generalizes the attack, which can be applied to mnemonic PINs and One-time Passwords (OTPs). Moreover, we collected the environmental audio (exploiting the webcam microphone) and the keylogs of the PIN pad through the USB interface during the experiment.

In particular, for each digit entered, we collect both the key down and key up events. We synchronized the video recordings with the timestamp of the key events. This information was collected to build the ground truth for the conducted experiments. The dataset is available at https://spritz.math.unipd.it/projects/HandMeYourPIN.

Figure 3.3: Our experimental setup. The cameras are visible but they can be hidden into the frame of an ATM. In all other aspects, we reproduced a common ATM layout in detail.

3.3.3 Pre-processing Video

Once the data acquisition phase is done, we need to pre-process the videos. For each video frame, we applied the following steps: i) convert the video frames to grayscale; ii) normalize the input so that all pixel values lie in the range $[0, 1]$; iii) crop the frames by centering the PIN pad, cutting off the irrelevant part of the background; (iv) resize the image to 250 x 250 pixels. After these steps, we applied a segmentation on each PIN video to obtain sub-sequences of frames corresponding to a single keypress (e.g., 5 sub-sequences for a 5-digit PIN). We extracted the keypress's timestamp from the recorded feedback sound of the PIN pad following the procedure explained in [36]. In particular, we filtered the audio signal using a band-pass filter, centered on the specific frequency of the feedback sound (i.e., 2 900 Hz

for Model *D-8201 F* and 2 500 Hz for Model *D-8203 B*). By identifying the peaks of the filtered signal, we could detect the timestamp of the target key (TK). This allowed us to extract a set of frames in each TK neighborhood. For each TK, the maximum number of frames (full-neighborhood) consists of all the frames ranging from the key preceding the TK to the key following the TK. If the TK corresponds to the first digit of the PIN, we consider only the frames between the TK and the next keypress. Analogously, if the TK corresponds to the last digit of the PIN, the frames considered are only those between the TK and its previous keypress. Since our model requires all input samples to have the same length, we decided to keep 11 frames for each sample. This value corresponds to the average number of frames in the full-neighborhood after removing the outliers over 3σ. To keep the TK at the center of the frames' sequence, we decided to consider five frames preceding the target keypress and five frames succeeding it, for a total of 11 frames per sample (including the target frame). There are three borderline cases: the TK is the first digit in the sequence, the TK is the last digit in the sequence, and the full-neighborhood has less than 11 frames. We apply black frame padding to keep the TK at the center of the sequence for these cases. In particular, if the TK is the first digit of the pin, five black frames are added at the head of the sequence, while if TK is the last digit of the PIN, we add five black frames at the end of the sequence. Finally, if there are not 11 frames in a sequence, we pad both the head and the tail (so that the TK is at the center).

3.3.4 Machine Learning Setup

For our experiments, we used a machine equipped with a CPU Intel(R) Xeon(R) E5-2670 2.60GHz, 128GB of RAM, and three Tesla K20m where each GPU has 5 Gb of RAM. To implement the machine learning models, we used Keras 2.3.0-tf (Tensorflow 2.2.0) and Python 3.8.6.

3.3.5 Prediction Models

Our approach aims to predict which key has been pressed on a PIN pad, exploiting only the video of a user covering the typing hand with the other hand. Since we deal with sequences of images, we implemented a model using Convolutional Neural Networks (CNNs) [92] and a Long Short-Term Memory (LSTM) [72]. The CNNs perform spatial feature extraction for each frame of a sequence, while the LSTM exploits these features to extract temporal patterns for the whole sequence of frames. The output of the LSTM passes through a multilayer perceptron (MLP) and a final Softmax

activation function layer with ten units (as there are ten digits). This model is known in the literature as Long-term Recurrent Convolutional Network (LRCN) [51]. In Keras [86], such architecture can be implemented using the TimeDistributed wrapper throughout all the CNNs layers, which causes the same convolutional filters to be applied to all the timesteps (i.e., the frames) of the input sequence.

We split our dataset into train, validation, and test sets. Each set's size depends on the attack scenario and is discussed in detail in Section 3.4. We explored different hyperparameters by using the randomized grid search. Based on a preliminary assessment, we set the ranges for specific hyperparameters (i.e., we limit the upper value for specific hyperparameters) to speed up the search. In particular, for the CNNs, we tested [3x3, 6x6, 9x9] kernel sizes. We also varied the number of convolutional layers in the range [1, ..., 4]. In the following dropout layer, we varied the dropout rates in the range $[0.01, 0.05, 0.1, 0.2]$. For the LSTM architecture, we varied the number of layers in the range [1, ..., 3], and the unit size in $[32, 64, 128, 256]$. We also assessed our network's performance using a Gated recurrent unit (GRU) instead of the LSTM. Finally, we examined the number of layers for the MLP in the range 1 to 4 and the number of units in the range $16, 32, 64, 128$. We tried two types of architectures for MLP: i) all the layers have the same number of units, ii) layers with decreasing number of units (funnel architecture), with every next layer having half the units of the previous one.

After a tuning phase, we selected a structure consisting of four convolutional layers (Conv2D in Keras) with ReLU activation functions, each followed by a pooling layer (MaxPooling2D in Keras). Three convolutional layers have a filter size of 3x3, and one (the second one) has a filter size of 9x9. Each pooling layer has a filter size of 2x2. The number of filters in the convolutional layers doubles at each layer, starting from 32 filters in the first layer, ending up at 256 filters in the fourth layer. We added a dropout layer (dropout rate 0.1) after the last pooling layer to prevent overfitting. The output is then flattened, preserving the temporal dimension to provide a sequence of temporal features to the following LSTM. A single layer LSTM with 128 units resulted in the best validation with a hyperbolic tangent activation function. Finally, for the MLP, we used four fully connected layers, with 64 units each, followed by the Softmax activation layer with ten units (i.e., the number of classes we want to predict). We used the categorical cross-entropy loss function and the stochastic gradient descent (SGD) optimizer. Finally, we set the model to evaluate the accuracy metric. We set the batch size to 16 and the learning rate to 0.1. We tested for 70 epochs since we found that the model always converged within this number of epochs. In

Appendix A.1, we provide additional details. Our experiments indicate that the classification task we conduct is relatively difficult, and one needs to use sophisticated deep learning architecture for good results. Still, we note that the architecture we use is in line with the state-of-the-art results for hand tracking problem [76, 90]. Finally, we observed significant changes in the performance depending on the specific hyperparameter choice, indicating a need for detailed tuning for the respective tasks.

In a real-world context, it might not be possible to reproduce precisely the experimental conditions (e.g., the camera might be rotated/tilted slightly concerning the PIN pad, or the distance to the PIN pad might not be the same). Thus, we also used data augmentation to generate synthetic measurements (20% of the training dataset) that cover more scenarios to account for such issues. In particular, we used the following video-based transformations:

- **rotation** for a maximum of 7 deg both clockwise and counterclockwise;
- **horizontal shift** for a maximum of 10% of the width;
- **vertical shift** for a maximum of 10% of the height;
- **zoom** between 0.9 and 1.1.

Synthetic samples were generated by randomly combining the transformation techniques listed above. We emphasize that data augmentation is also helpful as it makes the predictive model adaptable to different types of ATMs.

3.4 Experimental Results

In this section, we evaluate the performance of our approach for the three attack scenarios described in Section 3.1. We adopted a user-independent split strategy since, in a realistic context, the attacker does not have labeled videos of victims entering PINs. In this way, we guarantee that videos from a participant appear only once among the three sets. Moreover, since we are interested in evaluating the PINs reconstruction accuracy, we removed all non-5-digit sequences entered by mistake by participants (i.e., the "enter" key was pressed after a sequence longer or shorter than 5-digits.) The removed non-5-digits sequences account for 2.2% of the total PINs entered. We conducted the experiments on both 4-digits and 5-digits PINs. To experiment on 4-digit PINs, we removed the last digit of each 5-digit sequence in our dataset.

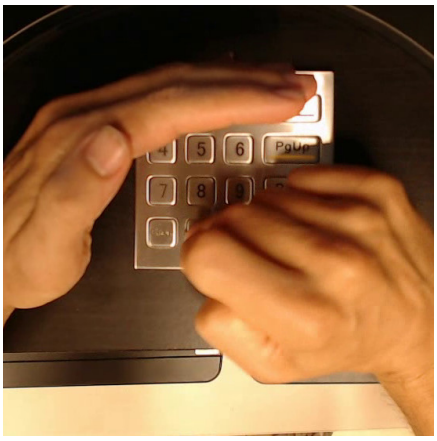
We define that a PIN is covered when there is no direct view of the entered keys and their surrounding. Still, we observed that some participants

failed to obtain a satisfactory coverage level with the non-typing hand despite our instruction before starting the data collection. Since this study aims to infer covered PINs, we decided to exclude the videos of participants that entered badly covered PINs from the validation and test sets. In this way, the validation and test sets consist of videos of covered entered PINs, while the training is composed of videos containing both covered and badly covered PINs. Note that badly covered PINs are still difficult to "read" by simply looking at the video, so we consider such data useful in building a training set. For the test set, we aim for the most difficult scenario where PINs are properly covered. Under these assumptions, we "blacklisted" 16 participants that badly covered the PIN pad: 14 for the first data collection and two for the second data collection. These participants have been excluded from validation and test sets described in the below scenarios [7]. In Figure 3.4, we provide an example of a badly covered PIN and a covered PIN.

To obtain a further indication of the quality of coverage and the difficulty of reconstructing a PIN by a human, we surveyed a random sub-sample of videos of covered PINs (Section 3.6). Finally, there is a question of how to predict the PIN that is not guessed correctly from the first attempt. Since we consider each digit independently, we consider a mechanism where our best guess comprises of individual best guesses (for each digit). If that PIN is incorrect, we consider the digit where the two best guesses have the smallest difference. We change that digit to the second-best guess in our PIN, and we try again. The same procedure is repeated for the third attempt if the second PIN is wrong.

1. **Single PIN pad scenario**. To evaluate the scenario where the adversary knows the target PIN pad model and owns a copy, we considered only the first data collection composed of 40 participants. We applied a user-independent split of the dataset in training, validation, and test sets with the proportions 80/10/10%.
2. **PIN pad independent scenario.** In this scenario, the adversary trains the machine learning model on a PIN pad with a similar layout to the target one. This scenario occurs when the attacker cannot obtain the same PIN pad model to collect data. Under these assumptions, we used for training and validation the first collected dataset (composed of 40 participants). We included the videos from 35 participants in the training set and the remaining 5 participants' videos in the validation set. We used the second collected dataset as the test

[7] Results are in Appendix A.3



(a) *Badly covered PIN that we excluded from the validation and test tests.*

(b) *Covered PIN, where there is no direct view of the pressed key and the surrounding digits.*

Figure 3.4: Badly covered vs. covered PINs.

(a) *True digit = 7*
Pred = 7 (0.999), 4 (0.000), 8 (0.000)

(b) *True digit = 3*
Pred = 3 (0.979), 2 (0.012), 6 (0.005)

(c) *True digit = 6*
Pred = 6 (0.819), 9 (0.170), 8 (0.009)

(d) *True digit = 3*
Pred = 3 (0.809), 2 (0.092), 5 (0.069)

(e) *True digit = 3*
Pred = 2 (0.329), 3 (0.315), 6 (0.185)

Figure 3.5: PIN 73633 entered by a user in our test set in the *Single PIN pad* scenario. Our algorithm suggests 73632 as the most probable PIN (probability = 21.32%), 73633 as the second most probable PIN (probability = 20.43%), and 73636 as the third most probable PIN (probability = 11.96%). The algorithm predicts the correct PIN in the second attempt.

set. We included only the videos of 16 out of 18 participants of the second data collection in the test set since two were in the group that badly covered the PIN pad.

3. **Mixed scenario.** This scenario corresponds to how the attacker owns both a copy of the target PIN pad and a PIN pad similar to the target one. In this case, we merged the two collected datasets and applied a user-independent split in training, validation, and test sets with the proportions 80/10/10%.

We begin the discussion on results by providing an example of a successful PIN attack in Figure 3.5. We consider the 5-digit PIN case and the *Single PIN pad* scenario. We provide an image for each digit. We give the top three digits and the corresponding accuracy values. Notice how the first and second digits are predicted correctly with high probabilities. This happens as the person sets the hand to allow an easy start of typing. Already for the third digit, we observe a significant drop in the accuracy value for the best prediction. Still, the value is significantly larger than the second-best prediction, so there are no issues in getting the correct prediction. This trend continues for the fourth digit and gets very pronounced for the last (fifth) digit. Indeed, the best guess is not correct anymore, but the second-best guess is correct (the difference in probability between those two guesses equals 0.014).

For all three scenarios, Figure 3.6 shows the results for the single key accuracy, while Figure 3.7 reports the results considering 5-digit and 4-digit PINs. Considering the single key accuracy (averaged over all digits), notice that even in the most difficult *PIN pad independent* scenario, our Top-3 accuracy reaches 63.8%, which is significantly higher than the result one would reach with random guessing (30%). At the same time, the results for the *Single PIN pad* scenario and the *Mixed* scenario are rather similar, and the Top-3 accuracy reaches up to 88.7%. Interestingly, we observe somewhat better results for Top-2 and Top-3 accuracy for *Single PIN pad* scenario than the *Mixed* scenario, which is the opposite of the results for 4-digit and 5-digit settings. We hypothesize this happens as we consider independent digits as naturally, the best results happen when the training and test are done on the same device. On the other hand, the *Mixed* scenario gives slightly better results for the PIN reconstruction scenarios as we need to consider a sequence of PINs with the movement between digits. Then, having different devices in the training set allows (slightly) better generalization.

In Figure 3.7a, we observe that the most difficult case is when the attacker does not have access to the same keypad as used by the victim. There, the accuracy for the Top-3 case equals 11.4%. Having access to the same

type of keypad improves accuracy in Top-3 to more than 20%. Finally, considering the *Mixed* scenario, we can improve the accuracy for Top-3 to almost 30% (29.7%). Next, in Figure 3.7b, we present results for 4-digit PINs. The results are significantly better than for the 5-digit scenario. The lowest accuracy happens for the Top-1 *PIN pad independent* scenario setting and it equals 10.6% (cf. 6.7% for the 5-digit scenario). The highest accuracy reaches 41.1% for the Top-3 accuracy in the *Mixed* scenario.

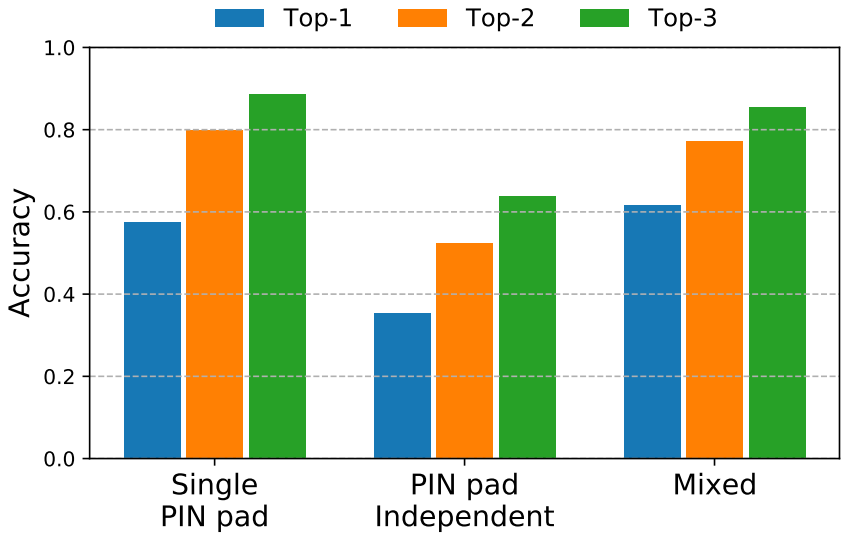


Figure 3.6: Single key accuracy of our algorithm for the three considered attack scenarios. Top-N means that we guessed the digit within the N attempts.

In Figure 3.8, we depict detailed results for the digit 1. We selected this digit since heat maps for others look similar and exhibit similar dispersion. First, in Figure 3.8a, we show the PIN pad layout. Figure 3.8b gives results for the *Single PIN pad* scenario. Notice that the heat map indicates that guess 1 is the most likely one with 67% probability. The digits 4 and 3 are recognized as the second and third best guess, respectively. Still, their probability is significantly lower. For the *PIN pad independent* scenario, we observe that the probabilities are more spread over all digits, which comes at the expense of a lower prediction probability for the correct digit. The second and third best guesses maintain the probabilities, indicating that Top-3 guesses are sufficient to guess a large number of PINs in the most difficult scenario. Finally, Figure 3.8d gives results for the *Mixed* scenario, where we see that the best guess is on the level with the *Single PIN pad* scenario. Interestingly, now the second and third best guesses are swapped

Attacker Information Source				4-digit PINs TOP-N Accuracy (%)		
KT	**OD**	**TT**	**Our Attack**	**TOP-1**	**TOP-2**	**TOP-3**
				0.01	0.02	0.03
	§			0.10	0.20	0.30
§				0.02	0.35	0.72
§	§			3.02	3.72	4.36
		§		3.76	7.52	11.28
§		§		15.54	27.79	33.63
			§	**29.61**	**37.06**	**41.12**

Table 3.1: Comparison of our attack with other unobtrusive attacks on ATM PIN pads. Note that we need to extract the frame for our attack, while for KT, one needs to use the timestamp, which is more precise information.

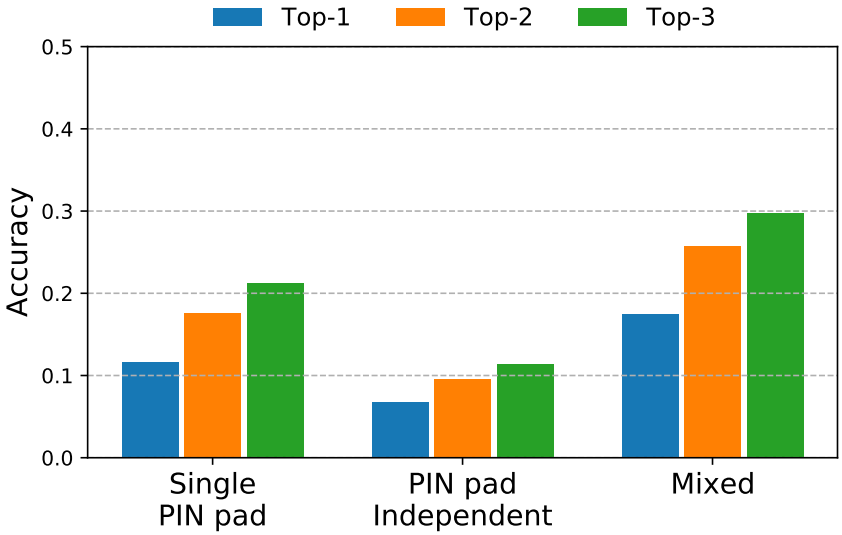
compared to the previous scenarios. All the other digits have 0 or negligible probability of being the correct digit. Appendix A.2 provides additional results for the key accuracy.

Based on our results, we provide several observations that we believe generalize beyond these experiments:

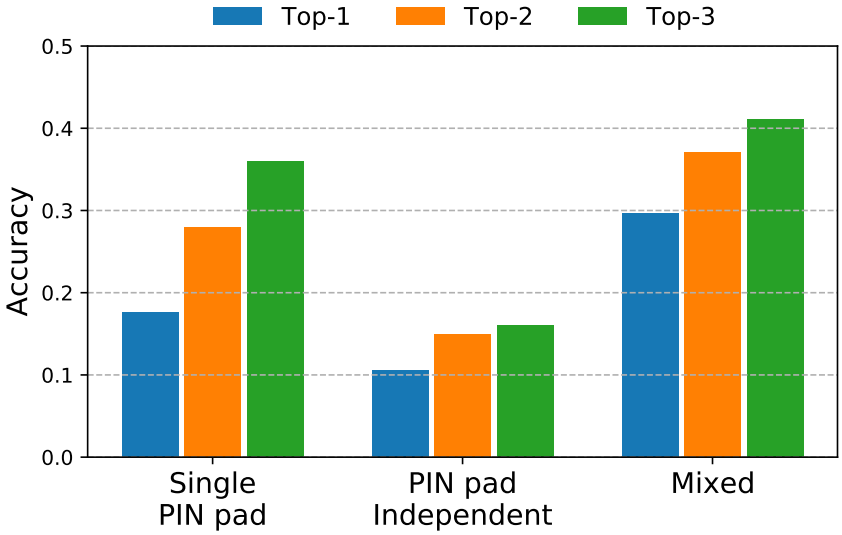
- Covering the PIN pad with the other hand is not sufficient to defend against deep learning-based attacks.
- Portability aspect (keypad differences) is quite significant, and the attacker should obtain the same type of keypad for a high probability of success in attack.
- There are three prevailing ways how users cover the typing hand: raised hand not touching the surface, hand resting on fingers and vertically covering the PIN pad, and hand resting on the side of the palm. The examples of all three covering strategies are shown in Figure 3.9.

Finally, Table 3.1 provides a comparison between our attack and several unobtrusive attacks on 4-digit PINs from the literature [36]. We divided the attacks according to the information that the attacker has: keystroke timing (KT), one digit of the victim's PIN (OD), and the thermal trace (TT) left on the PIN pad by the victim [3]. From the results, it is clear that our attack performs the best for all considered TOP-N accuracies.

Appendix A.3 provides experiments where we: i) resize the images, ii) consider different camera positions, iii) consider setup without data augmentation, and iv) consider the training set that includes the blacklisted participants. Finally, we also provide experiments for the frame detection error (when the feedback sound is not properly synchronized).



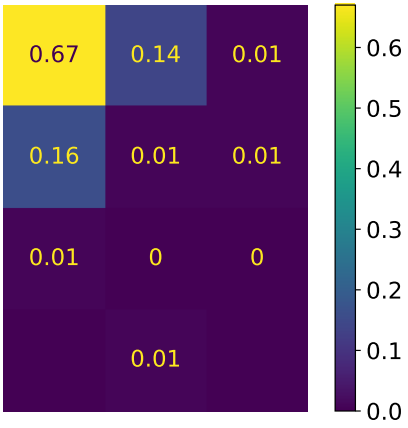
(a) *5-digit PINs.*



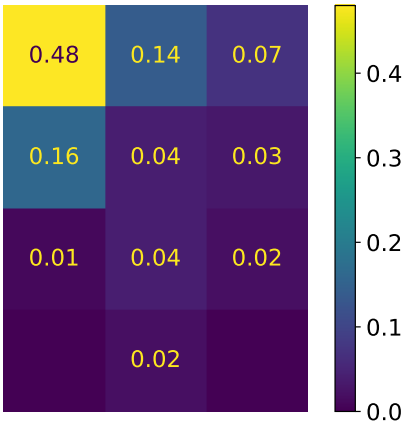
(b) *4-digit PINs.*

Figure 3.7: PIN accuracy of our algorithm in the three considered attack scenarios. Top-N means that we guessed the PIN within the N attempts.

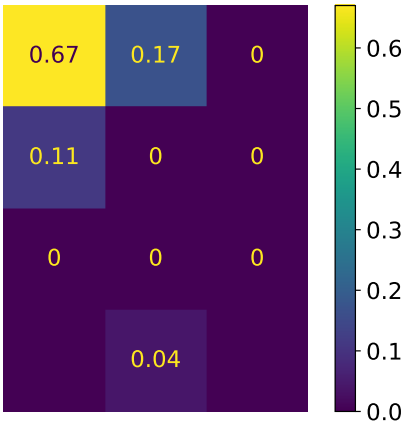
(a) *Layout of a generic PIN pad.*



(b) *Single PIN pad scenario.*



(c) *PIN pad independent scenario.*



(d) *Mixed scenario.*

Figure 3.8: Digit 1 predictions heat maps for the three considered attack scenarios.

(a) *Side: hand resting on the side of the palm.*

(b) *Over: raised hand not touching the surface.*

(c) *Top: hand resting on fingers and vertically covering the PIN pad.*

Figure 3.9: Different covering strategies using the non-typing hand.

3.5 Countermeasures

Different countermeasures could make the attack more difficult to succeed. For instance:

1. Longer PINs. This countermeasure would make the attack more difficult, as evident from the comparison for 4- and 5-digits PINs. This countermeasure would be relatively easy to support from a technical perspective. At the same time, it would have usability drawbacks as longer PINs take more time to type and are more difficult to remember.
2. Virtual and randomized keypad. Instead of using a mechanical keypad, one could consider using a touchscreen where the digits are randomized. More and more ATMs (but not PoS) have this feature, so implementing it would not be too difficult. Unfortunately, we believe this would seriously damage the usability aspect as people are accustomed to digits occurring in the natural sequence, and any changes would probably result in wrongly entered PINs.
3. Screen protectors. On many ATMs, there are already various types of screen protectors that occlude the typing hand. To maintain usability, many screen protectors are short and will not cover the whole typing hand. Making the screen protectors larger would impair usability as it will become more difficult for the user to read the keypad. This countermeasure is potentially not easy to deploy as it requires physical changes to the ATMs.

Next, we analyze how a PIN shield could affect the performance of our attack. We simulated the presence of the shield by applying a black patch to cover the PIN pad. In Table 3.2 we report the performance of our attack

Coverage percentage	Key accuracy	PIN TOP-3 accuracy
25%	0.54	0.22
50%	0.55	0.22
75%	0.50	0.17
100%	0.33	0.01

Table 3.2: PIN shield experiments.

in the *Mixed* scenario, applying four different levels of coverage (Figure A.5, Appendix A.3). The coverage of the PIN pad is larger than the percentage shown in Figure A.5 since the coverage given by the non-typing hand is not included in the given percentage. The results show that our attack remains effective even when 75% of the PIN pad is covered, while the performance decays significantly beyond this level of coverage. As such, it becomes clear that our deep learning attack uses information about the whole hand position and movement, and not only the tip of the fingers. Since the last row of the PIN pad has only one number (0), 100% coverage has poor attack results not only because of hiding all the numbers on the keypad but due to hiding of proximal interphalangeal, metacarpophalangeal, and carpometacarpal joints of the fingers. Thus, only PIN shields that offer full PIN pad coverage can be considered effective countermeasures to our attack.

We provide additional results with different covering strategies (`Side`, `Over`, and `Top`) in Appendix A.3. Those results again show that covering the PIN pad from the `Side` gives insufficient protection. On the other hand, using the `Over` strategy significantly decreases the key accuracy and PIN accuracy.

3.6 Deep Learning vs Humans

If an attacker has direct visibility of the PIN pad, reconstructing a PIN from a video can be considered a trivial task. One of the classic countermeasures to the so-called shoulder-surfing attacks is to cover the hand entering the PIN with the non-typing hand. In this way, the victim obstructs the attacker by removing the direct visibility of the keypad. We designed a questionnaire to evaluate how much the covering with the non-typing hand effectively prevents the PIN reconstruction.

3.6.1 Methodology

The questionnaire consists of 30 videos of people entering 5-digit PINs by covering the PIN pad with the non-typing hand as we noticed that for longer questionnaires, the participants' attention significantly goes down toward the end. For each video, the participants had to indicate the three most likely PINs in their opinion.

To assess human and model performance on both the PIN pads, we decided to use the test set of the *Mixed* scenario (i.e., the only one including both PIN pads). Since the test set was balanced in terms of samples per user, we randomly selected five PINs for each of the six users in the test set. We extracted 30 videos corresponding to the selected PINs from our dataset. We kept the original resolution of 720p and the original audio track containing the feedback sound emitted by the PIN pad for each video. The feedback sound helps the participants to recognize when a digit is entered. To avoid bias in the answers, we randomized the order of the videos in the questionnaire. Moreover, the participants were free to modify all their answers until the final submission. We did not apply any particular restriction to the participants during the filling of the questionnaire. In particular, there were no time restrictions to complete the task. The participants could freely apply the strategy they prefer to infer the PIN (e.g., write down the digits, pausing the video, restart the video any number of times, use the slow-motion option). Finally, we provided the users with the layout of the PIN pad.

To evaluate if people with specific knowledge about the task achieve a better performance, we pre-trained a group of participants. Specifically, we provided participants with a new set of 20 videos of users typing PINs by covering the PIN pad with the non-typing hand and the corresponding typed PIN. To make the training more effective, we decided to provide participants with videos of users included in the questionnaire (none of the videos are present in both training and questionnaire). Additionally, the questionnaire had suggestions on what to pay special attention. For a participant to be considered trained, the complete viewing of all 20 videos is required. In addition, trained participants could also watch the training videos while filling the questionnaire.

3.6.2 Evaluation and Discussion

A total of 78 distinct participants took part in our questionnaire experiment. In particular, 45 participants (14 female age 34.1 ± 10.4 years and 31 male age 29.7 ± 8.3 years) completed the experiment without any training, while

33 participants (10 female age 29.1 ± 3.3 and 23 male age 29.3 ± 5.6) completed the experiment after the training session. None of the questionnaire participants took part in the two data collections described in Section 3.3.2.

The proposed questionnaire's goal is twofold: i) investigate how effective the hand coverage is in preventing a PIN from being inferred by a human, and ii) compare the performance of our deep learning approach with that of a human. Although the coverage of the PIN pad provides an obstacle to the immediate identification of the typed PIN, a human can exploit various information (both local and global) to reduce the probability space about where to look for the entered PIN:

- Knowing the keys' spatial positioning thanks to the given layout of the target PIN pad.
- Understanding which finger pressed the key from the movements of the hand.
- Evaluating the topological distance between two consecutive keys from the feedback sound emitted by the PIN pad. Specifically, two topologically close keys have temporally close sound feedback [36].
- Excluding keys based on the non-typing hand coverage.
- Guessing the finger position based on the hand displacement between the insertion of a key and the next one.
- Deducing the fingers' position of the covered hand.

Although a human can exploit this information, the PIN pad coverage still partially prevents PIN reconstruction. In particular, the participants in our questionnaire could reconstruct on average (of both trained and non-trained humans) only 4.49% of the PINs entered in the videos on the first attempt and 7.92% within three attempts. The performance increasing between Top-1 and Top-3 accuracy suggests a certain ability in estimating the neighborhood of the keys pressed. This ability is also highlighted in Figure 3.11a, where the probability distribution shows how the error decreases with the increase of the topological distance from the target key. The heat maps for other keys look similar and exhibit similar dispersion.

Unlike humans, our algorithm focuses on target key classification and then reconstructs the entire PIN sequence. To compare the model's performance to that of humans on the same task, we evaluated our algorithm's accuracy on the videos included in the questionnaire. Recall that the questionnaire's videos are a sub-sample of the *Mixed* scenario test set, and therefore were not used in the model training phase. As reported in Figure 3.10, our model performs better than humans in all Top-N accuracy scenarios. To evaluate if our algorithm performance and humans' performance in reconstructing 5-digit PINs are statistically different, we applied

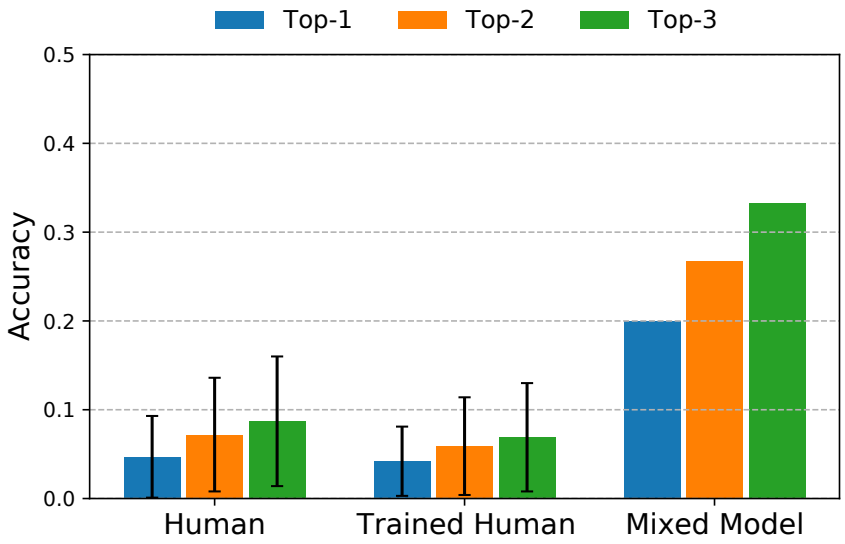
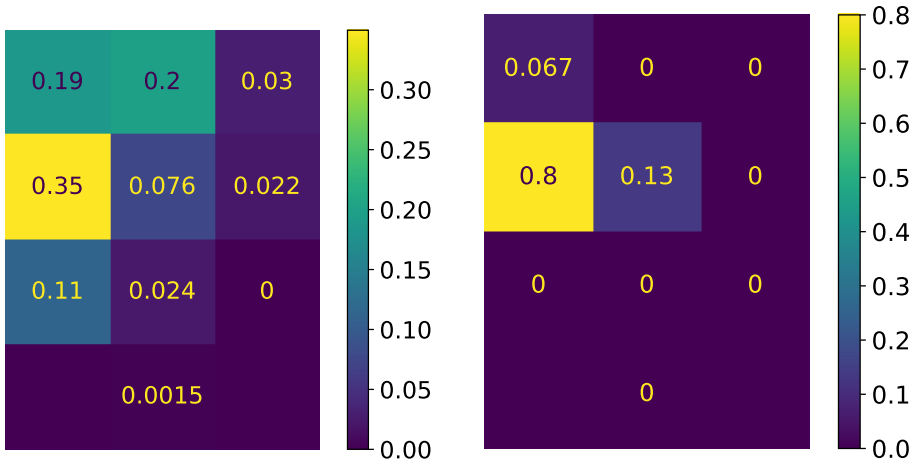


Figure 3.10: Comparison between human (non-trained and trained) and deep learning model performance in the sub-set of videos included in the questionnaire. Top-N means that participants guessed the PIN within the N attempts.

a series of Chi-square tests [105]. The Chi-square test resulted significant for all Top-1 ($\chi^2 = 14.19, p < 0.001$), Top-2 ($\chi^2 = 15.84, p < 0.001$), and Top-3 ($\chi^2 = 21.37, p < 0.001$) accuracy values for non-trained humans. In particular, our model outperforms humans showing a four-fold improvement in reconstructing a PIN in three attempts. Similarly, for trained humans, the Chi-square test resulted significant for all Top-1 ($\chi^2 = 16.12, p < 0.001$), Top-2 ($\chi^2 = 20.83, p < 0.001$), and Top-3 ($\chi^2 = 28.88, p < 0.001$) accuracy values.

This result comes from the difference in performance in the classification of single keys. The human average accuracy (considering both human data collections) on single key classification equals 0.351, approximately half compared to the model key accuracy of 0.687. The comparison of Figures 3.11b and 3.11a shows how the error in identifying a digit is significantly higher for humans, justifying why the increase in Top-2 and Top-3 PIN accuracy is greater for our algorithm. Finally, comparing trained and non-trained humans, the Chi-square test reported no significant differences with $p > 0.1$ for all Top-1, Top-2, and Top-3 accuracy values. This means that training does not improve a human's ability to identify a PIN within three attempts. Potentially, either a longer training could be required, or additional feedback from an expert should be provided to improve the performance. Ap-

pendix A.2 provides additional results for the comparison between our deep learning model and human performance.



(a) *Humans.* (b) *Mixed* scenario model.

Figure 3.11: Digit 4 predictions heat maps for the videos included in the questionnaire. We report an example from non-trained humans, since the heat maps for both non-trained and trained human are similar.

3.7 Related work

Side-channel attacks specifically target the information gained by the implementation of a system [101]. Most of the time, these attacks exploit channels like sound [64], timing [88], power consumption [87], and electromagnetic emanations [25] to learn the system's secrets in use. In [88], the authors managed to crack RSA keys by carefully timing the operations performed by the key-generating algorithm. Another example of a timing attack is reported in [125], where the authors measured the timing between keystrokes in interactive SSH sessions in an attempt to retrieve the typed passwords.

Human behavior can also be defined as a side-channel of a system, especially if the analyzed behavior directly results from the system's requirements. In [17], the authors analyzed the hand movements of people typing on a keyboard and, by using basic computer vision techniques, they tried to reconstruct the text being typed. In [124], the authors again analyzed the finger motion during the PIN-entry process on smartphones. They showed that 50% of the 4-digit PINs could be retrieved in just one attempt. Different from our work, where the target of the attack is a physical PIN pad, in [124],

the attackers could also exploit more information. In particular, the users typed the PIN using only one finger, and the attacker knew the finger the users are typing. The different contexts and assumptions make the works substantially different. In [128], the authors presented a side-channel attack on tablets, consisting of analyzing the backside movements of the tablet itself to infer what is being typed by the victim. To do so, they selected some peculiar features of the backside of the tablets (e.g., logos, side-buttons) and analyzed their movement throughout the frames to understand what area of the virtual keyboard is being pressed. Similarly, in [143], the authors presented an attack to infer the pattern lock of mobile devices from videos. Different from our approach, in [143], the attacker required a vision of the user's fingertip while drawing the pattern and a part of the device.

PIN and PIN pad attacks represent a branch of side-channel attacks that exploit information leakage from keyboards and numeric keyboards (i.e., PIN pads) to infer what the victim has typed (e.g., passwords or PINs). In this context, some works focused on exploiting the heat transferred from the hand to the keypad when the victim enters the PIN or password [84, 108]. The attacker points a thermal camera to the keypad as soon as the victim has finished entering the PIN. The thermal image shows which keys have been pressed and even highlights the order in which the victim pressed them. The main advantage of this attack is that it does not require the attacker to do anything while the victim is typing the PIN. On the other hand, the attacker must act quickly (i.e., within seconds) for a higher success rate as the heat on the keypad rapidly fades away. Another drawback of the attack is that its effectiveness depends on the keypad's material (e.g., metal PIN pads completely nullify the attack because of their high thermal conductivity).

Timing attacks against PINs represent another type of side-channel attack against this authentication method. In the scenario presented in [14], the attacker recorded the screen of an ATM while the victim is entering the PIN. When analyzing the recorded video, the attacker exploited the PIN masking symbols appearing on the ATM screen to extract timing information about the keystrokes. The attacker used predictive models to infer which keys were most likely typed by the victim, starting from the deduced inter-keystroke timing. In [36], the authors used the ATM's sound whenever a button is pressed. ATM's sound must be independent of which button is being pressed (i.e., a generic feedback sound). This consideration means that one feedback sound will not help the attacker. However, the sound gives enough information to extract a timestamp of the keys being pressed. Moreover, in [36, 97], the authors showed how combining timing, acoustic, and thermal information can significantly reduce the number of attempts to

guess a PIN (e.g., 34% of 4-digits PINs are recovered in three attempts). These attacks need to be reevaluated from a feasibility perspective in a real-world setting. In particular, as shown in [84], the heat signature is dissipated abruptly by metal PIN pads. The lack of this information limits the performance of the attacks presented in [36, 97], reducing the probability of guessing a 4-digit PIN in 3 attempts to 5%.

Our work shows several advantages over the state-of-the-art in ATM PIN inference. To the best of our knowledge, we are the first to investigate the security of hand covering protection methods for ATM's PIN entering. Further, our method shows a significant improvement in reconstructing the PIN compared to previous work on metal PIN pads, reaching 41% of success in reconstructing 4-digit PINs in three attempts (and correctly guessing every third PIN in the first guess).

3.8 Summary

This paper proposed a deep learning attack on PIN mechanisms reaching high accuracy even when the user covers the PIN to be entered. Our attack leverages the information from the hand position but also hand movements while entering the PIN. Our attack works in the profiling setup where the attacker uses a copy of the keypad to train the deep learning model and then attacks a different device while the victim is entering the PIN. For a 4-digit PIN, our attack reaches an accuracy of more than 40%, making it practically applicable and more powerful than the attacks from the related works.

Our data collection phase involved 58 persons, and our questionnaire involved 78 participants. While this required a significant effort and several months of data acquisition, one could still consider the datasets too small to allow general conclusions. Next, our analysis considered only two types of keypads. While most keypads do not have significant differences, including more keypad models in our analysis would be interesting. Additionally, there are several potential sources of bias in our data collection phase. While we managed to get a relatively good male and female participants ratio, we notice that data is skewed from several perspectives. Unfortunately, this was not possible to avoid as the participation was voluntary [8].

1. Our dataset has users ranging from 24 to 50 years. While this provides good variety, it would be good if it included older people. Still, we do

[8]The 2021 COVID-19 situation made data acquisition more challenging as participants needed to be in our lab during the data acquisition.

not expect any difficulties in running our attack. We consider it even somewhat easier as we noticed older people make more significant hand position adjustments when entering the PIN.

2. Our analysis includes only right-handed persons. We do not expect any issues due to the dataset's limitations as we use a camera positioned in the center. Still, we expect the attack to be more difficult when attacking left-handed persons if the training set does not contain such examples. Finally, from the real-world practicality, there are approximately 90% of right-handed persons vs. 10% left-handed persons [115], so our attack generalizes for the dominant part of the population.
3. All participants were Caucasians. We expect our attack will have difficulties working for people from other races. Still, this can be alleviated by expanding the training set to include more racial diversity.

Possible future work includes:

1. In our data collection phase, we allowed the users to select their covering strategies. Based on the current results, it would be interesting to explore if modifications in how the user covers the PIN would allow more protection.
2. We noted several potential sources of bias in our data collection phase. Including participants from other races and left-handed persons would allow us to make more general conclusions.
3. To avoid the need that the attacker should have different keypads, it would be beneficial to assess whether some more straightforward solution like a paper copy of the keypad would suffice (at the expense of losing information about the keypress sensitivity).
4. It would be interesting to investigate if it is possible to extract the timestamp directly from the video (when a person clicks a button, there is a specific movement).

Chapter 4

We Can Hear Your PIN Drop: An Acoustic Side-Channel Attack on ATM PIN Pads

The Automatic Teller Machines Industry Association estimates that over 300 million ATMs are deployed worldwide [12]. In the US alone, over 10 billion ATM transactions are performed every year [111]. ATMs have now become an indispensable part of the self-service banking ecosystem. An ATM typically uses a unique physical card (which a customer possesses) along with a PIN (which a customer remembers) to form a two-factor authentication system, wherein the card uniquely identifies the customer account and the PIN identifies the customer.

In recent years, there have been many attacks aimed at PINs and at information encoded on ATM cards. Such attacks are broadly referred to as skimming operations [132], whereby criminals usually install a card-reader-like device to trick customers into placing (or inserting) their cards and copy the information. This is often done in tandem with installing a video camera on the ATM (or in its vicinity) at an angle that allows the criminal to record PIN entry [122]. Recently studied attacks on PINs (e.g., [14, 36, 140]) went one step further and showed that the attacker does not even have to see the PIN. These side-channel attacks use a recording device (e.g., a video camera [14], a microphone [36], or a thermal camera [140]) placed near the ATM to collect information and use it to infer customers' PINs.

In this chapter, we present a new side-channel $\mathcal{P}inDrop$ attack on ATM PIN entry. It consists of two steps: (1) the attacker builds an acoustic profile

(a signature of click sounds) for each key on the target PIN pad, and (2) at PIN entry time, the attacker records audio emitted by each pressed key and compares them to the acoustic profile to infer the actual keys pressed, thereby learning the PIN. These two steps can be carried out in any order.

Intended Contributions The main contributions of this work are:

1. We describe a novel attack targeting PINs: $\mathcal{P}inDrop$, based on acoustic emanations from commodity ATM PIN pads. We demonstrate that $\mathcal{P}inDrop$ reconstructs up to 94% of 5-digit PINs and 96% of 4-digit PINs within three attempts. We show that the threat posed by $\mathcal{P}inDrop$ is higher compared to state-of-the-art acoustic side-channel attacks on ATM PIN pads [36, 97, 114].

2. We evaluate $\mathcal{P}inDrop$ via extensive experiments, collecting acoustic emanations for 5,800 5-digit PINs entered in a simulated ATM (though using real PIN pads) by 58 distinct participants. The resulting dataset is publicly available[1] to the research community. We believe it will be useful in studying the problem further and developing countermeasures.

3. We analyze the performance of $\mathcal{P}inDrop$ with two recording distances: 0.3 and 2 meters away from the PIN pad. At the distances of 0.3 and 2 meters, up to 96% and 57% (respectively) of 4-digit PINs were correctly learned in three attempts.

4. We demonstrate the feasibility of $\mathcal{P}inDrop$ on two commercially available ATM PIN pad models. The success rate of PIN guessing on both pads is about the same for each distance.

5. We analyze the impact of training set size on the performance of $\mathcal{P}inDrop$. We evaluated two important factors: the number of attackers participating in the Profiling Step, and the number of digits collected by each attacker. We showed that including training samples from multiple attackers is an effective strategy for appreciably improving attack success rate.

Organization: Section 4.1 discusses related work. ext, Section 4.2 describes $\mathcal{P}inDrop$ attack in detail. Then, Section 4.3 presents the design of our experiments and Section 4.4 discusses the results.

[1] Dataset link: https://spritz.math.unipd.it/projects/PINDrop

4.1 Related Work

This section overviews attacks based on acoustic emanations from user input devices. We first consider attacks targeting keyboards, followed by those targeting PIN pads. For a comprehensive discussion of keyboard side-channel attacks, we refer to [107].

Attacks on generic keyboards. The first extensive study on keyboard acoustic eavesdropping was conducted by Asonov and Agrawal [11]. It showed that each key can be identified by the unique sound that it emits when pressed. This work investigated the reasons for this behavior, demonstrating that it can be attributed to the placement of keys on the keyboard plastic plate. In particular, when different keys are pressed, the plate produces emits sounds with different timbers.

Subsequent efforts to infer key sequences from acoustic emanations are based on two types of approaches: (i) extraction of features that allow exploiting the uniqueness of acoustic emissions of pressed keys, and (ii) extraction of temporal information. The former tries to distinguish among keys by their characteristic sound, and relies on either supervised [11, 68, 70, 104] and unsupervised [22, 150] machine learning models, depending on the specific attack scenario. Supervised models exploit features, notably Fast Fourier Transform (FFT) coefficients and their derivatives, such as Mel-frequency cepstral coefficients (MFCCs). Supervised algorithms generally achieve better performance in identifying keystrokes. On the other hand, these models have a greater dependence on the keyboard used in training and the users' typing style. A further weakness of supervised algorithms is the need to collect a labeled dataset to be used as a training set. Indeed, the ground truth collection is not a trivial task and could significantly affect the attack's effectiveness. One possible solution is discussed in [8, 39]. which take advantage of the audio recorded during a VoIP call to collect a ground truth dataset directly. In this scenario, the attacker can exploit the text typed by the victim in a shared medium (e.g., in the VoIP chat or an email sent to the attacker during the call) to label the keystroke sound.

Unsupervised methods are used to group collected samples into unlabeled clusters. The label-cluster association is made by exploiting the characteristics of the input language. In particular, Zhuang et al. [150] perform labeling using letter frequency, while Berger et al. [22] make an association by selecting words from a dictionary that match specific constraints. Unsupervised approaches overcome the need for a ground-truth dataset. However, the scenarios where these attacks can be applied are limited by the strong as-

sumptions on input text and therefore their performance drastically declines on random letter sequences.

The second approach involves the extraction of temporal features of pressed keystrokes. To this end, many efforts focused on analyzing the Time Difference of Arrival (TDoA) information. They use one [96] or more [148] microphones positioned around the input device.

Pin Pad-focused Attacks. PIN pads are numeric keypads specifically designed for Point-of-Sale (PoS) terminals and ATMs, They facilitate users to enter their Personal Identification Numbers (PINs). Attacks on PIN pads tend to be different from those on regular keyboards. For instance, it is rather challenging to apply unsupervised techniques with PIN pads since the assumptions about the victim's language are no longer applicable. However, the other types of attacks, such as those based on the uniqueness of the acoustic emission and those based on the temporal information are usually applicable. PIN pads also prompt a new set of assumptions, usually dictated by the specific conditions under which they operate. This paves the way to new and more efficient side-channel attack scenarios. Below, we briefly discuss these attacks.

[14] demonstrates how to obtain PIN information by exploiting inter-keystroke timings. This information is leaked by recording the timing of appearance of masking symbols (usually, asterisks) on the screen while the victim is entering the PIN. On a related note, [36], shows how inter-keystroke timing information can be inferred with higher accuracy from the feedback sound emitted by the PIN pad when a key is pressed. It also shows that combining multiple side-channel information (e.g., inter-keystroke timing and thermal residue) can significantly improve the probability of reconstructing a 4-digit PIN. Similarly, [97], proposes an user-independent attack based on inter-keystroke timing on a plastic PIN pad.

PIN pad acoustic emanations can also be used to improve security of PIN-based authentication systems. For example, [114] shows that inter-keystroke features obtained from PIN pad-emitted audio, can be used as an additional layer of authentication. The same work also showed how to perform a close-by attack (i.e., with the microphone placed a few centimeters from the PIN pad) on an arbitrary subset of keys. Exploiting the inter-keystroke features on this subset, a 60% accuracy in the identification of the pressed key can be reached. Acoustic information is also used in [47], where a Point-of-Sale (PoS) terminal is tampered with by inserting multiple microphones into it. This allows identifying the pressed key position using triangulation, reaching the average accuracy of 88% for a single key, on three

PoS models. Although very effective, this approach requires full physical access to the PoS, thus reducing the attack's applicability and scalability.

4.2 $\mathcal{P}inDrop$ Attack

Assumptions: We assume that the victim interacts with a generic ATM, performing PIN-based authentication. The ATM is equipped with a PIN pad that emits a feedback sound when a key is pressed. The feedback sound (as perceived by the human ATM users) is the same for all keys. The attacker aims to learn the victim's PIN by placing a microphone near the ATM to record acoustic emanations of the PIN pad. The microphone stores recorded audio. How the microphone stores that audio is not relevant for $\mathcal{P}inDrop$, i.e., it can be stored locally or off-loaded to a remote site. $\mathcal{P}inDrop$ attack relies only on that recorded audio.

Preliminaries: To set up $\mathcal{P}inDrop$, the attacker must select a target ATM and hide a microphone nearby. The exact placement of the microphone can vary, though in the $\mathcal{P}inDrop$ setting the maximum distance form the PIN pad is 2 meters (just over 6′):

1. Concealed on the attacker's body, in case of a real-time attack. Albeit, strictly speaking, concealment is not required, since a regular smartphone microphone can be used, and it need not be hidden from view (as it is unlikely to arouse suspicion).
2. On any surface (walls, floor, ceiling) near the ATM. In this case, it might be in plain sight, especially, if its size and shape are inconspicuous enough not to be noticeable. It could also be partially hidden from view (e.g., behind a column or a light fixture), or even within or behind some normal-looking object, e.g., a vent, a light-switch or a garbage can.

As shown in Figure 4.1, $\mathcal{P}inDrop$ consists of four phases: 1) PIN Recording (Section 4.2.1), 2) Data Processing (Section 4.2.2), 3) Model Generation (Section 4.2.3), and 4) PIN Inference (Section 4.2.4),

4.2.1 PIN Recording

The goal of this this phase is to come up with two datasets (training and attack) with audio recordings of entered PINs. This takes two steps:

A.1 **Audio Recording** using a microphone placed near the ATM.

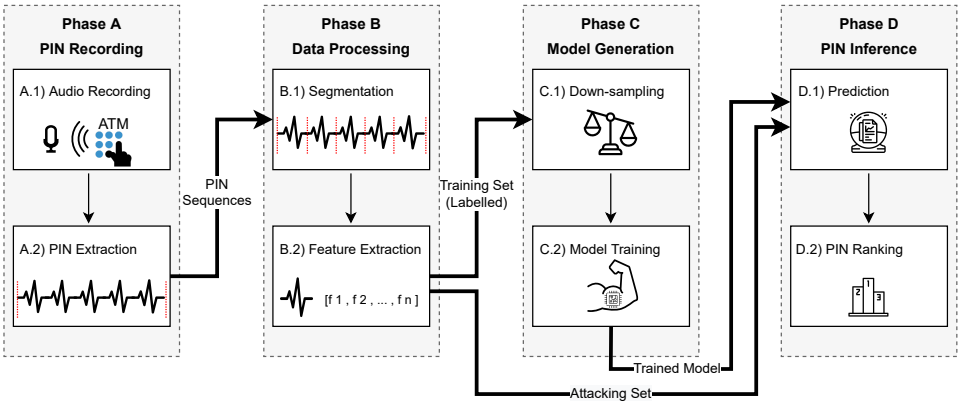


Figure 4.1: *PinDrop* attack phases.

A.2 **PIN Extraction**, i.e., isolation of the sequences of feedback sounds emitted by the PIN pad, given the knowledge of the number of digits in the PIN, e.g., the beginning and the end of the 6-digit PIN entry.

To build the *training set*, the attacker must enter a set of PIN sequences on the target PIN pad. The sequences must be representative of all ten numeric keys. Once this step is completed, the attacker has a table of entered PINs and their corresponding audio. The *attack set* consists of the audio recordings entered by the victim.

4.2.2 Data Processing

This phase is conducted on the data entered by both the attacker and the victim. It also consists of two steps: segmentation of the PIN audio signal into individual key-press sounds, and extraction of corresponding features.

B.1 **Segmentation:** The attacker uses the feedback sound emitted by the PIN pad as a signal that a key has been pressed. This can be achieved via the characteristic frequency of the feedback sound, as in [36]. The attacker segments the signal, using time windows centered at the detected key-press. The window size is chosen to comprise the entire audio segment related to a single key-press.

B.2 **Feature Extraction:** The attacker extracts features descriptive of a key-press sound. Prior results show that short-term power spectrum can be used for this type of a classification problem. In particular, [39] shows that mel-frequency cepstral coefficients (MFCC) [99] achieve the best performances for discriminating among the sounds of different keys. This step yields two feature sets: (1) a labeled training, and an (2) unlabeled attacker.

4.2.3 Model Generation

This phase is applied to the labeled training set in order to train a classifier.

C.1 **Down-sampling:** Since we make no assumptions about how often a victim uses a specific digit in the PIN, it may be necessary to undersample the data by classes before proceeding with training. The undersampling leads to a balanced dataset and mitigates over-fitting.

C.2 **Model Training.** The attacker trains a multi-class classifier to predict the digit based on its emitted key-press sound. The class labels output by the classifier are the keys (digits) of the PIN pad. Together with the predicted digit, classifiers also output the prediction probability of each class.

4.2.4 PIN Inference

In this phase, the attacker utilizes the trained classifier to guess a victim's PIN. The output is a sequence of all possible PINs ordered by probability. This ordering allows the attacker to minimize the number of attempts to guess the PIN. In a real-life setting, ATM cards are usually blocked after three failed attempts. This phase involves two steps:

D.1 **Prediction:** The attacker reconstructs the PIN entered by the victim applying the classifier trained in the previous phase to the attack set. As input to the classifier, the attacker feeds the features of a single key of the victim's PIN. This is repeated for each digit of the PIN.

D.2 **PIN Ranking.** The classifier yields a probability for each digit to be the one actually pressed by the victim. Combining the probability set of each input, the attacker builds a ranking of the most likely PINs. The probability assigned to a PIN is the product of the probability of each digit in that PIN.

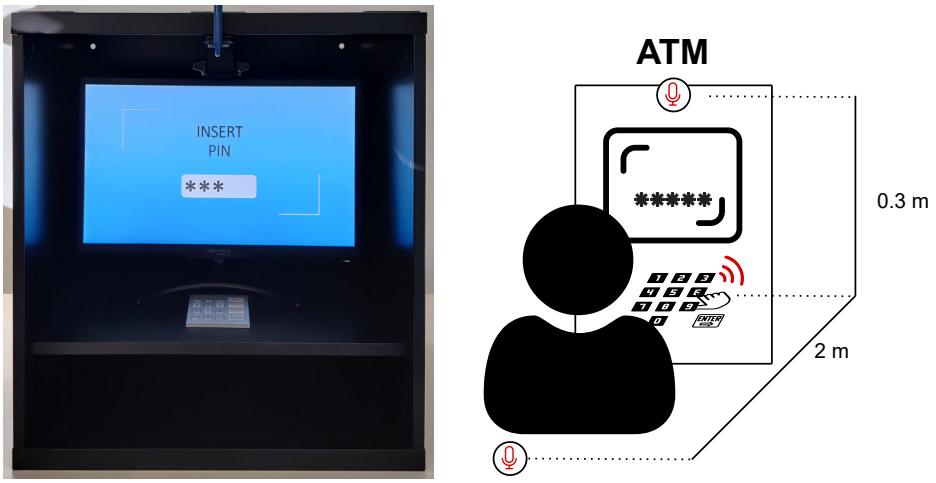
4.3 Experimental Setting

To assess the feasibility of $\mathcal{P}inDrop$, we collected a large dataset of keystroke sounds, as detailed in this section.

4.3.1 Data collection

We performed two separate data collection efforts on two commercially available (commodity) metal PIN pads: DAVO LIN Model *D-8201 F* (Fig-

ure 4.3a) [2] and Model *D-8203 B*(Figure 4.3b) [3]. For clarity's sake, we refer to *D-8201 F* as *PAD-1* and *D-8203 B* as *PAD-2*. For usability reasons, both pads emit a specific feedback sound (the same for all keys) when any key is pressed. In all experiments, we embedded each PIN pad into a simulated ATM (Figure 4.2a).



(a) *The simulated ATM.*

(b) *The testbed configuration used in the experiments.*

Figure 4.2: $\mathcal{P}inDrop$ experimental setup: The photo on the left shows the ATM layout. The figure on the right show the position of the microphone with respect to the ATM and the victim: the closer one at $0.3m$ is placed over the PIN pad, while the farther one at $2m$ is placed in front of the ATM, and behind the victim.

The simulated ATM size is based on a real ATM [74]. It is 0.6m wide, 0.64m high, and 0.4m deep. At 0.15m above the ATM base, we inserted a shelf upon which we placed the PIN pad and the monitor. The keyboard is 1.1m above the ground. To record keystroke sounds, we used the microphones of two *Logitech HD C920 Pro* webcams: one placed on the ATM's chassis 0.3m above the PIN pad, and another microphone – 2m in front of the ATM, as shown in Figure 4.2b.

The first data collection effort involved 38 participants (23 male and 15 female, average age 38.97 ± 11.36), while the second involved 20 participants (11 male and 9 female, average age 29.50 ± 5.74). Together, that makes the total of 58 participants who entered $5,800$ 5-digit PINs. We used both these

[2] https://www.davochina.com/4x4-ip65-waterproof-industrial-metal-keypad-stainless-steel-keyboard-for-access-control-atm-terminal-vending-machine-p00103p1.html

[3] https://www.davochina.com/4x4-ip65-stainless-steel-numeric-metal-keypad-with-waterproof-silicone-cover-p00126p1.html

(a) *PAD-1: DAVO LIN Model D-8201 F*

(b) *PAD-2: DAVO LIN Model D-8203 B*

Figure 4.3: Two commodity metal PIN pads we used.

data collections to obtain datasets of 4-digit PINs by removing the last key entered by the participants from each 5-digit PIN.

During the experiments, participants were asked to stand in front of the simulated ATM, and remain silent for the duration. A participant's task consisted of typing 100 5-digits PINs randomly generated, divided into four batches of 25 PINs. This split was made to allow for short breaks betwen batches in order to lower fatigue. PINs were displayed one at a time on the ATM screen: once a PIN is entered, the participant presses the Enter button to proceed to the next PIN.

Regardless of the individual's typing behavior and familiarity (or lack thereof) with a given PIN or the PIN pad, we decided to randomize the order of PINs, rather than ask users to enter the same PIN multiple times. This approach generalizes the $\mathcal{P}inDrop$ attack, which is actually applicable to both mnemonic PINs and One Time Passwords (OTPs).

We also collected the key logs of the PIN pad via the USB interface. In particular, for each pressed key, we collected both the "key-down" (press) and "key-up" (release) events. Moreover, we synchronized the recordings with the timestamp of these key events. All recordings were done with a sampling frequency of $44,100$Hz and then saved in the 32-bit WAV format.

4.3.2 Classification Methods

To identify the key pressed by the victim, we experimented with four well-known and popular classifiers: Support Vector Classification (SVC), k Nearest Neighbors (KNN), Random Forests (RF), and Logistic Regression (LR).

We applied a repeated nested crossfold validation to evaluate the performance of our approach. The pipeline varies on the number of attackers (i.e., a single attacker or a group) included in the training set.

In the outer loop, we randomly selected the attacker(s) among the participants in the dataset. This procedure was repeated 10 times generating 10 groups of attackers. The inner loop consists of a k-fold cross-validation, where k depends on the number of attackers. If the training set contains samples from a singke attacker, we used 5-fold cross-validation, since a user-independent split is not applicable. If samples from at least two attackers are present in the training set, we use a k-fold cross-validation user-independent where k is the number of attackers.

We varied hyper-parameters by using the grid search on all four considered classifiers. For SVC, we considered a linear kernel and varied C among: $[10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}]$. For KNN, we varied the number of neighbors to among: [1, ..., 20]. For RF, we considered from 10 to 100 estimators (steps of 10 and extremes included) and a max depth from 6 to 31 (steps of 5 and extremes included). Finally, LR was evaluated for li and l2 penalties, with C ranging from 10^{-4} to 10^{4}.

4.4 Experimental Results

We evaluate $\mathcal{P}inDrop$ in different scenarios, showing its performance in the different conditions in which the attacker may find himself. Section 4.4.1 describes how we evaluate different classifiers and consequently selected the best for our purpose. Sections 4.4.2 and 4.4.3 report the results for our algorithms on the key classification task and PIN classification task, respectively. Finally, Section 4.4.4 compares the performance of $\mathcal{P}inDrop$ with the results obtained in the state-of-the-art.

4.4.1 Model evaluation

To assess the performance of our classifiers, we evaluate different attack scenarios. In particular, we considered two settings: (i) number of distinct attackers and (ii) the number of digits entered by each attacker. We varied the number of attackers included in the training set between 1 and 10. This range has been selected to reflect a realistic attack scenarios. We varied the number of digits entered by each attacker in increments of 100, i.e., 100, 200, 300, 400, or 500.

The performance of our attack was evaluated on all possible combinations between the number of attackers and the number of digits entered by each attacker.

To select the best classifier, we compared the PINs validation accuracy of all the classifiers across different scenarios (i.e., PIN pads, and distances) and

	PAD-1		*PAD-2*	
	Distance 0.3 m	Distance 2 m	Distance 0.3 m	Distance 2 m
SVC	0.90±0.04	0.35±0.12	**0.86±0.06**	**0.21±0.07**
LR	**0.92±0.04**	**0.40±0.11**	0.85±0.06	0.19±0.04
KNN	0.65±0.07	0.13±0.07	0.17±0.05	0.02±0.01
RF	0.78±0.07	0.10±0.06	0.31±0.06	0.02±0.00

Table 4.1: PIN accuracies on the validation set for the investigated classifiers. The training set includes samples from five distinct attackers. The results show that for *PAD-1* the best performing model is the Logistic Regression (LR), while for *PAD-2* the best model is the SVC.

settings (i.e., number of digits per attacker, and number of attackers). SVC and LR achieved comparable performance, outperforming KNN and RF. In particular, LR achieved higher validation accuracy on *PAD-1*, while SVC showed better performance on *PAD-2*. In Table 4.1, we report a comparison of the validation accuracies for all the investigated classifiers, considering five attackers that train the classifiers with 500 digits each (i.e., training size = 2500 digits).

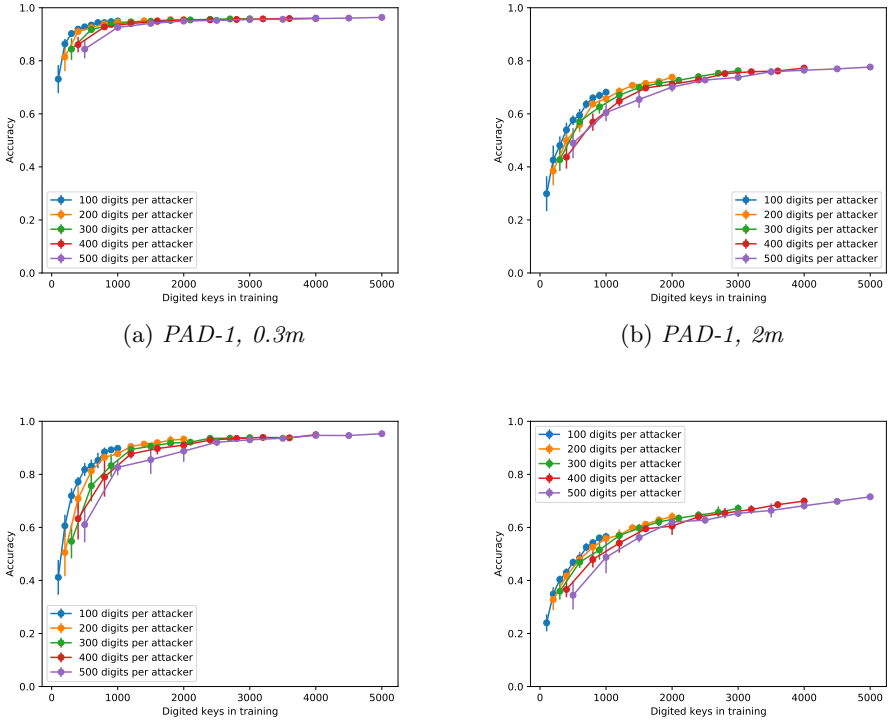
4.4.2 Single Key Inference

We report the LR classifier performance for the *PAD-1* and the SVC classifier performance for the *PAD-2* based on the validation results. In Figure 4.4 we show single key accuracy comparison for all the considered settings (i.e., the number of attackers and the number of digits entered by each attacker) in our four scenarios. Each graphic depicts how the accuracy varies in the considered scenario as the number of entered keys included in the training set varies. Further, each graphic shows five curves representing the number of digits entered by the attackers, while the bullets of a curve represent the number of attackers included in the training set. The bullets have an increasing value from left to right: the first bullet (from left) of each curve indicates that only one attacker has been included in training, the second indicates two attackers were included in training, and so on. Therefore, the number of numeric keys included in the training set varies according to the number of attackers and the number of digits entered by each attacker.

We note that the accuracy is significantly affected by the training set's size (i.e., entered keys in training) and the distance.

Interestingly, with the same number of entered keys in training, the accuracy improves due to the number of attackers. For example, if we set the number of entered keys in training at 400, we can see that in all scenarios, the accuracy obtained by four attackers typing 100 keys each (i.e., 20 5-digit PINs per attacker) is significantly higher than a single attacker typing 400

keys (i.e., 80 5-digits PINs). This may depend on the variability of the data used to train the classifiers. Each person has a slightly different typing style [114] (e.g., pressure strength, typing speed), and adding more attackers would introduce higher variance in the training set and helps our classifiers to generalize and improve their classification performance over a test set.



(a) *PAD-1, 0.3m* (b) *PAD-1, 2m*

(c) *PAD-2, 0.3m* (d) *PAD-2, 2m*

Figure 4.4: Key accuracy on the testing set for the best classifiers.

Furthermore, we analyzed how our classifiers mis-classify the true key to investigate how spatial locality interferes in the classifiers' predictions. In Figure 4.5, we report an example for the digit "3" for all the four scenarios (a similar behavior is shown by all the other keys).

Interestingly, we note a different distribution of classification errors between *PAD-1* and *PAD-2*. In the first case, the error is uniformly distributed over all digits, whereas in the second case, a higher concentration of errors is prominent around the true digit (i.e., digits 2, 5, and 6).

4.4.3 PIN inference

In a realistic context, an attacker generally has three attempts to guess the victim's PIN (i.e., the max number of incorrect PIN entries allowed before blocking the card). In this section, we report on the performance of our approach in PIN reconstruction in TOP 3-accuracy, i.e., only the three most probable PIN predictions. In Figure 4.6 we show the performance of the classifiers in the reconstruction of 4-digit and 5-digit PINs according to the different settings (i.e., PIN pad and distances). Further, similar to Figure 4.4, each graphic reports the performance for all possible combinations of the settings.

The results show that the effectiveness of the attack in each scenario. In particular, at 0.3m away, we can reconstruct correctly within three attempts up to 94% 4-digit PINs for *PAD-1* and up to 96% PINs for *PAD-2*. Although the performance worsens by increasing the distance at which the microphone is placed, $\mathcal{P}inDrop$ manages to reconstruct within three attempts up to 57% of the 4-digit PINs for *PAD-1* and up to 50% for *PAD-2* at 2m away. At 0.3m, the accuracy graphs reach a plateau at around 1500 digits in training. On the contrary, at 2m, the accuracy seems not to reach the plateau even with a training of 10 attackers and 500 digits per attacker (i.e., 5000 digits in training). This behavior is particularly marked in *PAD-2*, where the increase appears almost linear also with a high number of digits in training. This could be partially due to the classifier used in the specific scenario (i.e., LR for *PAD-1* and SVC for *PAD-2*) in addition to the physical differences between the two PIN pads.

Comparing the performance on two PIN pads (fixing the number of attackers and entered keys per attacker), the accuracy on *PAD-1* appears generally higher than the one on *PAD-2*. This applies to both distances. The number of attackers significantly affects performance with the same number of entered keys in training. For example, in *PAD-1* at 0.3m, the threshold of 80% of 4-digit PINs reconstructed in three attempts is reached with three attackers whom enter 100 digits each (i.e., 300 total digits), or two attackers whom enter at least 200 digits each (i.e., at least 400 total digits).

4.4.4 Comparison with the state-of-the-art

To evaluate $\mathcal{P}inDrop$, we compare its with that of state-of-the-art attacks exploiting acoustic emanations of PIN pads [36, 47, 97, 114]. Table 4.2 summarizes the results (with 10 attackers entering 500 digit each) in terms of key accuracy and PIN reconstruction accuracy within three attempts.

Both [97] and [36], exploit inter-keystroke timing. Although in [97] the distance at which the acoustic information is collected is unspecified, such attacks can be carried out from a distance over one meter, as demonstrated in [36]. The distance significantly decreases the risk of the attacker being detected. However, the reported performance is rather poor, since the PINs correctly reconstructed within three attempts were less than 1% for both attacks. However, from a greater distance (i.e., 2m) $\mathcal{P}inDrop$ outperform [36, 97] achieving the accuracy of 44% and 54% on 5-digit and 4-digit PINs, respectively.

Most effective attacks are those carried from a significantly shorter distance. In particular, [114] records acoustic emanations with a microphone placed at 0.05m from the PIN pad. This work obtains 60% key accuracy on a sub-set of keys (i.e., 6 on 10). Since we can not estimate the real accuracy considering all the 10 digits we decided for fairness, to leave this upper-bound. Under this assumption, we derived that this attack may achieve 4-digit and 5-digit PIN accuracies of 27.36% and 16.42%, respectively. Comparing these results with the performance of $\mathcal{P}inDrop$, we can see how $\mathcal{P}inDrop$ achieves better accuracy for both 0.3m and 2m.

The last method we consider was proposed by De Souza [47]. This attack assumes that two microphones are placed inside a PoS under the PIN pad. Unlike other methods, it uses the time of arrival of the acoustic signals. The performance achieved by the De Souza is slightly better to $\mathcal{P}inDrop$ from 2m. However, $\mathcal{P}inDrop$ has better performance from 0.3m (i.e., a 26% increase in 4-digit PINs and a 33% increase in 5-digit PINs). Moreover, $\mathcal{P}inDrop$ differs from [47] in that it does not require physical tampering with the device, even if the attack is performed from 0.3m away.

4.5 Potential Countermeasures & Future Work

The relatively high accuracy of $\mathcal{P}inDrop$ highlights its danger and the importance of robust countermeasures. Barring wholesale replacement of PINs with other login means, we consider the following possibilities:

- *Noise reduction*: This idea is simple, though challenging to deploy. It consists of masking the noise emitted by the PIN pad by covering it with soundproofing material. This approach could help in reducing the effectiveness of longer-range attack.
- *On-screen PIN pad*: An effective countermeasure could be to virtualize the PIN pad using a touch screen. (This is in fact already done on some ATMs). This countermeasure would also allow dynamic rearrangement of digits, making it much more challenging to implement

	Key Accuracy	4-digit PINs	5-digit PINs	Recording Distance
Liu [97]	NA	0.26% *	0.11% *	NA
Cardaioli [36]	NA	0.72%	NA	$1.50m$
Panda [114]	60.00%	27.36% **	16.42% **	$\sim 0.05m$
De Souza [47]	87.60%	68.40% **	59.92% **	$0.00m$ ***
$\mathcal{P}inDrop$	95.84%	94.64%	92.79%	$0.30m$
$\mathcal{P}inDrop$	74.58%	53.75%	43.99%	$2.00m$

* Performance derived from the proportion of human-chosen PINs and the accuracy of each PIN strength level reported in the paper.
** Performance estimated from reported key accuracy, assuming the prediction error to be equally distributed.
*** Multiple microphones are integrated in the device.

Table 4.2: Comparison between $\mathcal{P}inDrop$ and the state-of-the-art results on single key accuracy and percentage of guessed PINs within three attempts. If the score cannot be derived from the reference paper, we report N/A.

$\mathcal{P}inDrop$-like attacks. On the other hand, on-screen keypads are generally less user-friendly and can pose a problem for visually impaired users;

- *Feedback distortion*: If removing the characteristic sound emitted by each key is not possible, an alternative is to add noise that does not allow individual keys to be profiled. By emitting a masking sound at each key-press, $\mathcal{P}inDrop$ can be made more difficult, especially, its training phase;
- *Personal PIN pad*: Another possible countermeasure is to use a trusted device, such as a smartphone, to replace the physical PIN pad. The PIN could then be transmitted to the ATM using a wireless medium (e.g., NFC);
- *Behavioral biometrics layer*: An additional layer of security might be possibly via behavioral biometrics. One possibility is to involve user authentication based on keystroke dynamics. While this method can yield a high rate of false positives, it is completely transparent to the user (until or unless, a false positive occurs).

Possible future directions range from improving applicability of $\mathcal{P}inDrop$ to exploring its effectiveness on other kinds of PIN pads. An interesting direction might be to apply more sophisticated (e.g., parabolic) microphones. Such a microphone could significantly extend the effective recording distance of $\mathcal{P}inDrop$. Another direction is looking at $\mathcal{P}inDrop$ in the context

of screen-based PIN pads that are fairly common on modern ATMs. This setting is more complicated due to lack of physical keys the sound of which can be profiled. However, it would be interesting to study whether sounds emitted by the touchscreen still allow the attacker to infer information about keys pressed.

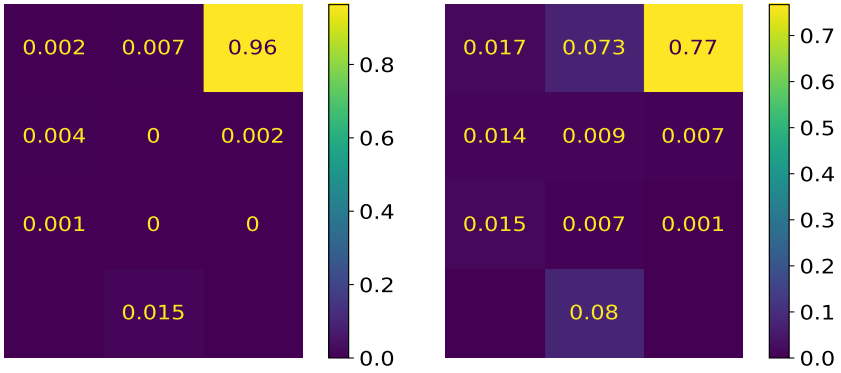
4.6 Summary

In this chapter, we demonstrated $\mathcal{P}inDrop$, a highly accurate acoustic side-channel attack on PIN pads. It takes advantage of acoustic emanations produced by ATM users entering their PINs into the commodity ATM's metal PIN pads. These emanations can be surreptitiously recorded and used to accurately profile all PIN pad keys, which allows $\mathcal{P}inDrop$ to yield the victim's PIN with high probability. Specifically, this work shows that $\mathcal{P}inDrop$ is effective when applied from a very short (and perhaps not always realistic) distance away from the PIN pad (0.3m) as well as from a rather safe and inconspicuous distance (2m).

We demonstrated effectiveness and robustness of $\mathcal{P}inDrop$ by conducting extensive experiments that involved a total of 58 participants and two commodity (commercially available) metal ATM PIN pads. We experimented with $\mathcal{P}inDrop$ in several configurations, showing how its performance can be optimized based on the training set size and the number of attackers.

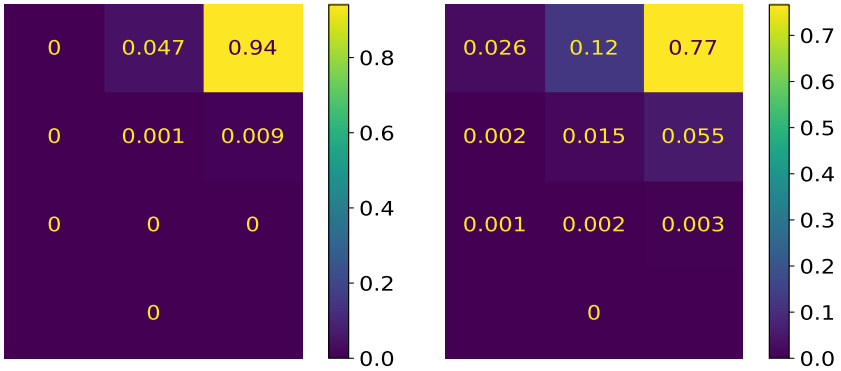
$\mathcal{P}inDrop$'s accuracy reaches 93% and 95% in reconstructing 5-and 4-digit PINs, respectively, within three attempts, from 0.3 meters away. Also, at 2m away, $\mathcal{P}inDrop$ outperforms state-of-the-art results, reaching over 44% accuracy. This translates into an average accuracy improvement of 44% and 53% in 5- and 4-digit PINs, respectively. We believe that, due to its real-world applicability and performance, this work significantly advances the state-of-the-art in acoustic side-channel attacks.

(a) *Generic PIN pad layout.*



(b) *PAD-1 at 0.3m*

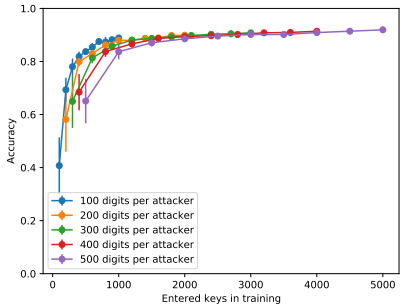
(c) *PAD-1 at 2m*



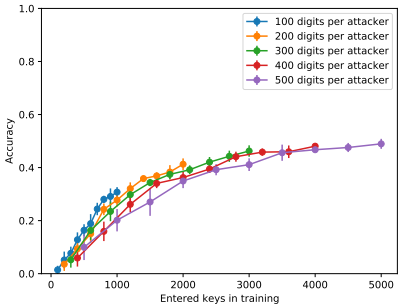
(d) *PAD-2 at 0.3m*

(e) *PAD-2 at 2m*

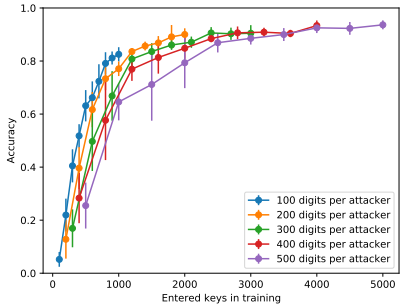
Figure 4.5: Digit "3" prediction heat maps for the four considered attack scenarios. We reported the results for the experiment with 5 attackers and 500 digits entered per attacker.



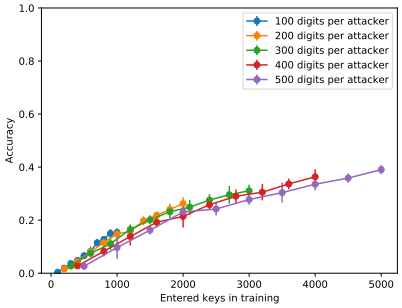
(a) *PAD-1 and microphone placed at 0.3m*



(b) *PAD-1 and microphone placed at 2m*



(c) *PAD-2 and microphone placed at 0.3m*



(d) *PAD-2 and microphone placed at 2m*

Figure 4.6: 5-digit PINs inference performance within 3 attempts for the best classifiers.

Part II

Novel Authentication Methods

Chapter 5

Eathentication: A Chewing-based Authentication Method

In the last decades, technological devices have become a real part of human life. These devices contain sensitive information like photos, message, bank applications. Therefore, authentication methods assumed a fundamental role, since they guarantee security and privacy to the actions we take. Among the most common authentication methods, we find passwords, PINs, and pattern. However, also, biometrics are regularly used, such as fingerprint and face recognition methods. In particular, smart devices widely used biometrics since they combine excellent levels of security and usability. However biometrics, still have some lacks: they require users to physically interact with the smartphone every time the authentication is needed. To remedy this problem, one of the techniques adopted is the use of wearable devices (smartwatches, headphones, etc.) as trust devices. This technique allows a user to unlock his smartphone without the need to continuously authenticate (under certain conditions). Thanks to these features, the usability of our smartphones can be significantly improved. However, there are still substantial limitations, since wearable devices do not generally provide biometric authentication methods such as smartphones. This lack is fundamental since it limits the usability of wearable in some daily life scenario. For example, it often happens that users need services provided by their smartphones, but they cannot because they have their hands busy (e.g., during a queue at the airport with many suitcases or while driving since this is a really common scenario). In many of these cases, people wear their

earphones, a device that many of us always use. If the earphones had an authentication method capable of recognizing their legitimate owner, new possible scenarios would open. A user could use a voice command to be recognized by their earphones. After the recognition, he can use again the voice commands to get access to all services of his smartphone – all this without having used his hands.

We propose *Eathentication*, a novel behavioral biometric authentication method that exploits earphones, a daily used device [2]. Our purpose is to provide an easy-to-use authentication method with a non-invasive interaction and economical device. In *Eathentication*, we consider as a unique behavior factor the ear channel movement during the chewing. A relation between the movement of the ear channel and the action of chewing exists thanks to the Temporomandibular Joint (TMJ) – two joints connecting the jawbone to the skull. TMJ is closely related to the external auditory canal and only separated from it by a thin layer of tympanic bone [5, 118]. A particular type of stimulation like talking, chewing, swallowing, yawning, laughing, coughing, breathing, eyelid reflex has ear response [117]. Furthermore, the relationship between the TMJ and temporal bone is confirmed [130].

In this work, we demonstrate our authentication method implementing a cheap and modified earphones. We believe that our method could improve smartphone usability, since it allows implicit authentication using earphones, a common device. We think this might have practical usage in several scenarios. Therefore, we submitted a patent application (IT patent n.102020000001192) for our method.

Contribution of our work In this chapter, we descibe the following contributions:

- To the best of our knowledge, we are the first to propose an authentication method based on external ear channel movements recognition during eating. We called this method *Eathentication*.
- To confirm our intuition, we designed and printed a cheap 3D prototype of modified earphones, integrating the sensors necessary to perform *Eathentication*. During our experiments, we collected data using this prototype, showing how it is possible to create a working and cheap prototype.
- We evaluate *Eathentication* designing an experiment where we asked participants to eat different types of food. To assess the feasibility of using *Eathentication* without eating food, we also collected data asking

participants to empty chew during the recording session. Our results show how *Eathentication* can be used as an authentication method, achieving good performance.

Organization This chapter is organized as follows: Section 5.1 discusses previous work; Section 5.2 introduces the model we choose for our authentication method; Section 5.3 describe the modified earphones we used for our experiments; Section 5.4 reports the data collection procedure we followed, the experiments we performed on these data and the results we obtained, in Section 5.5 we summarize our work and discuss the future possibility.

5.1 Related work

Nowadays people are increasingly connected and they continuously use smart devices to access their online services. One of the most important aspects for a smartphone is to provide transparency, usability, and security authentication. A way to obtain these properties is to use biometric authentication methods. These kind of metrics exploit human characteristics [58] to provide safe and usable access technique [126, 136]. In general, biometrics method can be classified respectively into two main categories: physical biometrics and behavioral biometrics. The first is based on users physiological features like fingerprint [33, 78, 79], ECG [26], face recognition [24], hand geometry [120]. Instead, the second is based on behavioral features, which involve human characteristics of everyday life like gesture [44], voice ID, mouse use characteristics, keystroke dynamics [20, 145], gait recognition [50] and signature verification [75].

About ear biometrics, nowadays there are several works focused on various features of the ear. An example is the recognition of ear shape. In [2, 41, 55, 116], authentication systems based on the recognition of the shape of the ear are widely introduced. The authors also described the datasets available in the literature. In [109, 110], Nanni propose a multi-matcher system based on a local approach. The image of the ear is divided into several windows and local features are extracted from each of them. Only the most discriminant window is kept to recognize a specific ear. In [80], Jamil proposed a technique that significantly reduce the problems related to the shape recognition with different lighting levels. This survey [32, 55] reports open questions and research directions for ear shape recognition. In [32] Burge describes some open issues like hair occlusion, low lighting levels, and changing of the ear shape over time. In [142] Yan reports

recognition problems related to the presence of earrings on the ear. Another important features for ear recognition is the acoustic proprieties of the ear channel. In [6], authors show how the acoustic properties of the outer flap of the ear can be exploited as a biometrics. Takayuki,in [9], propose instead an authentication method based on the ear reflection of the acoustic signal to recognize the ear shape. In [49], Derawi proposes a mapping and ranking of existing acoustic ear recognition systems.

Unlike physical biometrics, ear-based behavioral biometrics is still an undiscovered field. In [130] the authors propose a method to recognize human activities – like sleep, run, etc. – based on external ear channel movements. However, to the best of our knowledge, we are the first to propose a behavioral authentication method based on ear movements recognition during eating activity.

5.2 Our solution: Eathentication

To identify *Eathentication* use cases, we begin defining the model we adopted. Section 5.2.1 describes our assumption about the use of *Eathentication*. Section 5.2.2 then detailed presents our authentication method.

5.2.1 High level description

Our method is based on the assumption that the ear movement produced during chewing might be used as a biometric measure. We consider a chewing movement to start when the user begins the first chewing and to end with the swallow.

The model we adopt to describe *Eathentication* system is depicted in Figure 5.1. During the training and the authentication phases, we assume the user to wear our *modified earphones* (described in Section 5.3) – hereafter referred as earphones. The earphones is connected with the auth-device (authentication device). Our *Eathentication* algorithm, able to authenticate a user, is assumed to be installed on the auth-device. Furthermore, the user has different types of food available to be used for the registration. *Eathentication* aims to authenticate a *target user* based on his ear channel movements while chewing. If a user is recognized as legitimate, the *Eathentication* algorithm grants access to the auth-device.

To authenticate a user, the *Eathentication* algorithm must first carry out a training session on that specific user. During this training session, the target user wears the earphones and eats the provided food or empty chew. The auth-device collects the movements of the ears produced while the

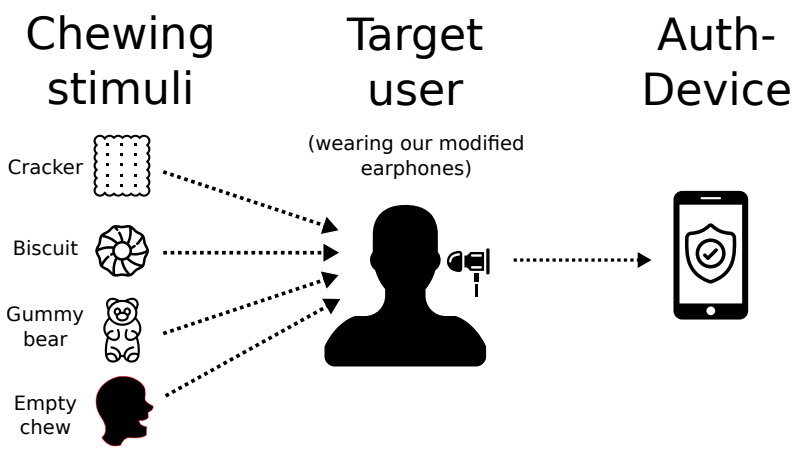


Figure 5.1: *Eathentication* model. In this scenario, a user tries to authenticate itself on a smartphone. First, the user eat or perform a stimulus. Consequently, the application on the smartphone evaluate if the user is legitimate or not.

user is chewing – hereafter referred as chewing-recordings. The *Eathentication* algorithm uses these chewing-recordings (from both this user and other users) to learn how to distinguish between the target user and every other user (even unknowns ones). To successfully authenticate, the user must eat the same kind of food eaten during the training phase or perform the same empty chew movement.

Figure 5.2 depicts the procedure to generate a trained *Eathentication* classifier able to recognize that specific user. We defined this procedure using the three phases described below.

Phase A – Data collection: During this phase, the target user wears the earphones, and the auth-device collects the chewing-recordings for the training phase. As described in Section 5.4.1, the target user collect the data during an eating session explicitly performed to recognize his eating behavior. We then join these data with a dataset of previously collected chewing-recordings from other users.

Phase B – Data processing: Starting from the collected data, in this phase, we extract the features used to train the machine learning model.

Phase C – Model generation: This phase selects the most relevant features from the previous extracted ones. We then train a model able to authenticate the target user based on the selected features.

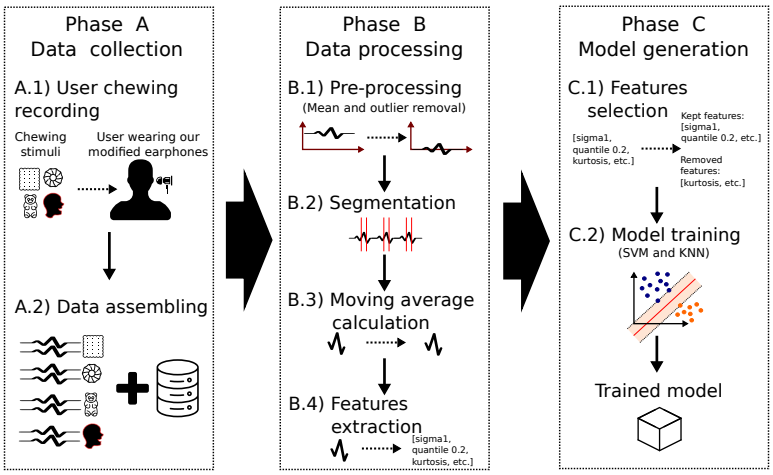


Figure 5.2: *Eathentication* processing steps: the chewing data collection (Phase A), the extracting of meaningful features (Phase B), and the model training process (Phase C).

5.2.2 Authentication description

Since *Eathentication* is an authentication method, we trained a binary machine learning classifier able to distinguish between a target user (the authorized one) and all other users (the unauthorized ones). This kind of choice is typical for machine learning-based authentication methods [53]. In this scenario, the *Eathentication* algorithm already knows the identity of the authorized user. After a user provides the secret information (the chewing-recordings in our case), the system answers with a binary decision: accepted if the user is recognized as the authorized one or rejected otherwise.

We now describe in more details all three phases and their different steps. Throughout the rest of the section, we refer to the phases and steps presented in Figure 5.2.

Phase A – Data collection This section describes in details the data collection phase. Below, we describe the data collection process:

A.1 **User chewing recording.** During this step, the target user wears the earphones. He eats different kind of food or empty chew by following the specific procedure described in Section 5.4.2. The collected chewing-recordings include data from the start of the first chew until the moment of swallowing. Because our earphones provides two different signals at the same time – one per ear –, chewing-recordings are always collected in pairs – keeping separated the signals and keeping information about their alignment.

A.2 **Data assembling.** In this step, a pre-collected dataset populated with several chewing-recordings from other users is joined with the data collected in the previous step. To make *Eathentication* as accurate as possible, this preliminary dataset should contain a wide range of users with the most disparate eating style. The obtained dataset is used as input for the following phase. Note that the data assembling step may be performed both in this moment or after the step B.4 – using a pre-collected dataset already converted into features. However, we place this step here to make it clear to the reader that both the data from the target user and other users are processed in the same way.

Phase B – Data processing This phase aims to extract meaningful features for the machine learning algorithm. In order to achieve better results we applied some filter and transformation to the original chewing-recordings. Below is a detailed explanation for each step.

B.1 **Pre-processing.** Raw collected data contains information which negatively affects the machine learning procedure by introducing bias or noise. First, we subtract from each chewing-recording its average value. Since, the data we collected are distances, the their average values represents the average distance from our earphones and the ear channel surface. Keeping this information in our chewing-recordings negatively bias the classifiers, since the average distance may change among different session. Finally, we also normalize the outlier values which may be caused by sudden movements of the earphones or the ear channel (e.g., swallow, head movements, but also shock to the earphones).

B.2 **Segmentation.** We perform data segmentation according to the following observation: the waveform of a chew presents a peak corresponding to the maximum opening of the mouth. Based on this assumption, we developed an algorithm able to detect this peak and extract the corresponding chew. In Section 5.4.2, we report the detailed description of the algorithm we used to extract chews.

This step divides a single chewing-recording into many chews. This optimization speeds up the time required to authenticate a user since the classifier learns to distinguish users based on single chews.

B.3 **Moving average calculation.** This step calculates the moving average of every chewing-recordings. As shown in [130], this transformation emphasizes users' specific chews property, allowing to extract useful features.

B.4 **Features extraction.** This step extract from the moving average several well-known features. The complete list of these features includes: mean, kurtosis, 0.2 quantile, 0.4 quantile, 0.6 quantile, 0.8 quantile, variance, skewness, sigma 1, sigma 2, sigma 3.

Phase C – Model generation In this phase, we trained a supervised machine learning model on the features extracted from both the *legitimate user* and many other *illegitimate user*. This phase is divided into two steps:

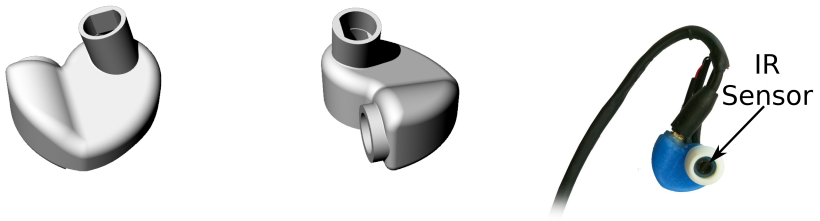
C.1 **Features selection.** To recognize the most relevant features for our specific purpose, we run the select K-Best algorithm on the features of the dataset. This method, provided by the library Scikit-Learn, select the best features according to the indicated metric (we used ANOVA F-value). We calculate the most frequently kept features among all the users of the dataset. Most kept features are quantile 0.8, quantile 0.2, sigma 3, quantile 0.6, sigma 1, variance, sigma 2, and quantile 0.4.

C.2 **Model training.** This step consists in training a binary supervised machine learning model able to distinguish between the *legitimate user* and all other *illegitimate users*. As a supervised model, we selected the support vector machine classifier, which shows better results compared to other tested classifiers – including linear regression, K-nearest neighbors, and random forest. The output of this phase is a trained model able to recognize if a chewing-recording comes from the authorized user or another (even unknown) unauthorized user. The complete procedure we followed is described in Section 5.4.2.

5.3 Prototype implementation

In this section, we describe our earphones prototype. We use this earphones to collect the dataset used in Section 5.4 to evaluate *Eathentication*.

We design our prototype using a 3D computer graphics application – a virtual rendering of the produced model is showed in Figure 5.3a and Figure 5.3b. To define the shape of our prototype earphones, we took inspiration from the shape of other commercial earphones. We selected this particular shape to optimize the stability of the device, however *Eathentication* is not bound to use this specific shape in order to operate. Unlike a commercial earphones, our prototype does not have a speaker. In fact, in the location where the speaker is usually present, we embedded an infrared LED. We used this sensor to collect data on the external ear canal movements. As sensor we used a Kodenshi reflective sensor combining a GaAs



(a) *Front perspective* (b) *Side perspective* (c) *Printed earphones with IR sensor*

Figure 5.3: Our built earphones.

IRED with a high-sensitivity photo-transistor. Our need to place the sensor where the speaker is usually positioned should not be considered a obstacle to include our method in future commercial earphones. In fact, there are sensors much smaller than the one we used, which could be integrated into normal earphones without particular problems. We printed our earphones using a 3D printer. In Figure 5.3c, we show the printed earphones used for our experiments.

Note, that the prototype we build use the same sensor proposed by [130]. However, we keep the hardware as simple as possible to show that is possible to monitor the ear channel movement with a cheap device.

5.3.1 Technical characteristics

Our earphones uses a sensor able to measure its distance from a specific surface. Precisely, the sensor consists of an infrared LED and a photo-transistor. The light emitted by the Diode is reflected from the adjacent surface (the external ear channel) to the junction of the photo-transistor. The distance is measured based on the variation of the light intensity using an Arduino Uno. Collected data consist of integer values that vary on a scale from 0 to 1023 Volts. In this representation, higher values mean lower distances.

The sampling frequency of the device is 100 Hz. We chose this value since higher sampling may create artifacts in the collected data – the Arduino we used may not be able to handle higher frequencies. Furthermore, we empirically observe that higher frequencies do not provide more information than those already collected at 100 Hz.

5.4 Evaluation

To show the effectiveness of *Eathentication*, we collected many chewing-recordings in a novel dataset. Section 5.4.1 presents the procedure we follow to collect our dataset. Section 5.4.2 describes then how we conducted the experiments to show the effectiveness of *Eathentication*. Finally, Section 5.4.3 shows and describe the results we obtained from our experiments.

5.4.1 Dataset

We carried out data collection in a laboratory. We standardized the procedure to be used to minimize any possible disturbance. During a registration session, we provided three kinds of food to each participant. The user performs all the experiments sitting on a chair and wearing our earphones. Before to start the data collection, we always asked the participants to follow these indications:

- Keep the posture homogeneous on the chair;
- Breathe through the nose while chewing;
- Minimize movements;
- Chew in the most natural way for their specific behavior – considering both chewing strength and speed.

Our procedure consists of two sessions:

- **Session one.** The participant performs a resting state registration. This session is necessary to stabilize the earphones inside the participant's ear. When a user wears an earphones, it takes a few seconds before reaching a position of stability within the ear. This settling process varies according to the specific physiology of the ear of the participant. During this session, the earphones signal is continuous checked. Once the earphones has reshaped the ear and found a stable position, the signal stabilizes on a specific level. The maximum time required for the signal stabilization never went beyond 30 seconds.
- **Session two.** This session consists of four steps during which the participant eats the provided food or performs a chewing movement. The first step is the only one that does not include food eating: the participants performs an empty chewing movement for a fixed time of 15 seconds. The next three steps consist in eating 1 potion of food

each – in order crackers, biscuit, gummy bears. For these three steps, the registration starts with the first chewing and terminates with the food swallowing. To complete the whole session two, the users repeat all four steps for three times: this means that for each step, we collect three distinct chewing-recordings per session.

For each participant, we repeat the above procedure 3 times - repeating each time both the two sessions. Between one session and the next, we have always waited at least 24 hours. We used this convention to collect data distributed over a medium-long period. This condition allows investigating the performance of *Eathentication* regardless of the variations that chewing may undergo this amount of time.

We collected data from 23 distinct participants, for a total amount of 828 chewing-recordings– 23 users that perform four chewing steps for three times (a session), repeated for three times. The dataset we collected is sensible bigger compared to the previous work that used similar recording devices [130].

Participants We recruited a total of 23 unique participants for our study. To provide results that as general as possible, we have collected data between people as diverse as possible. We randomly selected participants among graduate students and middle-aged people. All participants were healthy and never had serious health problems related to the ear canal or the masticatory system. Out of 23 participants, 6 were female, and 17 were male. The youngest was 24 years old at the time of the experiments, while the oldest was 64. Participants come from two continents and five distinct countries.

5.4.2 Experimental Setup

Once we collected our dataset, we evaluated the performance of *Eathentication* following the procedure described in *phases B* and *C* (Section 5.2.2). However, to evaluate the effectiveness of *Eathentication* for each participant, we repeated phase C considering each participant, one at a time, as the target user. To develop the authentication algorithm described below, we used Python 3 and its libraries.

As described in step B.2, we developed an algorithm to extract the individual chews performed by participants. The implemented algorithm works as described below. The first step consists in filtering the user's collected data. We analyzed all our dataset, and we identified the range of frequencies corresponding to all possible variant of chews performed by our participant. Obtained results showed a range between 0.5 and 1.5 Hz. We used this

range to apply a band-pass filter to every chewing-recording, thus obtaining a more defined chewing wave. Once filtered all chewing-recordings, we summed together all right and left channels – this operation is valid since the left and right channels are synchronized. The purpose of this operation is to minimize the influence on the final signal of any noise present on one single channel. We then identify every chew based on the position of its peak. Finally, we extracted each chew as a window of 110 sample (55 before the peak and 55 after the peak). Note that windows are extracted from the original data – the chewing-recordings without band-pass filter and from both the left and right channels.

To define the best window size, we conducted a preliminary experiment. We used the same procedure we are going to describe in the next paragraph of this section; however, we repeated the procedure several times using each time a different window size. Figure 5.4, shows the performances of *Eathentication* for each window size. We finally selected 110 as window size since it is a good trade-off between the FAR and FAR scores. Besides, 110 is approximately equal to the two sigma value for the average of all the chews we have detected. This value is therefore a good upper bound for the maximum size of most of the chews of our dataset.

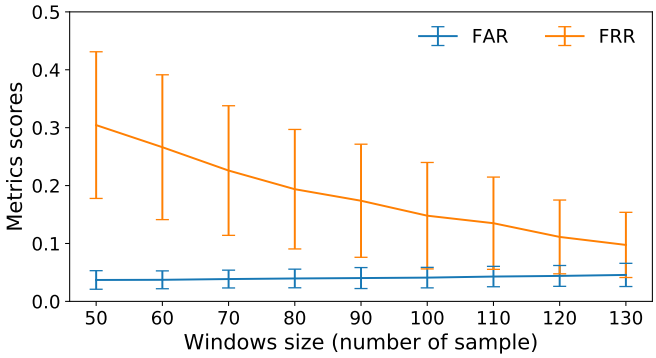


Figure 5.4: Results for a preliminary experiment conducted using different windows size during the Segmentation step (B.2).

Once phase B is complete, we have a dataset of features for each identified chew. We continue with phase C, but repeating it for each participant in the dataset and each time considering the current participant as the target user (the user to be considered as the authorized user). For each users we did the follow procedure.

First, we generate new labels for the current target user since the problem we need to solve is a binary problem. Therefore, the samples of the target user will receive the label 1, while the samples of the other users

will receive the label 0. Once generated the binary dataset and extracted the most significant features, we start the evaluation process. Note that, unlike in phase C, which returns a trained model, during the evaluation, we used a nested five-fold cross-validation. This technique returns the average results achieved by the tested model over five runs. Moreover, since we aim to train a machine learning model to authenticate chewing-recordings also from unknown users (users never saw in the training set), we define a user-independent k-fold cross-validation. This technique divides the dataset ensuring that training set and test set never contain chewing-recordings from the same users (except for the target user). Within each cycle of our k-fold cross-validation, we used a grid search to optimize the hyper-parameters of the machine learning model. Since the grid search further splits the training set into a training set and a validation set, we used another time our independent k-fold cross-validation (still of size five) to assure the two sets to be user-independent. Moreover, since the training set is highly unbalanced, at each cycle of the grid search k-fold, we balanced the training set. Note that balancing is not applied to validation sets and training sets, as it is realistic to test the classifier in unbalanced conditions. Once obtained the best classifier from the grid search, we evaluate it on the test set by calculating the False Acceptance Rate (FAR) and the False Rejection Rate (FRR) scores. FAR and FRR are two commonly used metrics for biometrics methods [53]. We calculate the final result for the current target user as the average of the five results obtained from the nested cross-validation. Once obtained the average results from the nested cross-validation, the evaluation on the single user is completed.

Once repeated the above procedure for each user of the dataset, we obtain the average performances of *Eathentication* for every user.

5.4.3 Results

In this section, we describe the experiments we conducted and we comment the obtained results.

5.4.3.1 Eathentication Performances

Hereafter, we reports and comments the results we obtained by training four machine learning models as described in Section 5.4.2. These machine learning models are support vector machine (SVM), K-Nearest Neighbors (KNN), Logistic Regression and Random Forest. In Table 5.1, we report the results we obtained for the SVM and KNN classifiers, as the other two methods did not show interesting results. Our results show that the SVM and

KNN classifiers achieve comparable performances for both the investigated metric. In follow experiments, we will reports results for the SVM model, since is achieve similar or better results, compared to the KNN model.

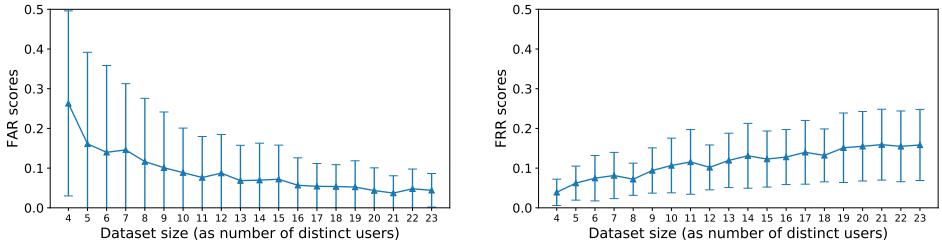
Table 5.1: Performances achieved by *Eathentication* for the nested five-fold cross-validation.

Classifier	FAR	FRR
SVM	0.041 ± 0.016	0.128 ± 0.043
KNN	0.046 ± 0.016	0.119 ± 0.053

5.4.3.2 Intuition Assessment

With this experiment we want to show how *Eathentication* performance changes according to the number of users involved in the train-test process. Similarly to Frank in [61], to evaluate this experiment, we generated a pool of sub-datasets, starting from our complete dataset. These sub-datasets are generated by randomly kept n users from the complete dataset, with $n \in \{4, ..., 23\}$. This process generates a total of 20 sub-datasets with an increasing number of users.

We evaluated the performances of *Eathentication* applying the procedure described in 5.4.2. However, we repeated the whole procedure for all the sub-datasets we considered. Figure 5.5 shows the results we obtained for each considered sub-datasets size. We performed the experiment using the SVM model. Moreover, since the random choice of the users may profoundly influence results, we repeated the experiment 50 times. Therefore, in Figure 5.5, the results for each size corresponds to the average performance on 50 random sub-dataset of that size. For both considered metrics, scores and their standard deviations tend to stabilize on specific values as the number of considered users grow. For the FAR metric, achieved results drop-down with the increase in the size of the dataset. For datasets with 20 or more users, the FAR scores start to stabilize achieving an average value of 0.047 and a stable variance of 0.05. For the FRR metric, the achieved results grow with the increase in the size of the dataset. Similarly to the FAR metric, for datasets with 19 or more users, the FRR scores start to stabilize achieving an average value of 0.15 and a stable variance of 0.10. Based on this experiment, we state that our authentication method can still achieve good performance even if trained on a higher number of users.



(a) *FAR* (b) *FRR*

Figure 5.5: Results achieved by *Eathentication* on sub-datasets of different sizes. The size of the sub-dataset refers to the number of distinct users.

5.4.3.3 Multiple chews Study

In this section, we evaluate the performances of *Eathentication* by varying the number of chews used to classify a user. The procedure used to train the SVM model and to classify the single chew is the same described in 5.4.2. However, this time the user's classification (as a target user or not) is no longer based on an individual chew, but on n distinct chews with $n \in \{1, ..., 5\}$. In case of ambiguity (same number of negative and positive classifications), we preferred safety over usability by rejecting the user. We performed this experiment using the SVM model.

Figure 5.6 shows obtained results. As, expected for both FAR and the FRR metrics performances improves. For FRR the improvement is quite sensible, passing from 0.128 ± 0.043 with a single chew to 0.04 ± 0.027 with only two chews.

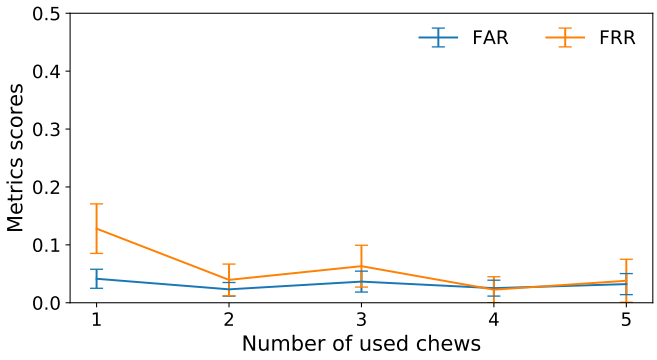


Figure 5.6: Results achieved by *Eathentication* considering more than one chewing in the classification process.

5.4.3.4 Foods Study

In this section, we show and describe the performances of *Eathentication* for each stimulus. This comparison takes place according to two different scenarios. In the first scenario, we consider a single classifier trained on the whole dataset. This scenario corresponds to the experiments conducted in Section 5.4.3.1. However, this time we divide the samples of the test set in four groups (each one containing only a specific stimulus), and we calculate the performance for each group independently. In the second scenario, we consider four distinct classifiers trained and tested each one on a different stimulus. The training procedure is the same used with one single classifier on the whole dataset.

In Figure 5.7, we show the performance for the first scenario, while in Figure 5.8 the performance for the second one. Compared with the performances reported in Table 5.1, the FAR metric generally shows good results in both scenarios. The variance for FAR remains stable in all cases and, in the second scenario, performance improves slightly. Contrarily, for the FRR metric, the results are different between the two scenarios. For the first scenario (one classifier), the FRRs are almost the same as Table 5.1 for Empty chew, crackers and biscuits – results get worse only the gummy bear stimulus. Instead, in the second scenario (four distinct classifiers), the performances drop for all stimuli and in particular for empty chew.

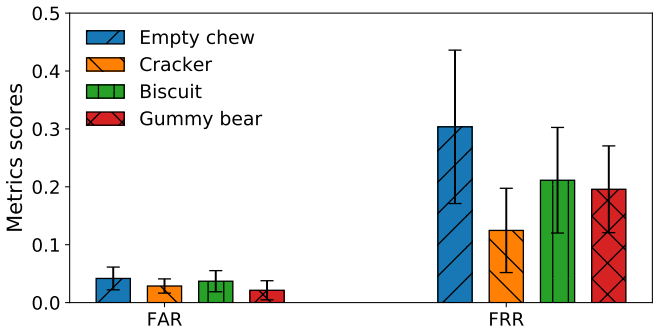


Figure 5.7: SVM performances training one classifier on all stimuli (the same experiments performed in Section 5.4.3.1). We divided the results to show the performances for each stimulus.

These results are significant, as they give us a way to understand how *Eathentication* works. The first thing to consider is that the performance is much worse for the second scenario (four classifiers). This result indicates that the used machine learning model generalizes better when trained on different stimuli. Another interesting observation is related to the results

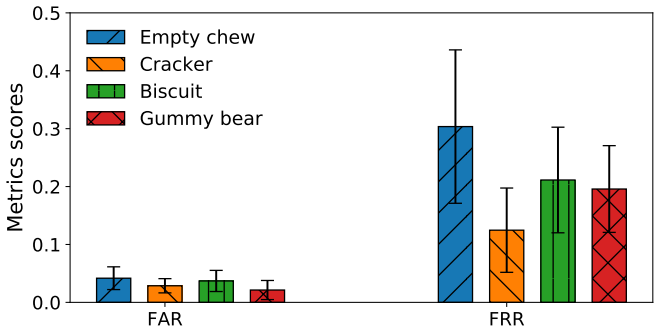


Figure 5.8: SVM performances training four distinct classifiers – one on each stimulus.

achieved by empty chew. This stimulus has the most significant difference in performance between the two scenarios. The low performances achieved in the second scenario can be explained based on the nature of this particular stimulus. Since chewing empty requires less effort than chewing food, the movements of the ear canal are also weaker. As consequences of this physical effect, the data we collected are negatively influenced. In Table 5.2, we reported the average variance of the features we extracted. As clearly visible, the variance for empty chew stimulus is much lower than the others. Therefore the classifier trained only on the empty chew stimulus, achieved worse results since the variance of the training data is not enough to well generalize. For the first scenario, we can instead justify the excellent performance, obtained on empty chew, based on our first consideration. Indeed, as previously deduced, the machine learning model generalizes better if trained on different types of stimuli. Therefore, for this scenario, the model probably well recognizes users also for empty chew based on other stimuli.

Table 5.2: Average features variance for all the stimuli considered in our dataset.

Stimulus	Features variance
Empty chew	9.37×10^{-5}
Cracker	3.84×10^{-4}
Biscuit	1.94×10^{-4}
Gummy bear	2.01×10^{-4}

In conclusion, we can say that *Eathentication* performs well on the single stimulus in at least one of the two scenarios. This result is essential for the empty chew stimulus, that is the most relevant for a usable authentication system. Therefore, this result shows how it is possible to train our method

on four stimuli and obtain a classifier capable of authenticating a user based on his empty chew. For the second scenario, its performances are not yet sufficient to guarantee an excellent authentication method. However, one possible solution could be to improve the hardware of our earphones. On the market, there are better sensors compared to that one we used. Therefore, at a higher cost, it may be possible to obtain a dataset of better quality.

5.5 Summary

In this chapter, we present *Eathentication*, the first method that explores the external ear channel during chewing to authenticates people. *Eathentication* takes advantage of LED proximity sensors placed on a modified earphones to recognize ear channel behavior. The authentication procedure we proposed is based on a supervised machine learning model.

During our experiments, we collected data from 23 participants. We used four well-distinct chewing stimuli – three different foods and one empty chew movement. To show the effectiveness of *Eathentication* with users never seen during the training phase, we conducted all experiments using user-independent test sets. Results show that *Eathentication* can effectively authenticate people based on their chewing behavior on the investigated stimuli. We experimented with different dataset sizes. We show that the performance of *Eathentication* tend to stabilize on the achieved results as the number of users considered increases. Among the investigated machine learning models, SVM and KNN achieved better results. The SVM classifier achieved, on average, a FAR score of 0.041 ± 0.016 and an FRR score of 0.128 ± 0.043. We also investigated the results for multi chew authentication showing that the FRR performances significantly improves with just two chew: SVM achieves under 0.039 of FRR with two chews. Finally, we investigated also the performance of *Eathentication* for each single stimulus. The most important results is that *Eathentication* can achieve excellent performances when it authenticates a user based on an empty chew (the training set must contain all stimuli). This features is very important, since it shows that *Eathentication* can effectively exploits the most usable stimulus (empty chew) to authenticate users.

5.5.1 Future Work

As future work, we first plain to extend our dataset by introducing new kind of stimuli and applying new machine learning model. Integrate new kind of stimuli would be interested, since it will be then possible to investigate the

performance of *Eathentication* in a stimulus independent test. Moreover, it would be also interesting to collect data in a more realistic scenario (e.g, when participants are engaged in other activities). If the this experiment were promising, it would be possible to authenticate users based on stimuli that have never been seen during the training phase. About the evaluation of the authentication method performance, a new important metric that we can consider is the one described in the work by Eberz et al.in [53]. In this work, the authors propose an additional metric to evaluate the error distribution of an authentication method. Thanks to the new data collections containing more data and of a more heterogeneous nature, we will calculate this additional metric, thus obtaining a more accurate definition of our method's safety level. Regarding possible new machine learning models to investigated, during our experiments we only took into account binary machine learning models. Other possibility may come from one class machine learning models, which are of particular interest for authentication methods. Possible example for these type of classifiers are SVM One Class and Autoencoders artificial neural network.

Chapter 6

TAMBUS: A Novel Authentication Method Through Covert Channels for Securing Industrial Networks

Modbus is a network protocol that enables communication between clients and servers in Cyber-Physical Systems (CPSs), in particular for industrial networks. Modicon (now Schneider Electric) developed this protocol in 1979 to communicate with its Programmable Logic Controllers (PLCs). *Modbus* is a *de facto* standard [52], and many different vendors still use it for its ease of use [18]. However, *Modbus* provides no security against unauthorized commands or interception of data, and consequently, the safety of facilities is at risk. Stuxnet [56], one of the most famous modern attacks on Industrial Control Systems (ICSs), taught the world that attacks on ICSs are possible and that they can also remain silent and undetected for a long time. More recent attacks, such as the attack against the Ukrainian power grid [38] in 2015, the Shanoon malware that infected Saudi Aramco in 2012 [31], or the leakage of data from NASA [4] in 2019, demonstrate that security on CPSs is a critical research field nowadays. The introduction of modern devices (e.g., Industrial IoT sensors) in legacy environments and the remote access to the operational network disrupts the assumption of isolated industrial systems. Due to these changes in the operational scenarios and to be compatible with the newer introduced protocols, a new version of *Modbus*, named *Modbus/TCP*, replaced the original one. *Modbus/TCP*, like its predecessor, is a plain text protocol and it provides no security features like authenticity and integrity (TCP may provide integrity, but a man in the

middle attack (MITM) may modify the a packet in each moment). This lack of security makes *Modbus/TCP* an extremely vulnerable protocol. In this work we focus on *Modbus/TCP* which over time has almost replaced the original *Modbus*.

Due to the implicit security issues (replay attack, MITM attack, etc.), many researchers tried to create secure versions of *Modbus/TCP*. Developed proposals focused on two possible types of solutions: encrypting each *Modbus/TCP* packet in a secure protocol (e.g., using TLS) or using covert channels to hide authentication information. However, these methods have implicit gaps which make their implementation difficult or even impractical in real cases. Indeed, previously proposed methods require to modify the *Modbus/TCP* protocol (with high costs to update the facilities devices), or alternatively they propose security by obscurity covert channels which can be easily bypassed.

In this work, we propose *TAMBUS*, an authentication method that allows trusted devices to verify the authenticity of industrial clients and servers and the integrity of the network traffic in *Modbus/TCP*. *TAMBUS* differs from the previously proposed methods since it is *not a security by obscurity* method and does not *modify the original protocol*. *TAMBUS* still uses covert channels but with a different purpose. In previous approaches, covert channels were the only way to provide security: the system's strength was that the attacker did not know there was a security system. If this coverage fails, the system becomes easily vulnerable. On the contrary, as it has been structured, *TAMBUS* continues to provide the same security level regardless of whether an attacker is aware of his presence and its architecture. The purpose of the covert channels in *TAMBUS* is that they do not contribute to the system's security, but they solely provide an alternative data transport channel. Thus, they ensure full backward compatibility with the *Modbus/TCP* protocol. Therefore, in *TAMBUS* the software present in the already installed devices (e.g., clients and servers) does not undergo any modification. These features are fundamental, as they justify the use of *TAMBUS* instead of a modern ICS protocol (e.g., OPC UA). Indeed, the transition to modern and safe protocols implies substantial upgrade costs (due to code rewriting, and device replacement) and it is often not justifiable in the face of possible security issues. Conversely, *TAMBUS*, which does not modify the existent code and guarantees a high-security level, can be a trade-off between facing huge upgrade costs and using an old unsafe technology. To provide these features, *TAMBUS* accepts a compromise between security and backward compatibility. Instead of providing a fully encrypted protocol, it improves the security of *Modbus/TCP* with an additional level of protection.

TAMBUS is not intended as a protocol to use when even modifying a single bit in a flow of packets can cause serious consequences to an ICS. However, it is essential to note that the probability of performing an attack, without being successfully detected by *TAMBUS*, is low. Attacks usually require the modification of several packages, and, as explained in Section 6.7, the probability of modifying n packages decreases exponentially with the increase of n. This makes it extremely difficult for an attacker to manage complex attacks (i.e., which require modifying multiple packets), as an alarm will be launched in the very first moments of the attack.

TAMBUS architecture adds new devices that embed unique codes to *Modbus/TCP* packets through secure covert channels. These codes allow data authentication and payload integrity information. In order to ensure the non-reproducibility and freshness of the generated codes, *TAMBUS* pre-shares a key and a counter between the novel added devices. However, unlike the previous proposed methods, *TAMBUS* generates the secret codes not only based on secret keys and counters, but also based on the content of packets, guaranteeing their integrity. The main benefit of using covert channels for *TAMBUS* is that they do not introduce protocol modifications or traffic overheads (i.e., extra bits or messages). Moreover, *TAMBUS* just needs the installation of three type of inexpensive devices, providing a *low-cost method* to secure a network.

Contribution of our work In this chapter, we discuss the following contributions:

- We propose *TAMBUS*, an authentication method that allows trusted devices to verify the authenticity of industrial clients and servers and the integrity of the network traffic in *Modbus/TCP*. *TAMBUS* exploits two covert channels to send a secret message on the *Modbus/TCP* transactions, without modifying the *Modbus/TCP* protocol and without adding traffic overheads. Moreover, *TAMBUS* is not implemented with a security by obscurity method;
- We evaluate the effectiveness and the performances of *TAMBUS* through the implementation of an industrial testbed including a real *Wago PLC*. Our results show that the used covert channels do not affect the regular *Modbus/TCP* functionality. Moreover, we show that the bit error ratios are always lower than 0.0012%, on the number of sent packets, for the timing-based covert channel on our testbed.

Organization In Section 6.1 we overview the state-of-the-art on methods

to securing the *Modbus/TCP* protocol. Next, in Section 6.3, we present the system model and the threat model for various attack scenarios. In Section 6.4, we present *TAMBUS*. Section 6.5 describes the developed methods and Section 6.6 the proposed implementation for these methods. In Section 6.8 we show the evaluation of the proposed method obtained using our testbed. Finally, in Section 6.9, we summarize this chapter and we propose possible future works.

6.1 Related work

CPS networks were initially developed with the concept of isolated networks, without connections beyond the production area where they operated. However, in the last years, this principle has become more unstable due to the interconnection of new technologies with legacy devices, paving the way for a series of possible attacks. Cyber-attacks against *Modbus/TCP* networks is an already well-investigated field. In [73] Huitsing proposed a taxonomy for the most famous attacks against the *Modbus/TCP* protocol. Many works [4, 31, 38, 56] provided reports and analysis of actual modern attacks against real ICSs.

Due to the catastrophic consequences that a cyber-attack could cause, researches attempted different ways to secure these protocols. One first attempt is dated 2009, when Fovino [60] proposed a new *secure* version of Modbus. The proposed new protocol satisfied traditional security requirements such as confidentiality, non-repudiation, and integrity. Enforcing security on the protocol is achieved by adding new SHA2 hashing function and RSA-based signature scheme to the original *Modbus/TCP* frame. In [59, 71, 141] researchers proposed similar techniques, always aiming to improve *Modbus/TCP* by changing the standard protocol. In 2018, the Modbus Organization (a group of independent users which promote Modbus) proposed a novel standard called *Modbus Secure*, which blend the old protocol with TLS to create a secure new one. All these proposals assume to modify the original *Modbus/TCP* protocol, but they consequently require the substitution of old devices with new compatible ones, which involves an expensive upgrade process. As previously mentioned, *TAMBUS* is much cheaper than this solution. Migrating to a new secure protocol, such as Modbus Secure, requires to change the PLCs intended to secure. Such devices are very expensive, for instance, our test PLC (WAGO 881) is a relatively small one, and it costs approximately €700. Conversely, *TAMBUS* uses cheaper devices, such as Raspberry Pi 3 B with a cost of €40 with additional €40 to install a TPM, and approximately in the same quantity of the PLCs intended to secure in

the previous solution. *TAMBUS* could also be implemented with less powerful devices due to its light computational burden, therefore a Raspberry Pi Zero worth €20 could be used instead.

The reader might think that the introduction of cryptography in *Modbus/TCP* can be obtained by adding two simple devices (between client and server) that encapsulate the protocol in an encrypted connection. Theoretically, this would be possible as the client and the server would send and receive regular *Modbus/TCP* traffic, however, such modification may imply several problems in a legacy environment. One problem could be the compatibility with the previously installed monitoring systems. Some companies make indeed use of passive ICS monitoring systems (e.g., as reported in [112]) and Intrusion Detection System (IDS). These solutions analyze the *Modbus/TCP* traffic by monitoring some core nodes in the network. If we assume to encrypt the *Modbus/TCP* traffic, we must also consider to update all already installed monitoring systems to keep them running. Furthermore, allowing these devices to decode the *Modbus/TCP* implies sharing the cryptography keys. Passive monitoring systems can be in exposed locations where they are exposed to physical attacks. Sharing the secret keys would therefore significantly increase the attack surface for an attacker, who could try to steal the secret keys. With these keys the attacker will be able to modify the encrypted *Modbus/TCP* traffic. Contrary, *TAMBUS* is completely transparent for a legacy environment, since it uses a covert channel to transmit extra security information in a legitimate *Modbus/TCP* packet. Therefore, even if *TAMBUS* transmits the *Modbus/TCP* packets in clear, on the other hand the modified *Modbus/TCP* packets are still readable by each monitoring system without any further changes. *TAMBUS* has therefore two essential advantages compared to the use of cryptography. First, it avoids spreading the cryptography keys on third non-secure devices, reducing the surface for an attacker. Second, even though the economic costs for the introduction of new devices would be the same, *TAMBUS* is more economical, compared to the use of cryptography, in the sense that it does not need to update existing monitoring systems, nor to manage the distribution and maintenance of the cryptographic keys also on these devices.

This obstacles have pushed the research to investigate covert channels in order to authenticate the *Modbus/TCP* transactions without modifying the original protocol. One of the first solutions appears in 2011, when [129] proposed a new scheme of authentication which exploits a timing-based covert channel. Conversely, storage-based covert channels in network protocols usually exploit unused header fields. Zander et al. in [146] collect several methods to transmit information, for example through unused header

bits, checksum fields or header extensions and padding. Among these, we illustrated the ones used in literature for *Modbus/TCP*. Similar work in 2014 [94] presented a storage-based covert channel to transmit hidden information using extra ad-hoc *Modbus/TCP* request. The communication occurs by pre-sharing a mapping of the addresses contained in the *Modbus* read requests (which are always between 0 and 9999) to the ASCII table. To communicate a single ASCII character, the client requests the address associated with that specific character. [93] gave an inspiring proposal, in which the authors proposed a storage-based covert channel to add more security to *Modbus/TCP*. The proposed scheme leverages least significant bits (LSBs) contained in read and write messages as a covert channel. The limitation of the method is that there is an explicit trade-off between the bandwidth and the stealth of the channel. Subsequent studies have shown that timing-based and storage-based covert channels can be used together in order to increase the security against possible attackers. In [131], the authors propose using storage-based and timing-based covert channels to provide some level of authentication to *Modbus/TCP* messages. The scheme assumes a client to send the server, through the covert channel, a lightweight CRC hash of the message that they want to authenticate. Upon reception of the CRC, the server decodes it and compares it with the reconstructed CRC of the received message to determine its authenticity. However, the authors do not provide any concrete implementation of this methodology.

In the current state of the art, no research accomplishes to introduce authentication and integrity control in *Modbus/TCP* without changing the original protocol. Indeed, methods that require protocol modifications are difficult to implement due to the enormous upgrade costs involved and problems related to IDSs and other software exploiting the plain text payload of a *Modbus/TCP* packet.

6.2 Background

In this section, we describe the *Modbus/TCP* protocol to provide the reader with the essential information to understand how this protocol works.

6.2.1 *Modbus/TCP* protocol

Modbus is a protocol developed by Modicon in 1979 to communicate with its Programmable logic controllers (PLCs). *Modbus/TCP* was developed in 1999 and, by design choice like its predecessor *Modbus*, is a plain-text protocol as its use was intended for in-industry networks. *Modbus/TCP* follows a

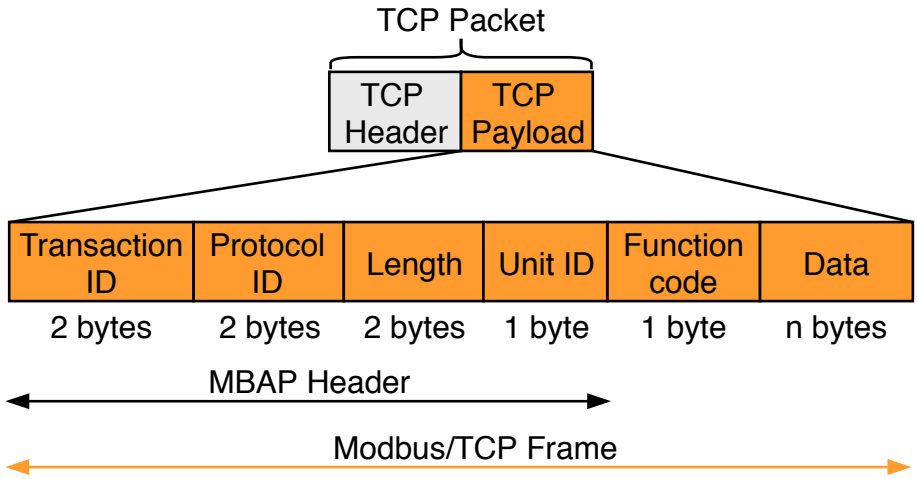


Figure 6.1: Encapsulation of a *Modbus/TCP* frame in a TCP packet. The represented frame corresponds to a *Modbus/TCP* request.

master-slave architecture: several slaves, also called *servers*, are connected with one master, also called *client*. A client, such as a Human Machine Interface (HMI), always starts the communication and sends *requests* to the server. *Modbus/TCP* servers correspond to PLCs, the actual devices capable of measuring and controlling the environment by using sensors and managing physical processes. The server sends *responses* to the corresponding *requests* of the clients. We will refer to a complete traffic flow, composed of a request and the corresponding response, as a *Modbus/TCP transaction*.

The *Modbus/TCP* frame is encapsulated inside the payload of TCP. The frame contains the MBAP Header (Transaction ID, Protocol ID, Length and Unit ID), the Function Code, and a *Data*. The Data field content varies based on the type of *Modbus/TCP* packet (see 6.2.2). The Protocol Identifier (Protocol ID) is filled with 16 bits of zeros (0) to indicate that the frame is a *Modbus/TCP* frame. Figure 6.1 depicts the TCP packet and the encapsulated Modbus Frame.

A *Modbus/TCP* server provides a client with four different object types to store and read data. In turn, we divide these objects into two subcategories: inputs and outputs. The inputs can be digital (Discrete Inputs) and analog (Input Registers), and they are used by a server to store the values read from sensors into the local memory. The outputs can also be digital (Coils) and analog (Holding Registers). Coils are used to open or close a digital switch, while Holding Registers are used to vary an analog

Table 6.1: *Modbus/TCP* object data types. The listed types of objects corresponds to the four primary types existing on a *Modbus/TCP* server.

Object-type	Access	Size
Coil	Read-write	1 bit
Discrete input	Read-only	1 bit
Input register	Read-only	16 bits
Holding register	Read-write	16 bits

output value within a range. All available object types are reported in Table 6.1.

Hence, two different macro-operations are possible for a client: read and write. There are different read and write operations, and they differ based on the type of object they are operating with. Usually, clients continually perform read requests at a fixed time interval (polling) to obtain data from sensors and display them on control monitors. Conversely, write requests are performed occasionally, when it is necessary to actuate something in the physical process.

6.2.2 Supported types in *Modbus/TCP*

The PLCs save the object types in 16-bits registers on the server. *Modbus/TCP* allows saving data in one or more registers using two types: floating-point (16, 32, and 64 bits) and integer (8, 16, 32 bit). Data types bigger than 16 bits are saved in two or more adjacent registers. Data types smaller than 16 bits are saved in a single register but padded with initial zeros. It is important to understand that a client, in order to correctly decode the values contained in one or more registers, has to be aware of the interpretation to associate with those specific registers. In a single request a client can request up to 61 floating-point values (32-bit per value). The values of the requested registers are all contained (and maintain the same order) in the *Data* field of the corresponding *Modbus/TCP* response, and bytes are transmitted in big-endian order: the most significant byte first and the least significant byte last.

6.3 System model and threat model

In this section, we describe the system (Sec. 6.3.1) and the threat model (Sec. 6.3.2) for an ICS that operates with *Modbus/TCP*.

6.3.1 System Model

The system model we used is depicted in Figure 6.2. We assume that the considered ICS follows the rules given by the Purdue Enterprise Reference Architecture model (or simply Purdue model) [139]. The Purdue model divides the ICS architecture into six levels and five zones, as shown in Figure B.1 in Appendix.

The IT network (the Enterprise Zone, where services and applications connected to Internet live) has no direct communication with the ICS environment (Manufacturing and Area Zone), and usually the access to the ICS environment is managed through a Demilitarized Zone (DMZ). Level 2 systems include manufacturing operations equipment for the production area, such as HMIs or control room workstations. Level 1 includes process control equipment that receives input from sensors or that sends data to process devices. In this level PLCs exist. Finally, level 0 includes the sensors and instrumentation that are directly connected and control the manufacturing process.

As showed in Figure 6.2, in our system model we consider only the first three levels, hereafter named as Client Area (Level 2/3), Server Area (Level 1), and Process (Level 0). Without loss of generality, we also assume the presence of three different switches in our system model. For sake of simplicity, we named these three switches respectively: "C.A. Switch" for the switch of the Client Area, "S.A. Switch" for the switch of the Server Area, and simply "Switch" a generic switch available anywhere on the network between the Client Area and the Server Area.

We assume the presence of three different types of switches in our system model. For the sake of simplicity, we named two of these three switches respectively: "C.A. Switch" for the switch of the Client Area, and "S.A. Switch" for the switch of the Server Area. Moreover, the Client and the Server Areas could be connected by an indefinite amount of switches (reported as WAN in Figure 6.2), since the facilities are geographically distributed. Without loss of generality, in rest of the chapter, we refer to the WAN network simply as "Switch": a generic switch available anywhere on the network between the Client Area and the Server Area.

The context that we consider in this work is a typical ICS scenario. The system could be made up to thousands of devices, and we assume that they are not installed in the same physical area since the facility could be geographically distributed. We assume that the installed devices are legacy devices, and replacing them with new equipment is not possible. These types of devices operate with *Modbus/TCP* as the transmission protocol.

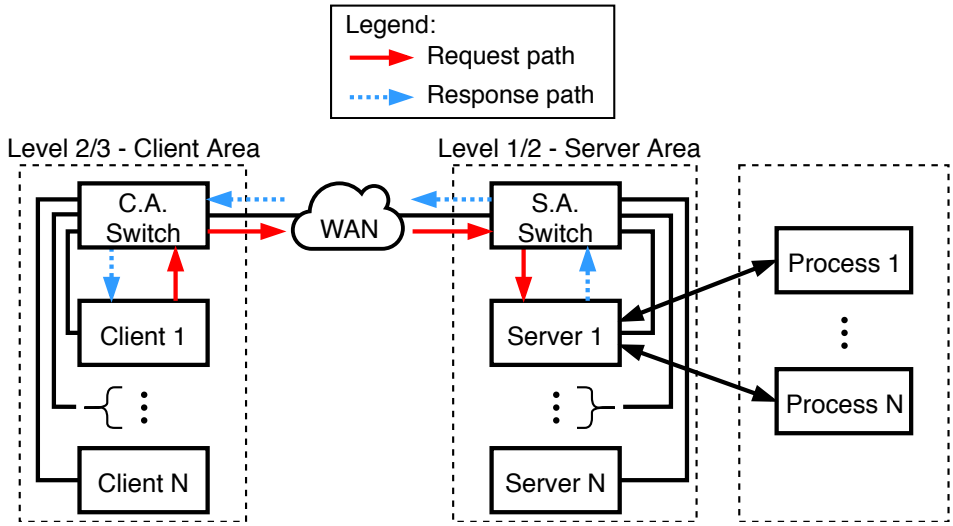


Figure 6.2: System model. We assume the system model to follow the rules provided by the Purdue Model. Continuous and dotted arrows describe an example of a *Modbus/TCP* transaction by showing the path followed by requests and responses packets

.

These legacy device may be updated with different information (such as firmware version, time clock), and they do not need to be synced in any way. Moreover, no internet connection is available because of the division of the manufacturing zone from the enterprise security zone. In fact, as explained in the Purdue Model, a DMZ separates the levels 3 and 4; thus, excluding the possibility of remote attacks. In addition to legacy devices, also modern ones could be installed in the facility and they have to be compatible with *Modbus/TCP* protocol. To conclude, we state that the considered system model describes an actual ICS without loss of generality.

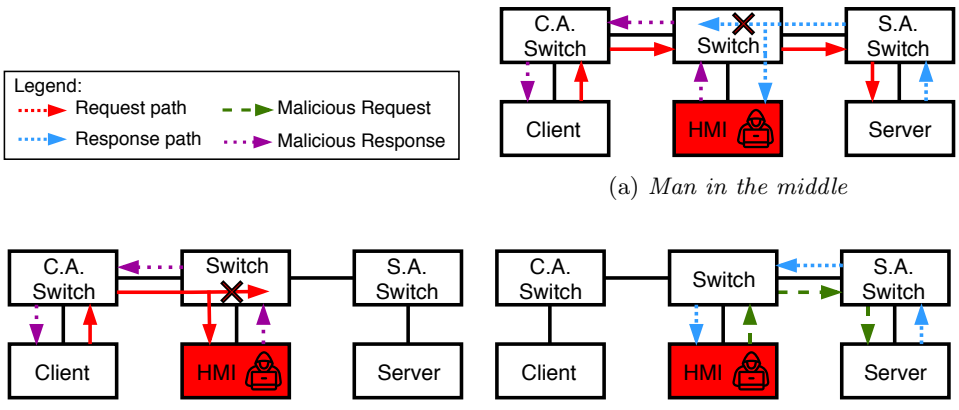
6.3.2 Threat model

We consider an adversary who aims to infiltrate the *Modbus/TCP* network and launches stealthy attacks without being detected. We assume that the adversary can passively monitor all ongoing *Modbus/TCP* transmissions, to acquire full knowledge of the ordinary transactions of the network. Furthermore, he is aware of the authentication method and he can observe the authentication messages that are being transmitted. There are two ways to acquire unauthorized access to the *Modbus/TCP* network: 1) compromise/control remotely an already plugged-in device, such as an HMI; 2) plug an external malicious device into one of the existing switches. We consider

that the adversary has access to all these pieces of information because he is an insider of the facility. More specifically, the insider can be any employee except a security officer. We expect companies that are security-aware to have basic physical protection around their PLCs (e.g., the PLC should be in a locked rack or a closed room). Under this condition, physical attacks are not among the possibilities available to the general employee, which only has the two above described ways to obtain unauthorized access.

In our threat model, we assume that an attacker has access to a compromised device (e.g., HMI) linked to a switch between the Client Area and the Server Area. Three types of attacks are possible: A) Man in the middle; B) Replay attack; C) Injection attack. A man in the middle attack allows an adversary to intercept packets along with the network and modify any field of the *Modbus/TCP* frame. Then, the replay attack allows an adversary to record network traffic and re-transmit it on the network in a later time. This attack lets the attacker to suspend the real transmission of packets and to send back recorded traffic, pretending to be authentic. In the injection attack, an adversary can forge and inject *Modbus/TCP* messages with arbitrary fields, such as Transaction ID, Function Code and Data.

We assume that the system is not leveraging other security techniques to prevent attacks such as man in the middle. These techniques, as described in [127], could be MAC address locking, static tables or ARPwatch.



(a) *Man in the middle*

(b) *Replay attack*

(c) *Injection attack*

Figure 6.3: Threat model. There are three possible attacks applicable to the *Modbus/TCP* network. All these attacks assume the attacker to be connected to a generic switch along the network (where *Modbus/TCP* is vulnerable)

6.4 TAMBUS

In this section we introduce *TAMBUS*. In Section 6.4.1 we introduce the new devices added by *TAMBUS* to a standard *Modbus/TCP* network. In Section 6.4.2 we describe how *TAMBUS* authenticates the previous devices and verify the integrity of their packets.

6.4.1 TAMBUS architecture

Figure 6.4 depicts the architecture of *TAMBUS* based on the system model presented in Figure 6.2. The main modules of *TAMBUS* are implemented in the Client Area (*Level 2*) and the Server Area (*Level 1*) of the industrial network. We also implemented the Process Level (*Level 0*) by using an emulated sensor on the real PLC server. We removed *Level 0* from Figure 6.4 as it is not necessary to understand the *TAMBUS* architecture.

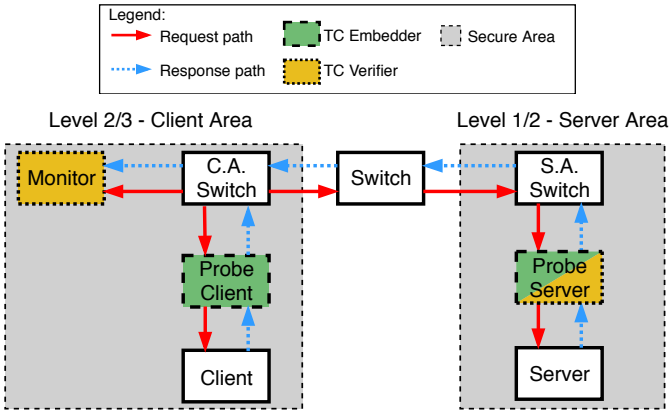


Figure 6.4: Illustration of *TAMBUS* architecture. This figure highlights the three new devices.

TAMBUS defines three innovative network nodes conceived to implement security in legacy industrial networks: the *Monitor*, the *Probe Client*, and the *Probe Server*. These nodes send and receive the *TAMBUS* code, which is necessary to authenticate packets and verify their integrity. The nodes can assume one or two roles among *TC Embedder* (*TAMBUS* code Embedder) and *TC Verifier* (*TAMBUS* code Verifier), which we define as follows:

- **TC Embedder** (with dashed borders in Figure 6.4): the i-th TC Embedder, with $i \in \{1, ..., n\}$, is appointed to embed the *TAMBUS* code in the received packets (transmitted from the device it is directly linked to) by using its own key K_i and counter C_i;

- **TC Verifier** (with dotted borders in Figure 6.4): it authenticates the i-th TC Embedder, with $i \in \{1, ..., n\}$. The TC Verifier checks the integrity of the packets, sent by the TC Embedder, by verifying the transmitted *TAMBUS* code. The verification procedure is done by using the key K_i and the counter C_i (that the TC Verifier already owns by system design) of the i-th TC Embedder.

We consider n as the total number of TC Embedders.

We present the three nodes in the following:

- **Monitor**: is a device that acts as a passive IDS. It is connected to the Client Area Switch by the mirroring port. In this way, the Monitor can access all packets from and to the Clients of its Client area. The Monitor assumes the role of TC Verifier for the Probe Server;
- **Probe Client**: is a device connected between a Client and the Client Area Switch. The Probe Client assumes the role of TC Embedder for the requests sent by the linked Client;
- **Probe Server**: is a device connected between a Server and the Server Area Switch. It can hold two roles: TC Verifier for all Probe Clients which communicate with its Server and TC Embedder for the responses sent by its Server.

We assume that all the connections inside the Client Area and Server Area (in light grey in Figure 6.4) are trusted and an adversary cannot compromise those devices. Moreover, we assume the new devices to be installed by trusted parties, to prevent potential tampering and compromises.

The *TAMBUS* architecture keeps the *TAMBUS* codes of requests and responses independent from each other. Consequently, if a specific scenario just needs to make the responses secure, the actual implementation of the system requires only the couple Monitor-Probe Server. Conversely, to make the requests secure requires only the couple Probe Client-Probe Server. For the reason that they are independent, it is possible to join the two scenarios by including all three devices. The joint scenario makes both the requests and the responses secure and all three the devices are included in the system.

6.4.2 *TAMBUS* code generation and verification scheme

Similar to [144], we assume that a key (K) is pre-shared, and stored in a trusted platform module (TPM) [113]. It is important to note that the TPM is required only on the new devices (the TC embedders and TC verifiers). The TPM is not required to be available on the legacy devices of the

network, since these devices do not store any keys in the *TAMBUS* architecture. An example of cheap new device may be a Raspberry Pi equipped with a TPM module. TPM module is vulnerable to the TPM-fail attack. However, we do not consider this attack a possible threat since it requires or a fast network or physical access to the TPM. About fast networks, this is not the case for the involved Modbus network. About physical access we already state in Section 6.3 that the Client area should be restricted to access. Moreover, over the last months, firmware resistant to TPM-fail attack have been released [1].

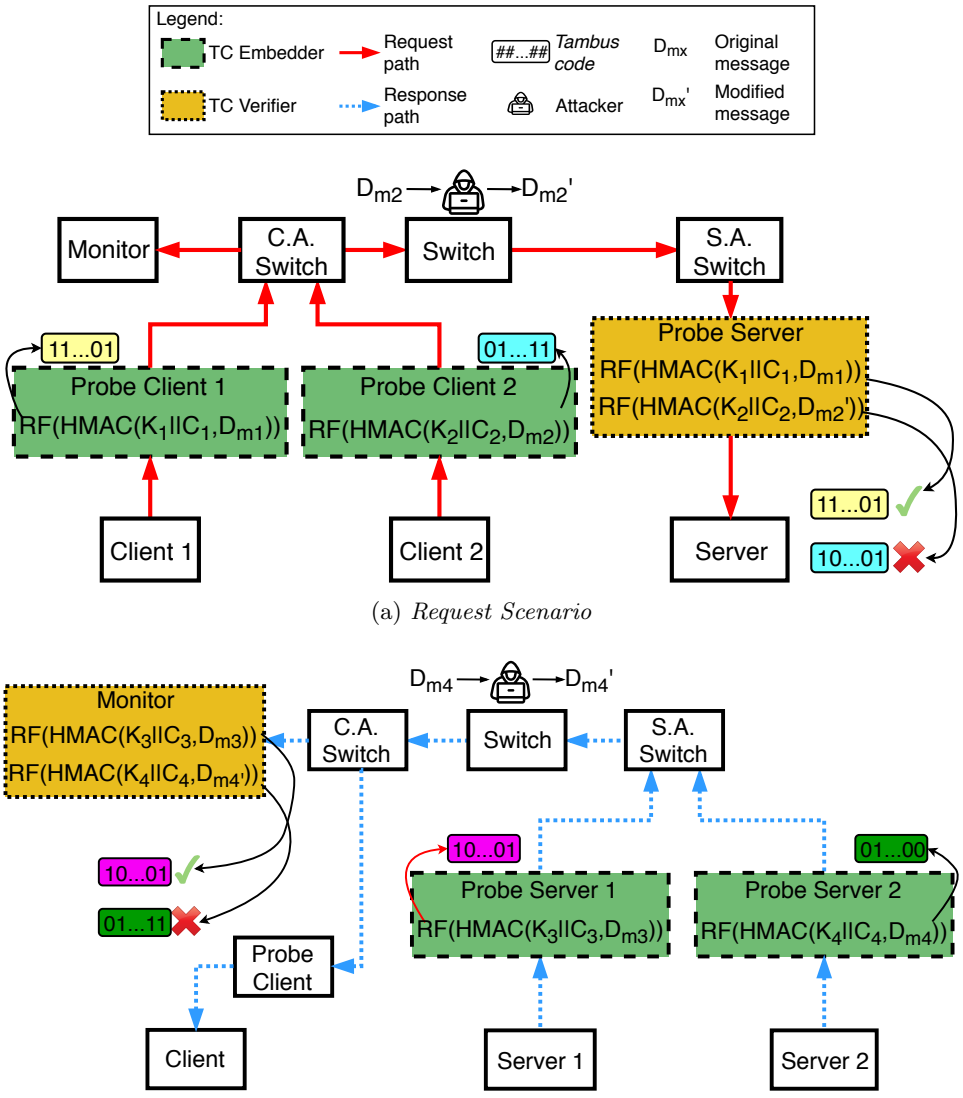
A key is always shared between two devices: a TC Embedder and the corresponding TC Verifier. The keys cannot be configured during the production of the devices, but they require to be installed by authorized parties during the securing process of the facility.

In Figure 6.5, we depict how the new devices introduced by *TAMBUS* interact during the Request scenario (Figure 6.5a) and the Response scenario (Figure 6.5b). We used a different pattern for the border of the rectangle only for the devices that play an active role in their specific scenario: dashed borders represents the TC Embedders, instead dotted borders are for the TC Verifiers. Note that a device playing one role in a specific scenario can play the opposite role or even no role in the opposite scenario. Inside every TC Embedder and TC Verifier device, we reported the elements involved in the *TAMBUS* code Generation and Verification: keys (K_x), counters (C_x), and processed packets (D_{mx}). For instance, in Figure 6.5a (Request Scenario) the Probe Server contains the same three elements used also by the Probe Client 1 to generate the *TAMBUS* code. The Probe Server preshares the key and the counter with the Probe Client 1. The counters are always synchronized between the devices, since their increase always happen together. The request packet, thanks which the *TAMBUS* code is transmitted, transits form the Probe Client 1 to the Probe Server using the normal *Modbus/TCP* network. We explain the implementations of the proposed covert channels in Section 6.5.

In the following we describe the *TAMBUS* code generation and verification scheme step by step:

- **Packet generation:** A Sender Device generates a *Modbus/TCP* packet and sends it to a specific Destination Device;
- ***TAMBUS* code generation:** The TC Embedder intercepts the packet and generates the corresponding *TAMBUS* code using its own

[1] Refer to https://tpm.fail/ for details.



(a) *Request Scenario*

(b) *Response Scenario*

Figure 6.5: Illustration of *TAMBUS* architecture showing keys and counters management for the Request and Response scenarios.

key (K) and counter (C). After the *TAMBUS* code generation, the local counter C is incremented. Once the TC Embedder creates the *TAMBUS* code, it sends the packet and the respective *TAMBUS* code to the original Destination Device: the first using the standard *Modbus/TCP* channel and the second by using the specific implemented *TAMBUS* covert channel;

- ***TAMBUS* code Verification:** The TC Verifier receives the packet (by the standard *Modbus/TCP* channel) and the corresponding *TAMBUS* code (by the *TAMBUS* covert channel). The TC Verifier recognizes which Sender Device generated the message by using its static IP (provided by the TCP level of the *Modbus/TCP* protocol). Using a preexisting map between a Sender Device and the corresponding TC Embedder, the TC Verifier recognizes which TC Embedder generated the *TAMBUS* code. Hence, the TC Verifier recomputes the *TAMBUS* code by using the local copies of the key (K) and the counter (C) of the recognized TC Embedder. After the verification process, the local copy of the counter C is incremented to stay synchronized with the counter of the TC Embedder.

Note that the operation of *TAMBUS* is not tied in any way to the use of static IP. It can work if the IPs are dynamic thanks to a simple shared table containing all the dynamic associations. However, the latter scenario is considered extremely rare, since ICSs widely use static IPs.

TAMBUS leverages two covert channels, storage-based and timing-based, to send the a *TAMBUS* code for both *Modbus/TCP* requests and responses. *TAMBUS* is fully functional with the use of storage-based covert channel. However, in this work we also investigated a solution offered by the timing-based method, as this can sometimes be useful for some particular use cases. These covert channels can be implemented singularly or jointly. Likewise, the packet verification procedures for requests and responses are independent. Consequently, according to specific needs, they can be implemented both or singularly. Considering only a single packet, *TAMBUS* is able to guarantee the transmitter to be trusted with a certain confidence (see Section 6.7 for a detailed explanation). This property can be successfully exploited to create a secure *Modbus/TCP* protocol, in that *Modbus/TCP* is used in situation where polling request are continuously performed to read data from PLC servers. In this contest, the probability of an attacker to successfully modify a large amount of packets decreases with the number of packets to modify. We will carefully describe the security details of *TAMBUS* in the next sections.

Here below, we describe the parameters used by *TAMBUS* to define the *TAMBUS* code generation and verification scheme. We consider n as the total number of TC Embedders. The parameters used by *TAMBUS* to generate the *TAMBUS* code are summarized as follows:

- The key K_i with $i \in \{1, ..., n\}$ is a pre-shared key saved in the i-th TC Embedder. A copy of the same key is also present on the corresponding TC Verifier;
- The local counter C_i with $i \in \{1, ..., n\}$ is an incremental value kept independently on each device. The counter is initialized and updated using the same rules on every pair of TC Embedder and TC Verifier. The counter is incremented each time the TC Embedder and the TC Verifier respectively generate and verify a *TAMBUS* code.

The local counters are independently updated by the TC Embedder and TC Verifier, so a de-synchronization of counters is possible. The counter size may vary depending on the number of operation per second. However, a 64-bit integer should always be more than enough to avoid security issues due to counter restart. More details are provided in the next sections where we give a re-synchronization procedure.

6.4.2.1 *TAMBUS* code generation

We define D as the TCP payload (corresponding to the entire *Modbus/TCP* frame). Starting from D we define a generic D_m, a modified payload generated from D. The specific rules used to generate D_m will be introduced in Section 6.5. We compute the HMAC, which generate an authentication message A_m based on K_i, C_i, but also on the message content D_m:

$$A_m = \text{HMAC}(K_i||C_i, D_m),$$

where HMAC function is the Hash-based message authentication code algorithm and || is the bit concatenation operator. Implementations of *TAMBUS* are free to use whichever hashing algorithm for HMAC and sizes of keys that are considered strong enough to achieve the desired system security.

A_m is usually hundreds of bits long but the *TAMBUS* codes used by *TAMBUS* require a fixed and low amount of bits. We define a general function that reduces the size of A_m to the desired number of bits. We name this function Reducing Function (RF) and we define it as follows:

$$Tambus\_code = \text{RF}(A_m).$$

The output of this function is the *TAMBUS* code, which is transmitted through the covert channel. For the specific definitions of the variable D_m and the function RF we refer to the Section 6.5, in which we detailed explain the used covert channels.

6.4.3 Initialization and counter Re-Synchronization

The initialization procedure for *TAMBUS* is quite simple. The *TAMBUS* devices must be connected to the original *Modbus/TCP* network as explained in Figure 6.4 – this is the only scenario in which *TAMBUS* requires to disconnect the Client and the Server from the working network. The TC Embedder and TC Verifier must be already initialized with the respective keys and the counters must be aligned. If the TC Verifier is an already installed Monitor, a new key must be added in order to allow the verification process for the new TC Embedder. Moreover, the counters must be initialized with the same random number on both the TC Embedder and the TC Verifier. After these steps, *TAMBUS* is ready to embed and verify *TAMBUS* codes.

We also consider possible situations when the counters (of a pair TC Embedder/TC Verifier) may go out of synchronization. This may happen in two different scenarios: first when an attacker is introducing or removing *Modbus/TCP* packets, second when an error occurs on the network or on *TAMBUS* devices (e.g., unexpected power outages). The first scenario consists of the usual way of operating of *TAMBUS*. When the attacker injects or replies one or more packets, *TAMBUS* goes out of sync. The consequence is that an alarm is raised, and the attack is correctly detected and notified to the security staff. At this point, the threat is handled by security experts. They will manage the attack in a way that protects the integrity and security of the ICS. About the second scenario, we precise that *Modbus/TCP* is based on TCP, which is a connection-oriented protocol. Consequently, the TC Embedders and TC Verifiers (which have access to all the packets passing through the network) are able to identify if *Modbus/TCP* failed to send a packet and so they can autonomously realign their counter if needed. Therefore, we can state that if *TAMBUS* detects a de-synchronization, it can assume that an attack is ongoing. Indeed, de-synchronization due to random factors are almost completely excluded thanks to the TCP protocol.

In any case, if it is necessary to realign the counters of *TAMBUS*, it is not possible to reset the devices by shutting them down, because ICSs require continuous availability (shutdowns are planned months or years in advance).

To realign them, it is possible to re-synchronize the counters by using predefined *Modbus/TCP* requests (irrelevant for non-*TAMBUS* devices). To accomplish this safely, each device has an additional maintenance key and a list of infinite maintenance counters (unique for each device) that are generated thanks to a dedicated hashing function. These specific requests indicate to the involved devices to use the *next maintenance counter* as the counter used by *TAMBUS*. However, the procedure just mentioned is outside the scope of this work. The reader should also note that even if *TAMBUS* devices are de-synchronized, the traffic is not blocked in any way and the system continues functioning normally. This transparent behavior by *TAMBUS* is necessary, as stopping packets could produce even more serious problems than an attack itself.

6.5 Covert channels

In this section, we describe the overall functioning of the proposed covert channels applied to a *Modbus/TCP* network. We refer to the next chapter for the individual implementations of the covert channels. In Section 6.5.1 we describe the storage-based covert channel, while in Section 6.5.2 the timing-based covert channel.

6.5.1 Storage-based covert channel

The storage-based covert channel exploits the *Modbus/TCP* frame to transmit information between two devices. *TAMBUS* works using a minimum of 1 dedicated bit of the *Modbus/TCP* frame as covert channel. In Figure 6.6 we show a scheme of a storage-based covert channel. The depicted scheme describes the procedure explained in 6.6.1, but it could be generalized by considering a different portion of the *Modbus/TCP* frame. The TC Embedder intercepts the packet P and replaces one or more bits of the TCP payload with a *TAMBUS* code. To generate the *TAMBUS* code, the TC Embedder generates A_m and considers D_m as D minus the bits reserved for the covert channel. After that, by using a reducing function, it obtains the *TAMBUS* code. Finally, the TC Embedder embeds the *TAMBUS* code into the reserved bits for the covert channel and sends it to the Destination Device. We call MP the modified packet.

The packet reaches the TC Verifier, which verifies if the *TAMBUS* code is authentic. To infer the *TAMBUS* code, the TC Verifier generates A_m by using the local copy of the key K and the counter C, and D_m. The TC Verifier considers D_m as D minus the bits reserved for the covert channel

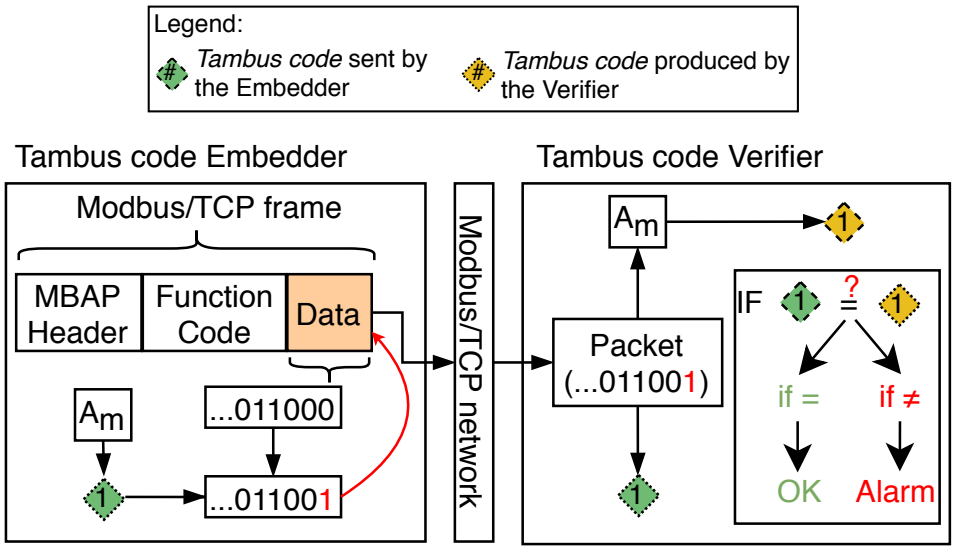


Figure 6.6: Storage-based covert channel authentication procedure.

(exactly the same considered by the TC Embedder). After the generation of A_m, the TC Verifier calculates the *TAMBUS* code by using the same reducing function used by the TC Embedder. Finally, it compares the calculated *TAMBUS* code with the one embedded in MP and checks if they are equal. If it detects a mismatch, it raises an alarm to inform the facility staff of a possible ongoing malicious attack.

6.5.2 Timing-based covert channel

In this scenario, the TC Embedder intercepts the packet P and it transmits a *TAMBUS* code by modulating timing events. In Figure 6.7 we show a scheme of a timing-based covert channel. To generate the *TAMBUS* code, the TC Embedder calculates A_m and it considers D_m as the entire payload D. After that, by using a reducing function, it generates the *TAMBUS* code as a 1-bit version of A_m, which only possible values are 0 and 1. Note that the reducing function is not the same used on the storage-based covert channel, since the timing-based and storage-based are completely independent and do not share information.

The TC Embedder, through the timing-based covert channel, transmits a *Tambus code = 1* by waiting a certain amount of time before sending the packet P to its Destination Device. Conversely, it transmits a *Tambus code = 0* by re-transmitting it without waiting.

We define the waiting time as WT and its value is empirically defined based on the network characteristics (transmission medium, number of con-

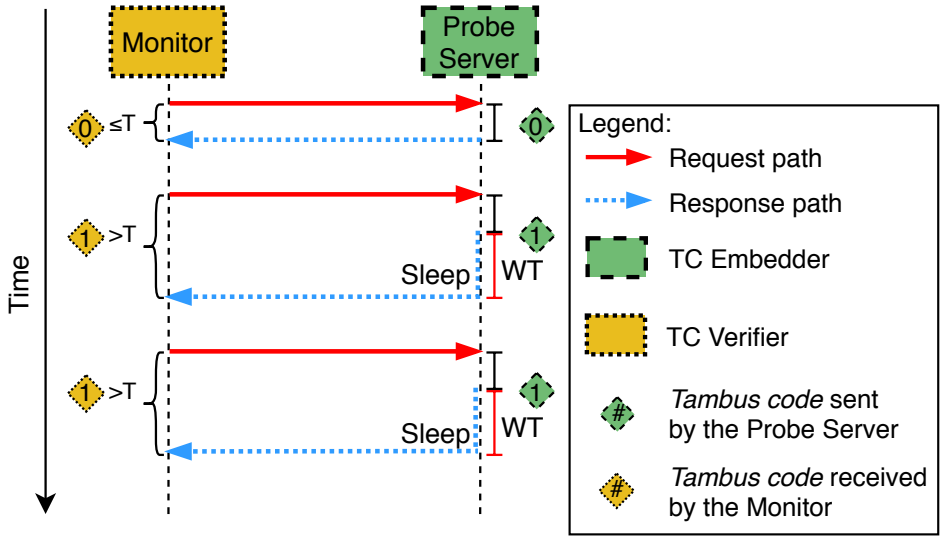


Figure 6.7: Timing-based covert channel authentication procedure.

nected devices, distance between the devices). To summarize, *TAMBUS* is configured as follows:

- *TAMBUS* code = 0 → no waiting;
- *TAMBUS* code = 1 → waiting time (WT).

We empirically define the value of WT by balancing two factors: it has to be big enough to minimize the detection of wrong *TAMBUS* codes, but it has also to be small enough to not introduce a significant delay on the network.

The TC Verifier, to recover the *TAMBUS* code transmitted through the covert channel, registers two different timestamps. The timestamps registered will be different based on the type of authentication (responses or requests). We call Δ the difference between the two timestamps and we define it as follows:

$$\Delta = T_2 - T_1. \tag{6.1}$$

Where T_1 and T_2 are respectively the first and the second timestamp registered by the TC Verifier. We set a threshold TH that the TC Verifier uses to infer the *TAMBUS* code transmitted by the TC Embedder. The TC

Verifier computes the *TAMBUS* code as follows:

$$Tambus\ code = \begin{cases} 0 & \text{if } \Delta \leq T \\ 1 & \text{if } \Delta > T. \end{cases}$$

The threshold TH is empirically defined by considering several parameters, such as the minimum and maximum Δ registered on the network. Theoretically, the value of TH should be inside a specific range: between the maximum possible Δ registered on packets with *TAMBUS* code = 0 (no waiting time) and the minimum possible Δ registered on packets with *TAMBUS* code = 1 (waiting time = ST). We discuss in detail how to set the value of WT and TH in the Section 6.8.2.1.

Finally, the TC Verifier generates A_m by using the key K and counter C related to the TC Embedder, and considers D_m as the entire payload D of the packet P to verify. After the generation of A_m, the TC Verifier calculates the *TAMBUS* code by using the same RF used by the TC Embedder. Finally, it compares the calculated *TAMBUS* code with the one transmitted with the covert channel (obtained by comparing Δ with the threshold TH) and checks if they are equal. If it detects a mismatch, it raises an alarm to inform the facility staff of a possible ongoing malicious attack.

6.6 Implementation

In this section, we describe how we applied the storage-based and timing-based covert channels. We separately show how these two covert channels operate for the response packets (Section 6.6.1) and the request packets (Section 6.6.2).

6.6.1 Covert channels for responses

6.6.1.1 Storage-based covert channel for responses

This covert channel exploits a particular property of the floating-point values to transmit one (or even more) bit using the normal *Modbus/TCP* channel. For this reason, in this scenario we make the following assumption: the type of the last two registers contained in the response packet must be floating-point (*Modbus/TCP* saves 32-bit floating-point using two 16-bit registers). The memory, on *Modbus/TCP* servers, is organized in big-endian [106] and it is possible to know in a deterministic way that the second register contains the LSB. This assumption is acceptable because in *Modbus/TCP* is realistic to assume that a response packet contains at least a floating-point number.

This happens because the PLC Servers collect data from industrial sensors, so it is ordinary that the PLCs save them as floating-point numbers. Moreover, despite our assumption, if a floating-point value is not present, then we assert that our method is still usable, but it would be more intrusive and its level of effective applicability may change from case to case (i.e., modifying integers). However, this scenario is outside the scope of this covert channel method. If any data type can be modified, the Protocol Identifier can be used to transfer data (exactly as the storage-based covert channel for requests 6.6.2.1).

In this scenario, the Probe Server as the TC Embedder intercepts the response packets and modifies the LSB, of the last floating-point value of these packets, to send a 1-bit *TAMBUS* code to the Monitor (i.e., the TC Verifier). In this context, we define the parameter D_m, used to generate A_m, as the complete payload D of the response packet minus the last bit of D itself. The reader should note that (as explained in 6.2 and considering our assumption) the last bit of a *Modbus/TCP* response is precisely the LSB of the last floating-point value contained in it (that is the value we want to exploit for our covert channel).

In this scenario, *TAMBUS* allows to transmit one bit through the covert channel. For this reason, the *TAMBUS* code must be 1-bit long and we defined the *Reducing_Function* to create a condensed 1-bit version of A_m. Hence, the Probe Server (as TC Embedder) replaces the last bit of the intercepted response with the *TAMBUS* code returned by the *Reducing_Function* (number ① in Figure 6.8a). Note that the replaced bit is the one that was discarded when creating D_m from D. We call MR the modified response packet.

The Probe Server sends back MR to the Client. When the packet reaches the C.A. Switch, the Monitor receives a copy of MR and checks if the *TAMBUS* code is valid (number ② in Figure 6.8a). To infer the *TAMBUS* code, the Monitor generates A_m by using the key K, the counter C and P_m. The Monitor considers D_m as the payload D of MR minus the last bit of MR itself (exactly as we defined earlier D_m defined from D). At this point, the D_m defined on the Monitor (i.e., the TC Verifier) must be exactly the same D_m defined on the Probe Server (i.e., the TC Embedder). Consequently, the generated A_m on the Monitor must be exactly the same of A_m contained in the Probe Server. Finally, the Monitor computes the *TAMBUS* code using the same *Reducing_Function* used by the Probe Server, and compares it with the corresponding *TAMBUS* code sent by the Probe Server in the last bit of MR. If the computed *TAMBUS* code does not match with the last bit

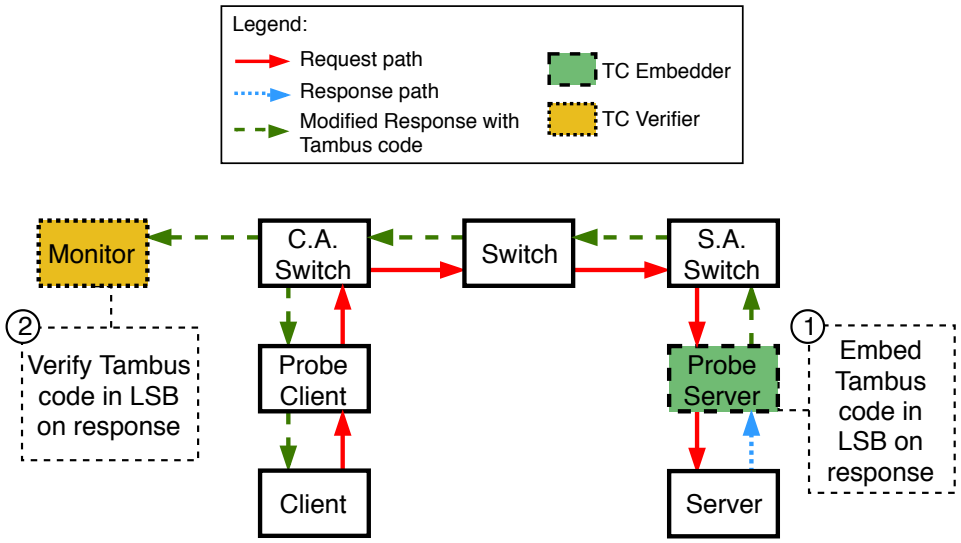
of MR, the Monitor raises an alarm to inform the facility staff of a possible ongoing malicious attack.

We propose the LSB as storage-based covert channel for responses since it has two main advantages compared to the implementation used on requests, explained in 6.6.2.1. First, *TAMBUS* introduces an error so small in the modified floating-point that it can be ignored by the applications on the Client. This error is equal to the machine epsilon relative to the modified floating-point number – i.e., $1.19 \times 10^{-5}\%$ in the case of a common floating-point 32 bits. Second, a change in the LSB is not directly noticeable by a possible attacker, and therefore it will be more difficult for him to understand whether a system is protected or not by *TAMBUS*. *TAMBUS* cannot use the LSB in three cases: when any floating-point value is available (and it is not possible modifying any data); when a response does not include return values (e.g, in write requests); when even the introduction of a small machine epsilon error can not be tolerated by the system. In all these cases, *TAMBUS* makes the responses secure by using the Protocol Identifier (exactly as explained in 6.6.2.1).

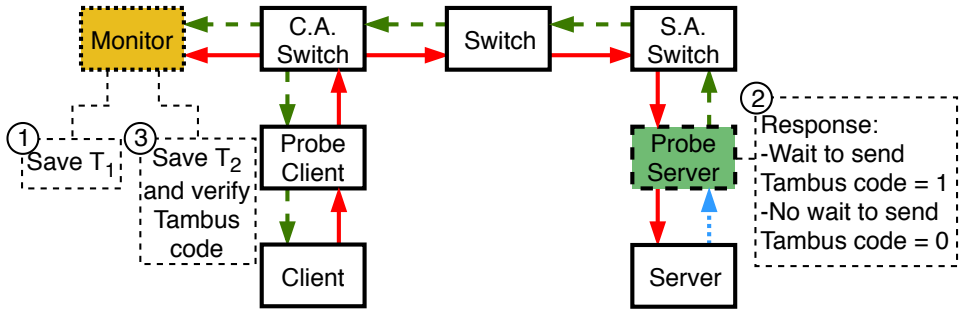
6.6.1.2 Timing-based covert channel for responses

The Monitor saves the arrival timestamp of a request in the C.A. Switch (number ① in Figure 6.8b). The Probe Server intercepts the response packet (number ② in Figure 6.8b). It calculates A_m and generates the *TAMBUS* code from that. Then, it transmits the *TAMBUS* code to the Monitor by following the rules explained in 6.5.2 (hence waiting or non waiting before sending back the packet). Then, the Monitor saves the arrival timestamp of the response in the C.A. Switch (number ③ in Figure 6.8b). To recover the *TAMBUS* code transmitted through the covert channel, the Monitor computes the difference between the two collected timestamps. Therefore, in this scenario we define Δ based on Equation (6.1). The Monitor uses as T_1 and T_2 the arrival collected timestamps of the request and the response respectively.

These operations are possible because the Monitor, thanks to the mirroring port, can observe all the traffic passing in the C.A. Switch. Note that the monitor does not receive a packet at the same time it passes through the C.A. Switch. This happens because the packet is cloned thanks to the mirroring port and it has to reach physically the Monitor. This phenomenon could bring some delay in the measurement of Δ, but it does not affect the process.



(a) *Storage-based covert channel on responses*



(b) *Timing-based covert channel on responses*

Figure 6.8: Implementation of the covert channels for the response packets.

Finally, the Monitor compares the *TAMBUS* code generated from A_m and the one recovered from the covert channel to check if they are equal.

6.6.2 Covert channels for requests

In this section, we describe how *TAMBUS* exploits the storage-based and timing-based covert channels to send *TAMBUS* codes in the request packets.

6.6.2.1 Storage-based covert channel for requests

In *Modbus/TCP*, two different types of requests exist: read and write. Read requests contain the address and the number of the registers, or coils, requested by the Client. Conversely, write requests contain the values to write in the Server and the memory address location in which the Server saves them. Hence, read requests do not contain floating-point values and *TAMBUS* cannot modify the values of write requests because they are critical (i.e., directly connected with the industrial process). For this reason, the request storage-based covert channel, differently from the response storage-based version, can not take advantage a floating-point value to hide a *TAMBUS* code.

We considered different possibilities and evaluated which was the best field in which *TAMBUS* could transmit the *TAMBUS* code. It is possible leveraging the following fields: address, function code, word count, protocol identifier and the payload which contains data. Several of them (address, function code, data) introduced assumptions on the *TAMBUS* architecture and consequently we discarded them. The word count field is exploitable without introducing assumptions, but it is available only on read and "multiple write" requests. The only modifiable field, that is in common between all type of *Modbus/TCP* requests and does not introduce assumptions, is the *protocol identifier*. This field is 16-bit long and it is filled entirely with zeros to state that the packet is a *Modbus/TCP* frame. *TAMBUS* use this field to transmit the *TAMBUS* code and restores it to 16-bits of zeros before the request arrives at the Server.

In this scenario, the Probe Client, as TC Embedder, intercepts the response packets and modifies the protocol identifier field to send a 16-bit *TAMBUS* code to the Probe Server (i.e., the TC Verifier). In this context, we define the parameter P_m, used to generate A_m, as the complete payload P of the response minus the 16 bits of the protocol identifier field.

Thus, *TAMBUS* allows to transmit 16 bits through the covert channel. For this reason, the *TAMBUS* code must be 16 bits long and we defined the *Reducing_Function* to create a condensed 16-bits version of A_m. Hence, the

Probe Client (as TC Embedder) replaces the bits of the protocol identifier of the intercepted response with the *TAMBUS* code returned by the *Reducing_Function* (number ① in Figure 6.9a). Note that the replaced bits are the ones discarded when creating D_m from D. We call MR the modified request packet.

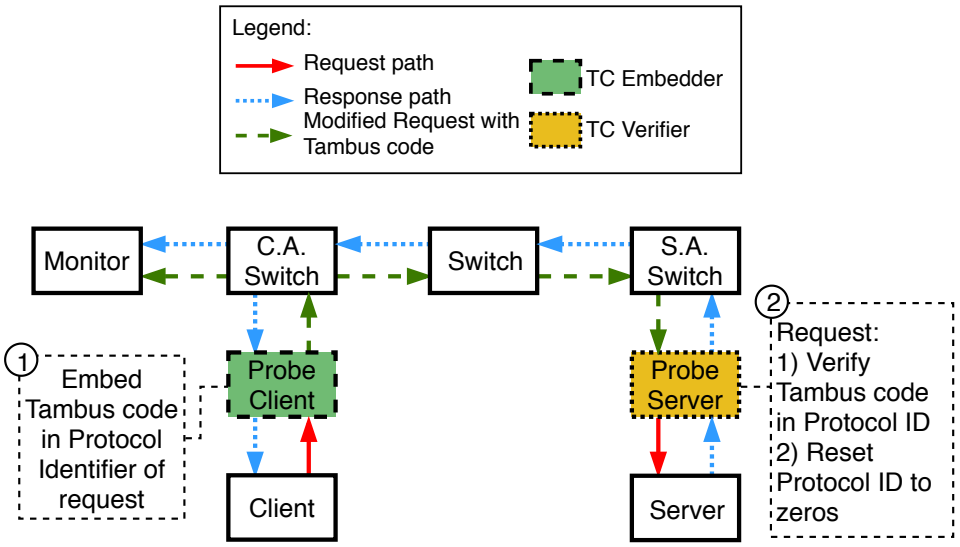
The Probe Client sends MR to the Server. The packet is intercepted by the Probe Server, that checks if the *TAMBUS* code is valid (number ② in Figure 6.9a). To infer the *TAMBUS* code, the Probe Server generates A_m by using the key K, the counter C and D_m. The Probe Server considers D_m as the modified payload MR minus the 16 bits of the protocol identifier (exactly as we defined earlier D_m starting from D). At this point, the D_m defined on the Probe Server (i.e., the TC Verifier) must be exactly the same D_m defined on the Probe Client (i.e., the TC Embedder). Consequently, the generated A_m on the Probe Server must be exactly the same of A_m contained in the Probe Client. Finally, the Probe Server computes the *TAMBUS* code using the same *Reducing_Function* used by the Probe Client, and compares it with the corresponding *TAMBUS* code sent by the Probe Server in the protocol identifier of MR. If the computed *TAMBUS* code does not match with the protocol identifier of MR, the Probe Server raises an alarm to inform the facility staff of a possible ongoing malicious attack. After the Verification process, the Probe Server restores all the bits of the protocol identifier to 16 zeros, to ensure the correctness of the *Modbus/TCP* frame. In fact, the Server would reject the packet if the protocol identifier is not set to 16 zeros.

6.6.2.2 Timing-based covert channel for requests

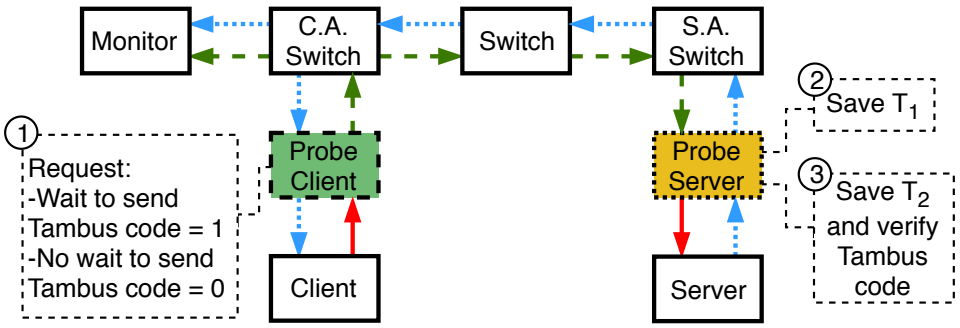
Firstly, the Probe Client intercepts the request packet (number ① in Figure 6.9b). It calculates A_m and generates the *TAMBUS* code from that. Then, it transmits the *TAMBUS* code by following the rules explained in 6.5.2.

The Probe Server, to obtain information to generate the *TAMBUS* code, registers two timestamps: the arrival timestamp of the request number X (number ② in Figure 6.9b) and the arrival timestamp of request number $X + 1$ (number ③ in Figure 6.9b). We define Δ based on Equation (6.1) and using as T_1 and T_2 respectively the arrival timestamps of the request and the response in the Probe Server.

The value of Δ is compared to the threshold TH and the comparison determines the value of the *TAMBUS* code.



(a) *Storage-based covert channel on requests*



(b) *Timing-based covert channel on requests*

Figure 6.9: Implementation of the covert channels for the request packets.

6.6.3 Possible options for *TAMBUS*

To reduce the number of new devices and to maximize the cost-benefit ratio, it is possible to choose between different variants of *TAMBUS*, based on the necessities of the facility. For example, we can limit the use of *TAMBUS* only to those clients allowed to send write requests (e.g., "Write Single Coil", "Write Single Holding Register"), so discarding merely visualization HMIs and clients used only to collect data. A company could target only write requests because they are the most dangerous ones since an unauthorized request could turn on/off machinery and cause disastrous consequences. Turning to the use of *TAMBUS* only for responses, we have two different options for the TC Verifier: using a Monitor or using a Probe Client. The work carried out by the Monitor could be performed, with any modification, by each Probe Client. Using a Monitor is preferable because it can be the TC Verifier for all the Clients attached to the C.A. Switch; on the other hand, using a Probe Client would require a new device for each installed Client. Moreover, the Monitor is a not intrusive device, and it behaves as a passive IDS, while the Probe Client requires adding a new device in the middle of already existing cable connections. The use of a Probe Client could be recommendable if *TAMBUS* is already implemented for requests on the Client.

6.7 Security Discussion

Compared to existing *Modbus/TCP* authentication methods, *TAMBUS* provides for the first time two essential features together in the same method: first, no changes are made to the *Modbus/TCP* protocol and second, the method is implemented in a provable secure design (while other proposed methods only provide the first features in security by obscurity design). *TAMBUS* can detect attacks that modify packets or that create new ones (man in the middle, replay, and injection) with an high degree of confidence. Moreover, the features provided by *TAMBUS* are independent from *Modbus/TCP*. Hence, *TAMBUS* could theoretically be applied to all unencrypted protocols: the assumption to have a Float value or an unused header field remains for the storage-based covert channel, but there are no limitations for the timing-based version.

Detecting Man in the middle: in the man in the middle attack, the adversary has access to one of the non-secure switches on the network. It intercepts, modifies, and sends again the packet to its Destination Device. Since the adversary has no access to the key (K) to generate the *TAMBUS*

code, by modifying at least one bit of the packet, it invalidates the *TAMBUS* code transmitted through the covert channel.

Detecting Replay Attack: an attacker has infiltrated into the network and is interrupting legitimate traffic, attempting to send previously transmitted messages. Its hope is to transmit traffic that embed a valid storage-based *TAMBUS* code. However, such attempt will be detected by *TAMBUS* due to the use of the counter C. This attack causes the de-synchronization between the counters of the TC Embedder and TC Verifier, and after the threat resolution, the staff has to initialize them again. We should also consider the possibility that the counter reaches the maximum values and restarts from zero. However, this possibility does not apply to realistic cases of *Modbus/TCP*. By using a 64-bit counter and considering an extremely high frequency of operations per second (e.g. 1000), it results that the time taken for the counter to return to zero would be far greater than the useful life of the PLC.

Detecting Injection Attack: an attacker has infiltrated into the network and it transmits fake requests to the Server. When the Probe Server receives the fake request, it recognizes that the request comes from a client which is not part of the group of *legitimate clients*. Consequently, the Probe Server raises an alarm. We do not consider the injection of fake responses since a Client accepts responses only if it made the corresponding requests. This attack causes the de-synchronization between the counters of the TC Embedder and TC Verifier, and after the threat resolution, the staff has to initialize them again.

TAMBUS allows adding to *Modbus/TCP* an additional security level able to detect attacks with high statistical confidence. *TAMBUS* cannot, therefore, be considered a solution in specific cases where it is necessary to provide the guarantee that not even a single bit in a long flow of packets has been modified. For instance, it is not the case in power or nuclear plants, where a single modification in a packet could lead to explosions and serious damages to the infrastructure. *TAMBUS* can be successfully used in all those cases in which an attack would require the modification of multiple messages to seriously damage a system. For instance, facilities with high security requirements would not use *TAMBUS*, but they would rather prefer more sophisticated and expensive protections. Therefore, *TAMBUS* is advisable to companies that want to increase the security level of their infrastructures while keeping the costs low.

For the sake of clarity, we remind the reader that *Modbus/TCP* continuously performs requests at regular intervals since it is used as a polling protocol. This implies that *TAMBUS* continually add *TAMBUS* codes

to the transmitted *Modbus/TCP* responses. If an attacker aims to supply fake measurements of a sensor to a Client, it should modify a stream of several packets. The probability of guessing a sequence of n packet is $P(response\_sequence) = \frac{1}{2^n}$. However, according to the needs of the facility, *TAMBUS* security may be improved by using additional available bits in the *Modbus/TCP* frame. For instance, it is possible to use more bits than only the single LSB, since in several circumstances the sensibility of sensors do not go beyond 10^{-3}. Moreover, if the security is a main concern, it is also possible using the 16 bits of the protocol identifier as in the request scenario described below. Therefore, an attack that aims to continuously modify the level of a sensor is not practicable in a real scenario.

The assumption of a stream of packets cannot be used for the *Modbus/TCP* requests. Indeed, even a single malicious *Modbus/TCP* request may have catastrophic effects on a facility. However, the number of bits available for a *TAMBUS* code in a request is 16, this means that an attacker has a possibility on 65.636 to guess the right *TAMBUS* code for a single packet. It is possible to further decrease the chances of a successful attack on the requests by adding the timing-based covert channel. In the joined scenario the probability is one more than 4 billions of attempts. *TAMBUS* is resistant to different types of attacks directed towards the devices added to the *Modbus/TCP* system. Indeed, the devices introduced in the system do not need an IP address on the *Modbus/TCP* network. This feature, combined with the prerequisite that the additional devices are not physically reachable, since they are within a secure area, makes really difficult to attack *TAMBUS* with both direct and remote attacks.

6.8 Evaluation

In this section, we provide the results of our experiments. We first describe in Section 6.8.1 the common approach used in the experiments, while in Section 6.8.2 we provide the actual results collected on our testbed.

6.8.1 Experiments on testbed

We evaluated the covert channels by running several experiments with the testbed in Figure B.2. During our tests, the Client is continuously sending requests to the Server. When it receives a response, it waits 50 ms before performing the next request. Since the scope of our experiments was not to optimize the implemented solution in term of performance, but to demonstrate the effectiveness of our method, we implemented *TAMBUS* us-

ing the Python language. During our experiments, we intercepted packets using *Linux iptables* and *netFilterQueues*, while we sniffed packets using the Python module *scapy*. To determine the impact of *TAMBUS*, we measured separately the time necessary to route and to modify a packet on 50.000 *Modbus/TCP* transactions. The collected measurements show that the Python implementation takes on average 19 ms to filter, intercept, modify, and resend a packet. This amount of time is composed by: 8 ms to intercept and filter the packet, 1 ms to modify the packet (i.e., the implementation of *TAMBUS*), and 10 ms to manage the packet payload (e.g., type conversions). Therefore, measurements show that *TAMBUS*, by itself, introduces a minimal overhead in a normal *Modbus/TCP* protocol. To implement the HMAC function we used a SHA-256 and a 32 byte key. The time of 1 ms spent to modify a packet is really important, since it allows us to compare *TAMBUS* with alternative methods. Indeed, the remaining 18 ms would be common to all possible alternatives, and only the time spent to elaborate the original packet is meaningful. Moreover, an optimized version of our script (e.g., based on C++), would certainly decrease the total execution time of our method, thus providing performance even better of the one presented in this section.

6.8.2 Covert Channels results and performance analysis

6.8.2.1 Evaluation of Timing-based Covert Channel

We evaluate the timing-based covert channel in two different scenarios. In the first one we considered the feasibility of implementing this covert channel on the proposed testbed. In the second one, we investigate the impact of external network traffic on the performance of the covert channel. In both scenarios, we evaluate the timing-based covert channel by performing two different experiments: one for requests and one for responses.

In the first scenario, we define the threshold TH and the waiting time WT based on the values of Δ recorded in the testbed. To define Δ we use the Equation (6.1). Note that, different values of TH and WT bring to different performances on the timing-based covert channel. For this reason, we conducted the experiments using two pairs TH/ST (respectively TH_1/ST_1 and TH_2/ST_2) to highlight the impact of such parameters. Note that, we calculate two thresholds for each experiment – two for the requests experiment and two for the responses experiment.

In both the experiments, we followed the below procedure:

Table 6.2: Error obtained transmitting the *TAMBUS* code with the Timing-based covert channel. We collected 100k packets for every experiments.

Methods	0→1 error (%)	1→0 error (%)	Total error (%)	TH (ms)	Waiting Time (ms)
a) Request 99.95	6.2e-4	6.1e-4	1.2e-3	20.90	9.92
b) Request 99.99	6.0e-5	1.8e-4	2.2e-4	25.90	17.05
c) Response 99.95	3.0e-4	8.4e-4	1.2e-3	19.09	9.78
d) Response 99.99	2.4e-4	1.4e-4	3.8e-4	20.90	13.10

1. We run 50K requests without leveraging the timing-based covert channel, and we calculate all the values of Δ (i.e., one value of Δ for each request).

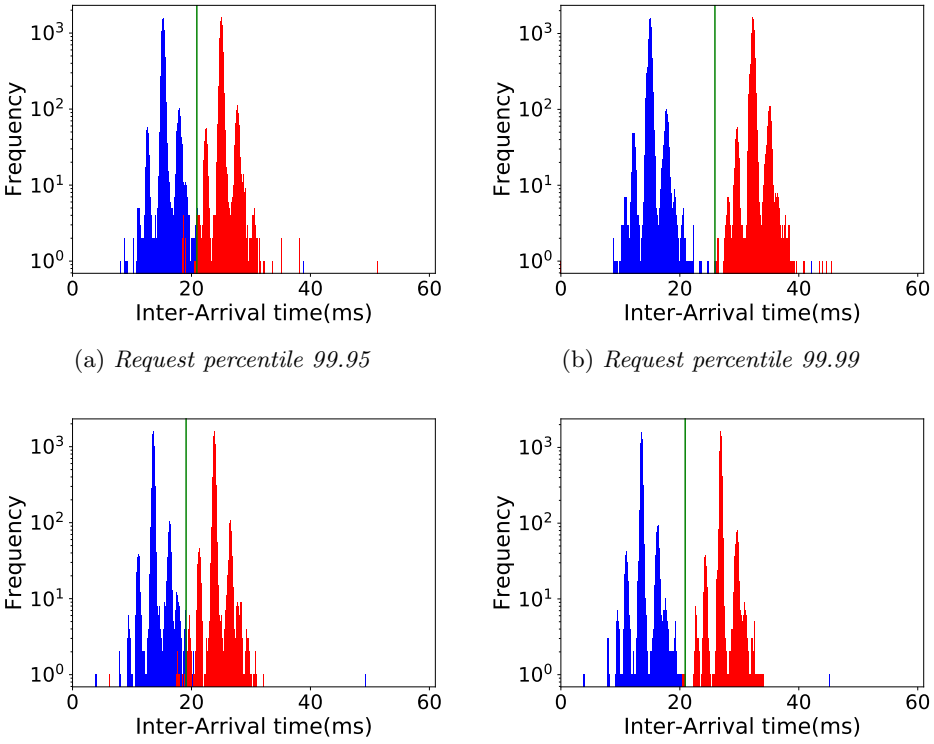
2. Once obtained the values of Δ, we calculate the thresholds TH_1 and TH_2 respectively as the 99.95 and the 99.99 percentile of the collected values. We define L_1 and L_2 respectively as the 0.05 and 0.01 percentile of the collected values of Δ. Finally, we define the waiting time ST_1 and ST_2 as $TH_1 - L_1$ and $TH_2 - L_2$ respectively.

3. We run two different experiments by including the timing-based covert channel, both made up of 500K requests. As parameters we used the pair TH_1/ST_1 for one of the experiment, TH_2/ST_2 for the other.

In Figure 6.10, we show the results for the timing-based covert channel on requests (6.10a and 6.10b) and responses (6.10c and 6.10d). We colored in blue the packets carrying 0 as *TAMBUS* code and in red the packets transmitting a 1 as *TAMBUS* code. We represent the position of the threshold TH with a green vertical line.

In Table 6.2, we report the detailed performances collected during the same experiments of the graphs of Figure 6.10. The total percentage of error is never greater than 10^{-3} as the order of magnitude. The first column indicates the type of *Modbus/TCP* message (request or response) and the percentile considered to generate TH and WT. We also report the letter of the corresponding graph in Figure 6.10. The second column shows the percentage of packets transmitting a *TAMBUS* code 0 but incorrectly classified by the TC Verifier as *TAMBUS* code 1 (late packets). Conversely, the third column shows the percentage of packets transmitting a *TAMBUS* code 1 but erroneously classified by the TC Verifier as *TAMBUS* code 0 (advance packets).

We evaluate *TAMBUS* using two different pairs TH/ST to show that it is possible to choose between different security levels of *TAMBUS*, based



(a) *Request percentile 99.95*

(b) *Request percentile 99.99*

(c) *Response percentile 99.95*

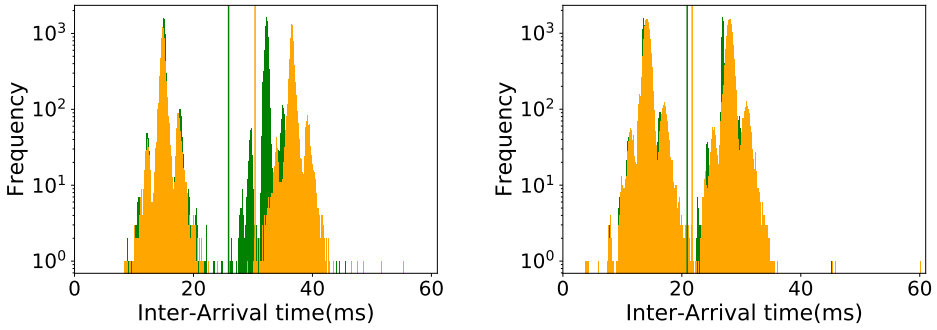
(d) *Response percentile 99.99*

Figure 6.10: Distribution of values of Δ for packets transmitting *TAMBUS* code = 0 (in blue) and *TAMBUS* code = 1 (in red). The green vertical line represents the used threshold TH.

on the necessity. A higher threshold would bring a lower error rate but a higher average delay. Conversely, a lower threshold would bring an higher error rate but a lower average delay. For instance, by choosing a threshold using the 99.95 percentile, a company allows the existence (in mean) of 0.05% of possible wrong packets over a certain amount of time. An high error rate means that, over a certain amount of time, an adversary has more chances to modify some packets without being noticed by *TAMBUS*. However, the timing-based covert channel is a statistical method and can detect anomalies if the percentage of errors goes above the chosen one.

The results of the first scenario show low error rates, demonstrating that the timing-based covert channel can effectively be implemented. However, even if the results in Table 6.2 evaluate a realistic situation, they do not consider any external traffic on the testbed. For this reason, in the second scenario, we performed two new experiments to test the timing-based covert channel in a more realist traffic scenario. To do this, we repeat the previous experiments adding to the testbed 10 extra clients (a single device simulates all of them). All extra clients perform requests every 50 ms to the Server. We perform the experiment only for the 99.99 percentile since this value get a lower bound on the performance of the timing-based covert channel. In Figure 6.11, we show the results of our experiment. We represented in green the time performance of a normal *TAMBUS* execution without extra clients (i.e., the same data of Figure 6.10b and Figure 6.10d), while in orange we represented the time performance with 10 extra clients. We observe that *TAMBUS* performances are still good: the new values for the waiting times are respectively 21.68 ms for the requests and 13.36 ms for the responses. Compared with the waiting time found in the first experiment (for requests and responses with percentile 99.99), the waiting time increases only of **4.63 ms** for the requests and **0.26 ms** for the responses. We can state that the greater values of the waiting times are due to the more significant workload on the TC Verifiers, which causes a higher variance of the collected Δ values. The TC Verifiers have to filter and check more packets, and this operation could influence the accuracy in the calculation of the arrival times, increasing so the variance of Δ. We have also found that the error percentage remains stable on the order of 10^{-4} for both the experiments: respectively 6.0×10^{-4} for the response scenario and 4.36×10^{-4} for the request scenario.

An important factor for the applicability of the timing-based covert channel is the average jitter of the network in which it is implemented. A complex ICS could have an average jitter higher of that one found in our testbed. To have further confirmation on the usability of the timing-based covert channel, we compared our threshold values with the maximum delay detected by



(a) *Request percentile 99.99* (b) *Response percentile 99.99*

Figure 6.11: Distribution of values of Δ for the packets transmitted in a *Modbus/TCP* network with only one Client (in green) and with 10 extra devices performing requests in the same network (in yellow). We represent the respective thresholds using the same colors.

Joelianto et al. in [82]. The results obtained by this previous work showed, on three different network topologies, a maximum delay of 16 ms. This value has the same order of magnitude of the thresholds that we found in our experiments with extra devices. Based on these results, we can assume that the timing-based method is implementable in most *Modbus/TCP* systems having a reasonable traffic load.

6.8.2.2 Evaluation of Storage-based Covert Channel

We evaluate the storage-based covert channel by performing three different experiments: one for requests, one for responses and one with both of them together. The TC Embedders intercept the packets and embed the *TAMBUS* code into the *Modbus/TCP* frame. As expected, in all the experiments the TC Verifier **never encounters a *TAMBUS* code error**. We also evaluate the storage-based covert channel timing performances. Results showed that responses take on average 20.04 ms to arrive at the destination, while requests take 38.97 ms. As reported in 6.8.1, about 18 ms of delay are introduced by the packet interception procedure (requests packets are intercepted twice). These results shows that generating the *TAMBUS* code and inserting it into the *Modbus/TCP* packet takes on average only 2 ms. Every security system that intercepts and modify packets introduce a timing overhead that could be not irrelevant for real-time systems. For this reason, the security staff has to decide the security method based on the necessities

of the facility. However, this problem affects every procedure that intercepts packets and it is not imputable to *TAMBUS*.

6.9 Summary

In this chapter, we presented *TAMBUS*, a method to provide authentication and packet integrity for *Modbus/TCP*. *TAMBUS* is the first method to provide at the same time the following features: (i) it is not a security by obscurity method, and (ii) it is compatible with legacy industrial devices. *TAMBUS* takes advantage of a storage-based and a timing-based covert channels to transmit authentication and integrity information between the *Modbus/TCP* clients and servers. This enhances the security features of *Modbus/TCP*, making it resistant to man in the middle, replay, and injection attacks with high statistical confidence.

We tested *TAMBUS* in a real testbed including an actual Wago PLC. We implemented both the storage-based and timing-based covert channels and tested them in separated experiments. For each experiment, we collected 100K packets. We also tested the timing-based covert channel in a scenario which included realistic external traffic on the *Modbus/TCP* network. To show the effectiveness of *TAMBUS*, we implemented and tested three different attacks against *TAMBUS*. Results show that *TAMBUS* can effectively protect *Modbus/TCP* by providing authentication and integrity of the packets. The timing-based covert channel achieved good results, obtaining an error rate on the *TAMBUS* code transmission never bigger than $10^{-3}\%$. For the experiments with a single client we got a maximum delay of 17.05 ms, while for the experiment including 10 extra clients the delay was of 21.68 ms. Since these delay are of the same order of magnitude as the maximum delay found by [82], we conclude that the timing-based covert channel can be implemented in most systems having a reasonable traffic load. The results for the experiments on the storage-based covert channel showed that this method correctly transmits the *TAMBUS* code without errors. Finally, the experiments showed that *TAMBUS* is an effective method to add a security layer to *Modbus/TCP* with a low-cost approach.

Chapter 7

Conclusion and Future Work

Computer authentication methods have grown tremendously over the last 60 years. These have developed from elementary methods over the decades, arriving today at biometrics and behavioral methods. Parallel to this development, the security authentication methods have also undergone many changes over the years: if we have never stopped improving them, it is also true that new techniques continuously arise to violate their security. In this thesis, we focus on the security of authentication methods. In Part I, we worked on possible novel attacks against the existing methods, while in Part II we developed novel authentication methods.

7.1 Summary of Contributions

In this section, we summarize the contributions of our work presented in this thesis.

7.1.1 Attack against existing authentication methods

In Part I, we investigate several attacks against the security of computer authentication systems. Over the years, research in this area has seen the development of many different kinds of attacks. This thesis focused on studying attacks to steal passwords by exploiting weaknesses in standard input methods. Our goal is to emphasize existing vulnerabilities and propose possible solutions to improve the security of these methods.

- *Eavesdropping on keyboards*: In Chapter 2, we investigate the security of Voice-over-IP (VoIP) against keyboard eavesdropping attacks. Nowadays, millions of people perform VoIP calls using their computer; However, people ignore the drawbacks of transmitting information along with their voice, such as keystroke sounds–as such sound can reveal what someone is typing on a keyboard. To this end, we present and assess a new keyboard acoustic eavesdropping attack that involves VoIP, called *Skype & Type attack* (*S&T*). Unlike previous attacks, *S&T* assumes a weak adversary model that is very practical in many real-world settings. We designed our experiments to recover standard and random texts (such as randomly generated passwords or PINs) typed during a VoIP call with the victim. We evaluate *S&T* over Skype, a very popular VoIP software, and we show it remains accurate in different attack scenarios. We showed it could significantly speed up brute force attacks on passwords and that it can be combined with textual information to recognize words with very high accuracy. We also focused on different experiments settings: we showed that our attack works well with different common and inexpensive recording devices and on a great variety of typing styles and speed and is robust to VoIP-related issues (low bandwidth and human voice). Finally, we proposed a transparent countermeasure that does not severely impact the quality of the voice during the call and can disrupt the spectral features needed by *S&T*.

- *Inferring ATM PINs of Users Typing with a Covered Hand*: In Chapter 3, we worked on keyboard eavesdropping security, with a particular focus on ATM PIN pad security. We proposed a novel attack to reconstruct PINs entered by victims covering the typing hand with the other hand. Our experiments showed that our method has high accuracy in reconstructing PINs within three attempts. In particular, our results showed that the proposed method outperforms humans, showing the effectiveness of such an approach. To evaluate our attack, we built a simulated ATM, thanks to which we collected 5800 5-digits PINs by 58 distinct participants. This dataset has been made publicly available, allowing the research community to study the problem further. Finally, we propose possible countermeasures against our attack and evaluate some of these.

- *Acoustic Side-Channel Attack on ATM PIN Pads*: In Chapter 4, we focused on ATM PIN pad acoustic eavesdropping attack (called $\mathcal{P}inDrop$). Like the previous attack, this aims to reconstruct the PINs

entered by a victim on an ATM PIN pad. However, this attack exploits a completely different technique based on recognizing the unique keypress sound emitted by a key when pressed. After collecting data with our testbed, we showed how this attack is effective. In particular, we show how our attack works, even placing the recording microphone at a significant distance from the target PIN pad. Furthermore, to evaluate our attack in the most realistic conditions, we evaluated the robustness of our algorithm in the presence of noise, showing how performance remains high.

7.1.2 Novel authentication methods

In Part II, we investigated novel authentication methods. In particular, our research aims twofold: (i) to propose user-friendly and secure methods that can easily be adapted to users' needs, and (ii) to propose solutions applicable in a real-world context to increase the security of legacy systems. We presented a novel behavioral authentication based on the ear channel movement recognition during chewing for the former. For the latter, we presented a backward protocol compatible with MODBUS that increases legacy CPS systems' security by leveraging two covert channels.

- *Eathentication*: To the best of our knowledge, our work presented in Chapter 5, is the first study that proposes a behavioral authentication method based on external ear channel movements recognition during eating. We called this method *Eathentication*. To this end, we designed and printed a cheap 3D prototype of modified earphones, and we integrated the sensors necessary to authenticate a user. To evaluate our method, we collected a dataset using our modified earphones prototype: we asked participants to eat different types of food and to empty-chew during the recording session. *Eathentication* achieved good performance showing how it can be effectively used as an authentication method.

- *Security network for ICS*: In Chapter 6, we presented *TAMBUS* a novel authentication method backward compatible with *Modbus/TCP*. To do so, *TAMBUS* improves *Modbus/TCP* security by exploiting trusted devices that verify the authenticity of industrial clients and servers and the integrity of the network traffic. We build an industrial testbed, including a real Wago PLC, to evaluate the effectiveness of our method. Thanks to this testbed, we showed that our method works without modifying the *Modbus/TCP* protocol and without adding traf-

fic overheads. Therefore, *TAMBUS* can guarantee a high security level for *Modbus/TCP* without affecting its regular functionality.

7.2 Lessons Learned

Thanks to the attacks we developed, we were able to study the principal vulnerabilities of the authentication systems analyzed. Despite their high level of usability, Pin-based systems are particularly vulnerable to different kinds of attacks. In particular, the research we conducted showed that the typical countermeasures applied by users are not as effective in preventing threats as one might think (e.g., covering with one hand while typing). Therefore, we suggest changing user habits or possibly implementing new countermeasures when possible. Thanks to our attack on VoIP call software, we investigated the security of our personal and business communication methods. It turns out that user privacy and security can be at risk by the careless use of these technologies. Therefore, it is desirable that the user is aware of these vulnerabilities and instructed to take simple countermeasures to protect himself.

We also investigated a new user-friendly biometrics authentication system. For this study, we demonstrate how the analysis of the auditory channel (while the user performs specific tasks) resulted in an effective method to ensure a secure and non-invasive access method. From the point of view of device authentication, we proposed a new method to secure an outdated authentication protocol. The techniques we developed give a new authentication method fully backward compatible with dated systems and at a very competitive cost compared to a complete facility upgrade.

7.3 Future Work

This section presents possible future directions of our research contribution presented in this thesis.

7.3.1 Attack against existing authentication methods

Despite in Chapter 2 we proposed a high effective countermeasure against our keyboard eavesdropping attack, it still might impact the VoIP call quality. As a future direction, we plan to conduct new experiments to validate our initial claim that voice is not affected by our countermeasure. Another interesting research direction might consider adversarial machine learning: leveraging these techniques, we might create a new effective countermeasure

technique that generates the minimum possible loss of quality for the VoIP call. For our attack introduced in Chapter 3, plans include the extension of our dataset to test the performance of our algorithm in new conditions. Thanks to this novel data collection, we will also evaluate how to improve the covering strategies. Other improvements for our attack concern the PIN pad reproduction to simplify the data collection and the investigation of the possibility to extract the timestamps without using the acoustic information. Extending our attack described in Chapter 4, an interesting extension regards the use of new kinds of microphones. In particular, we will evaluate if a parabolic microphone can be used to perform the attack: this would significantly increase the maximum recording distance. Finally, it would also be interesting to evaluate the performance of our attack on different PIN pads (e.g., touch screen PIN pads).

7.3.2 Novel authentication methods

Extending our work described in Chapter 5, we will collect new datasets by introducing novel kinds of stimuli. This extension will allow us to define more user-friendly procedures to train our authentication method, like authentication procedures based on new stimuli never seen in training. We will also evaluate new machine learning models on the dataset. In particular, we plan to investigate two new kinds of machine learning algorithms: one-class models that use a different paradigm to authenticate a user and autoencoders that require much more data to be trained and that may further improve our authentication performance. As a future direction of our authentication protocol proposed in Chapter 6, we will investigate its effectiveness in more realistic contexts. Indeed, even if our testbed included a real PLC, we plan to test our timing-based covert channel in more complex MODBUS networks. Possibility in this direction may include renting more sophisticated testbeds or collaborating with facilities to directly integrate our authentication system in controlled environments where it will be possible to test it.

Bibliography

[1] Oxford dictionary - which letters in the alphabet are used most often.

[2] Ayman Abaza, Arun Ross, Christina Hebert, Mary Ann F Harrison, and Mark S Nixon. A survey on ear biometrics. *ACM computing surveys (CSUR)*, 45(2):22, 2013.

[3] Yomna Abdelrahman, Mohamed Khamis, Stefan Schneegass, and Florian Alt. Stay cool! understanding thermal attacks on mobile-based user authentication. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pages 3751–3763, 2017.

[4] National Aeronautics and Space Administration. Audit cybersecurity management and oversight at the jet propulsion laboratory.

[5] John O Akers, Ramesh Narang, and Richard W DeChamplain. Posterior dislocation of the mandibular condyle into the external ear canal. *Journal of Oral and Maxillofacial Surgery*, 40(6):369–370, 1982.

[6] Anton HM Akkermans, Tom AM Kevenaar, and Daniel WE Schobben. Acoustic ear recognition for person identification. In *Fourth IEEE Workshop on Automatic Identification Advanced Technologies (AutoID'05)*, pages 219–223. IEEE, 2005.

[7] Kamran Ali, Alex Liu, Wei Wang, and Muhammad Shahzad. Keystroke recognition using wifi signals. In *ACM MobiCom*, pages 90–102, 2015.

[8] S Abhishek Anand and Nitesh Saxena. Keyboard emanations in remote voice calls: Password leakage and noise (less) masking defenses. In *ACM CODASPY*, 2018.

[9] Takayuki Arakawa, Takafumi Koshinaka, Shohei Yano, Hideki Irisawa, Ryoji Miyahara, and Hitoshi Imaoka. Fast and accurate personal authentication using ear acoustics. In *2016 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*, pages 1–4. IEEE, 2016.

[10] Dmitri Asonov and Rakesh Agrawal. Keyboard acoustic emanations. In *IEEE S&P*, pages 3–11, 2004.

[11] Dmitri Asonov and Rakesh Agrawal. Keyboard acoustic emanations. In *IEEE Symposium on Security and Privacy, 2004. Proceedings. 2004*, pages 3–11. IEEE, 2004.

[12] ATM Industry Association. .

[13] 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.

[14] Kiran Balagani, Matteo Cardaioli, Mauro Conti, Paolo Gasti, Martin Georgiev, Tristan Gurtler, Daniele Lain, Charissa Miller, Kendall Molas, Nikita Samarin, et al. Pilot: Password and pin information leakage from obfuscated typing videos. *Journal of Computer Security*, 27(4):405–425, 2019.

[15] Kiran S Balagani, Mauro Conti, Paolo Gasti, Martin Georgiev, Tristan Gurtler, Daniele Lain, Charissa Miller, Kendall Molas, Nikita Samarin, Eugen Saraci, et al. Silk-tv: Secret information leakage from keystroke timing videos. In *European Symposium on Research in Computer Security*, pages 263–280. Springer, 2018.

[16] Davide Balzarotti, Marco Cova, and Giovanni Vigna. Clearshot: Eavesdropping on keyboard input from video. In *IEEE S&P*, pages 170–183, 2008.

[17] Davide Balzarotti, Marco Cova, and Giovanni Vigna. Clearshot: Eavesdropping on keyboard input from video. In *2008 IEEE Symposium on Security and Privacy (sp 2008)*, pages 170–183. IEEE, 2008.

[18] Giovanni Barbieri, Mauro Conti, Nils Ole Tippenhauer, and Federico Turrin. Sorry, shodan is not enough! assessing ics security via ixp network traffic analysis. *arXiv preprint arXiv:2007.01114*, 2020.

[19] Dipak Basu. *Dictionary of pure and applied physics*. CRC press, 2000.

[20] Francesco Bergadano, Daniele Gunetti, and Claudia Picardi. User authentication through keystroke dynamics. *ACM Transactions on Information and System Security (TISSEC)*, 5(4):367–397, 2002.

[21] Yigael Berger, Avishai Wool, and Arie Yeredor. Dictionary attacks using keyboard acoustic emanations. In *ACM CCS*, pages 245–254, 2006.

[22] Yigael Berger, Avishai Wool, and Arie Yeredor. Dictionary attacks using keyboard acoustic emanations. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 245–254, 2006.

[23] Giuseppe Bernieri, Stefano Cecconello, Mauro Conti, and Gianluca Lain. Tambus: A novel authentication method through covert channels for securing industrial networks. *Computer Networks*, 183:107583, 2020.

[24] Charles Beumier and Marc Acheroy. Automatic 3d face authentication. *Image and Vision Computing*, 18(4):315–321, 2000.

[25] Shivam Bhasin, Anupam Chattopadhyay, Annelie Heuser, Dirmanto Jap, Stjepan Picek, and Ritu Ranjan Shrivastwa. Mind the portability: A warriors guide through realistic profiled side-channel analysis. In *27th Annual Network and Distributed System Security Symposium, NDSS 2020, San Diego, California, USA, February 23-26, 2020*. The Internet Society, 2020.

[26] Lena Biel, Ola Pettersson, Lennart Philipson, and Peter Wide. Ecg analysis: a new approach in human identification. *IEEE Transactions on Instrumentation and Measurement*, 50(3):808–812, 2001.

[27] Farid Binbeshr, ML Mat Kiah, Lip Yee Por, and Aws Alaa Zaidan. A systematic review of pin-entry methods resistant to shoulder-surfing attacks. *computers & security*, page 102116, 2020.

[28] Chris M. Bishop. Training with noise is equivalent to tikhonov regularization. *Neural Comput.*, 7(1):108–116, January 1995.

[29] Mike Bond, Omar Choudary, Steven J Murdoch, Sergei Skorobogatov, and Ross Anderson. Chip and skim: cloning emv cards with the preplay attack. In *2014 IEEE Symposium on Security and Privacy*, pages 49–64. IEEE, 2014.

[30] Stephen Boyd, Corinna Cortes, Mehryar Mohri, and Ana Radovanovic. Accuracy at the top. In *NIPS*, pages 953–961, 2012.

[31] Christopher Bronk and Eneken Tikk-Ringas. The cyber attack on saudi aramco. *Survival*, 55(2):81–96, 2013.

[32] Mark Burge and Wilhelm Burger. Ear biometrics in computer vision. In *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*, volume 2, pages 822–826. IEEE, 2000.

[33] Raffaele Cappelli, Dario Maio, Alessandra Lumini, and Davide Maltoni. Fingerprint image reconstruction from standard templates. *IEEE transactions on pattern analysis and machine intelligence*, 29(9):1489–1503, 2007.

[34] Stuart Card, Thomas Moran, and Allen Newell. The keystroke-level model for user performance time with interactive systems. *CACM*, 23(7):396–410, 1980.

[35] Matteo Cardaioli, Stefano Cecconello, Mauro Conti, Simone Milani, Stjepan Picek, and Eugen Saraci. Hand me your pin! inferring atm pins of users typing with a covered hand. In *31th USENIX Security Symposium (USENIX Security 22)*, 2022.

[36] Matteo Cardaioli, Mauro Conti, Kiran Balagani, and Paolo Gasti. Your pin sounds good! augmentation of pin guessing strategies via audio leakage. In *European Symposium on Research in Computer Security*, pages 720–735. Springer, 2020.

[37] Mattia Carlucci, Stefano Cecconello, Mauro Conti, and Piero Romare. Eathentication: A chewing-based authentication method. In *2020 IEEE Conference on Communications and Network Security (CNS)*, pages 1–9. IEEE, 2020.

[38] Defense Use Case. Analysis of the cyber attack on the ukrainian power grid. *Electricity Information Sharing and Analysis Center (E-ISAC)*, 2016.

[39] Stefano Cecconello, Alberto Compagno, Mauro Conti, Daniele Lain, and Gene Tsudik. Skype & type: Keyboard eavesdropping in voice-over-ip. *ACM Transactions on Privacy and Security (TOPS)*, 22(4):1–34, 2019.

[40] Yimin Chen, Tao Li, Rui Zhang, Yanchao Zhang, and Terri Hedgpeth. Eyetell: video-assisted touchscreen keystroke inference from eye movements. In *IEEE S&P*, pages 144–160, 2018.

[41] Michal Choras. Image feature extraction methods for ear biometrics–a survey. In *6th International Conference on Computer Information Systems and Industrial Management Applications (CISIM'07)*, pages 261–265. IEEE, 2007.

[42] Charles J Clopper and Egon S Pearson. The use of confidence or fiducial limits illustrated in the case of the binomial. *Biometrika*, 26(4):404–413, 1934.

[43] Alberto Compagno, Mauro Conti, Daniele Lain, and Gene Tsudik. Don't skype & type!: Acoustic eavesdropping in voice-over-ip. In *ACM ASIACCS*, 2017.

[44] Mauro Conti, Irina Zachia-Zlatea, and Bruno Crispo. Mind how you answer me!: transparently authenticating the user of a smartphone when answering or placing a call. In *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, pages 249–259. ACM, 2011.

[45] National Cash Register (NCR Corporation). The rise of emv and what it means for the magnetic stripe. https://www.ncr.com/blogs/payments/emv-magnetic-stripe, March 2021. [Online; accessed 7-June-2021].

[46] Anupam Das, Nikita Borisov, and Matthew Caesar. Do you hear what I hear?: fingerprinting smart devices through embedded acoustic components. In *ACM CCS*, pages 441–452, 2014.

[47] Gerson de Souza Faria and Hae Yong Kim. Differential audio analysis: a new side-channel attack on pin pads. *International Journal of Information Security*, 18(1):73–84, 2019.

[48] Belinda L Del Gaudio, Claudio Porzio, Gabriele Sampagnaro, and Vincenzo Verdoliva. How do mobile, internet and ict diffusion affect the banking industry? an empirical analysis. *European Management Journal*, 2020.

[49] Mohammad Derawi, Patrick Bours, and Ray Chen. Biometric acoustic ear recognition. In *Biometric Security and Privacy*, pages 71–120. Springer, 2017.

[50] Mohammad Derawi, Claudia Nickel, Patrick Bours, and Christoph Busch. Unobtrusive user-authentication on mobile phones using biometric gait recognition. pages 306–311, 10 2010.

[51] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2625–2634, 2015.

[52] Bill Drury. *Control Techniques Drives and Controls Handbook (2nd Edition)*. Institution of Engineering and Technology, 2009.

[53] Simon Eberz, Kasper B Rasmussen, Vincent Lenders, and Ivan Martinovic. Evaluating behavioral biometrics for continuous authentication: Challenges and metrics. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pages 386–399. ACM, 2017.

[54] Malin Eiband, Mohamed Khamis, Emanuel Von Zezschwitz, Heinrich Hussmann, and Florian Alt. Understanding shoulder surfing in the wild: Stories from users and observers. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pages 4254–4265, 2017.

[55] Žiga Emeršič, Vitomir Štruc, and Peter Peer. Ear recognition: More than a survey. *Neurocomputing*, 255:26–39, 2017.

[56] Nicolas Falliere, Liam O Murchu, and Eric Chien. W32. stuxnet dossier. *White paper, Symantec Corp., Security Response*, 5(6):29, 2011.

[57] Song Fang, Ian Markwood, Yao Liu, Shangqing Zhao, Zhuo Lu, and Haojin Zhu. No training hurdles: Fast training-agnostic attacks to infer your typing. In *ACM CCS*, pages 1747–1760, 2018.

[58] Marcos Faundez-Zanuy. Biometric security technology. *IEEE Aerospace and Electronic Systems Magazine*, 21(6):15–26, 2006.

[59] Liu Fei, Zhang Renbin, Li Gang, Li Sixian, and Xu Fuhao. Security authentication protocol of modbus/tcp based on hash chain and synchronization mechanism. *Application Research of Computers*, (4):45, 2018.

[60] Igor Nai Fovino, Andrea Carcano, Marcelo Masera, and Alberto Trombetta. Design and implementation of a secure modbus protocol. In *International conference on critical infrastructure protection*, pages 83–96. Springer, 2009.

[61] Mario Frank, Ralf Biedert, Eugene Ma, Ivan Martinovic, and Dawn Song. Touchalytics: On the applicability of touchscreen input as a behavioral biometric for continuous authentication. *IEEE transactions on information forensics and security*, 8(1):136–148, 2012.

[62] Jeffrey Friedman. Tempest: A signal problem. *NSA Cryptologic Spectrum*, 1972.

[63] Daniel Genkin, Mihir Pattani, Roei Schuster, and Eran Tromer. Synesthesia: Detecting screen content via remote acoustic side channels. *arXiv preprint arXiv:1809.02629*, 2018.

[64] Daniel Genkin, Adi Shamir, and Eran Tromer. Acoustic cryptanalysis. *J. Cryptol.*, 30(2):392–443, April 2017.

[65] Brij B Gupta and Shaifali Narayan. A survey on contactless smart cards and payment system: Technologies, policies, attacks and countermeasures. *Journal of Global Information Management (JGIM)*, 28(4):135–159, 2020.

[66] Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1-3):389–422, 2002.

[67] Tzipora Halevi and Nitesh Saxena. A closer look at keyboard acoustic emanations: random passwords, typing styles and decoding techniques. In *ACM CCS*, pages 89–90, 2012.

[68] Tzipora Halevi and Nitesh Saxena. A closer look at keyboard acoustic emanations: random passwords, typing styles and decoding techniques. In *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security*, pages 89–90, 2012.

[69] Tzipora Halevi and Nitesh Saxena. Keyboard acoustic side channel attacks: exploring realistic and security-sensitive scenarios. *International Journal of Information Security*, 14(5):443–456, 2015.

[70] Tzipora Halevi and Nitesh Saxena. Keyboard acoustic side channel attacks: exploring realistic and security-sensitive scenarios. *International Journal of Information Security*, 14(5):443–456, 2015.

[71] Garrett Hayes and Khalil El-Khatib. Securing modbus transactions using hash-based message authentication codes and stream transmission control protocol. In *2013 Third International Conference on Communications and Information Technology (ICCIT)*, pages 179–184. IEEE, 2013.

[72] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[73] Peter Huitsing, Rodrigo Chandia, Mauricio Papa, and Sujeet Shenoi. Attack taxonomies for the modbus protocols. *International Journal of Critical Infrastructure Protection*, 1:37–44, 2008.

[74] Nautilus Hyosung. cmax7600ta installation manual. http://www.tetralink.com/core/media/media.nl/id.46617/c.4970910/.f?h=d919934a85943438b8fe, 2015. [Online; accessed 30-December-2020].

[75] Donato Impedovo and Giuseppe Pirlo. Automatic signature verification: The state of the art. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(5):609–635, 2008.

[76] M. Islam, M. Hossain, R. ul Islam, and K. Andersson. Static hand gesture recognition using convolutional neural network with data augmentation. In *2019 Joint 8th International Conference on Informatics, Electronics & Vision (ICIEV) and 2019 3rd International Conference on Imaging, Vision & Pattern Recognition (icIVPR)*, pages 324–329, jun 2019.

[77] ISO. Iso: Financial services - personal identification number (pin) management and security - part 1: Basic principles and requirements for pins in card-based systems. https://www.iso.org/standard/68669.html, 2017. [Online; accessed 30-December-2020].

[78] Anil K Jain, Lin Hong, Sharath Pankanti, and Ruud Bolle. An identity-authentication system using fingerprints. *Proceedings of the IEEE*, 85(9):1365–1388, 1997.

[79] Anil K Jain, Salil Prabhakar, Lin Hong, and Sharath Pankanti. Filterbank-based fingerprint matching. *IEEE transactions on Image Processing*, 9(5):846–859, 2000.

[80] Nursuriati Jamil, Ali AlMisreb, and A Abdul Halin. Illumination-invariant ear authentication. *Procedia Computer Science*, 42:271–278, 2014.

[81] Kun Jin, Si Fang, Chunyi Peng, Zhiyang Teng, Xufei Mao, Lan Zhang, and Xiangyang Li. Vivisnoop: Someone is snooping your typing without seeing it! In *IEEE CNS*, pages 1–9, 2017.

[82] Endra Joelianto et al. Performance of an industrial data communication protocol on ethernet network. In *2008 5th IFIP International Conference on Wireless and Optical Communications Networks (WOCN'08)*, pages 1–5. IEEE, 2008.

[83] Tyler Kaczmarek, Ercan Ozturk, and Gene Tsudik. Thermanator: Thermal residue-based post factum attacks on keyboard password entry. *arXiv preprint arXiv:1806.10189*, 2018.

[84] Tyler Kaczmarek, Ercan Ozturk, and Gene Tsudik. Thermanator: Thermal residue-based post factum attacks on keyboard data entry. In *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security*, pages 586–593, 2019.

[85] Anastasis Keliris and Michail Maniatakos. Icsref: A framework for automated reverse engineering of industrial control systems binaries. *arXiv preprint arXiv:1812.03478*, 2018.

[86] Nikhil Ketkar. Introduction to keras. In *Deep learning with Python*, pages 97–111. Springer, 2017.

[87] Jaehun Kim, Stjepan Picek, Annelie Heuser, Shivam Bhasin, and Alan Hanjalic. Make some noise. unleashing the power of convolutional neural networks for profiled side-channel analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 148–179, 2019.

[88] Paul C Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In *Annual International Cryptology Conference*, pages 104–113. Springer, 1996.

[89] Tadayoshi Kohno, Andre Broido, and Kimberly Claffy. Remote physical device fingerprinting. *IEEE TDSC*, 2(2):93–108, 2005.

[90] K. Lai and S. N. Yanushkevich. Cnn+rnn depth and skeleton based dynamic hand gesture recognition. In *2018 24th International Conference on Pattern Recognition (ICPR)*, pages 3451–3456, 2018.

[91] Paul Lamere, Philip Kwok, Evandro Gouvea, Bhiksha Raj, Rita Singh, William Walker, Manfred Warmuth, and Peter Wolf. The cmu sphinx-4 speech recognition system. In *IEEE ICASSP*, volume 1, pages 2–5, 2003.

[92] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[93] Antoine Lemay, José M Fernandez, and Scott Knight. A modbus command and control channel. In *2016 Annual IEEE Systems Conference (SysCon)*, pages 1–6. IEEE, 2016.

[94] Carlos Leonardo and Daryl Johnson. Modbus covert channel. In *Proceedings of the International Conference on Security and Management (SAM)*, page 1. The Steering Committee of The World Congress in Computer Science, Computer . . . , 2014.

[95] Jian Liu, Yan Wang, Gorkem Kar, Yingying Chen, Jie Yang, and Marco Gruteser. Snooping keystrokes with mm-level audio ranging on a single phone. In *ACM MobiCom*, pages 142–154, 2015.

[96] Jian Liu, Yan Wang, Gorkem Kar, Yingying Chen, Jie Yang, and Marco Gruteser. Snooping keystrokes with mm-level audio ranging on a single phone. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, pages 142–154, 2015.

[97] Ximing Liu, Yingjiu Li, Robert H Deng, Bing Chang, and Shujun Li. When human cognitive modeling meets pins: User-independent inter-keystroke timing attacks. *Computers & Security*, 80:90–107, 2019.

[98] Beth Logan et al. Mel frequency cepstral coefficients for music modeling. In *ISMIR*, 2000.

[99] Beth Logan et al. Mel frequency cepstral coefficients for music modeling. In *Ismir*, volume 270, pages 1–11, 2000.

[100] Jan Lukas, Jessica Fridrich, and Miroslav Goljan. Digital camera identification from sensor pattern noise. *IEEE TIFS*, 1(2):205–214, 2006.

[101] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. Springer, December 2006. ISBN 0-387-30857-1, http://www.dpabook.org/.

[102] Philip Marquardt, Arunabh Verma, Henry Carter, and Patrick Traynor. (sp) iphone: decoding vibrations from nearby keyboards using mobile phone accelerometers. In *ACM CCS*, pages 551–562, 2011.

[103] Zdenek Martinasek, Vlastimil Clupek, and Krisztina Trasy. Acoustic attack on keyboard using spectrogram and neural network. In *TSP*, pages 637–641, 2015.

[104] Zdenek Martinasek, Vlastimil Clupek, and Krisztina Trasy. Acoustic attack on keyboard using spectrogram and neural network. In *2015 38th International Conference on Telecommunications and Signal Processing (TSP)*, pages 637–641. IEEE, 2015.

[105] Mary L McHugh. The chi-square test of independence. *Biochemia medica*, 23(2):143–149, 2013.

[106] Modicon. Modbus messaging on tcp/ip implementation guide v1.0b, 2006.

[107] John Monaco. Sok: Keylogging side channels. In *IEEE S&P*, 2018.

[108] Keaton Mowery, Sarah Meiklejohn, and Stefan Savage. Heat of the moment: Characterizing the efficacy of thermal camera-based attacks. In *Proceedings of the 5th USENIX conference on Offensive technologies*, pages 6–6, 2011.

[109] Loris Nanni and Alessandra Lumini. A multi-matcher for ear authentication. *Pattern Recognition Letters*, 28(16):2219–2226, 2007.

[110] Loris Nanni and Alessandra Lumini. Fusion of color spaces for ear authentication. *Pattern Recognition*, 42(9):1906–1913, 2009.

[111] NationalCash Systems. ATM Statistics.

[112] Nozomi Networks. Advancing ics visibility and cybersecurity with the nozomi networks solution – white paper.

[113] Justin D Osborn and David C Challener. Trusted platform module evolution. *Johns Hopkins APL Technical Digest (Applied Physics Laboratory)*, 32(2):536–543, 2013.

[114] Sourav Panda, Yuanzhen Liu, Gerhard Petrus Hancke, and Umair Mujtaba Qureshi. Behavioral acoustic emanations: Attack and verification of pin entry using keypress sounds. *Sensors*, 20(11):3015, 2020.

[115] Marietta Papadatou-Pastou, Eleni Ntolka, Judith Schmitz, Maryanne Martin, Marcus R Munafò, Sebastian Ocklenburg, and Silvia Paracchini. Human handedness: A meta-analysis. *Psychological Bulletin*, April 2020.

[116] Anika Pflug and Christoph Busch. Ear biometrics: a survey of detection, feature extraction and recognition methods. *IET biometrics*, 1(2):114–129, 2012.

[117] Luis Miguel Ramírez, Luis Ernestos Ballesteros, and German Pablo Sandoval. Tensor tympani muscle: strange chewing muscle. *Medicina Oral, Patología Oral y Cirugía Bucal (Internet)*, 12(2):96–100, 2007.

[118] Norman H Rappaport, Peter D Scholl, and Jr JH Harris. Injury to the glenoid fossa. *Plastic and reconstructive surgery*, 77(2):304–308, 1986.

[119] EH Rothauser, WD Chapman, N Guttman, KS Nordby, HR Silbiger, GE Urbanek, and M Weinstock. Ieee recommended practice for speech quality measurements. *IEEE Transactions on Audio and Electroacoustics*, 17(3):225–246, 1969.

[120] Raul Sanchez-Reillo, Carmen Sanchez-Avila, and Ana Gonzalez-Marcos. Biometric identification through hand geometry measurements. *IEEE Transactions on pattern analysis and machine intelligence*, 22(10):1168–1171, 2000.

[121] Nolen Scaife, Christian Peeters, and Patrick Traynor. Fear the reaper: Characterization and fast detection of card skimmers. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 1–14, 2018.

[122] Sean Kelly. Cell Phone Cameras Hidden Inside ATMs Cause Rise In Fraud, 2018.

[123] Diksha Shukla, Rajesh Kumar, Abdul Serwadda, and Vir Phoha. Beware, your hands reveal your secrets! In *ACM CCS*, pages 904–917, 2014.

[124] Diksha Shukla, Rajesh Kumar, Abdul Serwadda, and Vir V Phoha. Beware, your hands reveal your secrets! In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 904–917, 2014.

[125] Dawn Xiaodong Song, David Wagner, and Xuqing Tian. Timing analysis of keystrokes and timing attacks on ssh. In *USENIX Security Symposium*, volume 2001, 2001.

[126] Riccardo Spolaor, QianQian Li, Merylin Monaro, Mauro Conti, Luciano Gamberini, and Giuseppe Sartori. Biometric authentication methods on smartphones : A survey. 2017.

[127] Keith Stouffer, Joe Falco, and Karen Scarfone. Guide to industrial control systems (ics) security. *NIST special publication*, 800(82):16–16, 2011.

[128] Jingchao Sun, Xiaocong Jin, Yimin Chen, Jinxue Zhang, Yanchao Zhang, and Rui Zhang. Visible: Video-assisted keystroke inference from tablet backside motion. In *NDSS*, 2016.

[129] Yanan Sun, Xiaohong Guan, and Ting Liu. A new method for authentication based on covert channel. In *IFIP International Conference on Network and Parallel Computing*, pages 160–165. Springer, 2011.

[130] Kazuhiro Taniguchi, Hikaru Chiaki, Mami Kurosawa, and Atsushi Nishikawa. A novel earphone type sensor for measuring mealtime: consideration of the method to distinguish between running and meals. *Sensors*, 17(2):252, 2017.

[131] James M Taylor and Hamid R Sharif. Enhancing integrity of modbus tcp through covert channels. In *2017 11th International Conference on Signal Processing and Communication Systems (ICSPCS)*, pages 1–6. IEEE, 2017.

[132] United States Attorney's Office, District of Massachussets. Bulgarian National Pleads Guilty to ATM Skimming, 2021.

[133] Jean-Marc Valin, Koen Vos, and T Terriberry. Definition of the opus audio codec. *IETF, September*, 2012.

[134] Martin Vuagnoux and Sylvain Pasini. Compromising electromagnetic emanations of wired and wireless keyboards. In *USENIX Security*, pages 1–16, 2009.

[135] Junjue Wang, Kaichen Zhao, Xinyu Zhang, and Chunyi Peng. Ubiquitous keyboard for small mobile devices: harnessing multipath fading for fine-grained keystroke localization. In *ACM MobiSys*, pages 14–27, 2014.

[136] Alfred C Weaver. Biometric authentication. *Computer*, 39(2):96–97, 2006.

[137] RL Wegel and CE Lane. The auditory masking of one pure tone by another and its probable relation to the dynamics of the inner ear. *Physical Review*, 23(2):266, 1924.

[138] Teng Wei, Shu Wang, Anfu Zhou, and Xinyu Zhang. Acoustic eavesdropping through wireless vibrometry. In *ACM MobiCom*, pages 130–141, 2015.

[139] Theodore J Williams. The purdue enterprise reference architecture. *Computers in industry*, 24(2-3):141–158, 1994.

[140] Wojciech Wodo and Lucjan Hanzlik. Thermal imaging attacks on keypad security systems. In *ICETE*, 2016.

[141] Luo Xuan and Li Yongzhong. Research and implementation of modbus tcp security enhancement protocol. In *Journal of Physics: Conference Series*, volume 1213, page 052058. IOP Publishing, 2019.

[142] P. Yan and K. W. Bowyer. Biometric recognition using 3d ear shape. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(8):1297–1308, 2007.

[143] Guixin Ye, Zhanyong Tang, Dingyi Fang, Xiaojiang Chen, Kwang In Kim, Ben Taylor, and Zheng Wang. Cracking android pattern lock in five attempts. In *Proceedings of the 2017 Network and Distributed System Security Symposium 2017 (NDSS 17)*. Internet Society, 2017.

[144] Xuhang Ying, Giuseppe Bernieri, Mauro Conti, and Radha Poovendran. Tacan: Transmitter authentication through covert channels in controller area networks. *arXiv preprint arXiv:1903.05231*, 2019.

[145] Saira Zahid, Muhammad Shahzad, Syed Ali Khayam, and Muddassar Farooq. Keystroke-based user identification on smart phones. In Engin Kirda, Somesh Jha, and Davide Balzarotti, editors, *Recent Advances in Intrusion Detection*, pages 224–243, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[146] Sebastian Zander, Grenville Armitage, and Philip Branch. A survey of covert channels and countermeasures in computer network protocols. *IEEE Communications Surveys & Tutorials*, 9(3):44–57, 2007.

[147] Tong Zhu, Qiang Ma, Shanfeng Zhang, and Yunhao Liu. Context-free attacks using keyboard acoustic emanations. In *ACM CCS*, pages 453–464, 2014.

[148] Tong Zhu, Qiang Ma, Shanfeng Zhang, and Yunhao Liu. Context-free attacks using keyboard acoustic emanations. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*, pages 453–464, 2014.

[149] Li Zhuang, Feng Zhou, and Doug Tygar. Keyboard acoustic emanations revisited. *ACM TISSEC*, 13(1):3, 2009.

[150] Li Zhuang, Feng Zhou, and J Doug Tygar. Keyboard acoustic emanations revisited. *ACM Transactions on Information and System Security (TISSEC)*, 13(1):1–26, 2009.

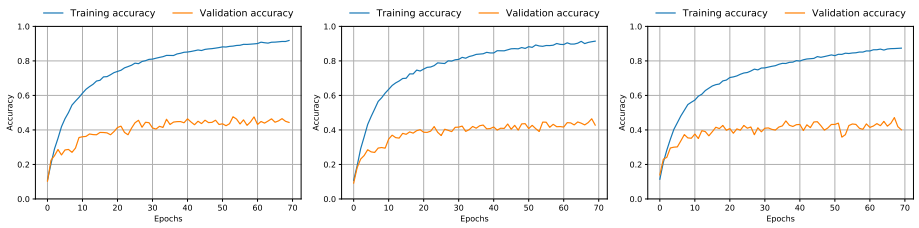
Appendices

Chapter A

Hand Me Your PIN

A.1 Neural Networks Additional Info

In Figure A.1, we show the training and validation accuracy for the three models selected after the random grid search. In the *Mixed* scenario, the validation accuracy grows faster than in *PIN pad independent* scenario and *Single PIN pad* scenario, reaching faster the plateau. Indeed, in the *Mixed* scenario, the validation accuracy stabilizes after 20 epochs, while we require more than 35 epochs for the other scenarios. This difference can be linked to a larger training size and a higher variance in the samples since the *Mixed* scenario is the only one to include videos from both PIN pads in the training phase.



(a) *Single PIN pad scenario.* We included 4 participants in validation, corresponding to 400 digits.

(b) *PIN pad independent scenario,* We included 5 participants in validation, corresponding to 500 digits.

(c) *Mixed scenario.* We included 6 participants in validation, corresponding to 600 digits.

Figure A.1: Training and validation accuracy for our three scenarios.

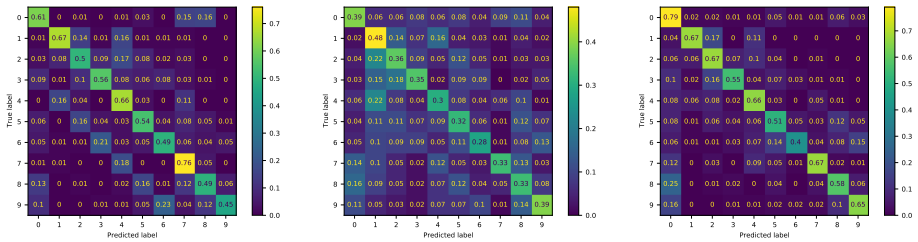
Next, we report some statistics about the training execution times for the three scenarios we consider.

- **Single PIN pad** scenario: the training set is composed of 32 participants, corresponding to 16 000 samples of 11 frames each. Our model takes 1 577 seconds to complete an epoch (i.e., approximately 34 hours to complete the entire training phase).
- **PIN pad independent** scenario: the training set is composed of 35 participants, corresponding to 17 500 samples of 11 frames each. Our model takes 1 598 seconds to complete an epoch (i.e., approximately 34 hours to complete the entire training phase).
- **Mixed** scenario: the training set is composed of 46 participants, corresponding to 23 000 samples of 11 frames each. Our model takes 2 240 seconds to complete an epoch (i.e., approximately 46 hours to complete the entire training phase).

A.2 Key Accuracy Analysis

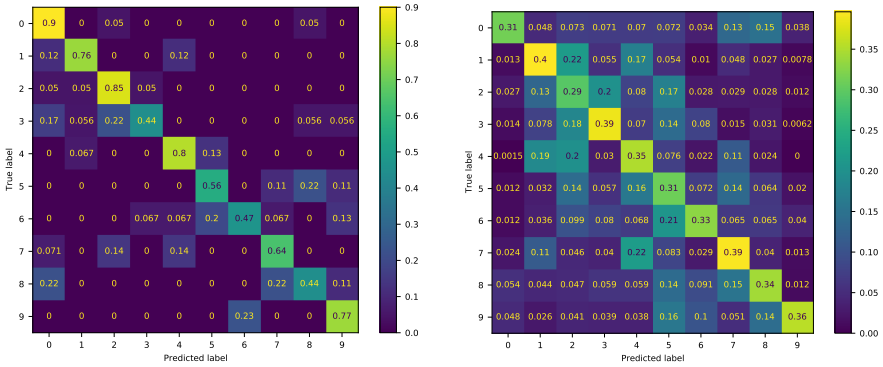
In this section, we provide further analysis on the key accuracy for our attack. Figure A.2 highlights that the accuracy on a single key is worse in the *PIN pad independent* scenario. Although the performance is considerably lower than the other two scenarios in Top-1 accuracy, it is interesting that the error dispersion affects the keys topologically close to the target one.

In Figure A.3, we compare our model and human performance on the key classification task. The misclassification error and the dispersion result are significantly lower for our algorithm. Moreover, it can be noticed how the four keys on which humans perform the best match those in the corners of our keypad (i.e., 1, 3, 7, and 9).



(a) *Single PIN pad* scenario. (b) *PIN pad independent* scenario. (c) *Mixed* scenario.

Figure A.2: Confusion matrices of key predictions (predicted labels) vs. true values (true labels) for our three scenarios.



(a) *Recalculated confusion matrix for our algorithm (Mixed scenario).* (b) *Confusion matrix for humans.*

Figure A.3: Confusion matrix comparison between our algorithm and humans.

A.3 Additional Experiments

To gain further insight into how coverage can affect the attack performance, we grouped the tested users by the coverage strategy:

- `Side`: The non-typing hand rests on the side of the palm and is angled to cover the keys of the PIN pad (40% of users applied this covering strategy).
- `Over`: The non-typing hand is raised completely off the surface, covering the PIN pad both with the entire back of the hand and the fingers (43% of users applied this covering strategy).
- `Top`: The fingers of the non-typing hand rest on the top of the PIN pad, and the back of the hand is used for the coverage (17% of users applied this covering strategy).

(a) *Left-corner camera.* (b) *Center camera.* (c) *Right-corner camera.*

Figure A.4: Same video frame recorded by three cameras.

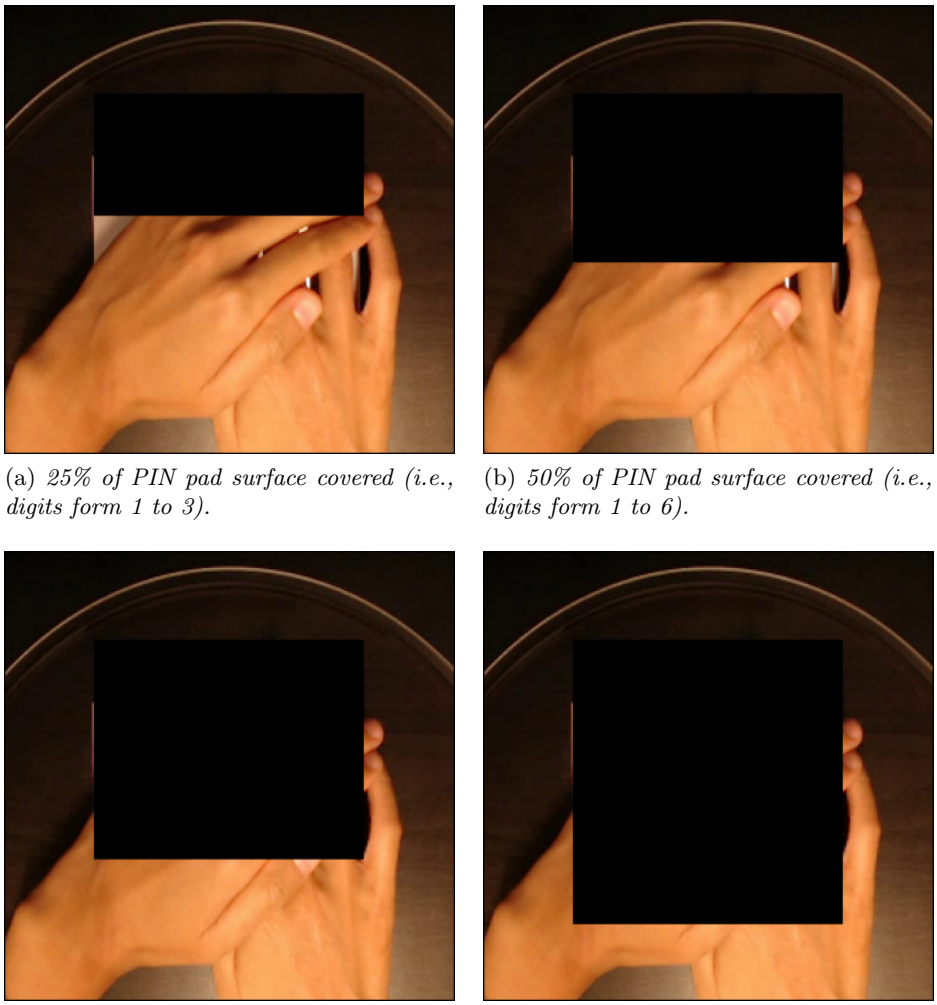
Covering strategy	Scenario	Key accuracy	PIN TOP-3 accuracy
`Side`	Single	0.64	0.30
	Independent	0.42	0.12
	Mixed	0.77	0.53
`Over`	Single	0.52	0.12
	Independent	0.31	0.10
	Mixed	0.46	0.07
`Top`	Single	NA	NA
	Independent	0.41	0.13
	Mixed	NA	NA

Table A.1: Performance of our attack for different covering strategies in *Single PIN pad*, *PIN pad independent*, and *Mixed* scenarios. `Top` covering participants were present in the *PIN pad independent* scenario only, as for the others, no data were available (NA).

In Table A.1, we report key and PIN TOP-3 accuracies for our approach. Clearly, `Side` covering strategy provides the least protection and should be avoided. At the same time, the `Over` and `Top` covering strategies provide much better protection. Interestingly, we see that with the `Over` covering strategy, the *Mixed* scenario reaches lower accuracy than the *Single PIN pad* scenario. We postulate this happens as this covering strategy makes it less "natural" for the user to type, deceiving the deep learning algorithm. Further attack improvements could be made with datasets having examples of one covering strategy only. For the `Top` covering strategy, there were no data for two out of three scenarios (denoted NA in Table A.1).

For the PIN shield countermeasure, we depict various levels of hiding in Figure A.5. There, 25% denotes that the first row of the PIN pad is covered (simulated with a black patch), 50% first two rows, 75% first three rows, and finally, 100% all four rows of the PIN pad are covered. Note that we do not include the covering with the other hand into these percentages.

Table A.2 provides results for several additional attack configurations. First, we performed two experiments simulating a lower camera quality or a larger camera distance from the PIN pad. For this purpose, we reduced the model input resolution from 250 `x` 250 to 125 `x` 125 and to 64 `x` 64. Results show that our model maintains an accuracy higher than 20%, even when halving the input resolution (i.e., doubling the camera distance). However, this is not to be considered as a physical limitation for our attack since if the attacker places a camera outside the ATM chassis, it is possible to use an optical zoom. Further, many pinhole cameras can record with a resolution



(a) *25% of PIN pad surface covered (i.e., digits form 1 to 3).*

(b) *50% of PIN pad surface covered (i.e., digits form 1 to 6).*

(c) *75% of PIN pad surface covered (i.e., digits form 1 to 9).*

(d) *100% of PIN pad surface covered (i.e., no digit is visible).*

Figure A.5: PIN pad shield configurations.

Experiment	Key accuracy	PIN TOP-3 accuracy
Input resolution 125 x 125	0.55	0.23
Input resolution 64 x 64	0.47	0.15
Left-corner camera	0.46	0.10
Right-corner camera	0.62	0.31
Multi-camera training	0.53	0.22
No data augmentation	0.44	0.11
Blacklisted excluded in training	0.54	0.18

Table A.2: Additional attack configurations and results in the *Mixed* scenario.

up to 1 080p [1], which is higher than the resolution we used to collect our dataset (720p).

Next, we investigated the accuracy of our attack leveraging different camera positions. In particular, we performed two experiments training and testing our model with the left-corner and the right-corner cameras, respectively. Figure A.4 shows the camera views used in our experiments. The results give a significant difference in performance if the camera is on the right or the left. This is because the participants in our experiment were right-handed, and therefore filming from the right had worse coverage of the PIN pad and typing hand. In contrast, the typing hand and the PIN pad were almost completely covered using shots from the left, significantly reducing the model's performance. We also evaluated whether using video from all three cameras in training (the experiment "multi-camera training" in Table A.2) could improve the accuracy of our model when compared with videos recorded from the center camera only. The results show a drop in performance, which we attribute to the higher variance in the data provided as input to the model.

Finally, we report the results of our model without data augmentation and without including the blacklisted users in the training set. In both configurations, the performance of our model drops, showing that reducing the training size is penalized heavily. Note that even in the worst case of a camera placed on the left corner (i.e., the one with less visibility), *our model still performs better than an average human.*

In this thesis, we used the feedback sound emitted by the PIN pad as a detection system for the frames containing a keystroke. To evaluate the im-

[1] https://www.dsecctv.com/Prod_telecamere_spioncino_porta_AHD.htm

Frame error confidence ($p < 0.01$)	**Key accuracy**	**PIN TOP-3 accuracy**
3	0.60	0.29
5	0.59	0.26
10	0.54	0.16
15	0.49	0.12
20	0.12	0.06

Table A.3: Performance of our attack in the *Mixed* scenario assuming different levels of frame detection error.

pact of other frame detection systems, we conducted an experiment varying the frame extraction precision. We simulated the detection error by adding Gaussian noise with mean zero to the ground truth (i.e., the frame position in the video). In Table A.3, we report the single key and the PIN TOP-3 accuracies for the *Mixed* scenario, simulating five levels of the frame detection error. Compared to the results obtained using the audio feedback (key accuracy 0.61, 5-digits PIN Top-3 accuracy 0.30), we see that our model works well even with small/medium levels of frame detection error (i.e., less than five frames). In particular, for a frame error confidence of three (i.e., when the frames are detected through the appearance on the screen of the masked symbols [36]), the performance drops only 1% both for key and TOP-3 PIN accuracies. Contrarily, when the detection error becomes high (i.e., more than 15 frames), the performance of our model decreases significantly. This happens since the frames considered by the model do not contain information related to the target key, as they are too temporally shifted. Naturally, if the attacker recognizes a situation like this, it would be possible to mitigate the effect of detection error by not using the feedback sound but observing the appearance of "*" symbols on the screen.

Chapter B

TAMBUS

B.1 Purdue Model

We design *TAMBUS* based on the Purdue Model. Figure B.1 depicts the Purdue model as defined by the standard ISA-99.

B.2 Experimental Testbed

The testbed that we used for our experiments includes a real industrial PLC and several Raspberry Pis. We used a Wago 750-881 controller. This device mounts an ARM microprocessor and uses WAGO-I/O-PRO V2.3, based on CODESYS V2.3, as programming environment. We decided to use a PLC from this vendor because Wago PLCs are very popular for small-to-medium installations, as Keliris states in [85]. Furthermore, the major part of the projects available on the web are written for Wago PLCs. Also, we considered experiments taken with a real testbed more accurate and precise than experiments run on a simulated environment (e.g., Mininet). We run a simulated physical process on the Wago PLC since the values, transmitted by the PLC, do not affect the operation of *TAMBUS*.

B.2.1 Testbed components

- Hardware information:
  - **Switch:** TP-Link TL-SG108E Switch Smart Managed, 8 Ports RJ45 Gigabit;

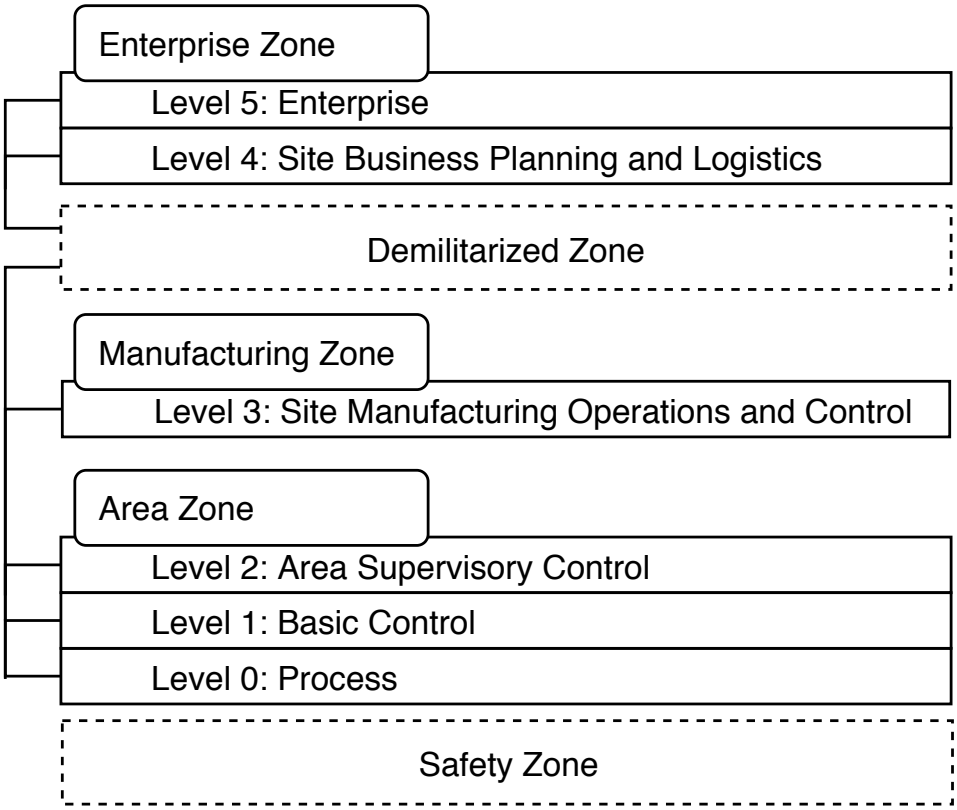


Figure B.1: Purdue Model overview.

  - **Client:** Raspberry Pi, model 3 B;
  - **Probe Client:** Raspberry Pi, model 3 B;
  - **Server:** Wago PLC 750-881;
  - **Probe Server:** Raspberry Pi, model 3 B;
  - **Monitor:** Raspberry Pi, model 3 B;
  - All the connections are made with Ethernet cables cat.5e, 3 meters.

- Software information for Raspberry Pis:
  - Raspbian Stretch Lite, minimal image based on Debian Stretch;
  - Version: April 2019;
  - Release date: 2019-04-08;
  - Kernel version: 4.14;

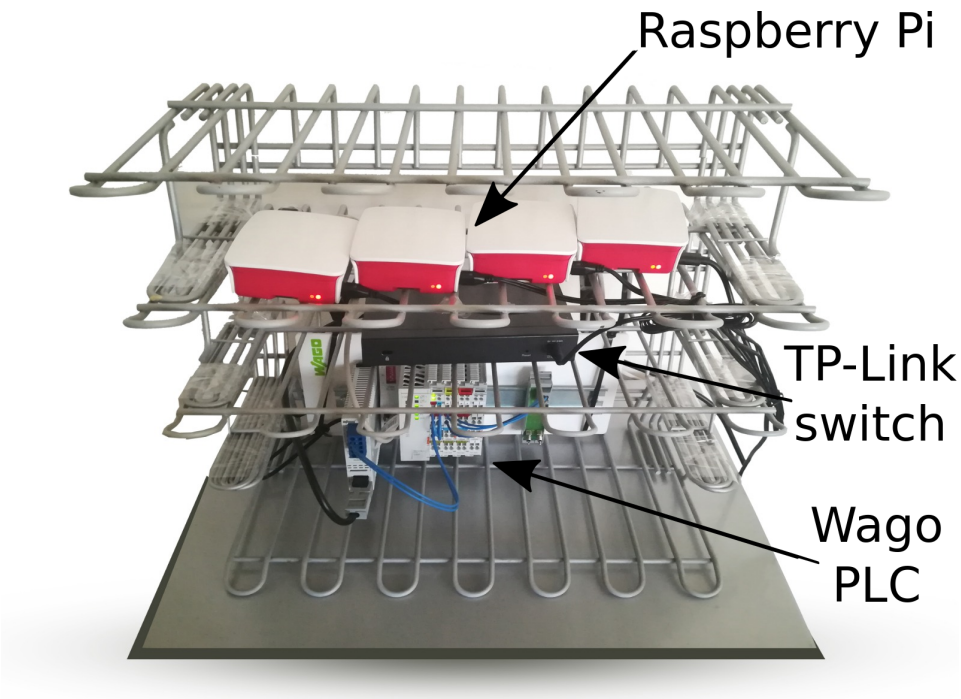


Figure B.2: Real testbed with Wago PLC. Top: four Raspberry Pi devices used as Client, Probe Client, Monitor and Probe Server. Middle: TP-Link switch. Bottom: Wago 750-881 PLC used as Server.

In Figure B.2 we show the actual testbed that we implemented. In the top of the figure there are four Raspberry Pis (the red ones), in the middle there is the TP-Link switch (the black device) and at the bottom there is the Wago PLC (the grey device).