Information Leakage Attacks and Countermeasures

Vasilios Mavroudis

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

of

University College London.

Department of Computer Science

University College London

2021

I, Vasilios Mavroudis, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Abstract

The scientific community has been consistently working on the pervasive problem of information leakage, uncovering numerous attack vectors, and proposing various countermeasures. Despite these efforts, leakage incidents remain prevalent, as the complexity of systems and protocols increases, and sophisticated modeling methods become more accessible to adversaries. This work studies how information leakages manifest in and impact interconnected systems and their users.

We first focus on online communications and investigate leakages in the Transport Layer Security protocol (TLS). Using modern machine learning models, we show that an eavesdropping adversary can efficiently exploit meta-information (e.g., packet size) not protected by the TLS' encryption to launch fingerprinting attacks at an unprecedented scale even under non-optimal conditions. We then turn our attention to ultrasonic communications, and discuss their security shortcomings and how adversaries could exploit them to compromise anonymity network users (even though they aim to offer a greater level of privacy compared to TLS).

Following up on these, we delve into physical layer leakages that concern a wide array of (networked) systems such as servers, embedded nodes, Tor relays, and hardware cryptocurrency wallets. We revisit location-based side-channel attacks and develop an exploitation neural network. Our model demonstrates the capabilities of a modern adversary but also presents an inexpensive tool to be used by auditors for detecting such leakages early on during the development cycle. Subsequently, we investigate techniques that further minimize the impact of leakages found in production components. Our proposed system design distributes both the custody of secrets and the cryptographic operation execution across several components, thus making the exploitation of leaks difficult.

Impact Statement

The problems addressed in this thesis, as well as the analyses, results, and tools we present herein, advance the state-of-the-art in areas of systems security and privacy. Moreover, researchers and industry practitioners will be able to use and adapt our techniques to secure their systems at various stages of manufacturing and operational lifecycle. The contents of this thesis have been published in conference proceedings and in open access repositories.

Our work on webpage fingerprinting provides novel insights on the privacy properties of the Transport Layer Protocol and shows that eavesdropping adversaries can reliably infer the webpage loaded based on traffic patterns. This work has been published as a preprint and has been presented at ScAINet 20' (collocated with the USENIX Security and AI Networking Summit). The visualization of the embeddings produced by the deep neural network received an honourable mention at the "Research Images Cross-Disciplinary Competition 2020" at UCL [1].

Our analysis of the ultrasonic communications ecosystem identified issues affecting hundreds of Android applications, used by millions of users at the time. Several frameworks modified their operation in line with our proposals. This work was presented at the Privacy Enhancing Technologies Symposium (PETS 2017), Black Hat EU 2016, RSA 2018 and several other venues and seminars. A non-exhaustive list of press coverage of this work can be found on the project's website: https://ubeacsec.org.

Our work on hardware leakages demonstrates that modern machine learning techniques enable adversaries to launch significantly more accurate side-channel

[1] https://www.grad.ucl.ac.uk/comp/2019-2020/research-images-competition/

attacks that can reliably extract keys from high-performance embedded processors. It was presented at the 25th Annual International Conference on the Theory and Application of Cryptology and Information Security (Asiacrypt 2019) in Kobe, Japan.

Finally, our trojan-resilient design, Myst, is one of the first practical system architectures that can mitigate low-level attacks (e.g., hardware trojans, side-channels) with only a small performance overhead. This work was presented at the ACM Conference on Computer and Communications Security (CCS 2017) and various other industrial and academic venues such as Defcon 25 in Las Vegas, US. Our prototype is currently used in production by Enigma Bridge and was a finalist in the CSAW Applied Research Contest 2018. Together with our work on location leakages, Myst introduces a new design approach that aims to break the detection-evasion cycle between manufacturers and adversaries. As part of this project, we also released the first open-source cryptographic library for Java Cards and presented it at Black-Hat US in 2017. This work, as well as our study of the ultrasonic communication ecosystem, were conducted as part of the Panoramix research project.

Acknowledgements

This thesis would not have been possible without the patience and kindness of my supervisor George Danezis. It has been an honor and a privilege working with George. I was also very lucky to have great people supporting me along the way; making me a better researcher and person: David Kohan Marzagao, Federico Maggi, Jamie Hayes, Jonathan Bootle, Katerina Tsarava, Kostas Papagiannopoulos, Petr Svenda, Sebastian Meiser and Yanick Fratantonio.

To all the great minds that I have had the pleasure of collaborating with - Alberto Sonnino, Andrea Cerulli, Aritra Dhar, Christos Andrikos, Dan Cvrcek, Dusan Klinec, Giorgos Rassias, Guilherme Perin, Hayden Melton, Kari Kostiainen, Karl Wüst, Lejla Batina, Liran Lerman, Lukasz Chmielewski, Nikolaos Samaras, and Srdjan Matic - thank you for all the lengthy discussions, the arguments, the late nights, the rejections, the rebuttals, and the successes. I am also indebted to Bernhard Esslinger, Carmela Troncoso, Chris Kruegel, Giovanni Vigna, and Srdjan Capkun for giving me opportunities I never knew existed. Furthermore, I would like to thank my officemates at UCL (6.22) and ETH Zurich; without their hospitality this Ph.D. experience would not have been the same.

Last but not least, I would like to thank Anastasia Gkigkoudi, Kostas Mavroudis, Vasileios Gkigkoudis, and Eva Bugallo Blanco for their endless support on this journey and beyond; your impact cannot be overstated.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis studies *information leakage* in interconnected systems, a pervasive problem that has been the subject of a large body of research on exploitation techniques and their corresponding mitigation approaches.

In 1937, the US Navy set up a complex arc of receivers (spanning from Philippines through Samoa, Midway, and Hawaii to Alaska) to track the movements of Japanese warships through their encrypted radio transmissions [1]. While the Japanese encryption cipher had not been broken yet [2], leakages enabled the allies to still extract valuable information. Using a network of sensitive antennas, US Navy officers would first find the direction at which a signal was the strongest and then use triangulation techniques to pinpoint the transmitter's location on the map. Successive measurements were used to extract information about the course and speed of warships, while the communication patterns revealed the lines of command (by ascertaining which radios talked to which) and impending military operations (through unusual spikes in the transmission volume).

In modern systems and protocols, information leakages take various forms and can occur for a wide range of reasons. One of the most well-studied areas are hardware side-channel attacks [3, 4, 5, 6, 7, 8], where the adversary extracts sensitive information (e.g., secret/private keys) from a chip by measuring and analysing its physical parameters (e.g., power consumption, electromagnetic emanations, operation latency). Through the analysis of those (noisy) measurements, the adversary deduces parts of the secret key or other sensitive information stored in the chip.

Such attacks pose a risk to several systems (cryptocurrency hardware wallets, TLS nodes, Internet-of-Things devices, anonymity network nodes) that need to retain their security properties even when the adversary has physical access to their hardware.

Besides physical leakages, eavesdropping adversaries may exploit leakages on the communication protocol layer. For example, fingerprinting attacks against anonymity networks (e.g., the Tor anonymity network [9, 10, 11]) that analyze the size, direction and inter-arrival times of the encrypted packets. Such attacks allow adversaries to infer the websites visited by the user, even if the underlying encryption scheme remains secure.

Overall, this work studies several types of leakages that occur when information or secrets are transmitted (Chapters 4 and 5), processed (Chapter 6) and stored (Chapter 7). We investigate how the advances in machine learning enable adversaries to attain very high accuracy in extracting sensitive information in various scenarios [12, 13] and launch fingerprinting attacks at an unprecedented scale [14]. Moreover, we show how the inter-connectivity of modern systems increases the risk for leakages as adversaries may exploit one application or platform to breach the security properties of another [15]. Finally, we expand on the idea of resilience to breaches and introduce a novel architecture for high-assurance systems that goes beyond the detect-evasion arms race [13].

1.1 Thesis Organization

In Chapter 2, we introduce fundamental concepts and ideas related to information leakage, machine learning and communication channels, as well as provide the background on techniques used later on. Related works are discussed in Chapter 3, while additional papers are introduced in the chapters when relevant.

Chapter 4 investigates whether machine learning models could enable adversaries to launch very accurate and large-scale *traffic fingerprinting* attacks. Such attacks enable adversaries to infer the websites or webpages loaded by users of the Transport Layer Security protocol or anonymity networks [16, 17, 18, 19, 20, 21, 22, 23]. We find that not only adversaries can achieve high inference accuracy but

that attacks are possible at an unprecedented scale even under non-optimal (for the adversary) assumptions. We thus revisit some of the main assumptions in past works and propose three realistic scenarios simulating non-optimal fingerprinting conditions. Then, we introduce an *adaptive* fingerprinting adversary and experimentally evaluate its accuracy and operation. Our experiments show that adaptive adversaries can uncover the webpage visited by a user among several thousand potential pages, even under considerable distributional shift (e.g., the webpage contents change significantly over time). Such adversaries could infer the products a user browses on shopping websites or log the browsing habits of state dissidents on online forums and encyclopedias.

We then investigate the real-world uses of the ultrasonic communications channel, analyze Android apps and examine their privacy and security characteristics (Chapter 5). We find that tracking techniques based on ultrasounds are susceptible to various information leakage attacks and expose to risks other unrelated privacy-sensitive applications that co-exist on the user's devices. In particular, we show that 1) an eavesdropping adversary could use beacon spoofing techniques to acquire sensitive information about a user, and 2) ultrasound cross-device tracking deployments can be abused to perform stealthy deanonymization attacks (e.g., to unmask users who browse the Internet through anonymity networks such as Tor). To curb those risks, we introduce two immediately deployable defense mechanisms that enable practitioners, researchers and everyday users to filter out ultrasonic emanations. Our Chrome browser extension and our patch for Android's permissions system allow for the suppression of the frequencies falling within the ultrasonic spectrum, thus preventing any unauthorized information transmission. Moreover, we discuss how the standardization of the channel and flexible OS-level APIs can both simplify app development and abuse.

Chapter 6 revisits the location side-channel and, given the fairly uncharted nature of such leakages, works to broaden the state-of-the-art on modeling and exploitation. Recent works have shown that near-field microprobes have the capacity to isolate small regions of a chip surface and enable precise measurements with high

spatial resolution. Being able to distinguish the activity of small regions has given rise to location-based side-channel attacks, which exploit the spatial dependencies of cryptographic algorithms in order to recover the secret key. Our work performs the first successful location-based attack on the Static Random-Access Memory (SRAM) of a modern ARM Cortex-M4 chip and investigates the effectiveness of neural network classifiers in distinguishing accesses to SRAM regions of varying size (in the context of single-shot attacks). Our results show that such machine learning models are capable of achieving high classification accuracy and suggest that the security of cryptographic implementations that may demonstrate location dependencies (e.g., RSA/ECC with large memory footprint, AES with lookup tables) needs to be carefully examined.

Finally, Chapter 7 proposes Myst, a practical high-assurance design, that uses commercial off-the-shelf (COTS) hardware, and provides strong leakage-tolerance guarantees, even in the presence of multiple compromised components. The key idea is to combine protective-redundancy with modern threshold cryptographic techniques to achieve tolerance to (intentional and unintentional) errors that could be exploited by an adversary to extract sensitive information from the system. To evaluate our design, we build a Hardware Security Module that provides the highest level of assurance possible with COTS components. Specifically, we employ more than one hundred COTS secure cryptographic coprocessors, verified to FIPS140-2 Level 4 tamper-resistance standards, and use them to realize high-confidentiality random number generation, key derivation, public key decryption and signing. Our experiments show a reasonable computational overhead (less than 1% for both Decryption and Signing) and an exponential increase in compromises as more ICs are added. This design takes a step in a different direction from the bulk of works in leakage-prevention, as it focuses on how critical systems could become resilient to compromised components. Myst can be also combined with any countermeasures that are available in the individual hardware components (e.g., masking), thus further minimizing the likelihood of successful attacks.

1.2 Publications & Works done in collaboration

In this section, we provide an overview of the research works comprising this report and outline the author's contributions.

Vasilios Mavroudis, Shuang Hao, Yanick Fratantonio, Federico Maggi, Christopher Kruegel, and Giovanni Vigna. On the Privacy and Security of the Ultrasound Ecosystem. *Proceedings on Privacy Enhancing Technologies*, 2017(2):95–112, 2017

To the best of our knowledge, this is the first study that was published on the ultrasonic communications ecosystem. It examines the different encoding techniques used in ultrasonic communications along with existing and future use cases of the technology. We introduce a number of novel attacks that abuse this newly established channel to breach the users' privacy and implement a series of countermeasures for end-users. The author designed and implemented all the attacks and defenses presented in this paper, with the exception of the Android patch that was conceived and implemented by Yanick Fratantonio. The original idea for the study was contributed by Federico Maggi. The software was released under the Apache 2.0 open-source license and is available at https://github.com/ubeacsec.

Vasilios Mavroudis and Jamie Hayes. Adaptive Traffic Fingerprinting: Large-scale Inference under Realistic Assumptions. 2020

This paper examines the adversarial assumptions made by past works on traffic fingerprinting and proposes additional parameters that should be considered when evaluating a fingerprinting methodology. The author was the primary investigator of this work and was responsible for compiling the three fingerprinting scenarios as well as the design and implementation of the fingerprinting adversary. The experiments were designed collaboratively with Jamie Hayes and implemented by the author.

Christos Andrikos, Lejla Batina, Lukasz Chmielewski, Liran Lerman, Vasilios Mavroudis, Kostas Papagiannopoulos, Guilherme Perin, Giorgos Rassias, and Alberto Sonnino. Location, location, location: Revisiting modeling and exploitation for location-based side channel leakages. In *Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part III*, volume 11923 of *Lecture Notes in Computer Science*, pages 285–314. Springer, 2019

This work revisits the location side-channel and examines novel exploitation techniques that outperform the current state-of-the-art. This project was conceived and led by Kostas Papagiannopoulos. The author handled the raw data (i.e., timeseries) collected from the microprobes, and compiled them into 2D matrices to be processed by the neural networks classifiers. Alberto Sonnino and the author, designed and executed a series of inference experiments with pre-trained neural networks on the ARM Cortex-M4's SRAM. Moreover, the author designed, trained and evaluated the custom neural network classifier based on convolutional layers.

Vasilios Mavroudis, Andrea Cerulli, Petr Svenda, Dan Cvrcek, Dusan Klinec, and George Danezis. A Touch of Evil: High-Assurance Cryptographic Hardware from Untrusted Components. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA*, pages 1583–1600, 2017

This work proposes a novel and practical high-assurance system architecture that retains its security properties in the presence of several malicious or faulty components. The author led the effort of designing and implementing the cryptographic protocols, prepared the performance and scalability experiments and processed the results. George Danezis had the original idea and direction of the project, and contributed in the protocol design. Petr Svenda and Dusan Klinec worked on debugging and

optimizing the JavaCard implementations to further improve our experimental results. Dan Cvrcek built our hardware prototype and provisioned the smartcards before the experiments. Andrea Cerulli contributed the security proof for our threshold signature scheme.

1.3 Other works

This section lists works that are not part of this thesis and were published by the author either individually or in collaboration with others. While not all the publications listed here are related to the problem of information leakage, many of them investigate relevant topics. Our research on market manipulation [24, 25, 26] is inspired by past works on the effects of information leakages on the market efficiency [27] and study how sophisticated traders exploit infrastructure inefficiencies to gain an informational advantage. Moreover, [28] proposed a privacy-preserving scheme that aims to improve the state of the art on data privacy and non-interactive publicly verifiable computations.

Vasilios Mavroudis, Karl Wüst, Aritra Dhar, Kari Kostiainen, and Srdjan Capkun. Snappy: Fast On-chain Payments with Practical Collaterals. In *27th Annual Network and Distributed System Security Symposium, NDSS 2020, San Diego, California, USA*. The Internet Society, 2020

Vasilios Mavroudis and Petr Svenda. JCMathLib:Wrapper Cryptographic Library for Transparent and Certifiable JavaCard Applets. In *Proceedings of the 1st International Workshop on lightweight and Incremental Cybersecurity Certification, CyberCert 2020, all-digital*. IEEE, 2020

Vasilios Mavroudis and Hayden Melton. Libra: Fair Order-Matching for Electronic Financial Exchanges. In *Proceedings of the 1st ACM Conference on Advances in Financial Technologies, AFT 2019, Zurich, Switzerland*, pages 156–168. ACM, 2019

Vasilios Mavroudis. Bounded Temporal Fairness for FIFO Financial Markets. In *Proceedings of the 26th International Workshop on Security Protocols*. Springer,

2019

Vasilios Mavroudis. Market manipulation as a security problem: Attacks and defenses. In *Proceedings of the 12th European Workshop on Systems Security*, pages 1–6, 2019

Vasilios Mavroudis and Michael Veale. Eavesdropping whilst you're shopping: Balancing personalisation and privacy in connected retail spaces. In *Living in the Internet of Things: Cybersecurity of the IoT*, pages 1–10. IET, 2018

Alexander Hicks, Vasilios Mavroudis, Mustafa Al-Bassam, Sarah Meiklejohn, and Steven J. Murdoch. VAMS: Verifiable Auditing of Access to Confidential Data. *CoRR*, abs/1805.04772, 2018

1.4 Responsible Disclosure & Ethics

In some of our works, we identified vulnerable or compromised production systems. In all such cases, we provided all the available information to the operators of the affected systems and worked with them to ensure that the problems had been patched before we publicly communicated our findings. We did not acquire or handle any user information, and all our experiments used data from publicly available sources that did not contain sensitive or personal information.

Chapter 2

Definitions & Preliminaries

Information leakage can be broadly defined as [32]:

> "The exposure of sensitive information to an actor that is
> not explicitly authorized to have access to that information."

While the above definition is generic and agnostic to the individual characteristics of the underlying system or protocol, leakages are better understood when studied within the context of the system or protocol affected. In fact, most works adopt definitions of leakage specific to the systems and protocols considered. For example, [33] studies the exposure of sensitive user data and introduces a metric that quantifies leakage by estimating it through the loss of privacy. In the same context, [34] uses differential privacy which defines and bounds leakage with regard to parameters ε and δ. Studies on hardware side-channel attacks focus on protecting a secret encryption key and thus define leakage as key bits recovered or the number of possible key candidates [35]. In the context of finance, information leakage is often considered to be the early exposure of information that can be exploited by a receiving trader, and not the release of the information in general [27].

We now provide the background on various technologies that are core to the systems and protocols we evaluate in later chapters.

2.1 User Tracking

Every time a user visits a website or other online resource, their behaviour provides highly specific information about their interests and needs. This information is

collected and analysed by advertisers who can then tailor their content accordingly.

2.1.1 Profiling

Behavioral targeting is a common practice in the advertisement industry, where the user's past activity is used to build a personalized profile on the user's interests. This profile, along with demographic data (e.g., gender, location), is then utilized by ad networks to better target their campaigns. A common way this information is passed on to advertisers is through *real-time bidding* (RTB) auctions where ad buyers compete for advertising inventory on a per-impression basis and decide their bids depending on the specific characteristics of the users (e.g., jewelry ads may have different prices depending on the user's gender) [36]. Since 2011, more than 80% of the North-American advertisers have switched to RTB for buying ad impressions [37]. Not surprisingly, this fine-grained data collection comes also with various privacy implications, and even though the data are not directly provided to the ad buyers, it has been shown that leakage of user profile information is hard to prevent [36, 38, 39].

2.1.2 Ultrasound Tracking Frameworks

Ultrasound tracking frameworks (e.g., [40, 41, 42, 43, 44]) are software components released by tracking service providers. These frameworks enable applications (e.g., an Android app) to perform user/device tracking using ultrasonic beacons (see Section 2.2). They are usually provided as libraries and are incorporated in the app owned by the client (e.g., supermarket loyalty apps). Such libraries expose proprietary methods needed for uBeacon-related operations (e.g., discovery, transmission demodulation, correctness verification) and almost always require access to the device's microphone.

2.2 Ultrasonic Beacons

Ultrasound beacons (i.e., uBeacons) are high-frequency audio tags that can be emitted and captured by most commercial speakers and microphones but are not audible by humans. These beacons encode a small sequence of characters and symbols, which in most cases serves as an identifier for fetching content from an external server or

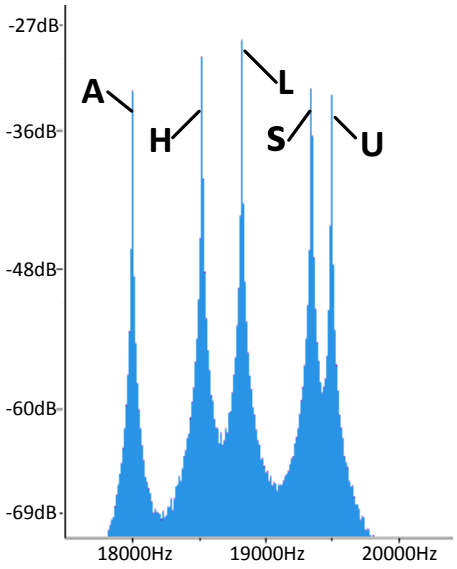


Figure 2.1: Spectrum plot of a uBeacon encoding five symbols using the MFSK frequency modulation scheme.

for pairing two devices together. Currently, there is no commonly accepted standard or specification, and thus each company uses its own beacon encoding format and communication protocol. As a result, there are multiple incompatible frameworks, providing varying levels of security. From a technical perspective, an ultrasound beacon has a duration of only few seconds but the exact encoding method varies greatly depending on the requirements of the application (e.g., range of transmission, volume of information to be sent) and the vendor.

One of the most commonly used modulation techniques is Multiple frequency-shift keying (MFSK) [45, 46] that divides the spectrum between 18,000 Hz and 20,000 Hz in smaller chunks and assigns each one of them to a symbol (e.g., a character). Parameter M determines the size of the alphabet. For instance, for a chunk size of $M = 26$ (75Hz), a tone in 18,000 Hz could correspond to character 'A', while one at tone in 18,075 Hz to character 'B' (see also Figure 2.1). Another parameter is the duration of each tone (i.e., symbol duration time), with many applications opting for a period of one second per symbol. Besides these, vendors often also incorporate some built-in error-prevention features. For example, many vendors use a set prefix to signal the beginning of a new uBeacon and disallow subsequent occurrences of the

same character. To decode an MFSK uBeacon, frameworks apply a fast Fourier transform and Goertzel's algorithm to each incoming signal. This process distinguishes the individual frequencies and retrieves the original characters/symbols [47]. It is computationally lightweight and can be performed on mobile devices with limited computational resources. In advertisement applications, the extracted sequence is then submitted to the company's backend and the corresponding resource (e.g., ad) is fetched over the Internet.

2.3 Internet Communications

A very large percentage of modern communications is conducted over the Internet with tens of thousands petabytes being transferred on a monthly basis. As a result, Internet communications have been the subject of an abundance of research works and a prime target for malicious parties that seek to breach the security or the privacy of the users. To prevent such attacks several protocols and systems have been established over the years.

2.3.1 The Transport Layer Security Protocol

The Transport Layer Security (TLS) protocol is a cryptographic protocol that is commonly used to establish secure two-party communication channels on the Internet. It is utilized in a wide range of applications such as web browsing, email, instant messaging and voice over IP, and employs end-to-end encryption between the two parties to protect the integrity and the confidentiality of the transmitted data. The two participants first negotiate the ciphersuite's parameters and then perform an one-time *handshake* to generate the cryptographic keys that will be used to protect the contents of their communication. Following a successful handshake, all the data exchanged is encrypted. To prevent man-in-the-middle attacks a client accessing a TLS-enabled server verifies the identity of the server through a public-key certificate issued by a trusted certification authority. For a detailed analysis of the TLS protocol please refer to [48, 49, 50, 51]. As the TLS specifications have undergone various changes over the years, we focus on the latest two versions of TLS: 1.2 [52] and 1.3 [50].

2.3.2 Fingerprinting Attacks

As Internet protocols matured, it became increasingly harder for adversaries to launch successful attacks against the underlying encryption schemes at a reasonable cost. Instead, another class of attacks emerged. Fingerprinting attacks aim to exfiltrate sensitive information from a given encrypted traffic stream by exploiting pattern in the encrypted data.

The majority of past works in the area has focused on website fingerprinting that targets users routing their traffic through anonymity networks (e.g., the Tor anonymity network) [23, 16, 17, 19, 20]. Such works aim to uncover the website visited by the user from a pool of possible websites that are of interest to the eavesdropping adversary.

Webpage fingerprinting is orthogonal to that goal as it aims to identify the specific page accessed by the user. Such attacks can be launched against both anonymity networks and standard end-to-end encryption protocols such as TLS (e.g., [53, 54]). So far, webpage fingerprinting has not received much attention in the literature, despite the fact that the Tor user-base is only a fraction of the total number of TLS users (see Section 3.1).

From a technical perspective, both adversaries exploit the leakages occuring through the various data-transmission patterns (i.e., byte counts, sender and recipient) to uniquely identify a loaded website or a webpage. However, *webpage* fingerprinting presents additional challenges, as websites tend to reuse the same template/theme in all their pages. Thus, webpages belonging to the same website exhibit only partially unique transmission patterns (i.e., reduced leakage volume), with the only differentiating factor being the content of each page. This limits the amount of useful identifying information one can extract from the traffic stream. In contrast, in website fingerprinting, the whole stream can be uniquely-identifying as websites usually use different themes/template.

2.4 Dimensionality Reduction

We now review a class of techniques commonly used to transform data from a high-dimensional space into low-dimensional representations. Traditional clustering algorithms such as k-means [55], Gaussian mixture models [56], and DBSCAN [57] operate on hand-crafted features designed to expose data structure and similarity. However, as the data dimensionality grows, uncovering the structure and designing reliable similarity metrics becomes a more difficult task. Transforming the data to a lower-dimension representation that retains structure is therefore an appealing goal.

Guo et al. [58] argue that the scope of shallow techniques for structure-preserving dimensionality reduction such as Principle Component Analysis (PCA) is limited. To counter this problem, a more recent line of work [59, 58, 60, 61] has focused on applying deep learning methods for dimensionality reduction upon which clustering can be applied. Broadly, deep learning-based clustering algorithms fall into two categories: (1) learning a lower-dimensional representation of the data and then applying clustering, and (2) jointly accomplish feature learning and clustering by defining an objective in a self-learning manner. One of the most widely used techniques is Stacked AutoEncoders (SAE) [61, 62, 63, 64, 59] which fall into the former category. For instance, Alom and Taha [65] used a SAE to learn a lower-dimensional data embedding and then use k-means clustering for intrusion detection. The drawback of SAE is that they often require layer-wise pretraining which can make their deployment expensive as a large number of training samples is required.

2.5 Neural Networks

Neural networks and stochastic gradient descent (SGD) are the core work horses behind the "deep learning revolution" [66, 67]. Given access to data, X with a label set Y, the goal of supervised machine learning is to learn the conditional distribution $p(Y|X)$. Neural networks define a parameterized function $f_w : X \to Y$ that when trained with SGD attempts to approximate the conditional $p(Y|X)$. A neural network is a composition of one or more layers of artificial neurons (i.e., perceptrons) such that given an input x, to layer i, the input to layer $i+1$ is given by:

$$\sigma(w_i^T \cdot x + b_i)$$

where w_i denotes the model weights that govern the strength of a connection between two neurons, b_i is a bias vector, and σ is a non-linear activation function. Popular choices of non-linear activation functions are ReLU(x) = max(0, x) and the sigmoid function. The final layer can be interpreted as a probability vector by applying the softmax function to outputs (sometimes referred to as logits). Given an input x, logits $f_w(x)$, where $f_w(x)_k$ is the logit value of the k^{th} class, and true label y, a *loss function* outputs a scalar based on how strongly an input would be assigned the true class label. The most common loss function to use in supervised neural network training is the log-loss defined as $-y\log(f_w(x)_y)$.

Model weights are then updated based on $\frac{\partial -y\log(f_w(x)_y)}{\partial w}$; averaging this loss over batches of inputs, computing the derivative with respect to model weights, and updating these weights in the opposite direction to this derivative is what is known as SGD and has produced state-of-the-art results on classification problems in a large number of fields [67, 68].

2.5.1 Convolutional Networks

Convolutional Neural Networks (CNNs) are defined as a mathematical workflow composed of a combination of convolutional, nonlinear, pooling (downsampling), and fully connected layers:

1. Convolutional layers: During the forward phase, input data are convoluted with some filters (features) to produce feature maps depicting where the features are actually located. During the backward phase, filter weights are readjusted (learned) so as to minimize a chosen loss function.

2. Pooling layers: The feature maps are downsized to reduce computational complexity and increase model robustness.

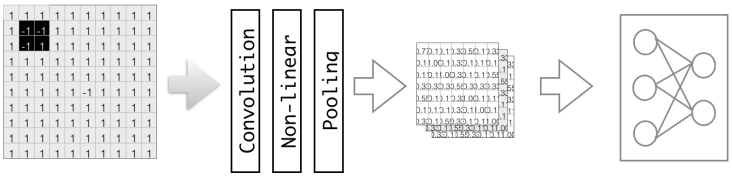


Figure 2.2: A basic convolutional neural network architecture, also featuring pooling, non-linear and fully connected layers.

3. Nonlinear layers: Nonlinear functions are invoked to provide normalization. Thus, scores generated by previous layers are converted to a probability distribution over the classes or are set to zero if they are negative.

4. Fully connected layers: Usually the final layers of the entire stack; a classic multi-layer perceptron that implements a voting scheme during the forward phase, while during the backward phase the weights are readjusted (learned) by minimizing a loss function.

Figure 2.2 depicts a sample CNN with the above basic layers. The key difference between CNNs and typical deep forward networks is that in CNNs, dimensionality reduction is an immediate result of the training process. Considering two or more inputs of high dimensionality, CNNs are able to learn the voting weights of the fully connected layers, as well as the features (filters) on the convolutional ones through back propagation. Due to their high performance, CNNs are one of the most well-studied neural network architectures and in many cases have been shown to outperform prior works in various problem domains [69]. [69]

2.5.2 Low-dimensional Embeddings

Neural network embeddings are learned representations of discrete variables (e.g., words or sequences of words) as continuous vectors in a low-dimensional space [70, 71]. Such representations significantly reduce the dimensionality of the input data, while they retain most of its information content. Embedding techniques are commonly used in recommendation systems as the reduction in the feature space makes learning easier. Similar benefits have been also observed in the context of classification, where the accuracy of a classifier and the volume of the training

data needed, depend heavily on the dimensionality of the input space [72, 73]. Besides these, dimensionality reduction can also provide additional application-specific advantages e.g., enhance the robustness of the classifier to perturbations [74, 75].

Chapter 3

Related Works

This section discusses prior works on research areas relevant to our contributions. As outlined in Chapter 2, the literature on information leakages is fragmented across several research areas as its definition varies depending on the specificities of each case. Additional research works, related to our findings, methodologies and the security of the schemes used are also presented in later chapters.

3.1 Traffic Fingerprinting

The literature on Internet traffic fingerprinting is very elaborate and covers several different manifestations depending on the context. Several works attempt to identify the browser/app or infer characteristics of the setup (e.g., operating system) used by the user [76, 77, 78, 79]. Such techniques are usually used to either identify malware initiating TLS connections or to keep track of various traffic and application trends on the Internet. For example, one of the most recent works in the area, [80] uses 8 billion unlabeled TLS sessions from several countries to identify popular enterprise TLS applications. This line of research does not aim to exploit any of the leakages present. In fact, many of these works argue that their techniques do not pose a threat to the end-users' privacy.

To the best of our knowledge, Cheng et al. [81] were the first to focus on privacy leakages on SSL 3.0 and introduced *webpage* fingerprinting in 1998. Their methodology was validated through a series of simulations on three small datasets, assuming static content. Mistry et al. [21] is another early work in the area which

manages to fingerprint a small scale website (<100 pages) by observing the transfer sizes of SSL packets. Following up on these works, Sun et al. [82] proposed a Jaccard-coefficient-based similarity metric between observed and collected encrypted traffic traces. This technique achieved a low false positive rate, however, their results were based on traffic from websites. Moreover, Danezis et al. [53] outlines a small-scale experiment on a static dataset (the exact size of the dataset is not reported), while Bissias et al. [22] and Cai et al. [23] propose improvements on the existing fingerprinting methodologies and verify their results on small (<100 webpages), static datasets too. Miller et al. [54] is one of the most recent works in the area. They use a dataset of webpages from various different websites and train and test their model on *subsets* of the whole set (up to 500 webpages each). They achieve a 90% accuracy on a top-15 adversary. They provide no experiments on larger websites or setups that would show how their technique could handle distributional shift. Finally, Dubin et al. [83] studies traces from video streaming services and proposes a technique that reaches 95% classification accuracy on a dataset of 100 Youtube videos.

Various insights used in webpage fingerprinting papers were motivated by works on *website* fingerprinting attacks against the Tor anonymity network [9]. Such attacks focus on inferring the website that the user has visited but not the specific webpage loaded [16, 17, 18, 19, 20]. Previous works on Tor-based website fingerprinting have employed standard machine learning techniques for classification such as k-NN [16], Support Vector Machines [17], random forests [19], and more recently neural networks [18, 20]. Overall, the state of the art in website fingerprinting is considerably more advanced compared to that of webpage fingerprinting with some works being able to fingerprint up to 3,000 separate classes [20]. However, to our knowledge, [84] is the only past work (both in website and webpage fingerprinting) that considers the problems of distributional shift and operational-cost and proposes a model that exhibits some adaptability. The main limitations of these works are: 1) they use the machine learning model for both feature extraction and classification, and 2) they test their techniques on sets of up to 100 classes. The former entails

that some form of retraining is still needed every time a webpage/website changes considerably, while the latter does not provide a reliable indication on the scalability of the technique to larger sets.

3.2 Audio Channels

It has been known for several years that audible and inaudible audio can be used for data transmission [85]. In fact, there are open-source implementations of software modems [86] and TCP/IP networking stacks [87] specifically tailored to the characteristics of the audio medium. Security research has focused primarily on the ultrasonic part of the audio spectrum (due to it being imperceptible by humans) and various studies have shown its capacity to operate as a low-cost covert or side channel [88, 89, 90, 91, 92, 93].

For example, [94] introduces an optimized modulation technique for ultrasonic frequencies that provides reasonable bandwidth and allows for inter- and intra-device communication. A noticeable side-effect of such channels are audible "clicks" that occur due to the sudden frequency changes. However, the authors in [94] manage to eliminate this effect too, making the channel completely imperceptible by humans. Petracca et al. [95] focus on modern mobile devices and conclude that a wide range of attacks that use audio as a covert channel are possible due to the lack of fine-grained access control. The authors demonstrate that a malicious app can use the phone's speaker to issue commands that would be picked up and blindly executed by other benign applications on the same phone through the microphone. For example, the app can trigger and control voice-activated apps (e.g., "OK Google, play some music" or "browse on evil.com"). In a similar vein, they show that the microphone can be also abused by malicious applications that eavesdrop sensitive information (e.g., while a screen-reading, text-to-speech application is "reading" a user's confidential email). In chapter 5, we also find that vulnerabilities in multi-application environments can adversely affect the security of other co-hosted applications.

Besides security research, audio channels have also found other uses in the industry. One instance is Intrasonics [96], a technology, which exploits the natural

echo filtering of the human brain to encode data in inaudible echo sequences.

3.3 Hardware Side-channels

In 2002, D. Agrawal et al. demonstrated that electromagnetic (EM) emanations can be used to retrieve information from cryptographic devices where the power side-channel is unavailable [3]. This work spawned a whole new branch of research on side-channel attacks, while much later served as the basis for location-based exploitation techniques. For example, the work of Sugawara et al. [97] demonstrates the presence of location-based leakage in an Application-specific integrated circuit (ASIC). In particular, they show that the power consumption of the chip's SRAM conveys information about the memory address that is being accessed. They refer to this effect as "geometric" leakage since it relates to the memory layout. Similarly, Andrikos et al. [98] performed preliminary analyses using the EM-based location leakage exhibited at the SRAM of an ARM Cortex-M4. The work of Heyszl et al. [99] manages to recover a secret scalar by exploiting the spatial dependencies of the double-and-add-always algorithm for elliptic curve cryptography. The experiments were carried out on a decapsulated FPGA, using near-field microprobes that identify the accessed register. Schlösser et al. [100] use the photonic side-channel in order to recover the exact SRAM location that is accessed during the activation of an AES Sbox lookup table. This location information can assist in key recovery, thus even cases of photonic emission analysis can be classified as location-based leakage. Moreover, countermeasures such as RSM [101] rely on rotating lookup tables to mask the data. Location-based leakage can identify which lookup table is currently under use and potentially weaken masking.

We distinguish between "location leakage" and "localized leakage". The former arises when knowledge of a component's (e.g., register, memory region) placement assists the recovery of the key. The latter occurs when the adversary is able to focus on analyzing the leakage of only a specific (usually small) region of the chip. For example, recovering the memory address accessed during an Sbox lookup implies a location leakage. Being able to measure the leakage right on top of a

processor's register file implies that the adversary is capturing localized leakage. Note that capturing localized leakage can be useful for data-based attacks as well as for location-based attacks. The works of Unterstein et al. [102], Immler et al. [103] and Specht et al. [104, 105, 106] acquire localized leakage via a microprobe in order to improve the signal-to-noise ratio of their data-dependent leakage. The work of Heyszl et al. [99] uses the same technique in order to improve the signal-to-noise ratio of their location-dependent leakage.

For clarity, we also distinguish between "location leakage" and "address leakage" [107]. Address leakage refers to leakages that occur through addressing mechanisms (e.g. the leakage of the control logic of a storage unit). Such leakages can be observed even far from the storage unit itself (e.g., at memory buses or at the CPU). In contrast, location leakages encapsulate *both* address-related and spatial effects. In location leakages, the leakage is caused by the address leakage as well as the leakage of the unit itself, which is often observed near it. We refer to the latter as "spatial leakage". For example, accessing a table in memory requires indexing and memory addressing in the CPU (address leakage). In addition, accessing causes the memory itself to be activated (spatial leakage). In most cases, the adversary is able to observe both types of leakage but it is often hard to distinguish the exact origin of the emanations measured.

3.4 Malicious circuitry

Malicious circuitry has been the subject of a large number of works with numerous new attacks and exploitation techniques being proposed in the last decade. For instance, the authors in [108] design two hardware trojans and implement a proof-of-concept attack against the PRINCE cipher [109]. The novelty of their attacks is that they use dopant polarity changes (first introduced in [110]), to create a hard-to-detect fault-injection attack backdoor. [111] also introduces a hardware trojan attacking RSA applications. In this attack, the adversary is able to use power supply fluctuations to trigger the trojan, which then leaks bits of the key through a signature. Another hard to detect class of trojans (inserted by fabrication-time

attackers) was introduced by Yang et al. in [112]. Such trojans leverage analog circuits and require only a single logic gate to launch their attack (e.g., switch the CPU's mode). Apart from these, evasive and stealthy triggering techniques have been proposed in [113, 114, 115, 116, 117].

Faulty components have been also reported in commercial and military hardware [118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129]. In all these cases, the errors were eventually attributed to honest design or fabrication mistakes but the systems were left vulnerable to leakages regardless. One instance of weak cryptographic hardware exploitation is introduced in [130]. In that work, Bernstein et al. study the random number generators used in smart cards and discover various malfunctioning pieces. For example, this attack allowed them to break 184 public keys used in "Citizen Digital Certificates" by Taiwanese citizens. Similarly, [131] uncovered several smart card chips with leaky cryptographic algorithm implementations.

To address the aforementioned threats, different approaches have been proposed. The most common ones attempt to either detect malicious circuitry or prevent its insertion. Detection techniques aim to determine whether any hardware trojans exist in a given circuit and involve a wide range of methods such as side-channel analysis [132, 133, 134, 135], logic testing [136], and trust verification [137, 138, 139, 140]. On the other hand, prevention techniques aim to either impede the introduction of trojans or make such circuitry easier to detect. Common prevention methods involve split manufacturing [141, 142, 143] which tries to minimize the circuit/design exposure to the adversary, logic obfuscation [144] and runtime monitoring [145, 146].

Other works consider verifiable computation architectures (such as [147]) to provide guarantees for the correctness of the computation on untrusted platforms. However, they come with a significant overhead and do not address the secure handling of secrets or the protection from side-channel attacks. The use of multi-party computation protocols to distribute trust between untrusted manufacturers during the fabrication process has been theoretically discussed in [148, 149]. Finally, a smaller body of work attempts to tackle the even harder problem of inferring

additional information about the malicious circuitry (e.g., its triggers, payload, exact location) [132, 133].

3.5 Fault-Tolerant Systems

Component redundancy and diversification are both key concepts of N-variant systems that aim to achieve high tolerance [150] against non-intentional faults. Such a system is the Triple-Triple Redundant 777 Primary Flight Computer [151, 152] that replicates all its computations in three processors and then performs a majority voting to determine the final result. The applications of N-variance in adversarial scenarios have been studied in only a few works that focused mainly on software attacks. In particular, [153] introduces a method for memory safety by generating randomized system copies which result in disjoint exploitation sets. This method can effectively thwart buffer overflow attacks. Similarly, [154] proposes a N-variant design that also aims to generate disjoint exploitation sets. Unfortunately, these methods can only protect against (potentially exploitable) unintentional errors and are not effective against fabrication-time attacks. Heterogeneous architectures with commercial off-the-shelf components have been also proposed in [155, 156]. However, the proposed designs provide protection against integrity attacks but not leakages as the computations are simply replicated between the different components of the system.

Another relevant line of work studies fault-tolerance in decentralized systems and the ability of the participating nodes to reach an agreement under specific latency and responsiveness assumptions. Notable problems in the area are state machine replication (SMR) [157, 158, 159, 160] and reaching consensus [161, 162] in the presence of *byzantine* (BFT) [163] or *fail-safe* failures. Interestingly, solving consensus and SMR is not equally expensive (computationally) in all cases [164]. Those problems are considered under three types of timing models (*synchronous*, *partially synchronous* [165] and *asynchronous* [166]) depending on the latency of delivering messages between the participating nodes. The asynchronous model is typically more challenging, in particular due to results such as the *FLP theorem* [167, 168] showing that it is not possible to guarantee the termination of a deterministic

protocol if at least one process may have a crash failure. In contrast, the same problem has solutions under the synchronous assumption [163].

Chapter 4

Internet Communications

This chapter studies leakages in the Transport Layer Security (TLS) protocol [52, 50], one of the most widespread protocols used by billions of people to protect the confidentiality and integrity of Internet communications. While TLS has been deployed in various applications (e.g., email, voice-over-IP, instant messaging), we focus on its primary use as part of the Hypertext Transfer Protocol Secure (HTTPS). In particular, we investigate whether encrypted traffic from TLS 1.2 and 1.3 leaks sufficient information for an adversary to attack the privacy of web users at a large-scale.

4.1 Introduction

The Internet has grown to host 1.2 billion websites and serves 161.3 Exabytes of traffic per month [169, 170, 171]. As a result, the security and privacy properties of communications over the Internet have been studied extensively, and various protocols and standards have been established. An abundance of works focused on the cryptanalysis of these protocols, improving their security significantly over time. In this chapter, we go beyond cipher cryptanalysis and study *traffic fingerprinting*, a class of attacks that exploit leakages in the traffic encryption protocols and extract sensitive information about the users or the contents of their communication. Unlike cryptanalytic attacks, traffic fingerprinting does not assume any flaws of the encryption scheme but instead uses patterns and meta-data leaked by the traffic stream itself.

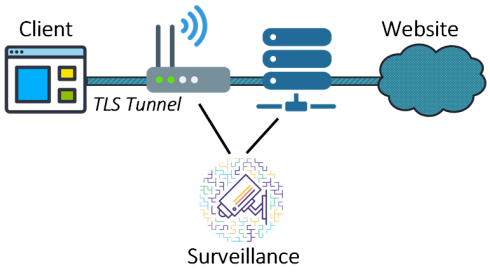


Figure 4.1: Illustration of a webpage fingerprinting scenario where the user loads a website through a TLS tunnel while an adversary eavesdrops on the encrypted traffic for surveillance or censorship purposes. Such a passive adversary can be someone sniffing traffic on a local wireless connection, an Internet router or the victim's Internet service provider.

Traffic fingerprinting is one of the most well-studied types of attacks against anonymity networks [23, 16, 17, 19, 20, 18, 84, 172, 173], where eavesdropping adversaries attempt to distinguish the *websites* visited by users. These adversaries extract features from the encrypted data exchanged between a targeted user and the entry node of the anonymity network without any assumptions of flaws on the network's encryption. For example, neural networks have been shown to be able to fingerprint websites based only on the sequence of packet lengths transmitted by each party and their respective direction [18, 84, 172].

Despite the fact that the TLS protocol is used by several billions of users, the literature on *webpage* fingerprinting is sparse and aged compared to that on *website* fingerprinting. This chapter extends the considerably smaller body of work on fingerprinting attacks against TLS [81, 53, 22, 23, 52, 54]. Fingerprinting adversaries against TLS have a different goal as they aim to uncover the specific *webpage* loaded by the user and not just its parent website. The latter would be trivial as TLS versions 1.2 and 1.3 reveal the IP address of the website visited by the user. However, TLS conceals the requested webpage by encrypting all transmitted data as well as the webpage path.

Nonetheless, both lines of research (webpage and website fingerprinting) share a mostly common methodology when evaluating techniques. In particular, they adopt scenarios that study the performance of the proposed fingerprinting model under the worse-case scenario for the user (optimal for the adversary conditions).

This is a common practice in the security literature as the accuracy of the model serves as a privacy upper-bound that the user can reliably assume under all possible circumstances. However, optimal fingerprinting scenarios are not always reliable indicators of a model's practicality. For example, a fingerprinting model that has a very high classification accuracy under some very specific optimal conditions is not guaranteed to remain as performant in other more realistic settings.

This leaves a gap in the literature on fingerprinting attacks and raises doubts about the degree of threat they pose [54, 76]. For example, despite the fact that webpage fingerprinting attacks are a known problem, generic claims about "privacy" on TLS appear in several specifications and industry documents [174, 175, 176, 177, 178, 179, 52]. Moreover, the TLS 1.2 Request for Comments (RFC) explicitly includes "privacy" in the primary goals of the protocol [52] but does not provide a definition for it:

> "The primary goal of the TLS protocol is to provide privacy and data integrity between two communicating applications."

Similarly, privacy is also mentioned in a variety of articles and technical reports [174, 175, 176, 177, 178, 179]. While TLS provides some "privacy" compared to plaintext communications, it cannot be assumed to reliably protect the user's browsing habits from sophisticated eavesdropping adversaries. On the other hand, the constrained scenarios considered in the literature ([53, 22, 23, 52, 54]) do not provide strong evidence that those attacks are practical and scale to larger websites, while they could potentially provide a false sense of privacy to Internet users.

In this chapter, we revisit those assumptions and argue that fingerprinting models should be also evaluated under non-ideal conditions. We compile a list of important factors that can directly affect the performance of a model and, consequently, its practicality. This list is not exhaustive but it includes some of the main difficulties faced by passive adversaries in modern deployments (e.g., large number of webpages, fast-changing contents). We then outline the basic properties that a fingerprinting adversary should have in order to be practical and argue that most of the existing

fingerprinting techniques fail to meet one or more of them. This is because those models were designed to operate optimally under static, non-changing conditions (e.g., constant webpage/website contents) and provide no adaptation mechanisms. Thus, an adversary has to constantly retrain the model in order to keep up with any form of distributional shift.

To evaluate if a realistic and performant adversary is possible, we use *adaptive fingerprinting* and study it under various adverse scenarios. This technique allows for rapid and inexpensive adaptation to distributional shift without the need for retraining. Our results show that it performs very well on static settings and that it retains similar levels of accuracy in scenarios where the fingerprinting targets evolve over time. Moreover, we find that neither TLS 1.2 and TLS 1.3 can reliably provide privacy in the presence of webpage fingerprinting adversaries, even in the case of websites with thousands of pages. Thus, users can rely on the TLS protocol to protect their credit card numbers and other private information (e.g., medical results) but not their browsing habits (e.g., eBay product pages, frequently visited subreddit webpages, online encyclopedias).

Besides attacks, fingerprinting techniques have also found applications in malware detection in network settings [180, 76, 77, 181, 182]. Network administrators employ fingerprinting techniques to identify malware, based on the TLS channels it establishes with its remote *command & control* servers (e.g., botnets using Twitter profiles to receive commands from their controllers [183, 184, 185, 186, 78]). In the rest of this chapter, we focus on surveillance scenarios due to the ramifications of such leakages to the privacy of individuals. However, we believe that our results could inform advancements in malware detection too. Overall, this chapter revisits the adversarial setting adopted in the fingerprinting attacks literature, discusses various new practicality factors and argues for scenarios that consider non-optimal fingerprinting conditions. While fingerprinting in such settings is more difficult, it allows for more reliable conclusions with regards to the practicality and the performance of the proposed models.

Adaptive fingerprinting can reliably classify thousands of webpages, has low

overhead and is robust to various forms of distributional shift (e.g., content, website, and TLS protocol version changes). Based on it, we show how an adversary can train their model *only once* and through a rapid and inexpensive adaptation procedure fingerprint webpages/classes that were not included in the original training set. Our findings indicate that the TLS protocol cannot provide privacy with regard to the users' browsing habits even in the case of large websites. Moreover, we demonstrate that fingerprinting attacks can be effective on websites with thousands of webpages, regardless of the website's details and the protocol version used (TLS 1.2 or 1.3).

4.2 Adversarial Setup

In this section, we introduce the threat model, the attack scenarios and the practicality constraints that we will consider in the rest of this chapter.

4.2.1 Threat Model

We assume a polynomially-bound *passive* adversary that can capture (but not tamper with) the packets exchanged between the client and the server. Such adversaries are used in the majority of the works on traffic fingerprinting [16, 17, 18, 172, 84, 23, 19, 20, 173]. The client communicates with a server over an encrypted channel established through TLS while the adversary intercepts some or all of the packets exchanged. For instance, the adversary may reside on the same home network, an intermediate Internet traffic router or the victim's Internet service provider (Figure 4.1). The goal of the adversary is to infer the specific *webpage* visited by the user (e.g., Wikipedia lemma, eBay product page). As neither TLS 1.2 nor 1.3 conceal the IP address of the webserver, we assume that the adversary is aware of the *website* that the user is visiting[1]. The above threat model is in line with the adversarial setup outlined in the specifications of the TLS protocol versions 1.2 and 1.3 [52, 50]:

[1]Even though an IP address may correspond to many websites (i.e., multihosting), this is neither guaranteed (e.g., large websites have dedicated servers) nor provides a provably large/secure anonymity set.

4.2.2 Realistic Fingerprinting Scenarios

We now focus on fingerprinting scenarios that provide a realistic representation of the conditions under which an adversary has to operate. While this may make it harder to design and implement effective attacks, it enables us to draw reliable conclusions about the capabilities of the adversary in real settings. In particular, we focus on three aspects of such scenarios: 1) Number of classes (e.g., webpages, websites), 2) Distributional shift (e.g., content updates), and 3) Shared resources (e.g., common HTML theme, shared images).

Number of classes

Past works on webpage fingerprinting considered scenarios where the user is assumed to visit a *fixed* set of known webpages while the adversary aims to infer which webpage was loaded [23, 81, 53, 22]. Unfortunately, their experiments were conducted on datasets of up to 500 webpages. Such datasets have been criticized as being unrealistically small [18] and led to doubts about the practicality of the proposed attacks, especially as many modern websites comprise of several hundred or even thousands of unique webpages. In comparison, recent works on website fingerprinting evaluated their proposed techniques to significantly larger sets (a few thousand websites) and showed that adversaries achieve a high performance under ideal conditions [18, 20]. Overall, we argue that fingerprinting techniques should be evaluated in at least one scenario with a moderate or large number of classes.

Distributional Shift

Another common assumption in past works on fingerprinting is *static* webpage/website contents. While assuming content invariability may look reasonable at first glance, it results in significant performance degradation in practice as pages change [172]. A model that is trained to classify a set of pages (e.g., Wikipedia articles, subreddits, eBay listings) will have to retain its accuracy as their contents get updated. This can be achieved either by retraining the classification model on the latest version of the webpages/websites or through other means. From an

adversarial perspective, the cost of keeping up with the ever-changing contents is directly connected to the practicality of the technique. For example, a model that needs to be retrained each time one or more webpages get updated is likely to incur large operational costs thus making the technique impractical, even if it achieves high accuracy. Overall, the degree of tolerance to distributional shift and the cost of adapting to changes are also important factors that must be considered when evaluating a fingerprinting technique.

Shared Resources

While the previous two factors concerned both webpage and website fingerprinting, webpage fingerprinting scenarios should also account for an additional parameter. It is common for the pages of a website to share a HTML theme (e.g., the same stylesheet, Javascript imports, background image files). This reduces the volume of unique information transferred in each page load, thus making it harder for the adversary to uniquely identify each webpage.

4.2.3 Practicality Considerations

We now introduce a list of requirements for a fingerprinting technique to be considered practical and realistic.

Accuracy & Scalability. An effective fingerprinting technique needs to provide high inference accuracy for at least medium-sized and preferably large-sized websites (with regards to their number of webpages). For example, a technique that achieves 80% accuracy on a set of 100 websites is not necessarily equally accurate when used on larger sets.

Adaptability. As discussed in Section 4.2.2, websites periodically add and remove webpages, as well as update their contents. Practical fingerprinting techniques must be resilient to such distributional shift and retain their accuracy [172]. Moreover, while adversaries may be able to cope with small page updates, it is not uncommon for webpages to have most of their content gradually replaced (through small but

frequent updates). This gradual process leads to a large distributional shift where the current version of a page has a very small overlap with the version the model was initially trained on. The practicality and the performance of a fingerprinting technique depend on its ability to adapt to such changes (e.g., frequent retraining, low generalization error) and the operational cost this entails.

Provisioning & Operational Costs. Making inferences from traffic traces should come at a reasonable operational cost (i.e., in time and computational resources), while provisioning the fingerprinting model may have a larger one-off cost. Minimizing these costs results in more practical and easily-applicable models

Protocol-agnostic. While past works have focused on a specific protocol version, it is advantageous for a practical adversary to be able to fingerprint webpages/websites regardless of the underlying protocol version used by the user. For example, a fingerprinting deployment that is tailored to only one protocol version of the TLS protocol could potentially be temporarily circumvented by switching to a different version (e.g., from TLS 1.2 to 1.3) or even to a different ciphersuite than the current one. This is not a strict requirement (protocol-specific attacks can be also very effective) but we consider this a desirable (albeit not necessary) feature for highly-transferable models.

4.3 Adaptive Fingerprinting

Our proposed methodology allows adversaries to fingerprint webpages from non-static, changing websites. The core components of our system (Figure 4.2) are the embedding neural network and the classification algorithm that attributes samples to classes (i.e., traffic traces to webpages). Its operation comprises of three processes: *provisioning*, *fingerprinting*, and *adaptation*.

The computationally demanding provisioning process takes place only once, while the lightweight fingerprinting and the adaptation processes are executed iteratively throughout the lifecycle of the deployment. This is primarily possible due to the generic nature of the *embeddings* generated as part of the *mapping* step (Section 4.3.2). The following sections provide the details of these operations.

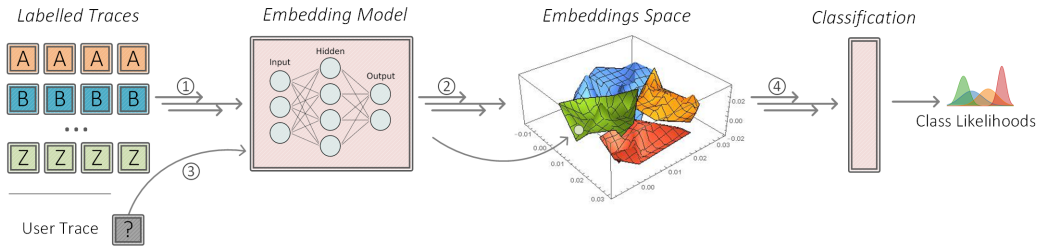


Figure 4.2: The eavesdropping adversary maintains a dataset of *labeled traces* from the webpages they monitor. These traces are processed by the *embedding neural network* and form the set of reference points. The reference points are then used to classify the user's traffic based on a proximity-based algorithm (e.g., k-nearest neighbours). Optionally, the adversary can keep populating the dataset with new reference points to stay up-to-date with the latest version of the webpages, without the need to retrain the embedding model.

4.3.1 Provisioning

Before the system is usable, the embedding neural network that reduces the dimensionality of the input traffic traces needs to be trained. Our training process is illustrated in Figure 4.4 and involves four steps.

Data Collection & Preprocessing.

Initially, the adversary compiles a list of webpages (preferably from the website(s) to be fingerprinted) and then proceeds to repeatedly load each webpage several times. For each visit, the network traffic between the client and the server is stored in a packet capture file (pcap file) and placed in a library of *raw* traces.

Following the collection of the raw traffic traces, the adversary processes them into sequences of integers (Figure 4.3). Each sequence corresponds to one of the IP addresses that transmitted data during the pageload and contains the byte-counts sent by that IP address over time.

In particular, each time an IP addresses sends out traffic, the new byte-count is appended to the corresponding sequence while the rest of the sequences are appended with a zero-count element. This is done to preserve the relative order of the transmissions. If an IP address sends more than one consecutive packets (i.e., no traffic from other IP addresses is interleaved), the byte-counts of those packets are aggregated and only their sum is appended.

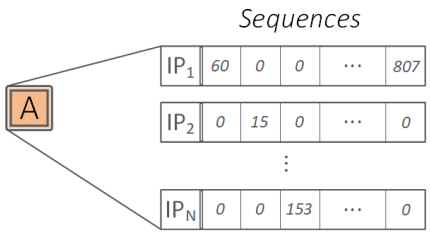


Figure 4.3: Illustration of how a network traffic of a pageload (labelled "A") is converted into IP sequences. Large websites often load various parts of their pages (e.g., JavaScript files, images) from different servers (e.g., for load balancing). Thus, each time the webpage is loaded, the client establishes TLS sessions with and fetches content from *several* different servers. Each sequence corresponds to the bytes sent by one of these servers while the first sequence always corresponds to the user.

Unlike our approach, prior works on website fingerprinting represent the data exchange as a single sequence where incoming packets are denoted by their byte-count and a negative sign, while outgoing by the byte-count with a positive sign [17, 19, 20]. This is equivalent to using only two IP sequences, one for incoming and one for outgoing traffic. The reduction in the number of sequences is because anonymity networks (e.g., Tor) conceal the IP addresses involved in a pageload as all the traffic is routed through an entry node of the network. In contract, TLS does not protect the IP addresses of the servers involved in a page load (e.g., user's client, main Wikipedia server, servers for auxiliary JavaScript files and images).

Following this step, the sequences can be optionally *quantized* to eliminate noisy artifacts (e.g., small differences in the byte counts). At the end of this process, the adversary has a dataset of labeled traces (each trace is a set of IP sequences corresponding to a single page load) that can be used to train the neural network (leftmost block in Figure 4.4).

Pair Generation

Given the dataset of labeled traces, the adversary generates *positive* and *negative pairs*. Positive pairs comprise of two traces corresponding to the same webpage, while negative pairs to different ones. The most straightforward strategy to generate pairs is at random, while more advanced techniques have been also proposed in the

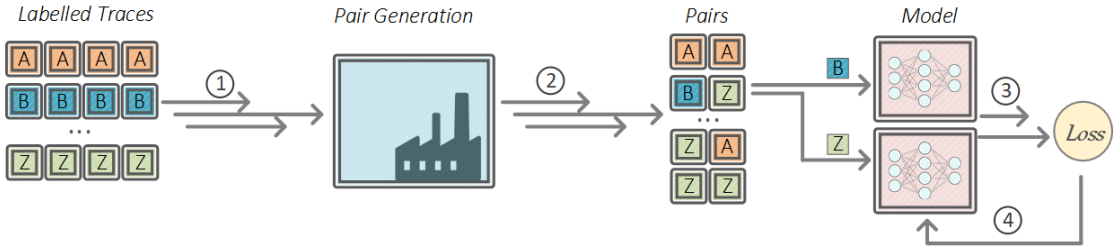


Figure 4.4: To train the embedding model, we use a dataset of labeled traffic traces that originate from the same website (e.g., Wikipedia). Using that set, we generate pairs of traces from the same class and from different ones (i.e., positive and negative pairs). These pairs are then used to iteratively train the model until sufficient accuracy has been achieved.

relevant ML literature (e.g., Hard-Negatives, Semi-Hard-Negatives [187, 188, 189]). The pairs are labeled based on the similarity of the samples (1 for similar, 0 for different) and are then used to train the embedding model.

Training

In this step, we train the machine learning model to produce embeddings that are in close proximity when the input traces originate from the same webpage, and far-apart otherwise. Intuitively, the role of the *embedding network* is to extract robust features that are less sensitive to artifacts (e.g., packet re-transmissions, non-deterministic resource loading order) and map the samples in the *embedding space* (Figure 4.2). As outlined in Section 2.5.2, classification algorithms (e.g., k-nearest neighbours) that rely on the distance between the samples (e.g., euclidean, cosine) perform significantly better in low-dimensional spaces compared to when they operate on the original high-dimensional feature space. The specific architecture of the neural network and its training details depend on the needs of the adversary and the use case.

Following the methodology outlined in [190, 191], for every training pair, we embed the two input sequences and compute the *similarity* of the two embeddings. For positive pairs, the similarity must be approximately equal to 1, while for negative pairs approximately equal to 0. To estimate the correctness of our model and update the network parameters accordingly, we compute the *contrastive loss* [191] given by

the formula:

$$\mathscr{L}(d,y) = yd^2 + (1-y)\max(margin - d, 0)^2 \tag{4.1}$$

where d is the (euclidean) distance between the two embeddings e_1 and e_2 ($d = ||e_1 - e_2||_2$), y is the known similarity label of the pair and the *margin* is a user defined parameter used to improve the separation between the different classes in the embedding space (i.e., dissimilar pairs should have a distance at least equal to the *margin*). The training process is completed once sufficient performance has been achieved and produces a model that can determine if two traffic sequences originate from the same website based only on the leakages of the cryptographic protocol used.

Initialization

Following the training of the embedding model, the system is populated with data that serve as reference points when classifying unlabeled traffic traces captured by the adversary. The adversary compiles a list of the webpages they intend to fingerprint, crawls them and and embeds the traffic sequences to generate a *reference set* of labeled embeddings (steps 1 and 2 in Figure 4.2). The reference set is then stored and used every time an unlabeled traffic trace is classified.

4.3.2 Fingerprinting

Given an initialized deployment with a populated *reference set*, the adversary can then proceed to fingerprint unlabeled samples captured from the user's traffic.

Capturing and Mapping

Depending on the setup, the adversary may capture the user's traffic at an Internet service provider (ISP) level or may reside in the same network and thus capture the traffic locally. Upon converting the packet capture into sequences, the adversary uses the embedding model to map the unlabeled sequence into the embedding space (step 3 in Figure 4.2). As outlined in Section 2.5.2, the embeddings generated for each sequence are continuous vectors that represent the packet exchange in

a low-dimensional space. It should be noted that, while this step determines the spatial proximity of the embeddings (based on their characteristics), the process is completely label-agnostic. This provides greater flexibility to the whole system as the embedding model does not need to be retrained if the labels change. In contrast, the majority of past works perform both the feature-extraction and the classification through the same model (e.g., convolutional neural networks [18]), thus fitting it specifically to the labels seen during the training. This is an important difference with past works as it minimizes the memorization of the specific characteristics of the webpages in the training set. In Section 4.5, we examine how accurately the embedding model can map sequences from webpages never seen during training.

Classifying

The adversary then classifies the embedding that corresponds to the user's traffic trace (step 4 in Figure 4.2). Intuitively, each captured sample is classified based on the labeled traces (reference points) that are in its proximity in the embedding space. The distance metric and the classification algorithm can be freely chosen by the adversary. In most cases, the algorithm outputs a list of the most probable labels for the examined sample and the frequency each one of them occurred (i.e., number of samples in proximity with that label).

4.3.3 Adaptation

Besides the initialization and the fingerprinting processes, our methodology involves an optional adaptation process. It provides a computationally lightweight process that brings the deployment up to date with changing webpages and prevents performance degradation [172].

Initially, the adversary crawls and identifies the webpages/websites that have been updated. The adversary can sequentially visit the webpages or in cases of larger websites, use techniques for monitoring and detecting changes in millions of webpages that were originally developed for web-archiving purposes [192, 193].

Given one such page, the adversary loads it, collects a traffic trace and fingerprints it as outlined in the previous section. If the accuracy of the classifier is not

adequate, the adversary crawls the page several times and updates the labeled traces in the reference samples dataset. The decision to update the reference samples of a particular class (in case the contents of the page have changed) can be taken based on a user-defined accuracy threshold (e.g., maximum discrepancy from the accuracy of the freshly-initialized deployment).

The main advantage of this process is that it does not require any retraining of the model or of any other component of the system (unlike the majority of past works on fingerprinting [16, 17, 19, 20, 18, 53, 54, 23]). Retraining a machine learning model is a costly operation and would impede the scalability of the attack if it was to be executed every time one of the thousands of pages/websites is updated. Instead, adaptive fingerprinting enables the adversary to remain up to date with fast-changing pages through a short sequence of inexpensive and low-complexity operations.

4.4 Datasets

To better understand the performance of fingerprinting adversaries under non-ideal conditions, we evaluate our proposed fingerprinting technique on two datasets with TLS traffic traces: one with traces from Wikipedia and the other with traces from Github. We focus on TLS as webpage fingerprinting attacks can affect many more users and have received little attention in the relevant literature. Moreover, webpage fingerprinting presents some additional practicality challenges (compared to *website* fingerprinting) that have not been studied thoroughly in the literature (e.g., the effect of shared HTML templates across all the pages of a website).

To the best of our knowledge, there are other no publicly-available datasets of that size with TLS traces, partially due to the little attention webpage fingerprinting has received. As outlined in Section 4.1, our goal is to enable further research into (adaptive) adversaries, scalability and webpage fingerprinting. For this purpose, we will publicly release both our datasets as well as our trained models. However, in order to limit potential abuse of our published data and models, we sought to crawl websites that:

- ❖ Do not have inherently sensitive contents (e.g., medical websites).

- ❖ Explicitly allow crawling (e.g., "crawl-delay" directive is present in the robots.txt).

- ❖ Have a large number of pages with varying types of content under a common HTML theme.

We identified *Wikipedia* and *Github* as services that fulfill the above requirements: Both websites have a large number of webpages that use the same theme but the text and media contents varying significantly between the webpages. They also explicitly permit crawling and their contents are generally not privacy-sensitive (we removed entries on potentially sensitive topics). Targets such as Amazon, eBay and Reddit do not permit crawling and public fingerprinting models trained on these websites have a high abuse potential.

Technical Details. Each dataset contains (encrypted) traffic traces as they would be captured by the eavesdropping adversary introduced in Section 4.2.1. We employed 100 Amazon EC2 instances distributed over five geographical regions (20 instances in each region). We opted for the "t3.small" instance type, which features 2 GBs of RAM and up to 5 Gbps network bandwidth.

The instances crawled a list of URLs, captured the generated traffic, stored it as a pcap file and processed it into sequences of bytes (Figure 4.3). To automate the crawling process, we used Python 3.7 with the Selenium automation framework[2]. To determine the browser to be used with selenium, we ran a small-scale experiment that did not indicate significant differences in the captured traces between Chrome and Firefox. However, instances using Firefox exhibited decreased stability. For this reason and due to the substantial difference in their market shares, we opted to use Google Chrome. The instances loaded the webpages strictly sequentially using incognito mode. In addition to this, we made sure that there were no prefetched resources, history or caches. No page loads took place in non-incognito browsing mode to prevent artifacts in our traces from cached favicons [194].

Each instance ran only one crawling process that visited each URL on the list sequentially in a random order. Before each visit, the crawler launched a Tcp-

[2]https://selenium-python.readthedocs.io/

dump [195] process and then proceeded to load the page with Google Chrome. Upon waiting 10 seconds for the contents to fully load, the Tcpdump process was terminated and the captured traces were stored on a pcap file.

The Wikipedia dataset. Our Wikipedia dataset (i.e., *Wiki19000*) consists of encrypted traffic traces from 19,000 distinct Wikipedia articles. We randomly chose 20,000 Wikipedia webpages and removed stub articles, articles on sensitive topics and indexing pages. The remaining ∼19,000 webpages where placed in a list to be used by the crawlers. To diversify our traces, each crawler shuffled the list and visited each article only once in a random order. The crawling process lasted approximately three days and costed approximately $300, thus making it relatively inexpensive to replicate our data collection and further extend the dataset.

Wikipedia uses TLS 1.2 and the page contents are usually loaded from two servers (one for text content and another one for media resources). We examined the contents of the Wikipedia articles crawled over the period these three days and found only minor changes on some articles. In total, the resulting dataset contains 1,900,000 traffic traces (100 traces for each URL). Capturing 100 samples per class is on the lower end and is consistent with in some recent works [84].

The Github dataset. For our second dataset (i.e., *Github500*), we chose Github as it was one of the few websites that had deployed TLS 1.3 at the time of the data collection and permits crawling of its pages. Moreover, it features a moderate number of webpages (i.e., projects) all sharing a common HTML theme.

Github allows projects to display a README page with information on the project as well as with installation and usage instructions. The overlaying Github template is common for all the projects but the contents of each page are managed by the project's contributors. Such pages include text, images and sometimes videos. Images and videos are stored either internally on Github or on external servers. Our dataset was generated by visiting the top 500 Github project pages[3], 1,000 times each. Each crawler instance shuffled the list of URLs and then visited each Github page 10 times over the span of several hours. We chose to use the top-500 projects,

[3]https://gitstar-ranking.com/repositories

as actively maintained projects with substantial contributions almost always have a detailed README page with information. In contrast, a random selection of pages (similar to that of Wiki19000) gave us mostly README pages with either no or minimal content (e.g., a single command line to compile the project).

Github uses TLS 1.3 and exhibits increased variability across various dimensions. It employs a significantly distributed infrastructure and advanced load balancing techniques causing various discrepancies between subsequent pageloads of the same page. Moreover, the number of servers involved is heavily dependent on the contents of each project page (e.g., externally hosted images, scripts and media). Due to this variability of the traffic patterns, we opted to collect 1,000 traces per class (in line with [18]). The dataset contains 500,000 traffic traces: 500 articles visited in random order 10 times by 100 crawler instances. Similarly with Wikipedia, we observed that Github project pages were not updated frequently (e.g., on an hourly basis) nor radically as they mostly provide compilation and usage details.

4.5 Experimental Evaluation

In this Section, we evaluate our proposed methodology by deploying and testing its performance on real data. We use three scenarios that simulate real-world fingerprinting setups with non-optimal conditions for the adversary. We focus on webpage fingerprinting scenarios as such attacks 1) have been systematically overlooked in the literature (cf. website fingerprinting attacks), 2) are more severe as they can affect many more users (i.e., the number of Tor users compared to that of Web users) and 3) pose a more pressing threat to the privacy of individuals. For example, a website fingerprinting attack could infer that the user is visiting Wikipedia, while webpage fingerprinting attacks uncover the exact article loaded.

4.5.1 Implementation & Parameterization

For the implementation of our neural network, we use the Python deep learning library Keras [196] as the front-end, and Tensorflow [197] as the back-end. For the data preprocessing and classification algorithm, we use Numpy [198] and Scipy [199], respectively.

Parameter	Value(s)
Input layer	30 LSTM units
# hidden fully connected layers	4 layers
Size of hidden fully connected layers	100 to 2000 neurons
Activation for hidden layers	ReLU [202]
Size of output layer	32 neurons
Activation for output	Leaky ReLU [203]
Optimizer	Stochastic Gradient Descent [68]
Dropout	0.1
Learning rate	0.001
Batch Size	512 pairs
Distance Metric	Euclidean distance
Contrastive Loss Margin	10

Table 4.1: The hyperparameters (top half) and the training parameters (bottom half) of our embedding neural network.

As outlined in Section 4.3, we use *contrastive loss* [191] to train our model on both positive and negative pairs. The *margin* of the loss function was set to be 10 and was determined through grid search ([200, 201]) among smaller and larger values. To measure the proximity of the traffic embeddings, we use the euclidean distance. The sizes of the hidden layers and the dimensionality of the produced embeddings were determined through grid search (see Table 4.1). The architecture of the embedding model and its hyperparameters were chosen carefully so as to maximize the fingerprinting performance and accuracy. However, as explained in the intro, past works have already shown that modern machine learning techniques can achieve a very high accuracy [16, 19, 18]. Thus, our focus is not to outperform all previous works but to study whether an adversary can retain such a high performance in a considerably larger scale while simultaneously alleviating the need for static targets.

For our classifier, we used the *k-nearest neighbours* algorithm with $k = 250$ for the first three experiments. We were able to achieve better classification results by adjusting the k parameter depending on the testing set but $k = 250$ produced consis-

tently good results regardless of the number of classes. An advantage of maintaining the same configuration across all three of our experiments on webpage fingerprinting is that we can compare our findings more reliably. In the four experiment, due to the small dataset sizes we opted for $k = 10$. In all our experiments, we report the average performance of 10 runs of the testing phase.

4.5.2 Experiment 1: Static Webpage Classification

In this experiment, we assume an adversary that aims to fingerprint the pages of a small- or medium-sized website where all the pages share the same HTML template. This first experiment studies the performance of our proposed technique against a website with mostly-static webpages and a moderate percentage of shared content (the HTML template and the graphics).

Using our methodology from Section 4.3, we train the adversary's embedding model on pairs of samples from our Wikipedia dataset. In particular, we use Set A (Figure 4.5) that includes 90 samples for each of the 6,000 distinct webpages/classes included in that Set. Upon completing the training phase, we deploy the model and use it to classify the samples in set B (Figure 4.5). The samples in set B originate from the same 6,000 classes but correspond to traffic traces that were not used during the training phase (i.e., not included in Set A). During the classification phase, we use set A as the adversary's labeled sequences corpus ($\sim$90 samples per class) and then use the trained model to classify the remaining $\sim$10 samples per class from set B (60,000 samples in total).

To better study the performance of our model, we run our recognition task on different versions of Sets A and B containing 500, 1,000, 3,000 and 6,000 classes respectively. As seen in Figure 4.6, out of a pool of 500 possible classes/articles, a top-3 adversary (i.e., the adversary is allowed to guess up to three classes) is able to correctly identify the Wikipedia article visited in more than $>$90% of the cases. Moreover, top-1 adversaries have 58% probability of correctly labeling the encrypted traffic trace, while top-10 adversaries are always able to correctly identify the page loaded. In comparison, [54] reported a top-15 adversary with accuracy up to 90%.

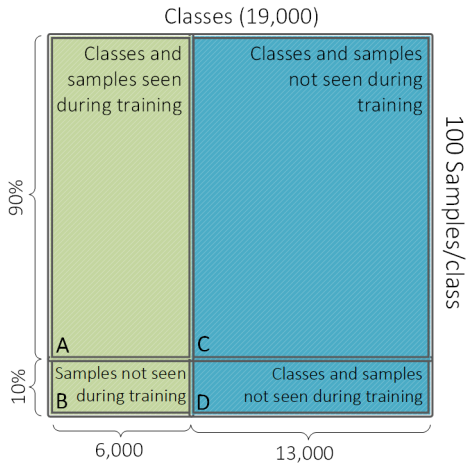


Figure 4.5: For experiments 1 and 2, we use our Wikipedia dataset. The dataset is split into four smaller sets, both across its classes and its samples. Experiment 1 trains the embedding model on Set A and then validates the accuracy of the produced embeddings on previously-unseen samples from the same classes (Set B). In contrast, Experiment 2 reuses the trained model from Exp. 1 (trained on set A) to embed samples from Set C as reference points. Experiment 2 uses Set D as its *test set*. Note that the classes in Sets C and D are not represented in sets A and B and vice versa. Moreover, no samples are shared between the Sets (e.g., no sequence from Set A is included in B, C or D).

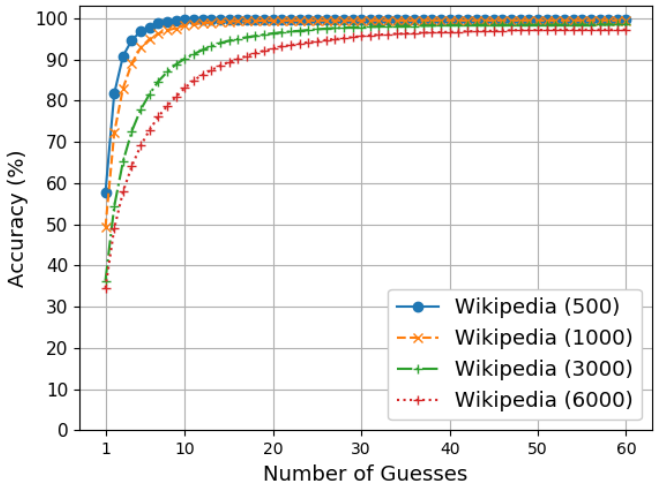


Figure 4.6: We evaluated the accuracy of the model in sets that required the adversary to attribute an encrypted traffic trace to a specific class from a set of 500, 1,000, 3,000 and 6,000 possible Wikipedia articles. For each class, we collected 100 samples, with 90 being used as reference points and the remaining 10 being classified by the model.

The top-15 adversary from [54] has been the state-of-the-art so far as later works did not report superior performance on datasets or similar size. Moving on to larger sets, we evaluate the classification accuracy of our model in slices of Sets A and B with 1000, 3000 and 6000 classes (Figure 4.5). In the scenario of 1000 classes, a top-1 adversary is able to correctly classify previously unseen samples with 50% accuracy, while in larger sets with 3000 and 6000 classes the same adversary achieves 35% accuracy. In the 1000- and the 3000-classes scenarios, the top-10 adversaries are able to correctly classify more than 90% of the samples. In the 6000-classes case, a top-20 adversary also achieved above-90% accuracy. In other words, an adversary who is allowed to choose 20 out of the 6000 labels (0.3% of the possible labels) has on average $> 90\%$ likelihood of correctly inferring the page visited by the user. In this and the following experiment, the classification of a single example in the testing phase required ≤ 2 seconds. Adversaries that need to improve the performance further can easily parallelize the mapping and classification steps.

Overall, we demonstrated that adaptive fingerprinting adversaries are scalable and can classify with high accuracy samples originating from a large pool of potential webpages. This result extends past works ([53, 54]) on webpage fingerprinting that presented adversaries capable of classifying up to 500 pages but did not evaluate on webpages with significant content overlap. We conclude that attacks against webpages/websites that share part of their content are realistic and can be launched even by adversaries with limited resources.

4.5.3 Experiment 2: Adaptability & Cross-class Transferability

One of the goals of our methodology is to investigate whether an adversary can retain their classification accuracy even in cases of distributional shift (e.g., content changes, addition of new classes) at a minimal cost. Such a characteristic would significantly exacerbate the severity of fingerprinting attacks as it would make it practical to fingerprint a dynamic set of webpages where classes are added, changed and removed. Our fingerprinting methodology decouples these two tasks and allows the ”encoding” model to remain class-agnostic, thus avoiding the need for any

costly retraining. Instead, the adversary can easily adapt to changes in the set of webpages/websites or the contents of the webpages by updating the reference samples in the corpus of labeled traces.

To simulate a scenario of extreme distributional shift, we design an experiment where the adversary is classifying a set of articles that is completely disjoint from the set that the model was trained on. This is the worst-case scenario for an adversary who classifies samples from a set of webpages that is completely disjoint to the set the training samples originated from. Such a difference between the training set and the testing set can occur in cases where the pages change drastically. For that purpose, we reuse the model trained in Experiment 1 (on Set A) to embed samples in Sets C and D. As shown in Figure 4.5, Set A does not overlap with Sets C (and D) as the former contains samples from 6,000 classes while the latter contain samples from 13,000 *different* classes. We consider our testing set to comprise Sets C and D, where Set C populates the adversary's dataset of reference samples and Set D contains the samples that need to be classified. As in Experiment 1, we investigate the accuracy of the model for slices of Sets C and D with different numbers of classes i.e., 500, 1,000, 3,000, 6,000, 13,000.

As seen in Figure 4.7, the classification accuracy of the adversary remains almost identical to the accuracy achieved with sets of the same size in Experiment 1 (i.e., without distributional shift). A top-1 adversary achieves 58% accuracy in the 500-classes set and a top-3 adversary $\sim$90% accuracy. Similarly, a top-1 adversary achieves almost 50% accuracy in the 1000-classes set and a top-4 adversary almost $\sim$90% accuracy.

This shows that the embedding model is learning the general leakage characteristics of TLS streams rather than simply memorizing patterns that apply only to specific pairs of samples or classes from the training set. For example, through manual inspection of the traffic traces collected, we observed that the transmission patterns of two samples from the same class can differ significantly. In one of them, the images were downloaded in multiple consecutive chunks of fixed length, while in the other they were fetched as a whole. Despite these differences, the model was

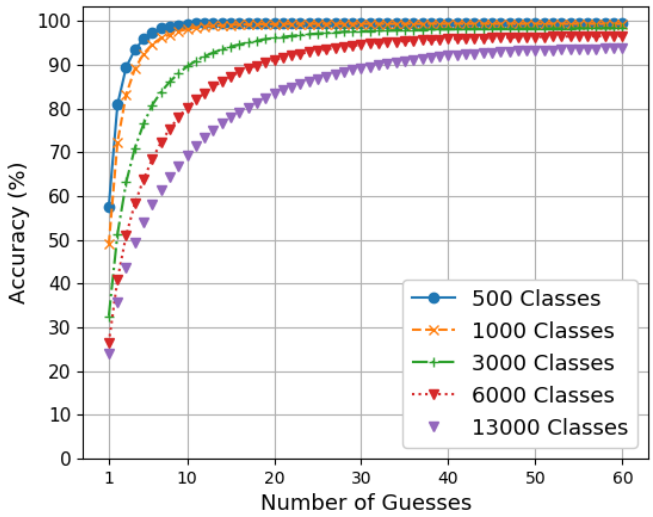


Figure 4.7: Accuracy of our fingerprinting model for varying numbers of classes (Wikipedia articles) that were never encountered during training. The model was trained on a fixed set of 6000 Wikipedia articles and evaluated on a completely disjoint set of articles whose size ranged from 500 to 13,000 classes. For each class, our dataset included 100 samples, with 90 being used as reference points and the remaining 10 being classified by the adversary.

correctly embedding the two samples in relative proximity.

Moreover, the adversary performs considerably well in even larger sets of new classes. In particular, a top-10 adversary achieved an accuracy of 90%, 80% and 70% in Sets with 3000, 6000, and 13000 classes respectively. This shows that our fingerprinting methodology can be reliably used to embed and classify samples from classes that were never encountered during training.

As seen in Figure 4.7, the adversary needs to increase their number of guesses (i.e., parameter n of a *top-n* adversary) as the number of classes increases in order for them to maintain the same level of accuracy (e.g., 90%). This is due to the increasing number of collisions between cross-class samples in the embeddings space. Intuitively, as the number of classes increases, the number of samples who are erroneously mapped in proximity to another class increases as well. However, as seen in Table 4.2, n increases slower than the number of classes. This implies that while the absolute number of collisions increases with the number of classes, the increase in collisions has a sublinear relationship with the increase of the number

Table 4.2: As the number of classes increases the accuracy of the embeddings decreases as cross-class collisions become more likely. Thus, adversaries need to increase parameter n to maintain the same level of accuracy. However, as seen in the rightmost column n has a sublinear relationship with the number of classes.

# Classes	Top-n	Accuracy	$\frac{n}{\#\text{Classes}}\%$
500	3	89%	0.6%
1000	4	89%	0.4%
3000	10	90%	0.33%
6000	20	92%	0.33%
13000	30	89%	0.23%

of classes. In other words, for any percent increase in the number of classes the adversary needs to increase their n by less than 1%.

4.5.4 Experiment 3: Sensitivity to Website themes and TLS versions

In this experiment, we examine the learning characteristics of our adaptive fingerprinting adversary. In particular, we evaluate 1) the effect of retaining multiple IP sequences, and 2) the degree that the model can sustain distributional shift across websites and TLS versions simultaneously.

As in Experiment 1, we train an embedding model on 6,000 Wikipedia articles (90 samples for each article) but we use it to classify traces from our Github README dataset (500 webpages from the top 500 open-source projects, Section 4.4). However, Wikipedia pageloads involve strictly 3 IP addressed (i.e., the client's browset, text server media server), while Github pages load resources from an arbitrary number of servers. As our model operates on a fixed number of sequences, we opted to represent the traffic as two sequences (i.e., traffic from and towards the user's browser). For this reason, we could not reuse our model from Experiment 1 (as it is trained to process three sequences) and had to retrain it to work on two sequences. We then ran the recognition task again on the original Wikipedia dataset (for a baseline) and on the Github dataset (both represented as two sequences).

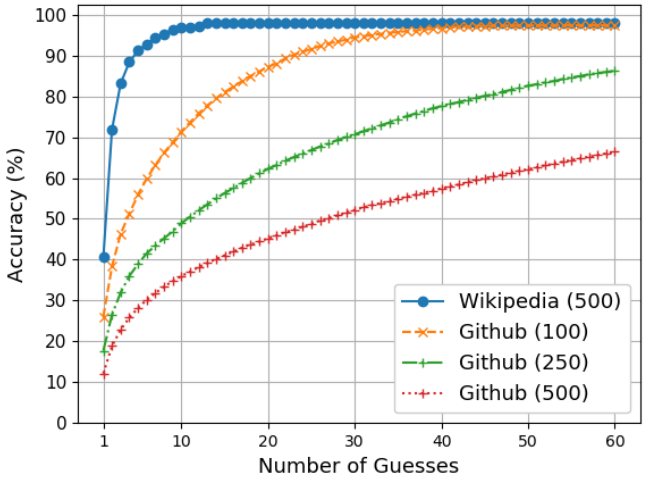


Figure 4.8: We trained our embedding model on two-sequence traffic traces from Wikipedia (TLS 1.2) and used it to embed and classify traces collected from Github (TLS 1.3). The model performs considerably better when operating on traces from the same website and with the same protocol version it was trained on. However, it still retains some of its accuracy. This indicates that some leakage characteristics are preserved even across very different setups.

Figure 4.8 illustrates the results of our experiment. We observe that the classification accuracy in the Wikipedia-500 set is reduced compared to that in the previous experiments where we used one sequence per IP. This shows that using just two traffic sequences (one for incoming and one for outgoing traffic) results in some information loss.

Moreover, the performance of the model on the three versions of the Github dataset (i.e., Github 100, 250, 500) shows that adversaries are able to retain a fair classification accuracy even in this case of extreme distributional shift across multiple dimensions. This indicates that some leakage characteristics persist across IP encoding, websites and protocol versions and can be exploited by sophisticated adversaries. Nonetheless, the reduced accuracy between Wikipedia-500 and Github hints that the embedding model is sensitive to at least one of the dimensions that were affected by the distributional shift.

4.5.5 Operational & Adaptation Costs

The above experiments study several aspects of modern fingerprinting attacks and show that adaptive fingerprinting adversaries are scalable and accurate, even under non-ideal conditions. We now discuss the costs of operating such a fingerprinting deployment.

As seen in Experiment 2, the adversary can use the *adaptation* process (Section 4.3.3) to swiftly swap the samples in the reference traces dataset with new ones so as to keep up with content updates or to include additional webpages in the set. This process does not involve any retraining as the embedding model can operate on any traffic trace even if it originates from a class not encountered during training. This simplifies the adaptation process to only a few low-complexity operations (i.e., collecting and embedding new samples) and enables the adversary to easily compensate for any distributional shift. Moreover, the deployment and the operation of the pipeline is inexpensive, as it requires only a small number of samples per class ($\sim$100) and only one training session for the embedding model. Nevertheless, the training of the model requires access to a computer with a capable Graphics Processing Unit card. However, this is an one-off cost (*provisioning* phase) that can be easily overcome with on-demand cloud computing resources.

In comparison, all past works on webpage fingerprinting assume a non-changing target set and would require some form of retraining to keep up with changes in the input distribution [53, 22, 23, 54]. While this cost may seem reasonable when considering small, fixed target sets ($<$ 500 webpages), it quickly grows (due to the constant retraining required) when considering several hundreds or thousands of changing pages.

4.5.6 Limitations & Open Challenges

Despite our best efforts, there are still aspects of our experimental evaluation that do not faithfully emulate all the challenges faced by a fingerprinting adversary. To begin with, the traffic traces were generated by our crawler and do not correspond to pageloads by real users. This is a common practice in traffic analysis works as the produced dataset is ensured to be diverse and balanced (e.g., all webpages are

loaded as many times). However, the true pageload distribution of real users is likely skewed with only a few webpages dominating the loads. Consequently, the success of a fingerprinting adversary will depend mostly on their inference accuracy on this small subset of frequently loaded webpages. In contrast, works that rely on balanced datasets examine the average accuracy of the adversary which may deviate significantly from the actual accuracy.

Moreover, all the traces were generated using the Chrome browser while in practice the users may use various other browsers too. To our knowledge, this aspect of webpage fingerprinting has not been studied. Interestingly, this factor is less important in *website* fingerprinting, as the Tor browser is used by the vast majority of the users. Additionally, our datasets do not include dynamically generated websites that may partially alter the contents on each pageload, thus affecting the accuracy of the embeddings.

Note also that we do not consider scenarios where the adversary has to classify out-of-distribution samples (open-world scenarios in website fingerprinting). Such scenarios are very common in website fingerprinting evaluations as the users can load any website available on the whole world wide web (WWW). The adversary cannot be assumed to have fingerprinted all WWW websites and thus needs to maintain a separate class for traces from an unknown origin. In contrast, webpage fingerprinting is concerned with websites that are practically finite. Thus, in many cases, it is possible for an adversary to fingerprint all the pages of a website. For example, Experiment 1 shows that an adversary could accurately fingerprint a website with up to 6,000 pages. However, this requires that the adversary keeps track of all the pages and updates their reference set each time a new one is added. This can become less practical in cases of websites with millions of pages. Thus, it would be beneficial to examine the performance of fingerprinting adversaries in scenarios with traces from pages that do not belong to the reference set.

4.6 Defenses

We now look into the space of potential defences against adaptive fingerprinting adversaries and examine the applicability of solutions from the existing literature. While introducing a new defence policy is beyond the scope of this chapter, our findings allow us to rule out some approaches and draw attention on others that show potential in thwarting such attacks.

One important observation is that adaptive fingerprinting attacks can affect both the users of anonymity networks and the users of the TLS protocol (i.e., a very large portion of the Internet users). However, the scope of potential defenses for the TLS protocol is limited to only those countermeasures that have only a very light impact on the bandwidth used. Intuitively, a protocol-level countermeasure with a 10% bandwidth overhead, would result in an approximately equal increase in the web-traffic bandwidth worldwide. For this reason, the majority of the defenses proposed for Tor are not directly applicable to TLS.

In the rest of this section, we focus on defenses against *webpage* fingerprinting attacks. This is due to the widespread adoption of the protocol and the limited coverage that TLS fingerprinting countermeasures have received in the literature (cf. fingerprinting defences for Tor).

As specified in Section 4.2.1, *webpage* fingerprinting aims to infer the specific page visited by a user from a set of pages all of which belong to the same website. This is a major difference to the website fingerprinting setup. In particular, each website can be treated as a separate entity and thus the defenses can be deployed and adjusted on a per-website basis. For example, a website with non privacy-sensitive pages (e.g., a list of hardware drivers) could decide to not deploy any countermeasure or optimize the deployment for low bandwidth impact (cf. for privacy). On the other hand, a website with sensitive content could use a more conservative configuration.

Being able to configure the countermeasure on a per-website basis, allows us to achieve protection without increasing the bandwidth overhead disproportionately. In comparison, defenses for website fingerprinting attacks rely on a cross-website anonymity set and thus require the deployment of the specific countermeasure by

several websites in order to be effective.

Furthermore, each website could configure the selected countermeasure so as to always guarantee a minimum anonymity set size (i.e., number of webpages that are indistinguishable) depending on the sensitivity of its content. We expect that smaller websites (< 500 webpages) could make all their pages indistinguishable at a relatively low bandwidth cost, while websites with more pages (e.g., Wikipedia) will have to split their content into smaller anonymity sets and aim for intraset indistinguishability.

Finding the optimal countermeasure, its policy and its configuration is an open problem that could be studied in future works. For example, a realization of such a per-website policy could be to use padding so as to conceal the byte length of the webpages loaded. This is approach conceals not only the length of each individual transmitted packet but also prevents timing attacks (e.g., with additional dummy packets). An advantage of this approach is that TLS already has this capability and thus would not require any protocol changes [50, 204]. Moreover, given that padding is a well-studied technique, we could draw useful lessons from prior works in the area (e.g., Pironti et al. [205] have shown that random-length padding is not sufficiently effective).

4.7 Conclusions

The widespread adoption of encrypted communications (e.g., the TLS protocol on the Internet) significantly reduced the scope of eavesdropping attacks and increased the security of their Internet communications. However, it did not completely eliminate the attacks that passive adversaries could launch. This chapter investigates how leakages in the TLS protocol. can be exploited to launch webpage fingerprinting attacks that target TLS traffic streams. It shows that sophisticated adversaries can use embedding deep neural networks to infer the webpages loaded by the user and for the first time demonstrates that they retain their accuracy under various types of distributional shift. Based on these findings, we argue that leakages in TLS is a pressing issue that has received disproportionately low attention. Especially,

when considering the number of users that are exposed to such attacks and the nature of the data that can be leaked (e.g., health information from users browsing condition-specific articles on medical websites).

A common solution to the privacy shortcomings of the TLS protocol is to use anonymity networks which offer greater privacy by concealing the users' communication meta-data. In the following chapter, we show how leakages in a seemingly unrelated channel can be exploited to launch attacks against even such networks. In particular, Chapter 5 focuses on leakages affecting the ultrasonic communications channel and discusses how adversaries can utilize them to either breach the privacy of its users directly or to launch deanonymization campaigns against anonymity networks.

Chapter 5

Ultrasonic Communications

In this chapter, we study the security of the ultrasonic communications channel and discuss how its shortcomings could be exploited to attack its users and seemingly unrelated anonymity protocols such as Tor.

In chapter 4, we investigated how an adversary can use embedding models to analyse the leakage of the TLS protocol and breach the users' privacy. Several such fingerprinting attacks have also been proposed for the Tor anonymity network. In fact, due to the emphasis Tor places on the users' privacy, the majority of the past fingerprinting works have focused exclusively on it. This chapter expands the literature on Tor towards a different direction. Instead of exploiting leakages in the protocol itself, we show how an adversary could exploit other communication channels to launch side-channel attacks against Tor.

Overall, the contributions of this chapter are twofold: 1) we study the security and privacy properties of the less-studied ultrasonic communications channel, and 2) showcase how leakages on one channel could be used as a side-channel to attack protocols on another channel.

5.1 Introduction

The increasing number of personal devices (e.g., smartphones, laptops, wearables) [206, 207] has created a new need for technologies that track the users across their different devices. To meet the demand, a set of novel tracking techniques based on ultrasounds has emerged.

Such techniques use a specific part of the audio spectrum to transmit information in the form of inaudible sound *beacons* and link devices that belong to the same user. Most of the technologies in the market use *near-ultrasounds*[1] with frequencies between 18,000 Hz and 20,000 Hz, which are inaudible to humans and can be handled by standard computer speakers and microphones. For simplicity, in the rest of this thesis, we will refer to all sound waves with frequencies higher than or equal to 18,000 Hz as *ultrasounds*.

One example of such an app is Google Cast [208] that uses ultrasounds to pair the user's smartphone to Google Chromecast devices, even if the devices are not part of the same wireless network. Apps such as Lisnr [44], ShopKick [209] and CopSonic [42] use ultrasonic beacons to track the in-store position of the customers' smartphones, study their behavior and serve them with relevant information and ads. Cross-device tracking solutions require an even more complex setup and capture ultrasonic beacons embedded into online/TV content so as to identify the different devices owned by a user and subsequently push targeted ads (e.g., SilverPush [210]).

We first explore the security and privacy implications of these technologies and study the surrounding application ecosystem for information leakages. We uncover various security shortcomings and show that malicious third parties can abuse ultrasound beacons to launch a wide range of attacks against end-users. More specifically, we find that existing ultrasonic communication protocols allow eavesdroppers to passively collect information on the online activities and interests of a victim user, while active attackers can even alter the user's profile maintained by the advertisers. Additionally, we find that state-level adversaries can coerce tracking operators to launch deanonymization campaigns against individual anonymity network users and communities (e.g., Tor anonymity network). To mitigate these risks, we design and develop an extension for the Chrome browser that selectively suppresses the frequencies within the ultrasonic spectrum. Moreover, we release a patch for the Android permission system that enables apps to request access only to the ultrasonic spectrum (and not to audible frequencies), as well as allows the user to grant or revoke access

[1] Cf. *ultrasounds* with frequencies higher than 20,000 Hz.

to that part of the spectrum on a per application basis. Finally, we discuss the need for standardization of the ultrasonic beacon format, and we envision a new operating system-level (OS-level) application programming interface (API) (similar with the Bluetooth low energy beacons [211, 212, 213] in Android) that implements in a single, trusted location the functionality to detect and decode beacons.

Overall, we conduct a thorough study of the security and privacy aspects of the ultrasound-based tracking ecosystem, identify two information leakage vectors and showcase their severity. We find that the introduction of a new communication channel increases the risk of leakages and that vulnerabilities in multi-application platforms can even affect applications do not directly make use of the channel. We also discuss potential countermeasures and argue that fine-grained permission control along with protocol standardization can prevent leakages in emerging communication channels without impacting their usability.

5.2 Ecosystem Overview

In this section, we study products that use the ultrasonic communications channel and discuss their technical details and capabilities. We organize the discussion by grouping the apps and the techniques used depending on their objective: 1) proximity tracking, and 2) cross-device tracking.

5.2.1 Proximity Tracking

Proximity tracking aims to determine the precise (e.g., at the vegetables' isle in supermarket XYZ) or relative (e.g., two meters from the device emitting the beacons) location of a user. It has found applications in *device pairing* and *marketing* [40, 41, 42, 43, 44, 208, 214]. In a pairing scenario between two devices A and B, A encodes a random PIN in an ultrasonic beacon and broadcasts it to all devices in proximity. If device B is within the broadcast range, it captures the transmission, decodes it and submit the PIN back to A through another channel (e.g., the Internet). This process allows B to prove its relative physical proximity to A.

In marketing applications, ultrasounds are used track the customers' in-store behavior through their smartphones. Vendors often incentivise their customers (e.g.,

through discounts) to install new beacon-receiving apps or simply incorporate beacon capabilities in their already existing loyalty apps. Ultrasonic tracking is suitable for various types of venues (e.g., casinos, museums, retail shops, airports) and relies on a grid of beacon transmitters installed in the premises. As a visitor moves around the space, their mobile device captures the location-specific beacons and submits them to the operators' backend. The backend processes the information and pushes notifications relevant to the location of the visitor (e.g., discounts for products in proximity). Compared to Wifi and Bluetooth solutions, ultrasound tracking deployments have a considerably smaller cost as they do not require expensive transmitters and can make use of the already existing audio infrastructure of the venue.

5.2.1.1 Google Cast

Google Cast [208] is an app developed by Google that reportedly utilizes ultrasound beacons to facilitate device pairing between smartphones and Google Chromecast devices [215]. A Chromecast device is a digital media player that, among others, enables mobile devices to stream content on a television or on an audio system. In this case, ultrasound beacons are used to prove physical proximity to the device, taking advantage of the inability of ultrasonic beacons to penetrate through walls. Ultrasound-enabled pairing allows users to connect to a Google Chromast device and stream their content even if their smartphone is not connected to the same Wifi network. Any user in the vicinity of a Google Chromecast device can request to pair with it through the Google Cast app. The device then responds by broadcasting an ultrasonic beacon carrying a unique 4-digit sequence. The Cast app captures the beacon through the smartphone's microphone, decodes it and submits it back to Google's backend through the Internet [215]. The pairing process is always initiated by the user and the device supports also alternative manual pairing methods.

5.2.1.2 Lisnr framework

The Lisnr framework [44] is another product that uses the ultrasonic part of the spectrum for proximity marketing and location-specific content, and has been already incorporated in various smartphone applications. For example, an American football

team used the framework in its official app (with hundreds of thousands downloads) to deliver content to fans in its home stadium [216, 217]. Moreover, the "Made in America Festival" Android App [218] (also available for iPhone) used Lisnr to stream real-time information to the event's audience. In both cases, ultrasounds were chosen as a convenient technology that makes use of the existing audio infrastructure and requires no additional transmission equipment.

From a technical perspective, the Lisnr framework implements all the necessary methods to capture ultrasonic beacons and fetches location-specific content from the service provider. Upon execution, the app incorporating the framework runs on the background and periodically accesses the device's microphone to listen for beacons. This is because the user is not expected to keep the app on the foreground for the whole duration of the events (e.g., games, concerts). Once an ultrasonic beacon is captured, the framework decodes it and extracts its content. If the content is a message, the app displays it on the device's screen. If the beacon encodes a content identifier, the app fetches the corresponding data from the company's backend through the Internet. While the ultrasonic channel could be used to transmit the data directly, its limited bandwidth and moderate transmission error rate make this alternative impractical. For users that do not want to receive constant notifications, some applications provide an option to deactivate the default listening behavior. However, this is an application-specific feature and is not enforced by the Lisnr framework.

5.2.1.3 Shopkick

Another real-world deployment using ultrasounds is the shopping application Shopkick [209] (available on Google Play Store). Shopkick aims to incentivise customers to visit and purchase products from specific stores and brands. The app is listening for ultrasonic beacons (i.e., walk-in tones) emitted by speakers installed in businesses and stores cooperating with Shopkick [219]. When a user visits a store, the app gets activated and reward points are credited to the user's account. These points can be later spent by the user for discounts or products at a reduced price.

To realise this functionality, Shopkick has developed its own framework for

ultrasonic communications. Shopkick uses beacons to encode the unique identifier of the store and the precise in-store location. When the app captures a beacon, it decodes the encapsulated identifiers, and submits them to the Shopkick servers, along with the user's details. Then, the company's backend verifies the validity of the "walk-in" and credits the user with the points. In contrast to the majority of the other implementations, the user must manually launch the app to access the microphone and start monitoring the spectrum for beacons. This is possible in this particular use case because the customers are strongly incentivized to use the app but would likely result in very low usage rates in scenarios where the users are not directly rewarded.

5.2.2 Cross-device Tracking

Cross-device tracking (XDT) aims to "follow" a user accross their different devices, and is currently used by many major advertisement networks to track users across different platforms. These techniques provide various degrees of precision, depending on the use case and the user identification method used. For instance, *probabilistic* XDT techniques are used in cases where the identification method does not guarantee high accuracy (e.g., when user fingerprinting techniques are used) and thus there is a degree of uncertainty regarding the set of devices owned by the user. On the other hand, *deterministic* XDT techniques achieve much higher precision but are often cumbersome. For example, deterministic techniques may require users to sign in to the advertiser's service from all the devices they own (e.g., "Sign in with Facebook").

Ultrasound cross-device tracking (uXDT) is also classified as deterministic but has considerably fewer requirements compared to other deterministic techniques. Figure 5.1 illustrates the actors participating in the ultrasound-based mobile advertising ecosystem and their interactions during an example tracking scenario. At first, the *advertising client* sets up a new advertising campaign and provides the ads to the *uXDT provider* along with the profile of the users to be targeted (❶). The advertising client may be a company or an individual that is interested in promoting their services or products (e.g., a supermarket chain), while the *uXDT provider* is the vendor providing the infrastructure for user-tracking. The uXDT provider then generates a unique inaudible ultrasonic beacon b and associates it with the client's

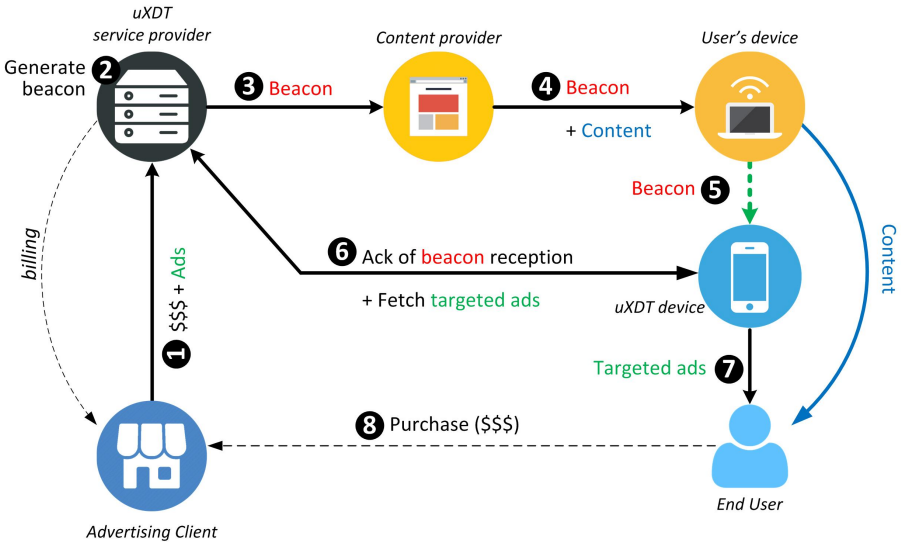


Figure 5.1: The interactions between the actors of the mobile advertising ecosystem during a beacon-enabled cross-device tracking scenario.

campaign (❷). *b* is then forwarded to the *content provider* (❸) chosen by the client (e.g., TV station, news website, media portal), who then embeds it in its content (❹). When a user accesses that content, *b* is emitted and captured by any of the uXDT-enabled devices in proximity (e.g., the user's smartphone) (❺). Those devices then report *b* to the backend of the uXDT service provider and receive ads that are targeted to the user's interests (❻). The uXDT device displays those ads (❼) prompting the user to visit the advertising client's store (❽).

uXDT is one of the most advanced uses of ultrasonic beacons, as it requires both sophisticated infrastructure (e.g., profiling algorithms processing millions of submissions) and a network of publishers who incorporate beacons in their ads/content. Thus, only a few companies are able to provide uXDT services. Additionally, the use of uXDT techniques is a controversial topic (e.g., [220, 221]) and is common for companies to not publicize the details of their tracking deployment.

5.2.2.1 SilverPush uXDT framework

One of the most widespread uXDT frameworks was developed by SilverPush and in April 2015 was tracking more than 18 million devices [221]. To study the operation of the framework, we reversed-engineered the History GK application [222] (an app incorporating the SilverPush uXDT framework), as neither the source code nor their

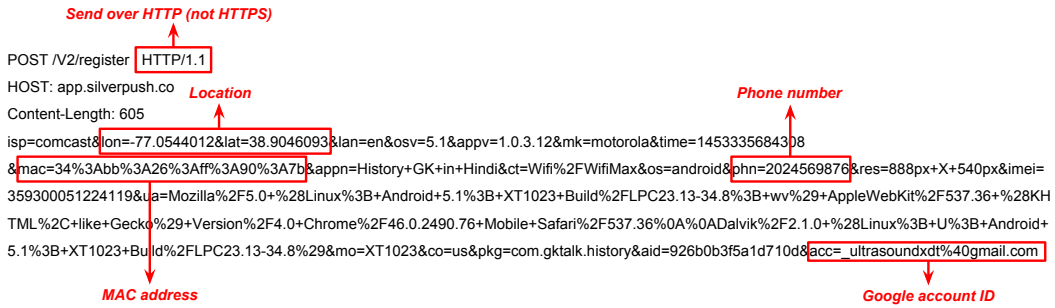


Figure 5.2: Example of an insecure, non-standardized tracking implementation: The framework collects sensitive information on the user (e.g., Google account ID, phone number, geolocation, the device's MAC address) and submits them over a plaintext HTTP connection to the tracking service provider's servers.

uBeacon specifications were made public.

The framework runs in the background and monitors the spectrum for ultrasonic beacons (every 20 seconds) though the device's microphone. To assist in beacon discovery, the app starts up automatically each time Android boots. When a uBeacon is captured, its sequence is decoded, verified and then submitted to the company's backend. Each such submission contains the Android ID of the device and the beacon sequence, and is performed through a plaintext HTTP connection. If there is no access to the Internet at the time, the information is stored and reported later. When the app is first executed, the framework extracts a wealth of identifiers including the user's phone number, longitude and latitude, the IMEI of the device, the Android ID, and the Google account ID (email address) and reports them to the company's backend (Figure 5.2). This information is used (1) to build and maintain the user's profile, (2) to display targeted ads, and (3) as filtering parameters for real-time bidding auctions (e.g., targeting users in a specific city).

During the installation process, the user grants permission to the app to use the microphone and to be launched at boot. However, the user is neither explicitly notified that a uXDT framework is being installed, nor that the microphone will be periodically active. We were not able to find any way for the user to disable the tracking functionality, apart from completely uninstalling the app.

5.3 Vulnerabilities & Attacks

This section examines different security and privacy shortcomings of the ultrasonic communications ecosystem and investigates whether they can be exploited to leak or corrupt user data. We first outline how an adversary can *inject* and *replay* beacons and introduce the concept of *bacon traps*. We then outline how these simple constructs can be utilized by sophisticated adversaries to realise more complicated attacks (Table 5.1). In all the scenarios considered, we assume that there is at least one user that has an ultrasound-enabled app installed on their device (e.g., a game app embedding a uXDT framework).

Beacon Injection & Replay. Due to the lack of authentication and replay-protection mechanisms in many of the apps examined, the ultrasonic channel is prone to injection and replay attacks. For instance, an adversary can trivially capture and replay a uBeacon from an existing advertisement campaign to any uXDT-enabled device. Capturing an ultrasonic beacon does not require any sophisticated equipment and can be performed using the recorder app of most commercial smartphones. Alternatively, the adversary can craft their own beacons and inject them to nearby ultrasound-enabled devices. To emit a uBeacon, the adversary can use a commercial smartphone, the audio infrastructure of a venue or any other device with speakers that support frequencies up to 20,000 Hz. Beacon replays have been already reported in production systems. For instance, users of the shopping application Shopkick [209] (available on Google Play Store), soon after the deployment of the system, started uploading collections of near-ultrasonic *walk-in tones* that other users could replay from their computer speakers to get the reward points. Large-scale injection attacks could also be used to influence *TV analytics*, where inaudible beacons are often used to track user engagement and behavior. An adversary could replay the beacons corresponding to a specific TV program to a large number of ultrasound-enabled devices resulting in inaccurate measurements and results. Unfortunately, replay-prevention mechanisms (e.g., monotonically increasing nonces) are impractical due to the very limited bandwidth of the channel.

Beacon Traps. A beacon trap is a maliciously crafted resource (e.g., webpage)

that an attacker uses to inject beacons into a user's ultrasound-enabled device. The advantage of beacon traps compared to plain beacon replay is that they do not require physical proximity to the victim's device. The attacker can use a small snippet of code that, when loaded on a browser, automatically reproduces one or more attacker-chosen ultrasonic beacons. For this technique to be effective, the adversary needs to attach the snippet to a resource that is accessed by the user. For example, an attacker could set up an innocuous-looking web page that, once visited, would play the audio beacon in the background. An attacker could also use existing cross-site scripting vulnerabilities present in third-party websites to emit beacons. Alternatively, an attacker could inject beacon-playing JavaScript (or HTML) in the users' traffic by launching a man-in-the-middle attack or by setting up a (malicious) Tor exit node [223, 224]. As a last example, an attacker could also send beacon through an audio message. Beacons emitted by traps are captured and handled by ultrasound-enabled devices as valid beacons (e.g., a uXDT app would *report* the beacon back to the uXDT backend).

Attack	Goal	Attacker capabilities
Unauthorized Audio Monitoring	Monitor parts of the audio spectrum abusing the user's consent.	◇ Developer of an ultrasound-enabled app, *or* ◇ Corrupt the source code of an ultrasound-enabled app.
Deanonymization	Retrieve the identity of anonymity network users.	◇ Physical proximity *or* lure the user to a *beacon trap*. ◇ Gain access to PII collected by the ultrasound service provider.
Profile Inference	Collect information on a user's interests and activities.	◇ Access to network traffic.
Profile Confluence	Collect information on a user's interests and activities.	◇ Physical proximity to the user.
Profile Corruption	Alter the user's profile maintained by the service provider.	◇ Physical proximity to the user or lure the user to a *beacon trap*.

Table 5.1: Types of attacks exploiting ultrasound-enabled tracking to leak and corrupt user information.

5.3.1 Unauthorized Audio Monitoring

Due to the granularity of the Android permission system, beacon discovery currently requires full access to the device's microphone (requesting access to a specific part of the audio spectrum is not possible). Consequently, this practice violates the least privilege principle as ultrasound-enabled apps gain also access to audible frequencies which are not relevant to their operation.

This could be exploited by a malicious developer who may request access to the microphone under the pretense that their app performs ultrasound pairing and then update their app to monitor the audible spectrum too. Such practices are not uncommon and researchers have found hundreds of malicious apps disguising as benign apps [225, 226]. Moreover, while the initial version of the app may be benign, a security breach in the development firm could allow third-party actors to exploit the permission by stealthily pushing an eavesdropping update. While such a security breach may occur in any app, the fact that the microphone permission has already been granted, makes the attack much easier for the adversary as the users will not be prompted for the permission again during the update.

Another side-effect of overprivileged apps is that benign vendors who use ultrasound technologies risk to be perceived as "potentially malicious" by users. Users are more concerned about their privacy after realizing that their decisions have put them at risk of data exfiltration [227], and could, thus be more hesitant with apps that require sensitive permissions such as access to the device's microphone. This could be further exacerbated by the wide discrepancies in the practices followed by companies when it comes to informing the users and providing opt-out options. In particular, our examination of ultrasound-enabled apps showed that in several cases no notice or opt-out option is given to the user (apart from the mandatory microphone permission request during installation).

5.3.2 Deanonymization

This attack enables an adversary to deanonymize one or more users of an anonymity network (e.g., Tor, I2P, VPN). For example, a journalist could uncover the identity of a whistleblower who uses the Tor anonymity network to leak highly confidential

documents. To perform the attack, a malicious journalist would need to either be in proximity to the whistleblower or lure them to visit a *beacon trap* (by social engineering or other means).

Figure 5.3 shows the stages of the attack and how the different entities interact within the ultrasound tracking deployment. First, the adversary *A* starts a campaign with an ultrasound tracking provider and captures the inaudible beacon associated with it (❶). Then, *A* creates a beacon trap that emits the uBeacon (❷). An example of a beacon trap can be seen in Figure 5.3 where the adversary incorporates the trap in a hidden service targeting Tor users. Subsequently, the adversary lures the anonymous user to visit the trap (❸). This step may involve social engineering or it may simply be that the trap uses a resource that the user visits often (i.e., a watering-hole attack). Alternatively, if the adversary is a Tor exit node operator, they can inject the source code of the trap on the requested web pages without any need to interact with the user. The process of becoming a Tor exit node is similar with that of a normal relay, and does not require many resources or a lengthy reputation-building period.

In the next step, the victim uses their "anonymous" device to load the resource (❹). Once this happens, the beacon trap is triggered and the device starts to periodically emit the beacon from its speakers (❺). The beacons are inaudible and thus the user is completely oblivious to the fact that ultrasounds are being emitted and that a deanonymization attack is being carried out. Simultaneously, any ultrasound-enabled device in proximity (e.g., the user's smartphone) is actively monitoring the spectrum for ultrasonic beacons. Once it detects one, it verifies its validity and reports it (along with a number of unique user/device identifiers) to the provider's server, where the data get stored (❻).

Adversaries then seek to obtain the unique identifiers of the user who reported their beacon at the specific time and day the beacon trap was triggered (❼). State-level adversaries (who are within Tor's threat model) can achieve this through a subpoena or a court order to the tracking service provider. Once the provider responds with the information, the real identity of the user can be trivially uncovered. For instance, uXDT frameworks often use the *international mobile equipment identity*

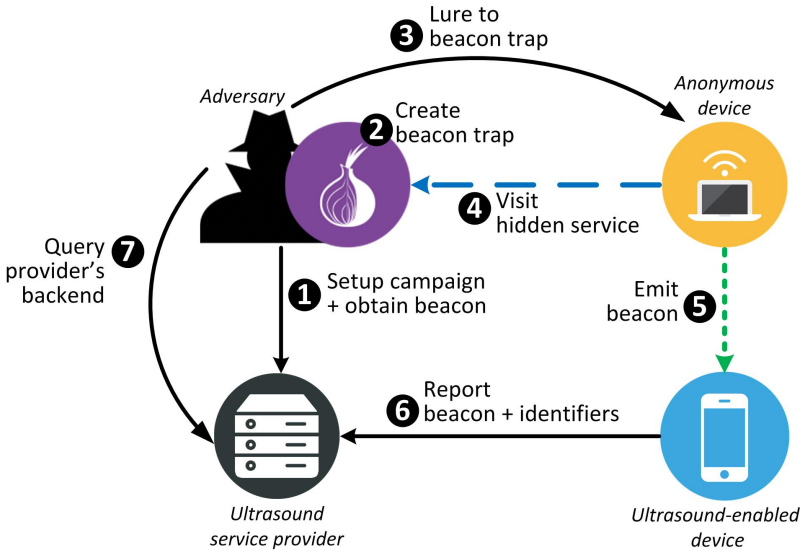


Figure 5.3: Steps of a deanonymization attack against a user of an anonymity network.

(IMEI) number as unique device an identifier. An IMEI can be easily attributed to a real person by the telecommunication operators [228, 229]. Moreover, uXDT service providers often store multiple unique identifiers, which make it even easier for the adversary to infer the identity of the user (e.g., the Google account name).

To practically validate our proposed attack and evaluate its effectiveness, we conducted a proof of concept attack against a demo user. This enabled us verify that a state-level adversary with access to the ultrasound service provider's data could indeed trap and deanonymize a Tor user. We hosted our beacon trap in malicious hidden service under our own .onion domain and had our PoC victim load the webpage using the Tor browser (version 6.0.1) configured with the default settings (step ❹). We provided to the victim a smartphone with one uXDT-enabled app from the Google Play store installed. However, since we didn't want our experiments to interfere with production systems, we redirected the traffic from the app to our own server. This allowed us to accurately and safely conduct numerous experiments. Figure 5.4 contains a visual representation of all the user information that our attacker was able to extract during the attack. Upon retrieval of this information, the identity of the previously-anonymous user has been fully uncovered.

The impact of the deanonymization attack is substantial as it relies on minimal assumptions, which are compatible with the threat models of anonymity networks

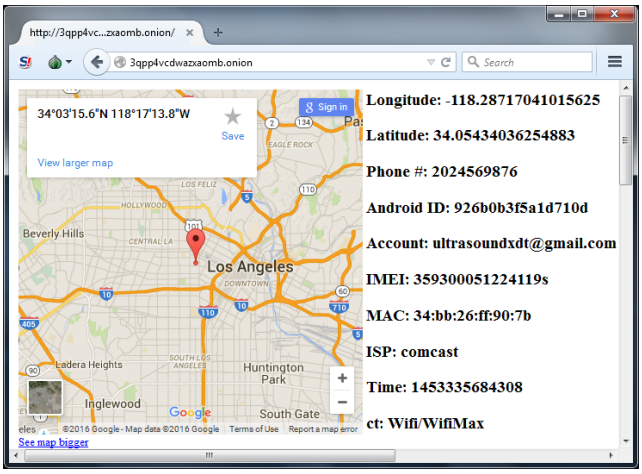


Figure 5.4: Screenshot of the proof-of-concept webpage upon a successful Tor deanonymization. The proof-of-concept attack can extract the user's location, ISP, phone number, Google account ID, and other sensitive information, even though the user is browsing through the Tor browser.

(e.g., Tor) [9]. Moreover, the majority of the ultrasound-enabled apps are vulnerable to this attack as most of the existing frameworks report the uBeacons captured (often along with user data) to an server operated by the service provider. We thus believe that the deanonymization attack poses an immediate threat for the users of anonymity networks and services and it is critical that mitigation measures are deployed by both network operators and service providers.

5.3.3 Profile Inference

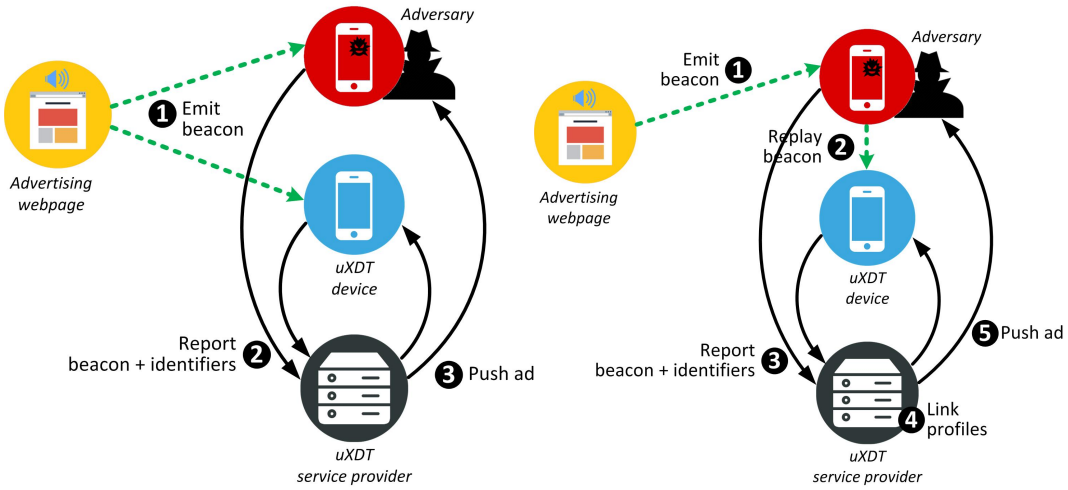
This attack is an adaptation for the ultrasound ecosystem of the attack introduced in [39]. The adversary exploits the lack of confidentiality and opt-out mechanisms to collect sensitive information on the user's online activities and interests. For example, such an attack could enable a malicious employer to acquire information on the online purchases and the shopping habits of their employees.

As in [39], we assume that the user operates in two or more environments with different behavioral requirements and expected levels of privacy (e.g., a user who uses a home computer and a workplace computer). There is a clear separation between the activities the user carries out in each of these environments, while the adversary is assumed to be present in only one of them (e.g., employer at work). We also assume that the user owns a smartphone with an ultrasound-enabled app that is

monitoring the ultrasonic spectrum for beacons. We recall that uXDT deployments use ultrasonic beacons to push user-specific ads across their different devices (see also Sections 2.1 and 5.2). An advertising network may push ads on both the smartphone and the home computer of a user, while searches and activities on the home computer will affect the ads displayed on the smartphone (and vice versa).

Consequently, a user that carries their uXDT-enabled smartphone at work risks having their work computer associated with their personal ad profile. The processes is as follows: When browsing the Internet on the work computer, ultrasound-enabled ads emit inaudible beacons that are captured by the user's smartphone. The uXDT framework on the phone then reports those beacons to the ultrasound service provider's backend and the user's ad profile is updated based on their work-related searches, purchases and activities. The advertiser can now push ads at the devices of the user based on the overall user's profile built based on their both home and work activities. Depending on the ultrasound service provider's configuration, it may take several ads before a device is associated with an existing user profile, however, the prolonged use and repeated uBeacon emissions from the work computer will eventually lead to an association of the two devices (i.e., the user's smartphone and the work computer).

Once such a link has been established, the work computer will display ads related to activities or searches the user conducted at their home computer or their smartphone. This enables anyone with access to the local area network traffic (e.g., network administrator, employer) to launch a privacy attack and infer the user's interests profile. In particular, as shown in [39, 230, 231, 38], even the analysis of a small number of targeted ads suffices for an adversary to reliably infer the user's Google interest categories with very high accuracy. In a nutshell, the adversary first removes the noise by filtering for contextual and location-specific ads and then rebuilds the user's profile by attributing each of the remaining ads to one broad category (e.g., car rental, travel) [39, 38]. As we will see in Chapter 4, even if all the advertising networks encrypted the transmitted data (e.g., TLS encryption), sophisticated adversaries may be still able to make accurate inferences about the



(a) Passive profile confluence attack. **(b)** Active profile confluence attack.

Figure 5.5: An adversary with physical proximity to a victim user could exploit the lack of authentication and replay protection in ultrasonic beacons to infer sensitive user information by reconstructing the user's interests profile.

content loaded.

While this user interests leakage occurs due to a combination of factors, ultrasonic pairing techniques and cross-device tracking make it considerably harder for the user to control the use of their personal information and maintain a clear separation between their work and personal interests profiles.

5.3.4 Profile Confluence

As discussed in the previous section, an adversary with access to the network traffic of a user can reconstruct the interests profile of the user by analysing their ad traffic. While such profile inference attacks had been introduced in prior works [39], we found that cross-device tracking made it especially hard for users to maintain their home and work interests profiles separate. We now outline how an adversary without access to the user's traffic could also launch an interests inference attack by (1) being periodically in proximity of the user and (2) exploiting the lack of authentication and replay protection in ultrasonic beacons.

Figure 5.5a illustrates the passive version of the attack where the adversary eavesdrops the uBeacons emitted by the work computer of a user with a uXDT-enabled smartphone. As before, the user maintains a separation between the activities,

online searches and purchases they conduct at home and at work, while the adversary tries to infer the overall interests profile of the user. Every time the user comes across an ultrasound-enabled ad, their work computer emits the corresponding uBeacon which is captured by their smartphone (❶). Then, the ultrasound tracking framework running on the phone parses the beacon and reports it to the ultrasound service provider (❷). Based on the beacons reported, the provider updates the user's interest profile and pushes targeted ads to the devices of the user (❸).

Due to the lack of confidentiality and authentication mechanisms, an eavesdropping adversary can intercept the uBeacons emitted by the user's work computer (e.g., with another uXDT-enabled smartphone). Those beacons do not necessarily contain sensitive information by themselves, as they are relevant to the work-related activities of the user. However, by repeatedly capturing and submitting such beacons the adversary can work towards 1) having their device associated with the user's profile and 2) gradually adapting their own profile to match that of the user.

In the former case, once the adversary's device is classified by the cross-device tracking service provider as belonging to the victim user, the adversary will start receiving ads based on the overall interests profile of the user. In fact, it is common among vendors that utilize behavioral profiling methods to match identically behaving users [232, 233, 234]. This information suffices to reconstruct the interests profile of the user [39, 38].

The latter scenario is relevant if the tracking service does not maintain a strict list of devices per user but instead updates and queries an ad or product recommender system. As both the user and the adversary repeatedly capture and submit (to the ultrasound service provider's backend) the same beacons, the recommender system will, with increasing likelihood, start pushing ads to the adversary's device influenced by the interests and choices of the victim user. While the actual precision of this variant of the attack depends on the technical details of the deployment (e.g., model used, number of users, possible recommendations), several works on recommender systems have shown that systems are prone to such attacks [235, 230, 231, 236].

To further increase the effectiveness of the above attack, the adversary may

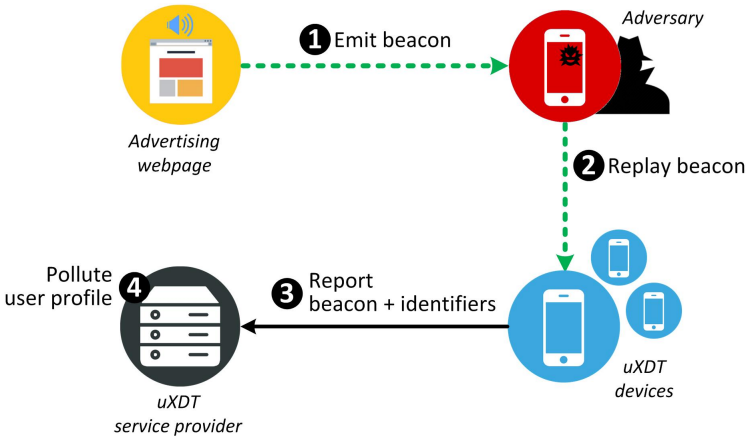


Figure 5.6: Operational steps of a profile corruption attack.

actively interfere with the submission of uBeacons by the victim (Figure 5.5b). Initially, the adversary collects a large number of beacons from various sources (❶) and then replays them to the victim's device (❷). Ultrasonic beacons have not replay protection and thus the victim's device cannot distinguish them from those emitted by valid sources. Simultaneously, the adversary submits the same beacons to the ultrasound service provider's backend server (❸). This practice aims to artificially increase the overlap between the interest profiles of the two actors thus making it more likely that the two profiles will be linked and that the adversary would receive recommendations influenced by personal activities of the victim (❹).

While authentication, replay-prevention or confidentiality mechanisms could potentially prevent the above attacks, the ultrasonic channel has very limited bandwidth and thus it is common for vendors not support only basic integrity checks.

5.3.5 Profile Corruption

An adversary can exploit the lack of replay protection to influence the user interest profiles maintained by the ultrasound service provider. For example, an attacker equipped with a simple beacon-emitting device (e.g., a smartphone) in a crowded venue can inject beacons to several ultrasound-enabled smartphones simultaneously, thus influencing their interest profiles and the ads served to them en masse.

As seen in Figure 5.6, the adversary initially acquires one or more valid beacons. To do this, they can set up an advertising campaign with an ultrasound service

provider and capture the beacon associated with it (❶). Alternatively, they can record the beacons associated with existing campaigns ran by others. Then, the adversary replays the captured beacons to the ultrasound-enabled devices of the victim users (❷). This step of the attack can be realized in many ways, depending on the goals of the adversary. For instance, the beacons can be replayed through the adversary's phone, the speakers of a venue, a computer or mobile device botnet or by injecting JavaScript as a Tor exit node (as discussed also in Section 5.3.2). Once the ultrasound-enabled devices of the users capture the beacons, they perform some basic integrity checks and forward the beacon sequence to the ultrasound service provider's backend server (❸).

Ultrasonic beacons have no replay protection and thus there is no way for the devices to detect that the beacon is replayed. Upon reception, the ultrasound service provider analyzes the uBeacon to determine the ad campaign it is associated with and updates the user's interest profile accordingly (❹). As a result, an adversary can pollute the user's profile to include activities or interests that are unrelated to their habits [237, 238]. The effectiveness of the attack can be further amplified by running multiple beacon injection rounds. In some implementations, beacons also act as a trigger for their corresponding ad to be displayed. This provides additional control to the adversary who can now directly influence the ads shown to the user's device. For example, malicious parties could use this side-effect to launch targeted malvertising campaigns [239, 240]

As in previous attacks, the bandwidth constraints of the ultrasonic channel [94] make it impractical to deploy on-channel authentication and/or replay-prevention mechanisms. With this knowledge, solutions should be sought in other parts of the ecosystem (e.g., smartphone's operating system).

5.4 Information Flow Control Mechanisms

We now discuss potential solutions to the security and privacy issues identified earlier, and evaluate their effectiveness and applicability. A jamming device emitting ambient ultrasonic noise may appear as a straightforward way to prevent ultrasonic

tracking but such a solution has several drawbacks (e.g., contributes to noise pollution, impractical to carry at all times). Instead, we propose two immediately deployable security mechanisms designed to mitigate the leakage risk for browser and smartphone users, while still retaining any ultrasonic-related functionality (if the user chooses so). Moreover, we propose a new OS-level API that would enable developers to implement ultrasound-based mechanisms without having to request full access to the microphone, thus abiding by the principle of least privilege.

5.4.1 Ultrasound-filtering Browser Extension

We develop an extension for the Google Chrome browser that filters out all ultrasounds from the audio output of the websites loaded by the user. Instead of simply detecting beacons, the extension proactively prevents webpages from emitting inaudible sounds and completely thwarts unsolicited attempts to stealthily broadcast beacons (unless the user opts to allow the specific tab to emit ultrasounds).

From a technical perspective, the extension mediates all the audio outputs of the page and filters out the frequencies that fall within the range used by inaudible beacons. To do this, each time a new webpage is loaded, a JavaScript snippet is inserted and executed in the page. The snippet uses the Web Audio API, which is part of the HTML5 specification. This API models all the audio modules of the page as `AudioNodes`. An `AudioNode` can be an audio source (e.g., an embedded YouTube video), an audio destination (e.g., the speakers) or an audio processing component (e.g., a low-pass filter). HTML5 also introduces the concept of *audio graphs*, where different `AudioNodes` are linked together to create a path from the audio source to the audio destination. This path can be arbitrarily long and may include multiple filters that process the signal before it reaches the destination node.

Our extension uses the `AudioNode`'s linking capabilities and prepends a new filter before the audio destination of the page. More specifically, upon execution, the snippet generates a *high-shelf* filter `AudioNode` and sets its base frequency and gain to 18kHz and -70db respectively. As a result, the filter attenuates all the frequencies above 18kHz, while leaving lower frequencies unaltered. Subsequently, the snippet identifies all the audio sources of the page and modifies the audio graph

so that their signal passes through the high-shelf filter before it reaches the speakers. From this point on, any audio played by the page is first sanitized by the high-shelf filter and then forwarded to the system's speakers. This procedure happens in real-time and it has minimum impact on audible frequencies. Our extension provides additional settings that allow the user to adjust the filter's parameters so as to optimize the coverage to their needs.

5.4.2 Android Ultrasound Permission

We now outline a modification to the Android OS that realises a filtering mechanism for the Android permissions system used by the apps. In particular, we extend Android and expose new functionality that allows users to selectively filter out high-frequencies from the signal captured by the smartphone's microphone. Our idea is to provide *two* microphone permissions: one for accessing the *audible* part of the spectrum and an additional one that allow access to the inaudible, high-frequency (i.e., ultrasonic or near-ultrasonic) parts too. This separation forces apps to clearly communicate their intention to capture inaudible emissions during the installation of the app, thus ensuring that the user is informed.

To achieve this, we modify the Android OS and extend the existing permission `RECORD_AUDIO`. An app that needs to record audio from the microphone can now selectively request only for the `RECORD_AUDIO` permission, whereas an ultrasound-enabled app will have to also request `RECORD_ULTRASOUND_AUDIO` that provides access to the higher frequencies too.

Since the vast majority of apps does not require access to the high-frequency parts of the spectrum, we believe that the any request for access to higher-frequencies could indicate that a more careful inspection of that app (by the app store) is needed.

The implementation of this new permission requires the modification of two Android Open Source Project (AOSP) parts. First, we define a new permission, which is done by modifying an OS configuration file.[2] The second modification relates to the AudioFlinger component. AudioFlinger is the main sound server in

[2]In AOSP, the file is found under the following path: `./base/core/res/AndroidManifest.xml`.

Android: when an app wants to obtain a data *read* from the microphone, the app communicates through the Binder Inter-process communication (IPC) mechanism with the AudioFlinger component. This component, in turn, reads the data stream from the kernel sound driver (e.g., ALSA, OSS, custom driver) and makes the content accessible to the requesting app (again through the Binder IPC mechanism).

Our modification in the AudioFlinger component implements the following logic. Consider an app that wants to acquire data from the microphone. If this app has both the `RECORD_AUDIO` *and* the `RECORD_ULTRASOUND_AUDIO` permissions, then the stream is not modified in any way. However, if the requesting app does not request the `RECORD_ULTRASOUND_AUDIO` permission, our patched AudioFlinger would *filter out* frequencies above a certain threshold. The filter implemented by our current prototype is a standard low-pass filter that attenuates signals with frequencies higher than the cutoff frequency (i.e., 18,000 Hz). Our current implementation filters sound in the time domain and can operate in real-time as it only requires a few bytes of extra memory. Furthermore, our patch is not invasive and comprises less than one hundred lines of codes.

5.4.3 Standardization & uBeacon API

The countermeasures proposed above can significantly reduce the likelihood of leakages giving end-users better control of their devices. However, such application-agnostic filtering mechanisms can also potentially interfere with benign use cases. For this reason, we argue for the standardization of an ultrasound beacon format and envision a new OS-level API that implements in a single, trusted place the logic for detecting, decoding and emitting uBeacons.

The first step towards this goal is to specify the format and structure of ultrasonic beacons. Based on this specification, an API for handling uBeacons can be then designed and implemented. Such an API should expose calls for: (1) uBeacon discovery and capturing, (2) uBeacon decoding and integrity validation, and (3) uBeacon generation and broadcasting. We note that a similar API for Bluetooth low energy (BLE) beacons is already used in Android [211, 212, 213].

From a technical perspective, such an API would require a privileged process

(e.g., a system process on Android) that realises and exposes the relevant uBeacon functionality. This would eliminate the need for apps to gain direct access to the device's microphone and would instead require access only to the relevant functions of the API. Besides protecting end-users from eavesdropping developers, apps would no longer need to request a security-sensitive permission (i.e., microphone access), thus avoiding being labeled as "potentially malicious". Moreover, such an API would allow for a central point in the operating system that keeps track and monitors all ultrasound-related activity (e.g., provide a full list of beacons captured). By delegating the task of monitoring and capturing uBeacons to the operating system, we improve the transparency of the ecosystem's operations and ensure that the user can easily enable and disable the channel.

In order to enforce the use of such an API, the ultrasonic spectrum must be accessible only to privileged components of the system. To achieve this, the system module handling the microphone should filter out ultrasonic frequencies by default and the user should be able to grant access to that part of the spectrum on a per-app basis. Our Android filtering countermeasure presented earlier, shows how such a feature can be technically realised.

As third-party apps would never get direct access to any signal in the ultrasonic spectrum (even when requesting the microphone permission), it would not be possible for developers to use beacon formats that do not abide by the standard. If a non-standard beacon is captured, the system's parsing process will fail to decode it and return an error. Such an API could prevent the attacks outlined in Section 5.3 as it would allow users to keep the channel disabled permanently or during privacy-sensitive tasks. This functionality can be easily exposed through a central switch, similar to that of Bluetooth or Wifi.

5.5 Tracking the Ecosystem

To better understand the evolution of ultrasound-enabled frameworks and relevant technologies over time, we now discuss developments that followed our initial study of the ecosystem [15]. Our initial study took place in 2016 when we identified

several companies developing solutions based on ultrasonic beacons. Since then, we kept track of the adoption of ultrasonic beacons and conducted periodic searches for Android apps that use the SilverPush framework. In 2016, we identified approximately 100 Android apps that used the SilverPush framework, which was also in line with the findings of other sources [241]. The download counts for the most popular of these apps ranged from several hundreds thousands to a few millions (e.g., [242]). In subsequent searches, we observed that the number of applications incorporating the specific framework declined. At this stage, the decline was likely due to both the community backlash forcing developers to remove the framework from their apps [243, 244, 245, 221]. Moreover, Silverpush reportedly withdrew its product from the US market and the framework steadily declined further in popularity in all markets until it was discontinued in 2017. This was also indicative of a change in the whole ecosystem as most companies moved from offering tracking products into device pairing, payment products and indoor navigation.

On the standardization side, our proposal for more fine grained control over the audio channel in Android has not been implemented. However, Android 11 introduced "one-time" permissions that make monitoring the audio spectrum on the background considerably harder [246]. In particular, the user has now the option to grant a permission "Only while using the app" which allows apps to use the microphone only while the app is in the foreground or has a foreground service that is visible to the user.

A detailed list of companies and frameworks participating in the ecosystem can be found in [247], where City Frequencies maintain an frequently updated report (last updated May 2021).

5.6 Conclusions

This chapter evaluated the privacy and security aspects of the ultrasonic communications channel. In particular, we found that the lack of fine-grained permissions in mobile devices, combined with poorly implemented communication protocols can expose users to a series of privacy attacks. Such attacks range from leakage of

sensitive user information to targeted deanonymization campaigns against anonymity network users. Such networks aim to provide stronger privacy guarantees compared to browsing over TLS, which we studied in Chapter 4. Despite this, they are also vulnerable to fingerprinting attacks (Section 3.1) and, as we saw in this chapter, side-channel attacks where an adversary exploits one application or channel to breach the security properties of another.

The latter attack vector is particularly difficult to address as the inter-connectivity of modern systems and the multitude of apps hosted on a single device make accurate threat modeling complex. In the case of ultrasonic communications, many of the issues identified were due to the immaturity of the protocols and we observed a self-correction in the ecosystem over time. Countermeasures at the operating-system level (e.g., OS-controlled APIs) have the potential of mitigating such leakages and their exploitation even if the app developers do not have access to the channel directly. A similar approach was used in Bluetooth Low Energy where the channel is managed exclusively by the OS.

In the following chapters, we move our focus on leakages that affect the physical layers of the computing stack. Such leakages concern most devices that handle sensitive keys including Internet servers and traffic relaying nodes. We first study how modern machine learning techniques can contribute to the detection of leakages in that context (Chapter 6) and then investigate designs for leakage-tolerant systems (Chapter 7).

Chapter 6

Hardware Side-Channels

In Chapters 4 and 5, we treated networked devices as monolithic components that could reliably protect secrets and information. On that basis, we examined the impact of leakages in communication channels (e.g., ultrasonic communications, secure connections through the Internet) and showed that security and privacy shortcomings can be utilized by passive and active adversaries.

We now switch on to the physical layer and investigate leakages in intra-device communication channels (e.g., buses) and low-level components. Such leakages have the potential to affect critical devices such as web servers, embedded nodes, hardware security modules, anonymity network relays, and hardware cryptocurrency wallets

This chapter puts forward a side-channel exploitation model and evaluates its efficiency against a commercial ARM core that is used in both commercial products and academic works. As in chapter 4, we find that the advancements in machine learning have significantly enhanced the analysis capabilities of the adversaries and alleviated the need for in-depth knowledge of the protocol or system.

6.1 Introduction

Side-channel analysis enables adversaries to leak sensitive information by monitoring and analyzing the physical characteristics and emanations of a cryptographic implementation. Usually, physical observables (e.g., power consumption, electromagnetic emissions) of a device [248, 249] are closely related to the data accessed,

stored or processed. Such *data-based leakages* compromise the device's security and may allow an adversary to infer the secret key or other confidential information the device stores or operates on.

In this chapter, we focus on location-based leakages that exploit the electromagnetic (EM) emanations which occur when certain chip components such as registers, memory regions, storage units and their respective addressing mechanisms (control logic, buses) access stored data. The power of the EM side-channel potentially conveys information about the *location* of the accessed component, i.e. it can reveal the particular register or memory address that has been accessed, regardless of the data stored in it. In cases, where there is dependence between the secret key and the location of the activated component, a side-channel adversary can exploit it to recover the key or other sensitive information.

Unlike the well-established power and electromagnetic data leakage models [250, 251], high-resolution EM-based location leakage remains less explored. The main reason is the semi-invasive nature of location attacks (often requiring chemical decapsulation), the time-consuming chip surface scanning and the lengthy measurement procedures involved. We argue that such attacks are increasingly relevant as they allow adversaries to circumvent established leakage protection mechanisms [252, 253] at a moderate, or even low, cost.

Overall, this chapter performs the first practical location-based attack on the SRAM of a modern ARM Cortex-M4 and investigates its degree of vulnerability to such leakages. We introduce a cipher-agnostic adversary that can make high-accuracy inferences on the SRAM regions accessed during the execution of a cryptographic algorithm. Our model allows countermeasure designers to gauge the amount of experimental work an adversary would need to breach the device, as well as identify certain security hazards and fine-tune their protection mechanisms. We find that a sophisticated adversary can distinguish consecutive SRAM regions of varying byte-length with high accuracy, which indicates that EM location-based leakages are potent enough to compromise the security of AES implementations that use SRAM lookup tables. While the adversary may not be always able to fully recover the secret

key, they can significantly reduce the number of key candidates.

6.2 Threat Model

We assume a secure chip that stores sensitive information or secrets in its memory and an adversary who aims to extract as many bits, as possible. For example, for a device that implements a key-dependent cipher operation using lookup tables [254, 255], the adversary aims to infer which parts of the table were active and in which order, so as to recover the key. Such lookup tables (LUTs) are commonly used to implement the substitution-boxes for symmetric key algorithms (e.g., AES, DES).

The adversary has full physical access to the chip, and is capable of passive (e.g., measurements through microprobes) and semi-invasive (e.g., die etching) attacks with an upper equipment cost of $\sim$\$30,000. This cost is moderate as power analysis attacks that exfiltrate information based on the chip's power consumption (e.g., [256]) require up to $\sim$\$5,000. We do not consider high-cost invasive attacks (e.g., Focused Ion Beam) [257]. Such a scenario is relevant to various adversarial settings such as key extraction attacks against cryptocurrency hardware wallets [258], Internet-of-Things devices [259], embedded TLS deployments [260] and confiscated anonymity network nodes.

Moreover, the adversary can execute instructions that involve memory accesses but cannot use them to retrieve information through any channel other than the EM emanations of the chip. Importantly, the adversary is able to calibrate their equipment and optimize their attacks (e.g., build a leakage template) on the same chip they intent to breach. This is a common assumption in the side-channel attacks literature (e.g., [261]) as it considers the worst-case scenario for the defender (i.e., best-case for the adversary).

In practice, most adversaries will not be able to build their profiles on the same physical chip they aim to attack but will be able to obtain another instance of the exact same chip model. It is thus likely that the accuracy of the adversary's leakage templates will be reduced due to the (minor) physical differences between the two chips. However, from the defender's perspective, these physical differences cannot

provide reliable protection as they are mere artifacts of the fabrication process. As a result, it is common practice to assume perfect template *transferability* between chips of the same model, even if it is not always the case. The problem of template transferability across different chips has been studied in other past works [262].

6.3 Experimental Setup & Dataset

Our measurement setup consists of a Riscure Pinata board [1] featuring a decapsulated STM32F417IG System on Chip (SoC) by ST. The STM32F417IG embeds an ARM 32-bit Cortex-M4 CPU clocked at 168 MHz fabricated using 90 *nm* technology and features 1,024 KB of Flash and 196 KB of RAM.

The Cortex-M4 core is commonly used in works investigating side-channel attacks [263, 264, 265, 266] as well as implementation of classic and post-quantum algorithms. For example, the official implementations of SIKE [267, 268] and the *pqm4* [269] post-quantum cryptography software library target this family of micro-controllers. Besides these, Cortex-M4 has found application in various other use cases involving key management such as a cryptocurrency hardware wallet [270, 258, 271], Internet of Things cryptography [259] and TLS deployments on constrained ARM processors [260]. While our experiments are conducted only on the STM32F417IG SoC, our findings likely transfer to other SoCs from the same family or even SoCs from other manufacturers.

The decapsulated chip surface (roughly 6 $mm^2 \approx 2.4\ mm \times 2.4\ mm$) is scanned using an ICR HH 100-27 Langer microprobe with diameter of 100 μm (approximately 0.03 mm^2). The decapsulated SoC on the Pinata board is shown in Figure 6.1 and the Langer microprobe in Figure 6.2.

The scan is performed on a rectangular grid of dimension 300, resulting in 300×300 measurement spots. The near-field probe is moved over the chip surface with the assistance of an XYZ-table with positioning accuracy of 50 μm. At every spot of the scan grid, a single measurement is performed, using a sampling rate of 1 Gsample/sec while traversing the memory sequentially. This process resulted in

[1] https://www.riscure.com/product/pinata-training-target

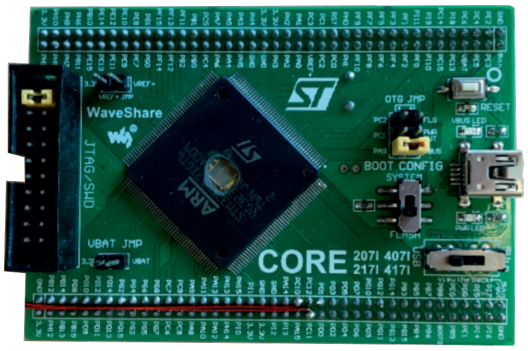


Figure 6.1: The modified Riscure Pinata board with the decapsulated STM32F417IG SoC.

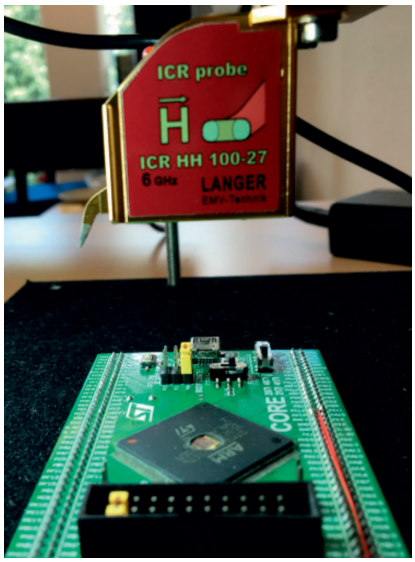


Figure 6.2: For our measurements, we used the ICR HH 100-27 Langer microprobe with a diameter of 100 μm.

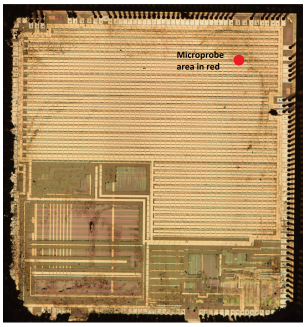


Figure 6.3: The surface of the chip after removing the plastic layer. For comparison, the approximate area of the microprobe we used for our measurements is illustrated as a red circle (0.03 mm^2).

approximately 170,000 *samples* per spot on the grid. Overall, a single measurement on a specific spot of the grid produces a *trace* with 170,000. When probing the chip, the adversary performs 300 $\times$ 300 measurements which produce as many traces.

Due to the complex and non-homogeneous nature of modern chips, several types of EM emissions are present on the surface, most of which are unrelated to the SRAM location. In this particular case study, the signals of interest were observed in amplitudes of roughly 70 mV, so we set the oscilloscope voltage range accordingly. In addition, several device peripherals (such as USB communication) have been disabled in order to reduce interference. The decapsulated surface where the scan is performed is visible in Figure 6.3 with the approximate microprobe area overlaid (in red) for comparison.

We consider the spatial leakage emitted by the ARM Cortex-M4 SRAM, while accessing an AES lookup table (LUT) of 256 bytes and excluding any data-based leakages. The processor uses a 32-bit architecture, thus the 256-byte lookup table uses 64 words (4 bytes each) which are stored consecutively in the SRAM. The adversary aims to distinguish accesses to different LUT regions (consisting of one or more words). Being able to infer which LUT/SRAM region was accessed can substantially reduce the number of AES key candidates. For instance, an adversary may template separately the leakage of all 64 words in order to reduce the possible AES key candidates from 256 to 4. Alternatively, they can partition the LUT to two regions (words 0 until 31 and words 32 until 63), profile both regions and recover the activated 128-byte region and reduce the AES key candidates from 256 to 128.

We fix the data in all accessed memory positions to value zero in order to prevent any data-dependent leakage, and perform sequential accesses to the SRAM. We opt to access the SRAM using ARM assembly instead of a higher-level language in order to avoid compiler-induced optimizations that could alter the side-channel behavior.

In the rest of this chapter, we use the terms "location leakage" and "spatial leakage" interchangeably. They include leakage from both the unit itself (e.g., accesses cause the memory to be activated) and the addressing mechanisms involved (e.g. leakage from the control logic of a storage unit). The adversary is usually able to measure both types of leakage but is unable distinguish between them.

6.4 Leakage Detection

Based on the dataset we compiled, we now investigate whether ARM Cortex-M4 exhibits leakage patterns. While the mere detection of EM leakages does not necessarily imply that an adversary can launch a successful region inference attack, it provides useful information to security auditors regarding the leakage characteristics of the device.

For our analysis, we partition our samples into two classes: The first 85,000 samples collected for each spot on the grip are labelled as class 1 and the remaining 85,000 as class 2. Intuitively, class 1 correspond to accesses in the first 2047 words in the SRAM, while class 2 to accesses from word 2048 until word 4096. In total, each class corresponds to 8 KB of SRAM.

We average the leakage samples from class 1 and class 2 for every position on the grid (x,y) by computing $\bar{l}_{class1} = \frac{1}{85k}\sum_{j=1}^{85k} l_{x,y}^{j}$ and $\bar{l}_{class2} = \frac{1}{85k}\sum_{j=85k}^{170k} l_{x,y}^{j}$ respectively, and compute the difference of means $\bar{l}_{class1} - \bar{l}_{class2}$. We, then, perform a Welch t-test with significance level of 0.1% for every spot on the grid. The t-test enables us to determine if location-based leakages are present.

The results are shown in Figure 6.4, which is cropped to show the specific part of the chip surface that exhibits significant differences. We observe two regions at close proximity (yellow and blue), where the yellow ones indicates positive difference

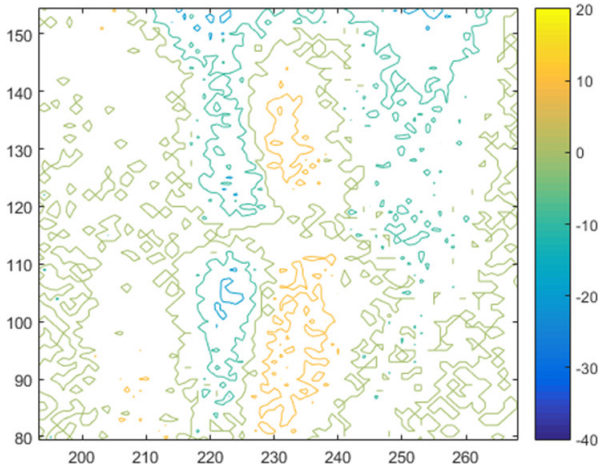


Figure 6.4: Distinguishing two 8 KB regions of the SRAM with the difference-of-means. The yellow region indicates stronger leakage from class 1 while the blue region from class 2.

between class 1 and 2, while the blue region negative difference between class 1 and 2. To investigate this further, we performed additional chemical etching on the chip surface in order to remove the top metal layer (Figure 6.5).

The different regions (yellow, blue) shown in Figure 6.4 are observed directly above the chip area enclosed by the red rectangle of Figure 6.5. Interestingly, after the removal of the top metal layer, we see that the red rectangular region contains large continuous chip components, possibly indicating that SRAM circuitry is present at this location. This hypothesis is corroborated by the following fact: when we perform the difference-of-means test for 4 KB regions, the yellow and blue regions shrink, indicating that the leakage area is *proportional* to the memory size that is being activated.

The expected surface area of an SRAM component can be approximated as

$$a = \frac{m \cdot a_{bit}}{e}$$

where m is the number of bits in the memory region, a_{bit} is the area of a single-bit memory cell and e is the array layout efficiency (usually around 70%) [272]. The value of a_{bit} ranges from $600\lambda^2$ to $1000\lambda^2$, where λ is equal to half the feature size, (i.e. for the current device-under-test $\lambda = 0.5 * 90\ nm$) thus the area of a 32-bit

Figure 6.5: Chip surface of the STM32F417IG after removing the top metal layer. The red rectangular region corresponds to the difference-of-means plot of Figure 6.4, i.e. it shows the location where the highest differences were observed.

word is between 55 and 92 μm^2. Likewise, an 8 KB region of the ARM Cortex-M4 amounts to an area of between 0.12 and 0.19 mm^2, depending on the fabrication process. Notably, this area estimation is quite close to the area of the yellow or the blue region of Figure 6.4 (approximately half of the red rectangle). Similar spatial characteristics have been observed by Heyszl et al. [99] in the context of FPGA registers.

Thus, our findings suggest that 1) there are leaky regions in close proximity, and 2) the area of leaky regions is approximately proportional to the memory size that we activate. In the following section, we follow up on these observations and investigate whether classification models can be used to exploit *spatial leakages* and infer the memory regions accessed.

6.5 Leakage Exploitation

In Section 6.4, we conducted a preliminary investigation that indicated the existence of EM leakage in our decapsulated ARM Cortex M4 die. We now attempt to infer the memory regions that were accessed so as to evaluate whether an adversary could exploit the identified EM leakages to extract secrets from the SRAM.

Works on side-channel attacks often assume that EM leakages follow a multivariate normal distribution and build their inference templates accordingly. We follow an alternative approach and employ distribution-agnostic techniques which have been recently shown to produce very promising results [273, 274, 275, 276, 277, 278, 279]. We introduce two adversaries that use neural networks to analyse the spatial leakages of the die, and we evaluate their performance under the security assumptions outlined in Section 6.2. In both experiments, we attempt to determine whether accessing different SRAM regions in a modern ARM-based device produces distinguishable signals. In other words, we examine the device's susceptibility to location-based attacks during key-dependent memory lookups, similar to AES LUT (i.e., look-up table). We formulate this as a classification task, where the adversary determines which class (i.e., memory region) a given EM snapshot belongs to.

As outlined in Section 6.3, we collected $\sim$170,000 EM measurements for each of the 300 $\times$ 300 spots on the chip. We treat this dataset as 170,000 snapshots of the chip's emanations captured while the adversary traversed all the memory positions sequentially (similar to the datasets from Magnetic Resonance Imaging [280]). We, then, get versions of our dataset with varying granularity by gradually splitting the 256 bytes of the AES LUT into classes. In particular, we prepare datasets with 2, 4, 8, 16, 32, 64, 128, and 256 partitions (of 128, 64, 32, 16, 8, 4, 2, and 1 bytes each, respectively). Each dataset is then split into training, validation and test sets with a 40-30-30 ratio. The first two sets are used for configuring and training our models, while the testing set is used for evaluating the actual performance of the adversary on data they have not encountered during training.

6.5.1 Transfer Learning

Before developing and customizing our own model, we evaluate the performance of existing, state-of-the-art pre-trained neural networks. Pre-trained models are usually large networks that have been trained for several weeks over vast datasets. As a result, their first layers tend to learn very good, generic discriminative features. Transfer Learning [281] is a set of techniques that, given such a pre-trained network, repurposes its last few layers for another similar (but not necessarily identical) task.

The objectives of our spacial identification task appear to be very close to those of standard image classification, while our data is formulated as 300×300 grid snapshots, which makes them compatible with the input format of several computer vision classification networks. We thus approach this as a contextual image classification problem where the EM emanations of pixels in proximity are related.

For this first attempt at region inference, we use several state-of-the-art networks, namely Oxford VGG16 and VGG19 [282], Microsoft ResNet50 [283], Google InceptionV3 [284] and Google InceptionResNetV2 [285]. It should be noted that the input format of these networks is often RGB images, while our 300×300 heatmaps resemble single-channel, grayscale images. To address this and recreate the three color channels that the original networks were trained for, we experiment with two techniques; (1) we assemble triplets of randomly chosen heatmaps, and (2) we recreate the three color channels by replicating the heatmaps of the samples three times.

We first re-train the pretrained models for our specific tasks and then use them to classify samples they have not encountered before. In accordance with the standard transfer learning methodology, during the re-training we freeze the first few layers of the networks to preserve the generic features these layers represent.

Despite various attempts and multiple hours of training, none of the aforementioned models achieved high classification accuracy. We attribute this to the nature of our task that, despite our initial intuition, may require a substantially different set of features. In particular, the features extracted by the pretrained models may not be useful for our task as they were primarily generated for computer vision tasks.

6.5.2 Convolutional Neural Networks

We now evaluate the performance of a custom model that uses a convolutional neural network (CNN) to infer the activated regions of a 256-byte, data-independent lookup table on the ARM Cortex-M4. The architecture of our neural network and the model's hyperparameters were determined through trial and error on the training and validation sets, while the test set was used to evaluate the accuracy of the final model (Figure 6.6). Our network is relatively small and features only three convolutional

Parameter	Value(s)
Input layer	Convolutional layer: filters 8 and kernel size (3,3)
# hidden convolutional layers	2 layers: filters 4 and kernel size (4,4)
# hidden fully connected layers	2 layers: 1024 and 512 neurons
Activation for hidden layers	ReLU [202]
Size of output layer	# memory regions
Activation for output	Softmax
Optimizer	Adam Stochastic Optimizer [286]
Dropout	0.1
Learning rate	0.0001
Batch Size	512 pairs

Table 6.1: The hyperparameters (top half) and the training parameters (bottom half) of our leakage-classification neural network.

layers, followed by three dense layers. After each layer, we use a Relu activation unit with the exception of the final classification layer that uses softmax activation. Table 6.1 lists the hyperparameters and the parameters of our model.

The attack success rates for the test traces for 2, 4, 8, 16, 32, 64, 128, and 256 partitions are presented in Figure 6.6; the accuracy values are 98%, 97%, 95%, 92%, 87%, 79%, 64%, and 35%, respectively. We observe that the accuracy of the adversary is significantly higher than the success rate of a randomly choosing adversary (i.e., 50%, 25%, 12.5%, 6.25%, 3.125%, 1.56%, 0.78%, 0.39%), even in the high granularity scenario with 256 partitions. We conclude that the STM32F417IG SoC (ARM Cortex-M4) is vulnerable to location-leakage attacks and that sophisticated adversaries can substantially reduce the security level of LUT-based cipher implementations unless appropriate countermeasures are deployed (e.g., address randomization).

6.6 Conclusions

This chapter revisited the important, yet often overlooked, problem of location-based leakages. Our findings demonstrate that modern STM32F417IG (ARM Cortex-M4) dies are vulnerable to location-based attacks when operating on secrets and show that an adversary can launch high-accuracy inference attacks using convolutional

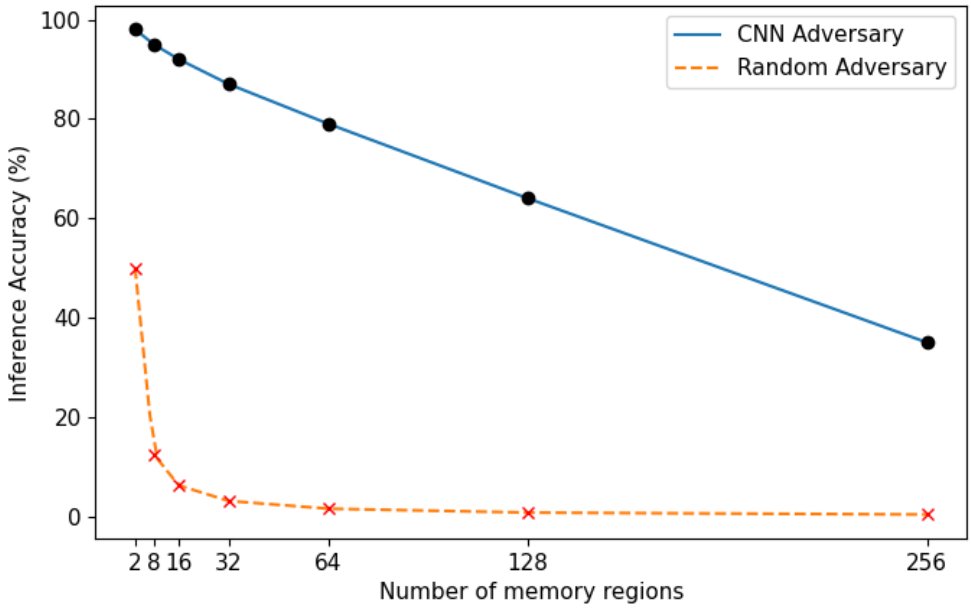


Figure 6.6: The blue line illustrates the testing accuracy of our leakage exploitation model and the orange line denotes the success rate of a randomly guessing adversary.

deep neural networks. Consequently, such attacks should be considered in the threat model of systems (e.g., cryptocurrency hardware wallets, embedded TLS-enabled servers) that rely on this core with sensitive tasks such as key management. Moreover, the success of the deep neural network in this task is also in line with the results of Chapter 4 where we used deep neural networks to exploit leakages in the TLS protocol. In both cases, the attacks outperform past models, are low or moderate in cost and do not require deep knowledge of the chip or the protocol to calibrate and fine-tune.

From a defense perspective, our leakage modeling technique is suitable for countermeasure designers and auditors who need to inexpensively gauge the amount of experimental work an adversary would need to breach a device. Designers can also use ML-based adversaries to fine-tune their protection mechanisms, provide customized security, avoid lengthy design-evaluation cycles and capture certain security hazards at an early stage. This allows for the time-consuming and expensive leakage certification of the physical device to be performed at a later stage, once any obvious defects have been already addressed. Chapter 7 further expands on

the space of possible defenses but opts for a less explored direction. Instead of trying to detect or prevent leakages, we study designs that tolerate (i.e., retain their security properties) leakages occurring due to vulnerable hardware components or inter-component channels.

Chapter 7

Leakage-tolerant Systems

In the previous chapters, we discussed how adversaries can exploit intra- and inter-device leakages to exfiltrate secrets and information. Chapters 4 and 6 revisited known problem areas, while Chapter 5 studied leakages in an emerging communications channel. Our findings highlight the impact leakages can have on end-users and the need for robust countermeasures.

Following up on these, this chapter discusses defenses and, more specifically, leakage-resilient system designs. Such designs complement classic leakage prevention and detection techniques and alleviate the requirement for all of the system's components to be leakage-free. Consequently, a system can retain its security properties even when some of its core components are intentionally or unintentionally leaky. This is particularly relevant to complex systems where it is hard or expensive to validate the security of all of their components.

7.1 Introduction

In this chapter, we introduce Myst, a novel approach to the problem of building trustworthy cryptographic hardware. Instead of attempting to detect or prevent hardware trojans and errors, our proposed architecture enables a critical system comprised of untrusted hardware components to retain its security properties, as long as at least one of them is not compromised, even if benign and malicious components are indistinguishable.

Many critical systems rely on high-integrity hardware to carry out sensitive

security tasks (e.g., key generation and storage, digital signature issuance, code signing) and to provide a protection layer against leakages and security breaches. Such systems typically handle data for banking, communications, military, space and other applications. In most cases, they use Hardware Security Modules, Trusted Platform Modules and/or Cryptographic Accelerators that are designed and fabricated under strict specifications to ensure that the final system will be both secure and reliable.

However, manufacturers are not always able to strictly oversee all parts of their components' supply chains [287, 288, 289]. For example, it is increasingly common for components to be manufactured in overseas foundries as the constant reduction in transistor size makes integrated circuit (IC) fabrication an expensive process for many IC designers [290, 291, 292]. While vendors are still able to conduct some quality control and audits of the outsourced manufacturing, those processes are not as rigorous as those of in-house production lines. For this reason, vendors conduct a wide array of post-fabrication tests on the finished components to find potential faults before they are used in their products.

Post-fabrication tests are very effective in detecting common defects but do not provide equally strong guarantees for security-related errors. For instance, in past incidents, cryptoprocessors with defective RNG modules and hardware cipher implementations have made it into production [293, 294], mainstream processors were found vulnerable to memory leakages [295, 296, 296, 297, 298] and smart card chips featured leaky cryptographic algorithm implementations [131]. Similarly, the Intel Software Guard Extensions [299, 300, 301] failed to protect the contents of the enclave's memory, while critical security vulnerabilities have been also reported in military [118, 119], networking [120, 121] and various other applications [122, 123, 124, 125].

Besides unintentional defects, parts of the IC's supply chain could be also exposed to attacks from malicious insiders [302, 303, 287, 304]. A large body of research works has introduced various types of hardware trojans (HT) and backdoors that demonstrate the extent of the problem and discuss potential countermea-

sures [113, 114, 115, 116, 117, 111, 110, 109]. In particular, the relevant literature spans across detection techniques and more proactive prevention measures against leaky and/or malicious circuitry. Detection techniques aim to determine whether any errors exist in a given circuit [132, 133, 134, 135], while prevention techniques either impede the introduction of intentional errors or assist with their detection at a later stage [305, 141, 142, 143, 146].

Unfortunately, fabrication-time countermeasures come at a high manufacturing cost [306, 142, 141], which contradicts the motives of fabrication outsourcing and cannot be used by vendors that rely on commercial off-the-shelf (COTS) components. Furthermore, in the literature new hardware trojans have been repeatedly able to circumvent state of the art countermeasures [142] triggering an arms race that seems favorable to the attackers [307, 142]. For instance, ICs fabricated using split manufacturing (one of the most promising and effective prevention techniques) can be modified to include analog malicious components (e.g., capacitors) that allow for remotely-controllable privilege escalation and evade all known defenses [112].

Our key insight is that by combining established privacy-enhancing technologies with mature fault-tolerant system architectures, we can distribute the trust between multiple components originating from non-overlapping supply chains, thus reducing the likelihood of compromises. To achieve this, we deploy distributed cryptographic schemes on top of an N-variant system architecture and build a trusted platform that supports a wide-range of commonly used cryptographic operations (e.g., random number and key generation, decryption, signing). This design draws from protective-redundancy and component diversification [150], and is built on the premise that a single adversary could compromise several components (e.g., a state actor compromising the processors and communication controllers fabricated in a country). However, unlike N-variant systems, instead of replicating the computations on all processing units, Myst uses multi-party cryptographic schemes to distribute the secrets so that the individual components hold only shares of the secrets (and not the secrets themselves), at all times. Thus, as long as one of the components remains honest, the secret cannot be reconstructed or leaked.

Our proposed architecture is suitable for high-assurance system vendors who design in-house but outsource the fabrication of some of their ICs and vendors who rely exclusively on commercial components. The former category has increased control over the IC design and could even combine our design with existing detection [136, 137, 138] and prevention techniques [141, 142, 143, 144]. On the other hand, vendors that use only COTS components have limited control over the ICs themselves but can mitigate the risks by diversifying their suppliers (i.e., multiple non-overlapping supply chains).

Overall, this chapter outlines how cryptographic schemes can be combined with N-variant system architectures to build high-assurance systems and introduce a novel distributed signing scheme based on Schnorr blind signatures. We demonstrate the practicality of the proposed architecture by building a custom board featuring 120 highly tamper-resistant ICs, featuring secure variants of random number and key generation, public key decryption and signing. Through a series of experiments, we find that diversified system designs can achieve strong resilience to breaches while remaining competitive in terms of throughput and latency. Our implementation is based on (n,n) secret-sharing but can be easily generalised to (t,n) schemes. Follow up works have further enriched both the list of cryptographic schemes supported but also expanded the concept of component diversification [308].

7.2 Threat Model

We assume an adversary who is capable of introducing intentional errors (e.g., a hardware trojan, a backdoor) to hardware components (e.g., integrated circuits). To do that, the adversary needs to be able to breach at least one point of the component's supply chain. Consequently, given several components with non-overlapping supply chains (e.g., manufactured at different fabrication facilities, locations and by different vendors), for the adversary to compromise all of them, they need to breach at least one point in every one of the chains involved.

Upon introducing the malicious circuitry, the component is assumed to be fully controlled by the adversary with full access to the memory contents (e.g., private

keys), the execution stack and every other functionality available (e.g., the operation of the random number generator). The adversary is also able to remotely connect and transfer data to and from the component. This can be done through a side-channel (e.g., RF [309], acoustic [310] or optical [311] channels) or other means. They are also able to exfiltrate information from the device's buses and the channel controller, as well as interfere with the communication between the different components. We also assume that any component compromised by the adversary is completely indistinguishable from benign/non-faulty ones.

For the development and the provisioning of the system, we assume a trusted software developer that builds and signs the applications to be run on the processors. The integrity of the source code can be ensured though standard high-integrity software development practices (e.g., security audits, public code repository trees, deterministic builds). The system operator is trusted to choose components and initialize the device so as to minimize the likelihood of compromises (e.g., computation quorums from diverse components). Finally, the users' computers may be compromised by the adversary and may submit malicious service requests to the device or leak secrets stored there.

7.3 System Overview

We now introduce our proposed system which draws from fault-tolerant designs and N-variant systems [312, 150]. It is illustrated in Figure 7.1 and has three main components: (1) the remote host, (2) the untrusted controller, and (3) the processing integrated circuits. The design is based on the thesis that given a diverse set of k components sourced from k untrusted suppliers, a trusted system can be built, as long as at least one of them can be assumed to contain no errors. Alternatively, our architecture can tolerate up to 100% compromised or defective components if it holds that two or more *non-colluding* adversaries control them.

Processing ICs. The ICs form the core of Myst as they are collectively entrusted to perform high-integrity computations and provide secure storage space. The ICs are programmable processors or microprocessors. They have enough persistent memory

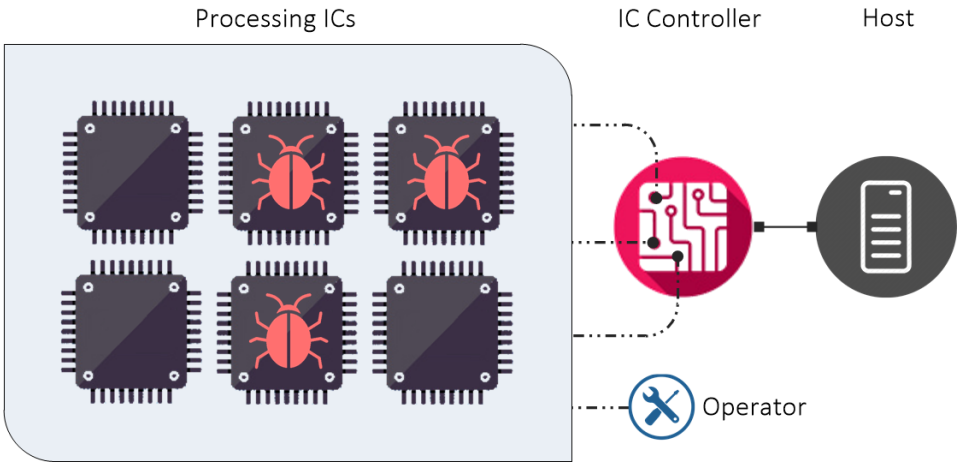


Figure 7.1: Overview of Myst's design, featuring all the integral components and communication buses. The gray area represents the cryptographic device, featuring several untrusted cryptographic processors (ICs). During the provisioning phase, the operator configures the individual ICs. Thereafter, the users (Host) interact with the device through the controller who coordinates and exposes all the available cryptographic operations.

to store keys and they feature a secure random number generator. Protection against physical tampering or side-channels is also desirable in most cases and mandated by the industry standards and best practices for cryptographic hardware. For this reason, our prototype (Section 7.5.1) is comprised of components that verify to a high level of tamper-resistance (i.e., FIPS140-2 Level 4) and expose a reliable random number generator (i.e., FIPS140-2 Level 3). There must be two or more ICs, of which at least one must be free of errors. We define this coalition of ICs as a *quorum*. The exact number of ICs in a quorum is determined by the operator depending on the degree of assurance needed (see also Subsections 7.3.2 and 7.6).

IC controller. The controller enables communication between the ICs and exposes the device's functionality to the users. It manages the bus used by the processing ICs to communicate with each other and the interface that receives the users' requests. As a minimum, it should support the following communication modes:

- *Unicast, IC-to-IC*: an IC sends instructions to another IC (and receives its response).
- *Broadcast, IC-to-ICs*: an IC broadcasts instructions to all other ICs (and receives their responses).

- *Unicast, Host-to-IC*: a remote client sends commands to a specific IC (and receives its response).
- *Broadcast, Host-to-ICs*: a remote client broadcasts commands to all ICs (and receive their responses).

The controller is also an IC and is also untrusted. Thus according to our threat model (Section 7.2) it may drop, modify or forge packets, in order to breach the device's integrity. It may also collude with one or more of the processing ICs.

Operator. The operator is responsible for setting up the system and configuring it. They source the ICs and make sure the quorums are as diverse as possible. Moreover, the operator determines the size of the quorums and sets the security parameters of the cryptosystem (Section 7.4). They are assumed to make a best effort to prevent compromises and may also be the owner of the secrets stored in the device.

Remote Host. The remote host is controlled by the user(s) and is connected to the IC controller. It allows users to submit requests and retrieve the results. The remote host can be any kind of computer either in the local network or in a remote one. In order for a request to be services by the processing ICs, the host must provide proof of its authorization. For instance, the requests could be signed with a public key associated with the user's identity (e.g., certificate by a trusted CA). More specifically, each request is comprised of the following information: 1) the operation requested, 2) any input data, and 3) the authorization signature (see also Section 7.3.1). As, we will discuss in Section 7.3.1), a compromised host may allow an adversary to use the device to perform cryptographic operations but would not enable the extraction of any secrets.

Communication Channels. At the physical level, the processing ICs, the controller and the users' hosts are connected through buses and network interfaces. As discussed in Section 7.2, the adversary is able to eavesdrop on the buses of the device and tamper with their traffic (e.g., inject or modify packets). We use established cryptographic mechanisms to ensure the integrity and confidentiality for the transmitted data. All unicast and broadcast packets are signed using the sender IC's certificate,

so that their origin and integrity can be verified by the recipients. Moreover, in cases where the confidentiality of the transmitted data needs to be protected, we use end-to-end encryption between the communicating parties.

7.3.1 Access Control

Access Control (AC) is critical for all systems operating on confidential data. In Myst, AC determines which hosts can submit service requests to the system, and the quorums they can interact with. Despite the distributed architecture of Myst, established AC techniques can be easily applied on a per-processor basis. Each IC is provided with a list of public keys of the hosts/users who are allowed to submit service requests. Optionally, this list may distinguish between hosts that have full access to the system and hosts that may only perform a subset of the operations. Once a request is received, the IC verifies the signature of the host and ensures that the public key is in the AC's list. The system's operator is responsible for configuring each quorum (i.e., size k, ICs participating) and initializing the processing ICs with the AC list. For example, the list could be initialized and distributed during the provisioning phase, and updated periodically when storing a secret or generating a new key pair. This is a critical procedure, as if one of the hosts turns out to be compromised, the device can be misused in order to either decrypt confidential ciphertexts or sign statements chosen by the adversary. However, the private keys stored in the quorum's ICs will remain safe as there is no way for the adversary to extract them (unless they use physical-tampering, which our prototype is also resilient against, Section 7.5.1).

7.3.2 Reliability Estimation

In the case of cryptographic hardware, in order for the operator to decide on the threshold k and the quorum composition, an estimation of the likelihood of hardware errors is needed. For this purpose, we introduce *k-tolerance*, which, given ICs from k supply chains, estimates the probability of their quorum being secure:

$$\Pr[\mathsf{secure}] = 1 - \prod_{i=1}^{k} \Pr[\mathsf{error}]_i \tag{7.1}$$

where, k denotes the number of ICs in the quorum and $\Pr[\mathsf{error}]_i$ the probability that IC i is defective. It is important to note, that this metric assumes that the components have non-overlapping supply chains and thus the probabilities of introducing errors are *independent*.

As there is no commonly accepted way of evaluating the likelihood of errors for a supply chain, the specific values largely depend on the information and testing capabilities the device's vendor has at their disposal. For instance, hardware designers that use split manufacturing [141, 142, 143] can estimate the probability of a backdoored component using the *k-security* metric [313].

On the other hand, COTS vendors cannot easily estimate the probability of a compromised component, as the security level at the fabrication facility is not always known. Despite that, it is still possible for them to achieve very high levels of error and backdoor-tolerance by increasing the size of the quorums and sourcing ICs from distinct facilities (i.e., minimizing the likelihood of overlapping supply chains).

7.4 Secure distributed protocols

In this section, we introduce a set of protocols that leverage the diversity of ICs in Myst to realize standard cryptographic operations on sensitive keying material. Our cryptosystem is comprised of a key generation operation (Section 7.4.1), the ElGamal [314] encryption operation (Section 7.4.2), distributed ElGamal decryption (Section 7.4.3) and distributed Schnorr [315] signature issuance (Section 7.4.5). For operational purposes, we also introduce a key propagation protocol that enables the distribution of secret shares between non-overlapping quorums (Section 7.4.6). These operations are realized using interactive secret-sharing protocols that rely on standard cryptographic assumptions over elliptic curve groups. The security guarantees of the protocols are those of secret-sharing schemes and are presented in detail in Section 7.6.4.

The ICs are initialized with the domain parameters $T = (p, a, b, G, n, h)$ of the elliptic curve to be used, where p is a prime specifying the finite field $\mathbb{F}_p$, a and b

are the curve coefficients, G is the base point of the curve with order n, and h is the curve's cofactor. This initialization procedure is further discussed in the Section 7.6.7 and must be completed prior to the execution of any protocol.

7.4.1 Distributed Key Pair Generation

The distributed key pair generation operation enables multiple ICs to collectively generate a shared public and private key pair with shares distributed between all participating ICs. More formally, a *quorum* Q of k processing ICs can *collectively* generate: 1) a random secret x which is an element of a finite field, and 2) a public value $Y_{agg} = x \cdot G$ for a given elliptic curve point G. At the end of the protocol, each IC in Q holds a share of the secret x, denoted as x_i and the public value Y_{agg}. In the presence of leaky processing ICs, none of the ICs have access to the full private key at any point. Given the produced public key, the remote host and anyone else can encrypt plaintexts or verify signatures produced by the device.

We opt for a threshold scheme that offers the maximum level of confidentiality (t-*of*-t, $k = t$), shown in Figure 7.2. However, there are numerous protocols that allow for different thresholds, such as Pedersen's VSS scheme [316, 317, 318]. The importance of the security threshold is discussed in more detail in Section 7.6.4.

Algorithm 7.4.1: TripletGen: Generation of a pair $x_i - Y_i$ along with the corresponding zero-knowledge proof of knowledge (k_i) of the discrete logarithm x_i of Y_i.

Input : The domain parameters λ
Output : An element triplet (x_i, Y_i, k_i)

1 $x_i \leftarrow \text{Rand}(\lambda)$
2 $Y_i \leftarrow x_i \cdot G$
3 $k_i \leftarrow \text{Sign}_{x_i}(Y_i)$
4 **return** (x_i, Y_i, k_i)

The execution of the protocol is triggered when an *authorized* host sends a service request (❶). Upon receiving the request, each member of Q runs Algorithm 7.4.1 and generates a triplet consisting of: 1) a share x_i, which is a randomly sampled element from $\mathbb{Z}_n$, 2) an elliptic curve point Y_i, and 3) a zero-knowledge proof of knowledge for the discrete logarithm x_i of Y_i, denoted k_i (❷). The proof-of-

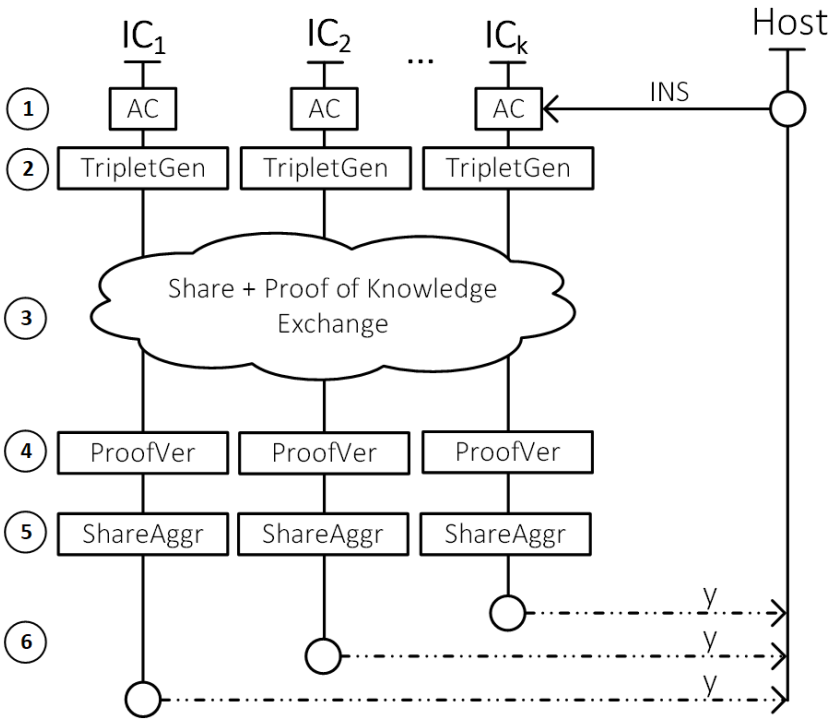


Figure 7.2: The interaction between the different participants during the execution of the distributed key pair generation protocol.

knowledge component (k_i) is necessary so as to prevent malicious processing ICs from launching rogue key attacks [319].

Algorithm 7.4.2: ProofVerify: Checks Y_i, against their respective proof-of-knowledge k_i.

Input : A list of the public key shares Y, and a list of the corresponding proofs-of-knowledge K
Output : Bool: Success/Failure

```
for i ∈ {1, |Y|} do
    if SignVerify(Y_i, k_i) ≠ True then
        return False
    return True
```

Then, the members perform a pairwise exchange of their Y_i shares and their corresponding k_i (❸). Once each member of the quorum receives shares from every other member, it executes Algorithm 7.4.2 to verify the validity of the shares they received (❹). If one or more proofs-of-knowledge fail the verification, then the member infers that an error (either intentional or unintentional) occurred and the protocol is terminated. Otherwise, if all proofs are successfully verified, the member

executes Algorithm 7.4.3 (❺) and reports it to the remote host (❻). The host accepts Y_{agg} as valid, if all the Y_{agg} received by all the ICs match.

Algorithm 7.4.3: ShareAggr: Aggregates elements in a set of shares (e.g., Elliptic Curve points, field elements).

Input : Set of shares Q
Output : The aggregate of the shares q

```
1 q ← 0
2 for q_i ∈ Q do
3 |   q ← q + q_i
4 return q
```

7.4.1.1 Replacing ZKP with Commitments

The above protocol (Figure 7.2) can be also realized using cryptographic *commitments* instead of zero-knowledge proofs. In principle, the ZKP version is more efficient as commitments require two rounds of communication i.e., one round where everyone "commits" to their public key share and another where all the parties "open" their commitments. In contrast, zero-knowledge proofs enable the members to broadcast their share Y_i and prove knowledge of its discrete logarithm within a single communication round.

Interestingly, in practice, a commitment-based protocol may have a lower execution time than a ZKP-based protocol, despite the fact that the former involves an additional round of communication. This is due to the complexity of the cryptographic operations required by the ZKP and the bit length of the information transmitted. Section 7.6.2 details one such case and compares the runtimes and the bit lengths of the two approaches.

To modify the protocol to use commitments (instead of zero-knowledge proofs), we first need to adapt Algorithm 7.4.1 to generate a commitment c_i to Y_i. This commitment replaces the zero-knowledge proof k_i in the triplet generated (Algorithm 7.4.4). Then, each member shares the commitment c_i with the rest of the quorum's members, while keeping the private key share x_i secret. Once all the commitments have been exchanged, an additional pairwise exchange is performed for members to "open" their commitments by sharing Y_i. The rest of the protocol

remains the same.

Algorithm 7.4.4: TripletGen: Generation of a pair $x_i - Y_i$ along with the corresponding commitment (c_i) to the share of the public key Y_i.

Input : The domain parameters λ
Output : An element triplet (x_i, Y_i, c_i)

$x_i \leftarrow \text{Rand}(\lambda)$
$Y_i \leftarrow x_i \cdot G$
$c_i \leftarrow \text{Commit}(Y_i)$
return (x_i, Y_i, c_i)

For example, in the IC use case, the signature k_i in Algorithm 7.4.1 could be replaced by the SHA-256 digest of Y_i. Once the triplet has been generated, the IC broadcasts its commitment c_i to the rest of the quorum. It is important, however, that at this step the ICs exchange only their commitments without revealing their share of public key. Once each IC has received commitments from all the other members of the quorum, it then broadcasts its share of the public key. If the commitments received verify correctly with the shares of the public key, the IC continues the execution as above, otherwise it reports an error to the host and aborts.

7.4.2 Encryption

For encryption, we use the Elliptic Curve ElGamal scheme [314, 320] (Algorithm 7.4.5). This operation does not use the private key and thus there is no need to perform it in a distributed manner. It can be performed directly on the host or remotely by any party holding the public key.

Algorithm 7.4.5: Encrypts a plaintext using the agreed common public key.

Input : The domain parameters T, the plaintext m encoded as an element of the group $\mathbb{G}$, and the calculated public key Y_{agg}.
Output : The ElGamal ciphertext tuple, $(C_1,\ C_2)$.

$r \leftarrow Rand(T)$
$C_1 \leftarrow r \cdot G$
$C_2 \leftarrow m + r \cdot Y_{agg}$
return $(C_1,\ C_2)$

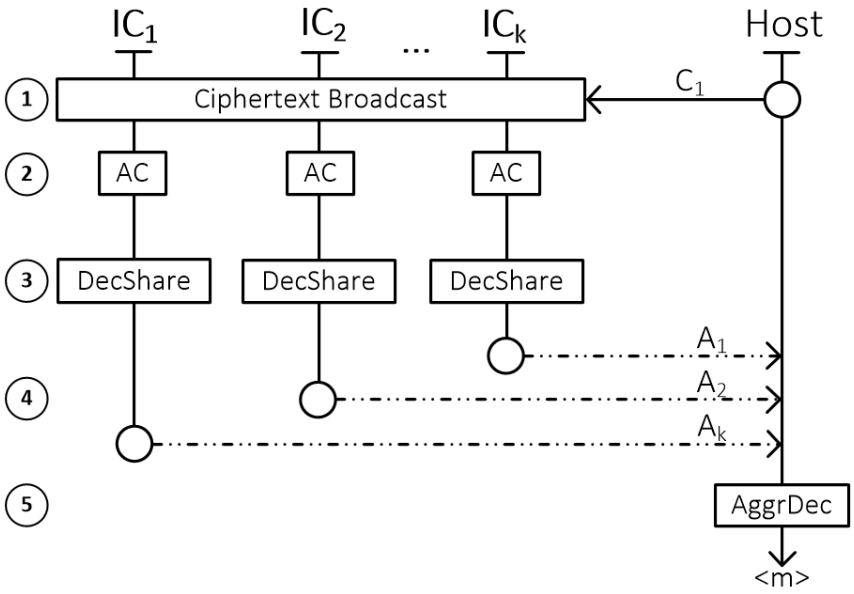


Figure 7.3: The interaction between the different ICs during the execution of the distributed decryption protocol.

7.4.3 Decryption

One of the fundamental cryptographic operations involving a private key is ciphertext decryption. In settings where the key is held by a single authority, the decryption process is straightforward but assumes that none of the components involved are compromised. Myst alleviates this requirement by distributing the decryption process between k distinct processing ICs that hold shares of the key (Figure 7.3).

The protocol runs as follows: Initially, the host broadcasts the decryption instruction along with the part of the ElGamal ciphertext C_1 to the processing ICs (❶). Upon reception, the ICs first verify that the request is signed by an authorized user (❷), and then execute Algorithm 7.4.6 to generate their decryption shares A_i (❸). Once the shares are generated, they are signed by their respective ICs, get encrypted under the host's public key and are sent back to the host (❹). Once the host receives k decryption shares, executes Algorithm 7.4.7 to combine them and retrieve the plaintext m (❺).

It should be noted that during the decryption process, the plaintext is not revealed to any other party except the host, and neither the secret key nor its shares ever leave the honest ICs. An extension to the above protocol can also prevent malicious

Algorithm 7.4.6: DecShare: Returns the decryption share for a given ciphertext.

Input : A part of the ElGamal ciphertext (C_1) and the IC's key share x_i.
Output : The decryption share A_i, where i is the IC's uid.

$A_i \leftarrow -x_i \cdot C_1$
return A_i

Algorithm 7.4.7: AggrDec:Combines the decryption shares and returns the plaintext for a given ciphertext.

Input : The ElGamal ciphertext C_2 and a set of decryption shares A.
Output : The plaintext m.

$D \leftarrow 0$
for $A_i \in A$ **do**
  $D \leftarrow D + A_i$
$m \leftarrow (C_2 + D)$
return m

ICs from returning arbitrary decryption shares, by incorporating a non-interactive zero-knowledge proof (ZKP) [321] in the operation output.

7.4.4 Random String Generation

Another important application of secure hardware is the generation of a random fixed-length bit-strings in the presence of active adversaries. The key property of such systems is that errors (e.g., a hardware backdoor) should not introduce bias in the generated bitstring. The implementation of such an operation is straightforward. The remote host submits a request for randomness to all the processing ICs of the quorum. Subsequently, each IC independently generates a random share b_i, encrypts it with the public key of the host, and signs the ciphertext with its private key. The host receives all the shares, combines them to retrieve b and then uses an one way function (e.g., SHA3-512 [322]) to convert b to a fixed length string.

7.4.5 Signing

To issue digital signatures, we introduce a multi-signature variant of the Schnorr signature scheme [323], which has also similarities with the construct in [324]. This scheme enables a quorum to collectively sign a single message (the private key remains distributed across the different processing ICs) and allows for easy signature

verification just with the aggregate public key (Y_{agg}). The main difference of our setup with the majority of the related works on multisignatures [325, 326, 327, 328, 324, 329] is that we can rely on the host to act as an aggregation and storage point (although not a trusted one). This reduces the communication complexity of our protocol as it enables us to replace rounds of pairwise communication between the processing ICs with simple unicast transmissions from all the ICs towards the host.

As discussed in Sections 7.2 and 7.3.1, the host can be fully compromised by the adversary. It is thus possible for an adversary to issue malicious service requests (e.g., for decryption or document signing). Electronic authentication methods such as multi-factor authentication could help reduce the risk of such breaches but this goes beyond the scope of this chapter. In our current design the device has no way of distinguishing between a honest host and a compromised one. However, even in the case of a fully compromised host the adversary should not be able to extract any secret shares from the processing ICs.

Algorithm 7.4.8: SigShare: Returns the signature share of the IC for a given plaintext and index j.

Input : The digest of the plaintext to be signed $H(m)$, the IC's private key of x_i and secret s, an index j, and the aggregated random EC point R_j.

Output : The signature share tuple $(\sigma_{ij}, \varepsilon_j)$

1 $\varepsilon_j \leftarrow Hash(R_j||Hash(m)||j)$
2 $r_{ij} \leftarrow PRF_s(j)$
3 $\sigma_{ij} \leftarrow r_{ij} - x_i \cdot \varepsilon_j \mod n$
4 **return** $(\sigma_{ij}, \varepsilon_j)$

Before the execution of the protocol, all k processing ICs cooperate to generate a public key y (using the distributed key generation operation, Section 7.4.1) and each store securely their own share x_i. Let $PRF_s(j)$ be a pseudorandom function with key s that takes j as input and instantiates it as $Hash(s||j)$. Each IC generates a secret s for the PRF and stores it securely. Once these have been completed, the signing protocol can be executed. The protocol is comprised of the *caching* and *signing* phases.

In the *caching phase* (Figure 7.4), the host queries each of the processing ICs

for random group elements R_{ij}, where i is the id of the IC and j a monotonically increasing request counter (❶). Upon receiving the request, each IC verifies that the host is authorized and then applies the keyed pseudorandom function on the index j to compute $r_{i,j} = PRF_s(j)$ (❷). The IC then uses $r_{i,j}$ to generate a group element (i.e., an EC Point in our case) as such $R_{ij} = r_{i,j} \cdot G$ (❸), and returns it to the host.

The host uses Algorithm 7.4.3 to compute the aggregate of the group elements (R_j) and stores it for future use (❹). It should be noted that the storage cost for R_j is negligible: for each round the host stores only 65 Bytes or 129 Bytes depending on the elliptic curve used (for R_j) and the corresponding round index j. This allows the host to run the caching phase multiple times in order to generate a list of random elements that can be used later.

The *signing phase* (Figure 7.5) starts with the host sending a Sign request to all processing ICs (❶). Such a request includes the hash of the plaintext $Hash(m)$, the index of the round j, and the random group element R_j corresponding to the round. Each IC first verifies that the host has the authorization to submit queries (❷) and that the specific j has not been already used (❸). The latter check on j is to prevent attacks that aim to either leak the private key or to allow the adversary to craft new signatures from existing ones. If these checks are successful, the IC executes Algorithm 7.4.8 and generates its signature share (❹). The signature share $(\sigma_{ij}, \varepsilon_j)$ is then sent to the host (❺). The host can combine all the shares σ_{ij} for the same j using Algorithm 7.4.3 to recover σ_j, thus obtaining the aggregate signature $(\sigma_j, \varepsilon_j)$ (❻).

The security proof of this protocol can be found in [13]. To verify the produced signature, a recipient of $\langle (m, j),\ \sigma, \varepsilon \rangle$ can check if it holds that

$$\varepsilon == Hash(R||Hash(m)||j)$$

, where $R = \sigma \cdot G + \varepsilon \cdot Y$.

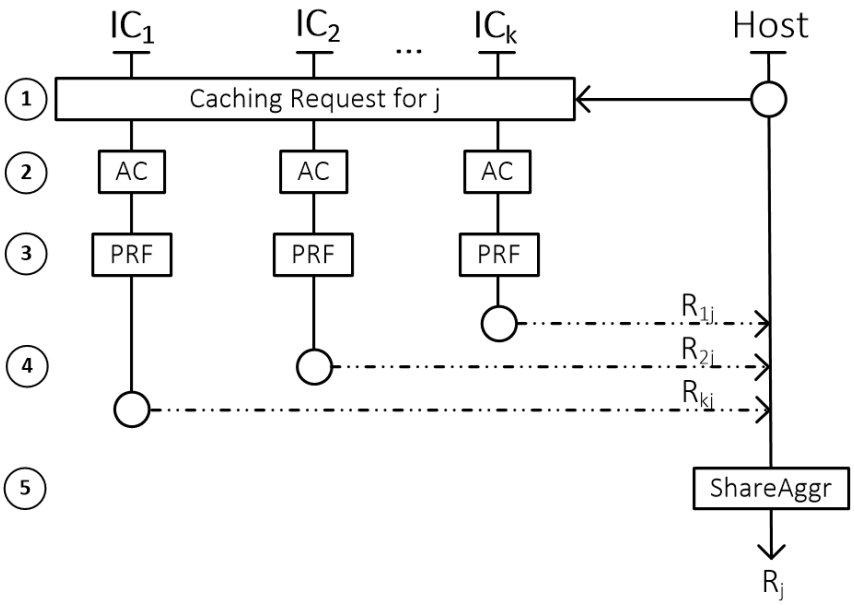


Figure 7.4: The interaction between the different actors during the caching phase of the distributed signing protocol.

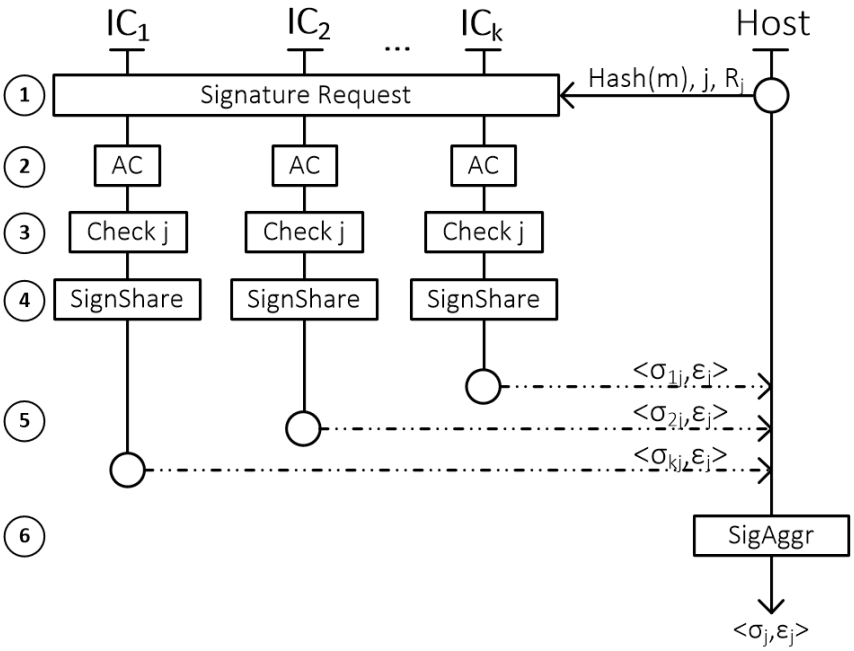


Figure 7.5: The interaction between the different actors when issuing a multi-signature based on cached randomness.

7.4.6 Key Propagation

In cases where more than one quorum is available, it is useful to enable them all to handle requests for the same public key. This is of particular importance for both the

system's scalability and availability, as we further discuss in Sections 7.6.3 and 7.6.5 respectively.

Once a quorum Q_1 has generated its public key Y (Section 7.4.1), the operator can specify another quorum Q_2 that is to be initialized with shares for Y. For this purpose, each member q_i of Q_1 splits its secret x_i in $|Q_2|$ shares and distributes them to the individual members of Q_2. To do that q_i follows the secret sharing method shown in Algorithm 7.4.9 but any t-of-t secret sharing scheme proposed in the literature [330, 331, 316] would do.

Algorithm 7.4.9: SecretShare: Returns a vector of shares from a secret.

Input : The domain parameters T, a secret s which is to be shared, and the number of shares k.
Output : A vector of shares $\vec{v}_s$

1 **for** *(*$i = 0$ *to* $k-2$*)* **do**
2 $\quad \vec{v}_s[i] \leftarrow Rand(T)$
3 $\vec{v}_s[k-1] \leftarrow (s - \vec{v}_s[1] - \vec{v}_s[2] - ... - \vec{v}_s[k-2])$
4 **return** $\vec{v}_s$

Once all members of Q_2 receive $|Q_1|$ shares, each one of them individually combines them to retrieve their share of the secret x'_i. To obtain x'_i, a member simply aggregates the incoming shares modulo p (p was provided with the domain parameters T). An additional benefit of such a scheme is that it enables a device to support quorums of varying sizes (i.e., $|Q_1| \neq |Q_2|$).

It should be also noted that the straightforward approach of having each member of q_1 send their share of x to a member of q_2 is insecure, as malicious members from Q_1 and Q_2 could then collude to reconstruct the public key.

7.5 Implementation

We now provide the implementation details of our Myst prototype. We first examine the components and capabilities of our custom hardware platform and subsequently we focus on the software of the untrusted ICs and the remote host.

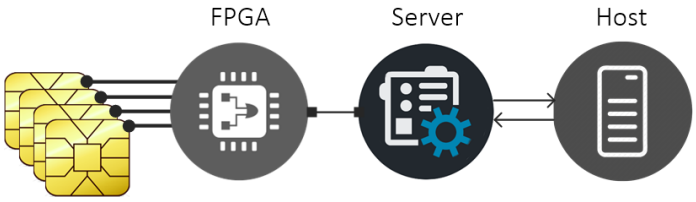


Figure 7.6: Overview of the components of our custom high-assurance device.

7.5.1 Hardware Design & Implementation

For our prototype, we designed our own custom circuit board, which features 120 processing ICs (set to use 40 quorums of 3 smart cards from different manufacturers) interconnected with dedicated buses with 1.2 Mbps bandwidth. The processing ICs are JavaCards (version 3.0.1), loaded with a custom applet implementing the protocols outlined in Section 7.4. JavaCard is a platform suitable for our purposes as it provides good interoperability (i.e., applets are manufacturer-independent), which contributes to IC-diversification and prevents lock-in to particular vendors. Moreover, they are also resilient to leakages as:

- ❖ They are tamper-resistant (FIPS140-2 Level 4, CC EAL4) and can withstand attacks from adversaries with physical access to them [332].
- ❖ They feature secure (FIPS140-2 compliant) on-card random number generators.
- ❖ They offer cryptographic capabilities (e.g., Elliptic curve addition, multiplication) through a dedicated co-processor.
- ❖ There are numerous independent fabrication facilities (see also Section 7.6).

In addition to these, they have secure and persistent on-card storage space, ideal for storing key shares and protocol parameters.

The host is implemented using a computer that runs a python client application, which submits the user requests (e.g., Ciphertext Decryption) to Myst using a RESTful API exposed by the device. The incoming requests are handled by a Spring server, which parses them, converts them to a sequence of low-level instructions and then forwards these to the IC controller, through an 1Gbps TCP/UDP interface. The

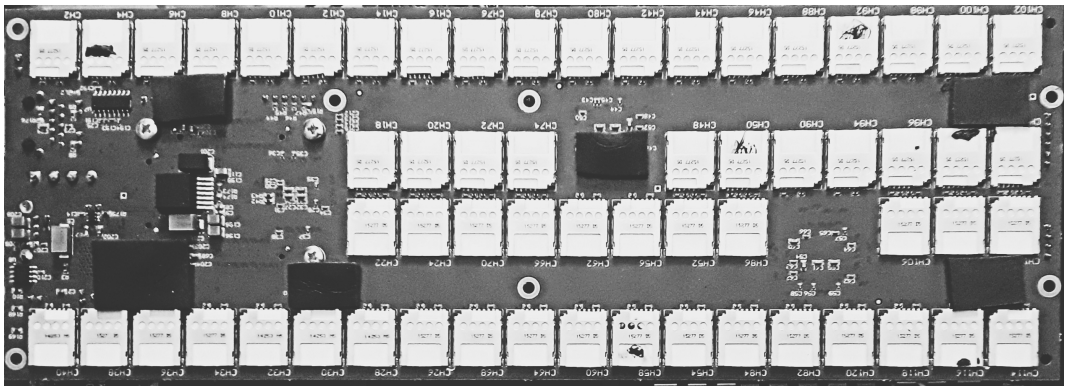


Figure 7.7: Myst's smartcard board supports 120 ICs (60 on each side). Our configuration splits them into 40 quorums of 3 diverse ICs each.

ICs controller is a programmable Artix-7 FPGA that listens for incoming instructions and then routes them to the processing ICs, through separate physical connections. We took special care that these buses offer high bandwidth (over 400kbps), to prevent bottlenecks between the controller and the ICs even under extreme load. Once the ICs return the results, the controller communicates them back to the server, that subsequently forwards them to the host.

7.5.2 Software

We now implement the protocols of Section 7.4 and provide the necessary methods for inter-component communication and system parameterization.

We develop and load the processing ICs with JavaCard applets realizing methods for card management, key generation, decryption, and signing. JavaCard is an excellent platform for use cases that require high-integrity cryptographic computations. However, we still faced various challenges implementing the cryptographic algorithms from Section 7.4. More specifically, the cards available in the market do not support Big Integer or elliptic curve arithmetic operations, even though the JavaCard API (as of version 2.2.2) specifies a BigNumber class. Moreover, the only software-based BigInteger library available (i.e., BigNat[1]) is not maintained and lacked essential operations. The only available option for developers that require access to low-level primitives is to request access to the manufacturer-specific proprietary APIs that realize some of these operations (e.g., EC point addition). However,

[1]https://ovchip.cs.ru.nl/OV-chip_2.0

gaining access to these APIs often requires signing a non-disclosure agreement and breaks the interoperability of the applet as the produced code will work with the products of a single manufacturer. In our case, this would prevent us from using cards from different manufacturers, thus reducing the diversity in the final quorum.

To overcome these limitations, we built a library that realises classes for Integers, Big Numbers and Elliptic curve points as well as the corresponding arithmetic operations. Our functionality is based solely on the public JavaCard API and incorporates various optimizations to utilize the cryptographic coprocessor as much as possible. The current implementation is based on the NIST P-256 [333, 334] curve that provides at least 128 bits of security. However, it can also be easily adapted to support other curves too. More information on the optimizations used can be found in [30]. The source code of the library has been publicly released at *https://OpenCryptoJC.org* and the applet for Myst at *https://github.com/OpenCryptoProject/Myst*.

7.5.3 Optimizations

Although JavaCard applets are compiled with a standard Java compiler, the computational limitations of the platform make common Java patterns (e.g., frequent array reallocation due to resizing) prohibitively expensive. Instead, we use various other optimization techniques to increase the speed and the side-channel attack resilience of our applet [335]:

- Where possible, use hardware-accelerated cryptographic methods from the JavaCard API instead of custom implementations interpreted by the JavaCard virtual machine.
- Store the session data in the faster RAM-allocated arrays instead of the persistent, but slower EEPROM/Flash.
- Use copy-free methods which minimize the transfer of data in memory, and utilize native methods provided by the JCSystem class for array manipulation like memory set, copy and erase.
- Pre-allocate all cryptographic engines and key objects during the applet installation. No further allocation during the rest of the applet lifetime is performed.

- ❖ Additional cryptographic engines and key objects are used so as to minimize the number of key scheduling and engine initializations for a particular key, as these operations impose non-trivial overhead [335].

7.5.4 System States

Initially, the system is in an non-operational state, where the processing ICs do not respond to user requests. To make it operational, a secure initialization process has to be carried out. During the initialization, the processing ICs and the other components described in 7.3 are loaded with verified application packages and the domain parameters for the distributed protocols. Moreover, the ICs are provided with their certificates, which they will later use to sign their packets and establish secure communication channels.

Once the initialization process has been completed, the system switches to an operational state and is available to serve user requests. Depending on the configuration, the system may be brought back to an uninitialized state, in order to load new software or change the protocol parameters. We further discuss the importance of the initialization process in Section 7.6.5.

7.6 Evaluation

In this section, we evaluate Myst by examining both its performance and its qualitative properties.

7.6.1 Experimental Setup

All evaluations were performed using the setup illustrated in Figure 7.6. The host is a single machine with CentOS 7 (3.10.0 Linux kernel), featuring a 3.00GHz Intel(R) Core i5-4590S CPU and 16GB of RAM. It uses a python client script to submit service requests to Myst, through a 1 Gbps Ethernet interface. Upon reception, the server uses the Java Spring framework [336] to parse and forward them to the Artix-7 FPGA, which subsequently routes them to the individual smart cards. In all our experiments, we collect response-time measurements from both the host and the Spring server. On average the round-trip from the host to the server takes 5*ms*. In

the rest of this chapter, we report the measurements from the host.

7.6.2 Performance Impact

This subsection evaluates the performance of our protocols and compares the throughput and latency of our multi-IC prototype with that of a single-IC system. Moreover, we examines the effect of our optimizations on the overall performance of the system.

Methodology. To better understand the overhead that the use of a distributed architecture entails, we run experiments that measure the latency as the size of the processing quorum grows. We sequentially submit 1,000 requests for each of the supported cryptographic operations in quorum with sizes from one to ten ICs, and measure the response latency on a per-operation basis. Simultaneously, to gain a more in-depth understanding of the causes of latency, we benchmark the execution time of each operation. For key generation measurements, we use the commitment-based version of the protocol (Section 7.4.1.1) as it is considerably faster than the zero-knowledge proof version. This is because the ECDSA signature verification has an average runtime of 262 ms with the signature being 512 bits long, while SHA-256 takes only 110 ms and the digest is 256 bits.

Results. Figure 7.8 shows the average response time for performing *key generation*, *decryption* and *signing* using quorums of different sizes. Decryption is the fastest operation with 119 ms runtime that does not increase with the size of the quorum. This is because the protocol has only a single communication round and does not require interaction between the processing ICs. This highlights that decryption is only CPU-bound and that the capacity of the prototype's buses and network interfaces does not pose a bottleneck. High-throughput decryption is of high importance in applications such as secure key derivation in mix-network infrastructures [337] that heavily rely on decryption.

Similarly, the runtime for signature issuance (signing phase) remains constant ($\sim$517ms) regardless of the quorum size. This is because our multi-signature protocol does not require interaction between the processing ICs. The caching phase is not included in Figure 7.8 and takes on average 169ms without being affected by the

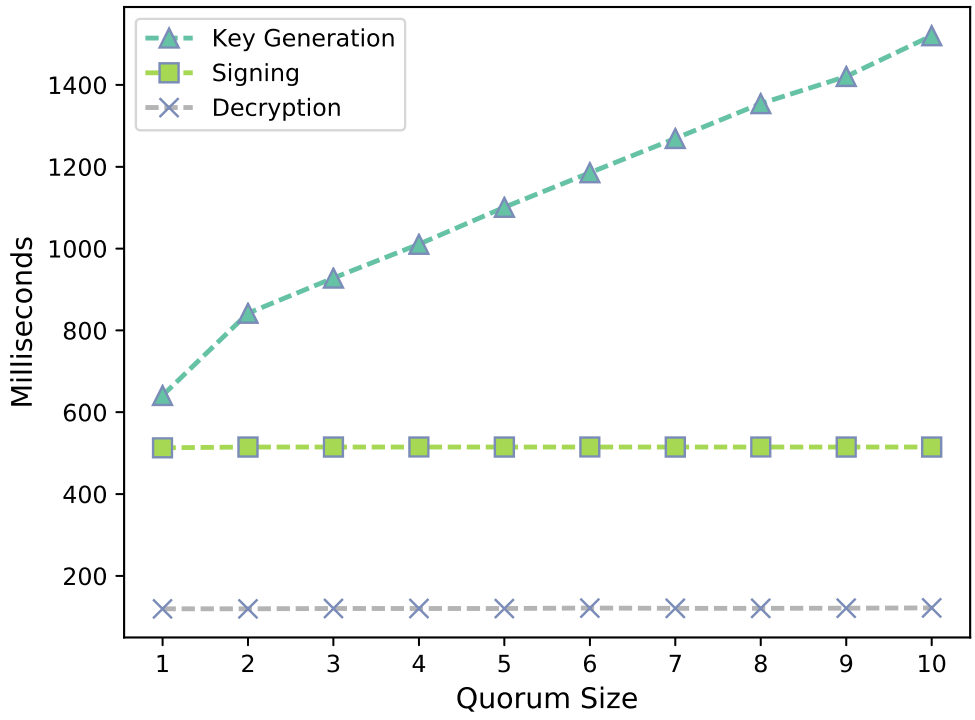


Figure 7.8: The average runtime of each distributed protocol, in relation to the quorum size (i.e., a coalition of multiple ICs) size.

quorum size. In practice, the operator can execute thousands of caching rounds when the device is idle (e.g., overnight) to pre-generate randomness for future signatures. The runtime discrepancy between the decryption and signing protocols is primarily due to the different operations required and the limited expressiveness of the JavaCard API. In order to implement the signing protocol in JavaCards, we need to make use of the general purpose processor of the card as certain operations are not available natively.

The key generation protocol has a latency of ∼650 ms for a single processing IC and each additional IC results in an runtime increase of approximately 90 ms. To better understand the causes of this latency, Figure 7.9 illustrates the runtime of the low-level operations involved in the protocol. For small quorums, the cost of key generation is dominated by the "INS_KEYGEN_INIT" operation that generates a secret share, derives a public key share and initializes the arrays for storing the shares from the other quorum members (624 ms). However, as the size of the quorum grows, the operations that involve pairwaise communication between the

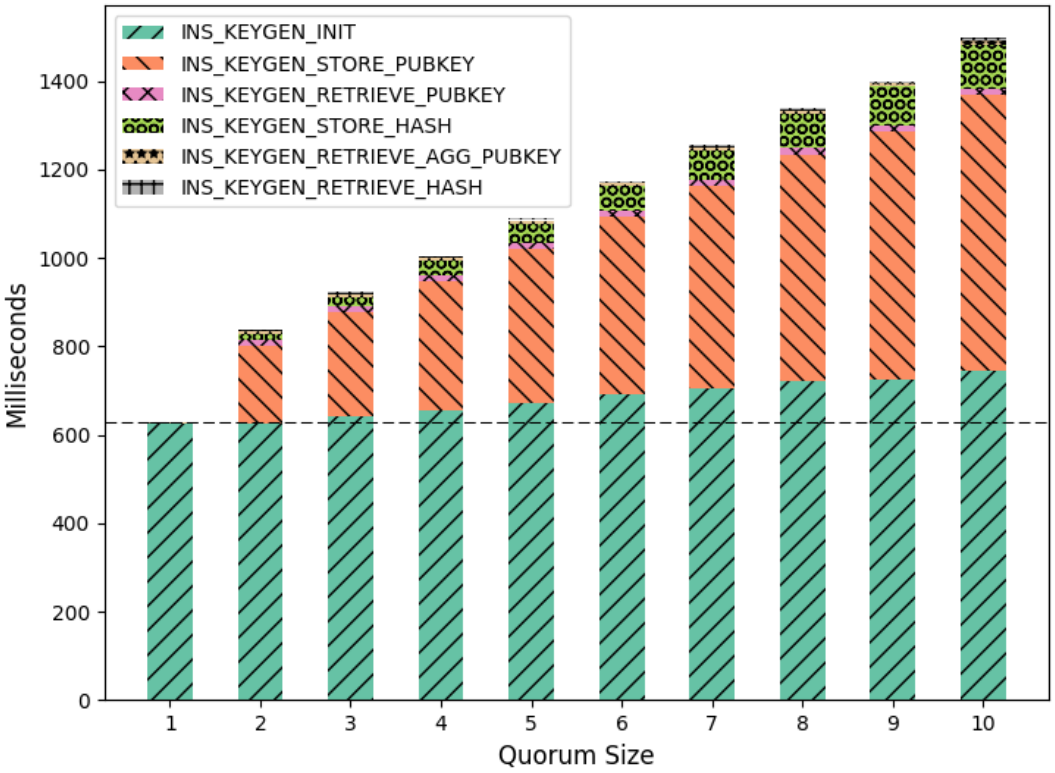


Figure 7.9: Breakdown of the runtime for low-level instructions that comprise the key generation operation, in relation to the quorum size. The horizontal reference line represent the cost of using a single IC.

different ICs (i.e., "INS_KEYGEN_STORE_PUBKEY" for public key shares and "INS_KEYGEN_STORE_HASH" for commitments) become significant. For instance, in a quorum of 10 ICs the "INS_KEYGEN_STORE_PUBKEY" operation accounts for nearly 50% of the protocol's runtime. However, for smaller quorums of 3, the impact on the runtime is merely 303ms, when compared to a single IC. The rest of the low-level operations have a negligible cost regardless of the quorum's size.

7.6.3 Scalability & Extensibility

This section examines how the throughput of our prototype grows as more processing power (i.e., quorums) is added. The absence of bottlenecks in the system is important to be able to provide high availability in production environments.

Methodology. To determine how efficiently our design scales in practice, we run a series of experiments that measure our prototype's throughput, for varying numbers of processing quorums. As described in Section 7.5.1, our board supports up to 120

processing ICs which can be divided into multiple quorums so as to serve requests simultaneously. To benchmark the scalability of the system, on each instance of the experiment, we submit 10,000 requests for each high-level operation supported and measure the overall throughput. We fix the quorum size k to 3 and measure the throughput of our prototype up to its maximum capacity of 40 quorums. For simplicity, we assign each processing IC to only one quorum. However, it is also technically possible for an IC to participate in more than one quorums and handle shares from multiple keys.

Results. Figure 7.10 illustrates the throughput of the system (in operations per second) for varying number of processing quorums. The maximum throughput is 315ops/sec for decryption and 77 ops/sec for signing when all 40 quorums are used. As the number of quorums grows, the decryption performance increases linearly at a rate of $\sim$9 ops/sec for every additional quorum. Similarly, every additional quorum increases the throughput by $\sim$1.9 ops/seq for signature issuance. This indicates that the system is CPU-bound and that the communication channels between the ICs and the host are not a bottleneck. Consequently, a system with this design is also applicable to high-throughput use cases.

In cases, where a lower threshold is used (i.e., $k < t$), we do not expect any performance degradation as the communication complexity will remain the same. However, this may result in some processing ICs being idle longer. This can be alleviated by having processing ICs that server requests for more than one quorum.

7.6.4 Tolerance levels

Our proposed system design aims to prevent various forms of leakage by being tolerant against both fabrication-time and design-time errors. In this section, we examine the relationship between the system's parameters and the tolerance levels achieved depending on the error type (Table 7.1).

In practice, the threshold k and the size of the quorums t express the trade-off between confidentiality and robustness for each particular quorum. The relationship between these two parameters determines how many ICs can cease to function before the quorum fails: When k equals the number of processing ICs t, the secrets are safe

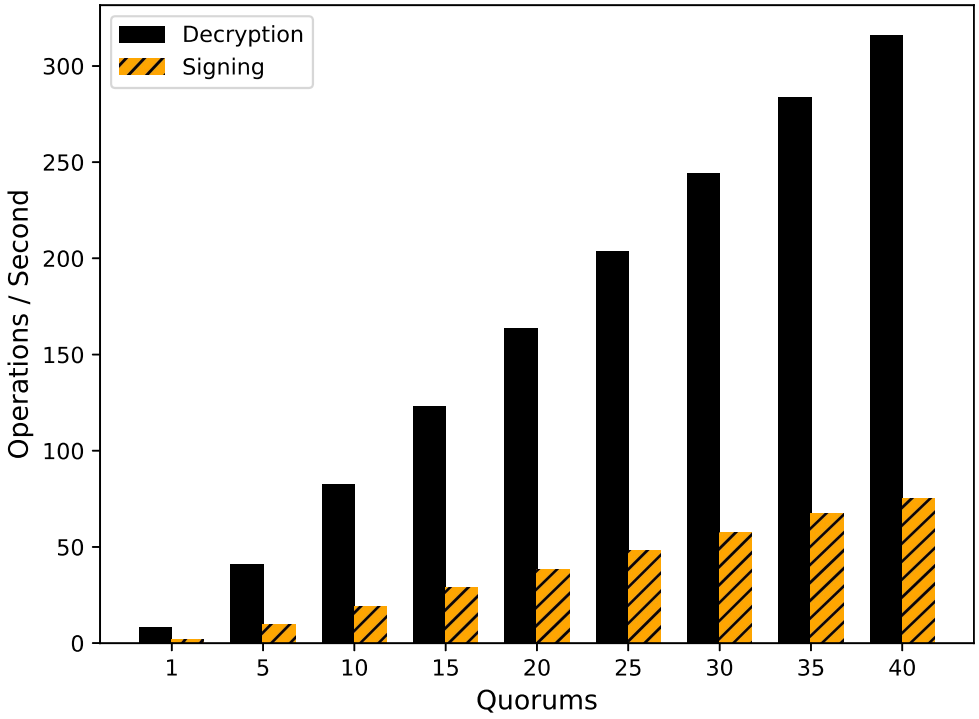


Figure 7.10: The average system throughput in relation to the number of quorums ($k = 3$) that serve requests simultaneously.

Parameters	Leakage	Denial-of-Service	IC Failures
Single IC	0	0	0
$k = t$	$t-1$	0	$n-1$
$k < t$	$k-1$	$t-k$	$(t-k)*n$

Table 7.1: Number of defective processors that different setups can tolerate under three error scenarios. The system is assumed to feature n identical quorums of size t with a secret-sharing threshold k.

in the presence of $t-1$ compromised and colluding ICs. On the other hand, a more "relaxed" threshold (i.e., $k < t$) enables the quorum to remain fully functional unless more than $t-k$ ICs fail (maliciously or through wear). Alternatively, $(k = t)$-systems can withstand multiple ICs failing (due to wear) using the technique introduced in Section 7.4.6. This technique enables several quorums to handle requests for the same public key, and thus even if multiple ICs (and consequently the quorums they belong to) fail, the remaining quorums can still handle incoming requests for those public keys. It should be noted that this technique provides robustness only in the presence of faulty and defective ICs, but does not mitigate denial of service attacks. This is because, all n quorums are identical (same manufacturer diversity) and thus a malicious IC from a breached supply chain will participate in all of them. In contrast, defective ICs will fail with a probability less than 1 (post-fabrication tests detect ICs that fail 100% of the times) and thus at least some of the quorums will remain functional.

From all the above, the security guarantees of Myst are determined by the threshold k, the IC diversity and the number of quorums. In our prototype, we chose to maximize resistance to leakage in the presence of $t-1$ actively malicious ICs. Malicious ICs launching denial-of-service attacks are easier to detect and replace, compared to those subtly leaking secrets or weakening keys. In cases where increased robustness and availability are paramount, the security level can be adjusted in favor of redundancy using the appropriate threshold schemes [316].

7.6.5 Other Considerations

In this section, we discuss various qualitative properties of high-assurance systems.

7.6.6 Physical Security & Diversity

Smartcards form the core of our prototype and have multiple benefits that make them a component suitable for leakage-tolerant systems. To begin with, they are designed to operate in a hostile environment that is fully controlled by the adversary [332]. For this reason, they come with very high tamper-resistance (FIPS140-2, Level 4) and secure storage capabilities that are constantly enhanced. Moreover, there are

numerous manufacturers owning foundries (e.g., NXP semiconductors, ST Microelectronics, Samsung, Infineon, Athena), as well as various independent fabrication facilities [338, 339, 340, 341]. This allows the system operator or the manufacturer to maintain a healthy fabrication and supply chain diversity for the quorums of their device.

7.6.7 Code & Parameter Provisioning

Even if the hardware components have been perfectly diversified, the security of the device can be compromised if the code deployed on the processing ICs during the provisioning phase contain errors. This can occur if the code is faulty or backdoored, and in cases where the party responsible for the provisioning suffers a security breach.

We now discuss three approaches that can help alleviate this risk. First, we can perform the provisioning step at the factory. This is a standard practice in the telecommunications industry where sim cards are loaded with a predefined applet before they are handed to the mobile network operator. This is in line with our assumption that some of the foundries are honest and the adversary will have to compromise all the fabrication facilities to successfully leak any secrets stored in the device. Alternatively, we can use a trusted off-line provisioning facility that is not under the control of an adversary. While this is a stronger assumption, such a facility would need to only to guarantee high integrity, as no secrets are involved in the provisioning step (secrets are generated within the ICs as part of protocols executed later). Finally, the software development can be diversified too, with several independent development vendors implementing and auditing the source code.

7.7 Myst Prototype Extensions

Our Myst prototype (as seen in Section 7.5.1) can also be extended to use multiple ICs boards, if higher throughput is needed. Figure 7.11 depicts an instantiation with 240 ICs: two boards holding 120 ICs each (60 ICs on each side) and an Artix-7 FPGA to facilitate the inter-IC communication within each board.

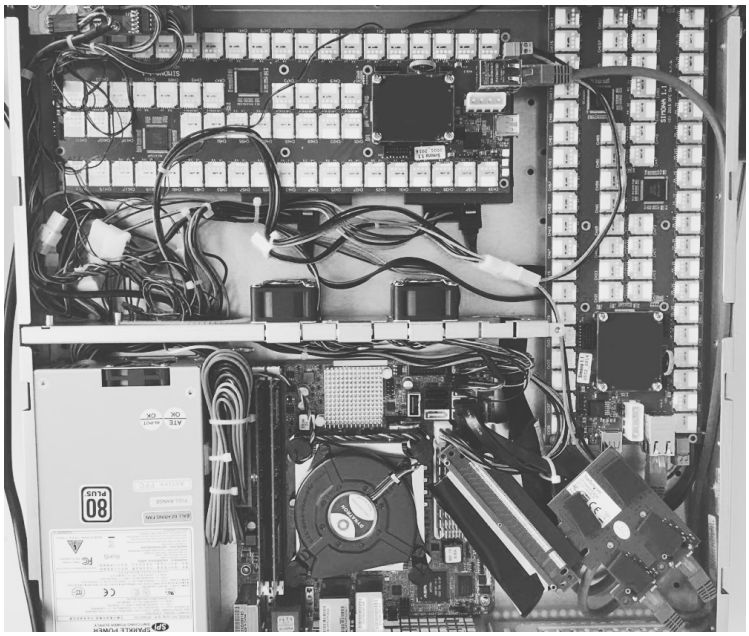


Figure 7.11: Myst's prototype with 240 JavaCards fitted into an 1U rack case.

7.8 Conclusions

This chapter introduced an alternative approach in system design, leveraging a diverse set of untrusted hardware components to minimize the risk of leakages by distributing trust between them. This design relaxes the common assumption that all the components of high-assurance cryptographic hardware must be secure and free from errors. For example, it can provide resilience to the location-based leakages seen in Chapter 6. Additionally, die-level countermeasures remain applicable and can be used in tandem to further increase the security of the final system. Overall, while a single countermeasure cannot effectively protect from all possible types of leakage, the diversification of a system's components can help substantially in the protection of the user's information and secrets stored or processed in a device.

The HSM outlined in 7.5.1 is currently used in production by Enigmabridge [342, 343] with the Myst schemes being also available to customers on request. Finally, [308] further extends our design by diversifying also across cryptographic schemes. The combined scheme will become applicable to cryptocurrency key management once the Bitcoin blockchain [344] introduces the Taproot [345, 346] consensus update.

Chapter 8

Conclusions & Future Work

In this thesis, we studied information leakages with an emphasis on their impact on networked and interconnected systems. We investigated their causes in different contexts and contributed to a better overall understanding of the problem. We found that a common cause of leakages, that traverses various layers of the computing stack, is the lack of a formal specification of the protocol and/or of its security properties. Chapters 4, 5 and 6 demonstrated how this manifests into inaccurate threat models that do not correctly represent all the information related to the security of the protocol. Moreover, we found that even minor leakages can be easily exploited by adversaries, thus making it very hard for complex, networked systems to remain secure. Chapter 7 builds on this conclusion and proposes a leakage-resilient system design that retains its security properties even in the presence of leakages.

We started our investigation from the TLS protocol, in chapter 4. Our examination of the specifications of the versions 1.2 and 1.3 [52, 50] uncovered that privacy is not clearly defined, even though it is listed as one of the protocol's primary goals. This, coupled with generic claims for "privacy" on TLS by articles and technical reports [174, 175, 176, 177, 178, 179, 52] has led to confusion as to the properties TLS can reliably provide. To better evaluate the level of privacy that a user should expect in real-life, we designed and implemented an adversary that exploits patterns in the encrypted TLS traffic stream to extract sensitive information about the user's activities.

Our experiments showed that adversaries have greatly benefited from the ad-

vancements in machine learning and can successfully launch large-scale attacks even under non-optimal conditions. Overall, the TLS protocol cannot reliably provide privacy with regards to the users' browsing habits. However, low-overhead countermeasures could potentially thwart the problem, without imposing an impractical overhead on bandwidth. A promising direction is countermeasures that guarantee a minimum anonymity set size that can be adjusted by webserver administrators depending on the sensitivity of the contents of their website.

Unlike adversaries targeting static targets, adaptive adversaries use models that learn equivalences in the traffic patterns without necessarily memorizing the specific patterns exhibited by each targeted webpage or website. As a result, they are more robust to changes in the page contents. However, this could potentially make them more sensitive to changes in the infrastructure of the operator hosting the page (e.g., use of CDNs). An investigation in this aspect of fingerprinting would shed light into how infrastructural changes affect an attacker but more importantly study how the centralization of hosting services (e.g., on Amazon EC2) could potentially enhance the transferability of a model across unrelated websites. Moreover, while experiments in our and past works provide a good indication of the capabilities of passive adversaries, there are very few works on *active* fingerprinting attacks against anonymity networks/websites [347, 348] and none (that we are aware of) against TLS/webpages. Such adversaries are potentially more capable as they have more actions at their disposal (e.g., trigger packet re-transmissions). It is thus a promising direction for future works to also study the performance and characteristics of such adversaries so as to reliably inform the selection of defenses.

In Chapter 5, we studied the ultrasonic communications channel and identified various security and privacy shortcomings. Unlike the TLS protocol, which had a specification but failed to define privacy, there was no commonly accepted specification for ultrasonic communication in the context of mobile devices. This led to various security issues that could be exploited to either directly breach the privacy of end-users or launch side-channel attacks against other applications and channels. Interestingly, one of our attacks deanonymizes anonymity network users,

even though such networks aim to provide greater privacy than standard Internet protocols (e.g., TLS from Chapter 4). Looking into the underlying causes and the evolution of the ecosystem over the years, we attribute the shortcomings primarily to the vendors' market competition in the early stages of the ecosystem that led to hastly-designed and poorly-secured implementations. As an immediate remediation, we introduced two user-applicable countermeasures and argued for finer-grained permissions and a commonly-accepted specification for ultrasonic communications. Recent changes in Android's permission system and the introduction of the Nearby Messages API helped on that front by 1) preventing apps from abusing user trust (e.g., monitoring the ultrasonic spectrum on the background, and 2) enforcing good practices.

In Chapter 6, we further expanded on how modern machine learning models enhance the adversarial capabilities and showed how they can be used in a commercial ARM core. Our experiments show that adversaries making use of convolutional neural networks can attack implementations of AES LUT with high-accuracy, thus reducing significantly the number of the key candidates. Such techniques provide a straightforward and low-cost way for manufacturers and auditors to quickly gauge the security of a given product. It is thus beneficial for the end customers as it reduces the need for expensive manual experiments and allows vendors to: 1) identify certain security hazards early on, and 2) better fine-tune their protection mechanisms.

Overall, based on the previous chapters and prior work, we argue that the detection-evasion arms race in the context of leakages is significantly skewed against the defender. Adversaries have access to very capable models that evolve rapidly, while designers and operators have to go through expensive and cumbersome processes to evaluate the security properties of a given system at a given time. Chapter 7 introduces an alternative that is orthogonal to existing approaches for the prevention and the detection of leakages. The proposed design assumes that some of the system's components may be vulnerable and attempts to reduce the trust placed on each component individually. This requires some replication of components which results in an (reasonable) additional cost but significantly enhances the resilience of

the system to leakages and attacks overall. While prevention and detection measures remain necessary, trust-distribution designs can serve as a final layer of protection against leakages in cases where everything else fails.

A recurring theme in all our chapters is the need of system designers and operators to reliably validate the security properties of their systems and products. Currently, most systems and protocols come with a manually-derived proof or a formal argument for their security. This practice is far from ideal as there have been several cases where designs were found to be less secure than initially thought or even completely flawed. Learning-based techniques could provide an alternative for the estimation of the lower security bound of a system or protocol. For example, as seen in Chapters 4 and 6, machine learning has significantly enhanced the capabilities of adversaries allowing for sophisticated attacks at a low cost. Based on these observations, we believe that learning-based models should be further studied for their potential to serve as *security approximators*. A promising direction could involve reinforcement learning techniques that (provably) converge to the optimal adversarial strategy, thus approximating the worst-case adversary for a given system. Such a tool could allow for quick iterations on product designs, validation of the correctness of analytical security proofs, and security audits of deployments and system implementations.

Bibliography

[1] David Kahn. The codebreakers-the story of secret writing. *New York, NY: Signet*, 1973.

[2] Wes Freeman, Geoff Sullivan, and Frode Weierud. Purple revealed: Simulation and computer-aided cryptanalysis of angooki taipu b. *Cryptologia*, 27(1):1–43, 2003.

[3] Dakshi Agrawal, Bruce Archambeault, Josyula R Rao, and Pankaj Rohatgi. The em side—channel (s). In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 29–45. Springer, 2002.

[4] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In *International workshop on cryptographic hardware and embedded systems*, pages 16–29. Springer, 2004.

[5] François-Xavier Standaert, Tal G Malkin, and Moti Yung. A unified framework for the analysis of side-channel key recovery attacks. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 443–461. Springer, 2009.

[6] Silvio Micali and Leonid Reyzin. Physically observable cryptography. In *Theory of Cryptography Conference*, pages 278–296. Springer, 2004.

[7] Paul Kocher, Joshua Jaffe, Benjamin Jun, and Pankaj Rohatgi. Introduction to differential power analysis. *Journal of Cryptographic Engineering*, 1(1):5–27, 2011.

[8] Markus Guenther Kuhn. *Compromising emanations: eavesdropping risks of computer displays*. PhD thesis, Citeseer, 2002.

[9] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. Technical report, DTIC Document, 2004.

[10] David Goldschlag, Michael Reed, and Paul Syverson. Onion routing. *Communications of the ACM*, 42(2):39–41, 1999.

[11] Michael G Reed, Paul F Syverson, and David M Goldschlag. Anonymous connections and onion routing. *IEEE Journal on Selected areas in Communications*, 16(4):482–494, 1998.

[12] Christos Andrikos, Lejla Batina, Lukasz Chmielewski, Liran Lerman, Vasilios Mavroudis, Kostas Papagiannopoulos, Guilherme Perin, Giorgos Rassias, and Alberto Sonnino. Location, location, location: Revisiting modeling and exploitation for location-based side channel leakages. In *Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part III*, volume 11923 of *Lecture Notes in Computer Science*, pages 285–314. Springer, 2019.

[13] Vasilios Mavroudis, Andrea Cerulli, Petr Svenda, Dan Cvrcek, Dusan Klinec, and George Danezis. A Touch of Evil: High-Assurance Cryptographic Hardware from Untrusted Components. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA*, pages 1583–1600, 2017.

[14] Vasilios Mavroudis and Jamie Hayes. Adaptive Traffic Fingerprinting: Large-scale Inference under Realistic Assumptions. 2020.

[15] Vasilios Mavroudis, Shuang Hao, Yanick Fratantonio, Federico Maggi, Christopher Kruegel, and Giovanni Vigna. On the Privacy and Security of the Ultrasound Ecosystem. *Proceedings on Privacy Enhancing Technologies*, 2017(2):95–112, 2017.

[16] Tao Wang, Xiang Cai, Rishab Nithyanand, Rob Johnson, and Ian Goldberg. Effective attacks and provable defenses for website fingerprinting. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 143–157, 2014.

[17] Andriy Panchenko, Fabian Lanze, Jan Pennekamp, Thomas Engel, Andreas Zinnen, Martin Henze, and Klaus Wehrle. Website fingerprinting at internet scale. In *23rd Annual Network and Distributed System Security Symposium, NDSS 2016, San Diego, California, USA, February 21-24, 2016*. The Internet Society, 2016.

[18] Payap Sirinam, Mohsen Imani, Marc Juarez, and Matthew Wright. Deep fingerprinting: Undermining website fingerprinting defenses with deep learning. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 1928–1943. ACM, 2018.

[19] Jamie Hayes and George Danezis. k-fingerprinting: A robust scalable website fingerprinting technique. In *25th USENIX Security Symposium*, pages 1187–1203, 2016.

[20] Sanjit Bhat, David Lu, Albert Kwon, and Srinivas Devadas. Var-cnn: A data-efficient website fingerprinting attack based on deep learning. *Proceedings on Privacy Enhancing Technologies*, 2019(4):292–310, 2019.

[21] Shailen Mistry and Bhaskaran Raman. Quantifying Traffic Analysis of Encrypted Web-Browsing. In *Project paper, University of Berkeley*, 1998.

[22] George Dean Bissias, Marc Liberatore, David Jensen, and Brian Neil Levine. Privacy vulnerabilities in encrypted http streams. In *International Workshop on Privacy Enhancing Technologies*, pages 1–11. Springer, 2005.

[23] Xiang Cai, Xin Cheng Zhang, Brijesh Joshi, and Rob Johnson. Touching from a distance: Website fingerprinting attacks and defenses. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 605–616. ACM, 2012.

[24] Vasilios Mavroudis. Market manipulation as a security problem: Attacks and defenses. In *Proceedings of the 12th European Workshop on Systems Security*, pages 1–6, 2019.

[25] Vasilios Mavroudis and Hayden Melton. Libra: Fair Order-Matching for Electronic Financial Exchanges. In *Proceedings of the 1st ACM Conference on Advances in Financial Technologies, AFT 2019, Zurich, Switzerland*, pages 156–168. ACM, 2019.

[26] Vasilios Mavroudis. Bounded Temporal Fairness for FIFO Financial Markets. In *Proceedings of the 26th International Workshop on Security Protocols*. Springer, 2019.

[27] Markus K Brunnermeier. Information leakage and market efficiency. *The Review of Financial Studies*, 18(2):417–457, 2005.

[28] Alexander Hicks, Vasilios Mavroudis, Mustafa Al-Bassam, Sarah Meiklejohn, and Steven J. Murdoch. VAMS: Verifiable Auditing of Access to Confidential Data. *CoRR*, abs/1805.04772, 2018.

[29] Vasilios Mavroudis, Karl Wüst, Aritra Dhar, Kari Kostiainen, and Srdjan Capkun. Snappy: Fast On-chain Payments with Practical Collaterals. In *27th Annual Network and Distributed System Security Symposium, NDSS 2020, San Diego, California, USA*. The Internet Society, 2020.

[30] Vasilios Mavroudis and Petr Svenda. JCMathLib:Wrapper Cryptographic Library for Transparent and Certifiable JavaCard Applets. In *Proceedings of the 1st International Workshop on lightweight and Incremental Cybersecurity Certification, CyberCert 2020, all-digital*. IEEE, 2020.

[31] Vasilios Mavroudis and Michael Veale. Eavesdropping whilst you're shopping: Balancing personalisation and privacy in connected retail spaces. In *Living in the Internet of Things: Cybersecurity of the IoT*, pages 1–10. IET, 2018.

[32] MITRE. Cwe-200: Exposure of sensitive information to an unauthorized actor. https://cwe.mitre.org/data/definitions/200.html, 2020.

[33] Steven Euijong Whang and Hector Garcia-Molina. A model for quantifying information leakage. In *Workshop on Secure Data Management*, pages 25–44. Springer, 2012.

[34] Chugui Xu, Ju Ren, Deyu Zhang, Yaoxue Zhang, Zhan Qin, and Kui Ren. Ganobfuscator: Mitigating information leakage under gan via differential privacy. *IEEE Transactions on Information Forensics and Security*, 14(9):2358–2371, 2019.

[35] Monodeep Kar, Arvind Singh, Sanu K Mathew, Anand Rajan, Vivek De, and Saibal Mukhopadhyay. Reducing power side-channel information leakage of aes engines using fully integrated inductive voltage regulator. *IEEE Journal of Solid-State Circuits*, 53(8):2399–2414, 2018.

[36] Lukasz Olejnik, Minh-Dung Tran, and Claude Castelluccia. Selling off user privacy at auction. In *21st Annual Network and Distributed System Security Symposium, NDSS 2014, San Diego, California, USA, February 23-26, 2014*. The Internet Society, 2014.

[37] Yong Yuan, Feiyue Wang, Juanjuan Li, and Rui Qin. A survey on real time bidding advertising. In *Service Operations and Logistics, and Informatics (SOLI), 2014 IEEE International Conference on*, pages 418–423. IEEE, 2014.

[38] Terence Chen, Imdad Ullah, Mohamed Ali Kaafar, and Roksana Boreli. Information leakage through mobile analytics services. In *Proceedings of the 15th Workshop on Mobile Computing Systems and Applications*, page 15. ACM, 2014.

[39] Claude Castelluccia, Mohamed-Ali Kaafar, and Minh-Dung Tran. Betrayed by your ads! In *Privacy Enhancing Technologies Symposium*, pages 1–17. Springer, 2012.

[40] Signal360. http://www.signal360.com/, 2021.

[41] Audible magic. https://www.audiblemagic.com/, 2021.

[42] Copsonic. http://www.copsonic.com/, 2021.

[43] Tchirp. http://www.tchirp.com/, 2021.

[44] Lisnr. http://lisnr.com/platform, 2021.

[45] Simon Haykin. *Communication systems*. John Wiley & Sons, 2008.

[46] Simon Haykin. *Digital Communication systems*. John Wiley & Sons, 2013.

[47] Daniel Arp, Erwin Quiring, Christian Wressnegger, and Konrad Rieck. Privacy threats through ultrasonic side channels on mobile devices. In *2017 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 35–47. IEEE, 2017.

[48] Adrienne Porter Felt, Richard Barnes, April King, Chris Palmer, Chris Bentzel, and Parisa Tabriz. Measuring HTTPS adoption on the web. In *26th USENIX Security Symposium*, pages 1323–1338, 2017.

[49] Ghada Arfaoui, Xavier Bultel, Pierre-Alain Fouque, Adina Nedelcu, and Cristina Onete. The privacy of the tls 1.3 protocol. *Proceedings on Privacy Enhancing Technologies*, 2019(4):190–210, 2019.

[50] Eric Rescorla. The transport layer security (tls) protocol version 1.3. 2018.

[51] Benjamin Dowling, Marc Fischlin, Felix Günther, and Douglas Stebila. A cryptographic analysis of the tls 1.3 draft-10 full and pre-shared key handshake protocol. *IACR Cryptology ePrint Archive*, 2016:81, 2016.

[52] Eric Rescorla. The transport layer security (tls) protocol version 1.2. 2008.

[53] George Danezis. Traffic analysis of the http protocol over tls, 2009.

[54] Brad Miller, Ling Huang, Anthony D Joseph, and J Doug Tygar. I know why you went to the clinic: Risks and realization of https traffic analysis. In *International Symposium on Privacy Enhancing Technologies Symposium*, pages 143–163. Springer, 2014.

[55] James MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability. Vol. 1. No. 14.*, 1967.

[56] Christopher M Bishop. *Pattern recognition and machine learning*. Springer, 2006.

[57] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise.

[58] Xifeng Guo, Long Gao, Xinwang Liu, and Jianping Yin. Improved deep embedded clustering with local structure preservation. In Carles Sierra, editor, *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 1753–1759. ijcai.org, 2017.

[59] Xifeng Guo, Xinwang Liu, En Zhu, and Jianping Yin. Deep clustering with convolutional autoencoders. In *International Conference on Neural Information Processing*, pages 373–382. Springer, 2017.

[60] Fei Tian, Bin Gao, Qing Cui, Enhong Chen, and Tie-Yan Liu. Learning deep representations for graph clustering. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[61] Xi Peng, Shijie Xiao, Jiashi Feng, Wei-Yun Yau, and Zhang Yi. Deep subspace clustering with sparsity prior. In Subbarao Kambhampati, editor, *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 1925–1931. IJCAI/AAAI Press, 2016.

[62] Bo Yang, Xiao Fu, Nicholas D Sidiropoulos, and Mingyi Hong. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3861–3870. JMLR. org, 2017.

[63] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *International conference on machine learning*, pages 478–487, 2016.

[64] Elyor Kodirov, Tao Xiang, and Shaogang Gong. Semantic autoencoder for zero-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3174–3183, 2017.

[65] Md Zahangir Alom and Tarek M Taha. Network intrusion detection for cyber security using unsupervised deep learning approaches. In *2017 IEEE National Aerospace and Electronics Conference (NAECON)*, pages 63–69. IEEE, 2017.

[66] Stephan Mandt, Matthew D Hoffman, and David M Blei. Stochastic gradient descent as approximate bayesian inference. *The Journal of Machine Learning Research*, 18(1):4873–4907, 2017.

[67] Léon Bottou. Stochastic gradient descent tricks. In *Neural networks: Tricks of the trade*, pages 421–436. Springer, 2012.

[68] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[69] Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, Gang Wang, Jianfei Cai, et al. Recent advances in convolutional neural networks. *Pattern Recognition*, 77:354–377, 2018.

[70] David Snyder, Daniel Garcia-Romero, Daniel Povey, and Sanjeev Khudanpur. Deep neural network embeddings for text-independent speaker verification. In *Interspeech*, pages 999–1003, 2017.

[71] Vladimir A Golovko, Leanid U Vaitsekhovich, Pavel A Kochurko, and Uladzimir S Rubanau. Dimensionality reduction and attack recognition using neural network approaches. In *2007 International Joint Conference on Neural Networks*, pages 2734–2739. IEEE, 2007.

[72] R. P. W. Duin. Classifiers in almost empty spaces. In *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*, volume 2, pages 1–7 vol.2, 2000.

[73] Laurens Van Der Maaten, Eric Postma, and Jaap Van den Herik. Dimensionality reduction: a comparative. *J Mach Learn Res*, 10(66-71):13, 2009.

[74] Tiago A Almeida, Jurandy Almeida, and Akebo Yamakami. Spam filtering: how the dimensionality reduction affects the accuracy of naive bayes classifiers. *Journal of Internet Services and Applications*, 1(3):183–200, 2011.

[75] Arjun Nitin Bhagoji, Daniel Cullina, and Prateek Mittal. Dimensionality reduction as a defense against evasion attacks on machine learning classifiers. *arXiv preprint arXiv:1704.02654*, 2, 2017.

[76] Martin Husák, Milan Čermák, Tomáš Jirsík, and Pavel Čeleda. Https traffic analysis and client identification using passive ssl/tls fingerprinting. *EURASIP Journal on Information Security*, 2016(1):6, 2016.

[77] Martin Husák, Milan Cermák, Tomá Jirsík, and Pavel Celeda. Network-based https client identification using ssl/tls fingerprinting. In *2015 10th International Conference on Availability, Reliability and Security*, pages 389–396. IEEE, 2015.

[78] Maciej Korczyński and Andrzej Duda. Markov chain fingerprinting to classify encrypted traffic. In *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*, pages 781–789. IEEE, 2014.

[79] J. Muehlstein, Y. Zion, M. Bahumi, I. Kirshenboim, R. Dubin, A. Dvir, and O. Pele. Analyzing https encrypted traffic to identify user's operating

system, browser and application. In *2017 14th IEEE Annual Consumer Communications Networking Conference (CCNC)*, pages 1–6, 2017.

[80] Blake Anderson and David McGrew. Tls beyond the browser: Combining end host and network data to understand application behavior. In *Proceedings of the Internet Measurement Conference*, pages 379–392, 2019.

[81] Heyning Cheng and Ron Avnur. Traffic analysis of ssl encrypted web browsing. *URL citeseer. ist. psu. edu/656522. html*, 1998.

[82] Qixiang Sun, Daniel R Simon, Yi-Min Wang, Wilf Russell, Venkata N Padmanabhan, and Lili Qiu. Statistical identification of encrypted web browsing traffic. In *Proceedings 2002 IEEE Symposium on Security and Privacy*, pages 19–30. IEEE, 2002.

[83] Ran Dubin, Amit Dvir, Ofir Pele, and Ofer Hadar. I know what you saw last minute—encrypted http adaptive video streaming title classification. *IEEE transactions on information forensics and security*, 12(12):3039–3049, 2017.

[84] Payap Sirinam, Nate Mathews, Mohammad Saidur Rahman, and Matthew Wright. Triplet fingerprinting: More practical and portable website fingerprinting with n-shot learning. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 1131–1148. ACM, 2019.

[85] Honda Electronics. Ultrasonic Aquatic Communication System. https://archive.is/3Ab5N, 2015-12-23T06:30:49Z.

[86] Mostafa Kamal. minimodem - general-purpose software audio FSK modem. http://www.whence.com/minimodem/, December 2011.

[87] Ultrasound Networking. https://www.anfractuosity.com/projects/ultrasound-networking/, May 2013.

[88] Michael Hanspach and Michael Goetz. On Covert Acoustical Mesh Networks in Air. 8(11):758–767, 2013.

[89] Venkatachalam Subramanian, Selcuk Uluagac, Hasan Cam, and Raheem Beyah. Examining the characteristics and implications of sensor side channels. In *Communications (ICC), 2013 IEEE International Conference on*, pages 2205–2210. IEEE, 2013.

[90] Mordechai Guri, Yosef Solewicz, and Yuval Elovici. Speaker-to-speaker covert ultrasonic communication. *Journal of Information Security and Applications*, 51:102458, 2020.

[91] Sebastian Zimmeck, Jie S Li, Hyungtae Kim, Steven M Bellovin, and Tony Jebara. A privacy analysis of cross-device tracking. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 1391–1408, 2017.

[92] Mordechai Guri, Yosef Solewicz, and Yuval Elovici. Mosquito: Covert ultrasonic transmissions between two air-gapped computers using speaker-to-speaker communication. In *2018 IEEE Conference on Dependable and Secure Computing (DSC)*, pages 1–8. IEEE, 2018.

[93] Nikolay Matyunin, Jakub Szefer, and Stefan Katzenbeisser. Zero-permission acoustic cross-device tracking. In *2018 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 25–32. IEEE, 2018.

[94] Luke Deshotels. Inaudible sound as a covert channel in mobile devices. In *Proc. 8th USENIX Conf. Offensive Technologies*, page 16, 2015.

[95] Giuseppe Petracca, Yuqiong Sun, Trent Jaeger, and Ahmad Atamli. AuDroid: Preventing Attacks on Audio Channels in Mobile Devices. In *Annual Computer Security Applications Conference*. ACM Press, 2015-12-10T18:43:21Z.

[96] Intrasonics-Artificial Echo Modulation. http://www.intrasonics.com/, 2016.

[97] Takeshi Sugawara, Daisuke Suzuki, Minoru Saeki, Mitsuru Shiozaki, and Takeshi Fujino. On measurable side-channel leaks inside ASIC design primitives. *J. Cryptographic Engineering*, 4(1):59–73, 2014.

[98] Christos Andrikos, Giorgos Rassias, Liran Lerman, Kostas Papagiannopoulos, and Lejla Batina. Location-based leakages: New directions in modeling and exploiting. In *2017 International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS)*, pages 246–252, July 2017.

[99] Johann Heyszl, Stefan Mangard, Benedikt Heinz, Frederic Stumpf, and Georg Sigl. Localized electromagnetic analysis of cryptographic implementations. In *Topics in Cryptology - CT-RSA 2012 - The Cryptographers' Track at the RSA Conference 2012, San Francisco, CA, USA, February 27 - March 2, 2012. Proceedings*, pages 231–244, 2012.

[100] Alexander Schlösser, Dmitry Nedospasov, Juliane Krämer, Susanna Orlic, and Jean-Pierre Seifert. *Simple Photonic Emission Analysis of AES*, pages 41–57. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[101] M. Nassar, Y. Souissi, S. Guilley, and J. L. Danger. Rsm: A small and fast countermeasure for aes, secure against 1st and 2nd-order zero-offset scas. In *2012 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1173–1178, March 2012.

[102] Florian Unterstein, Johann Heyszl, Fabrizio De Santis, and Robert Specht. Dissecting leakage resilient prfs with multivariate localized EM attacks - A practical security evaluation on FPGA. *COSADE*, 2017:272, 2017.

[103] Vincent Immler, Robert Specht, and Florian Unterstein. Your rails cannot hide from localized EM: how dual-rail logic fails on fpgas. In *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, pages 403–424, 2017.

[104] Robert Specht, Johann Heyszl, and Georg Sigl. Investigating measurement methods for high-resolution electromagnetic field side-channel analysis. In

2014 International Symposium on Integrated Circuits (ISIC), Singapore, December 10-12, 2014, pages 21–24, 2014.

[105] Robert Specht, Johann Heyszl, Martin Kleinsteuber, and Georg Sigl. Improving non-profiled attacks on exponentiations based on clustering and extracting leakage from multi-channel high-resolution EM measurements. In *Constructive Side-Channel Analysis and Secure Design - 6th International Workshop, COSADE 2015, Berlin, Germany, April 13-14, 2015. Revised Selected Papers*, pages 3–19, 2015.

[106] R. Specht, V. Immler, F. Unterstein, J. Heyszl, and G. Sig. Dividing the threshold: Multi-probe localized em analysis on threshold implementations. In *2018 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 33–40, April 2018.

[107] Kouichi Itoh, Tetsuya Izu, and Masahiko Takenaka. Address-bit differential power analysis of cryptographic schemes ok-ecdh and ok-ecdsa. In *Revised Papers from the 4th International Workshop on Cryptographic Hardware and Embedded Systems*, CHES '02, pages 129–143, London, UK, UK, 2003. Springer-Verlag.

[108] Raghavan Kumar, Philipp Jovanovic, Wayne P Burleson, and Ilia Polian. Parametric trojans for fault-injection attacks on cryptographic hardware. *IACR Cryptology ePrint Archive*, 2014:783, 2014.

[109] Julia Borghoff, Anne Canteaut, Tim Güneysu, Elif Bilge Kavun, Miroslav Knezevic, Lars R Knudsen, Gregor Leander, Ventzislav Nikov, Christof Paar, Christian Rechberger, et al. Prince–a low-latency block cipher for pervasive computing applications. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 208–225. Springer, 2012.

[110] Georg T Becker, Francesco Regazzoni, Christof Paar, and Wayne P Burleson. Stealthy dopant-level hardware trojans. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 197–214. Springer, 2013.

[111] Andrea Pellegrini, Valeria Bertacco, and Todd Austin. Fault-based attack of rsa authentication. In *Proceedings of the conference on Design, automation and test in Europe*, pages 855–860. European Design and Automation Association, 2010.

[112] Kaiyuan Yang, Matthew Hicks, Qing Dong, Todd Austin, and Dennis Sylvester. A2: Analog malicious hardware, 2016.

[113] Xinmu Wang, Tatini Mal-Sarkar, Aswin Raghav Krishna, Seetharam Narasimhan, and Swarup Bhunia. Software exploitable hardware trojans in embedded processor. In *2012 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems, DFT 2012, Austin, TX, USA, October 3-5, 2012*, pages 55–58, 2012.

[114] Xinmu Wang. *Hardware trojan attacks: Threat analysis and low-cost countermeasures through golden-free detection and secure design.* PhD thesis, Case Western Reserve University, 2014.

[115] Xinmu Wang, Seetharam Narasimhan, Aswin Krishna, Tatini Mal-Sarkar, and Swarup Bhunia. Sequential hardware trojan: Side-channel aware design and placement. In *Computer Design (ICCD), 2011 IEEE 29th International Conference on*, pages 297–300. IEEE, 2011.

[116] Sebastian Kutzner, Axel York Poschmann, and Marc Stöttinger. Hardware trojan design and detection: a practical evaluation. In *Proceedings of the Workshop on Embedded Systems Security, WESS 2013, Montreal, Quebec, Canada, September 29 - October 4, 2013*, pages 1:1–1:9, 2013.

[117] Swarup Bhunia, Michael S. Hsiao, Mainak Banga, and Seetharam Narasimhan. Hardware trojan attacks: Threat analysis and countermeasures. *Proceedings of the IEEE*, 102(8):1229–1247, 2014.

[118] Sergei Skorobogatov and Christopher Woods. Breakthrough silicon scanning discovers backdoor in military chip. In *Cryptographic Hardware and Embedded Systems - CHES 2012 - 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings*, pages 23–40, 2012.

[119] Subhasish Mitra, H-S Philip Wong, and Simon Wong. The trojan-proof chip. *IEEE Spectrum*, 52(2):46–51, 2015.

[120] Yier Jin and Yiorgos Makris. Hardware trojans in wireless cryptographic ics. *IEEE Design & Test of Computers*, 27(1), 2010.

[121] Sean Gallagher. Photos of an nsa "upgrade" factory show cisco router getting implant. *Ars Technica*, 14, 2014.

[122] S Skorobogatov. Hardware assurance and its importance to national security. https://www.cl.cam.ac.uk/~sps32/sec_news.html, 2012.

[123] Sally Adee. The hunt for the kill switch. *IEEE Spectrum*, 45(5):34–39, 2008.

[124] John Markoff. Old trick threatens the newest weapons. *The New York Times*, 27, 2009.

[125] Thomas Shrimpton and R Seth Terashima. A provable-security analysis of intel's secure key rng. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 77–100. Springer, 2015.

[126] FreeBSD Security Working Group. Freebsd developer summit: Security working group. https://wiki.freebsd.org/201309DevSummit/Security, 2013.

[127] RT. 'we cannot trust them anymore': Engineers abandon encryption chips after snowden leaks. https://www.rt.com/usa/snowden-leak-rng-randomness-019/, 2013.

[128] Dan Goodin. 'we cannot trust" intel and via's chip-based crypto', freebsd developers say. http://arstechnica.com/security/2013/12/we-cannot-trust-intel-and-vias-chip-based-crypto-freebsd-developers-say/, 2013.

[129] Bruce Schneier. Surreptitiously tampering with computer chips. https://www.schneier.com/blog/archives/2013/09/surreptitiously.html, 2013.

[130] Daniel J Bernstein, Yun-An Chang, Chen-Mou Cheng, Li-Ping Chou, Nadia Heninger, Tanja Lange, and Nicko Van Someren. Factoring rsa keys from certified smart cards: Coppersmith in the wild. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 341–360. Springer, 2013.

[131] Jan Jancar, Vladimir Sedlacek, Petr Svenda, and Marek Sys. Minerva: The curse of ecdsa nonces systematic analysis of lattice attacks on noisy leakage of bit-length of ecdsa nonces. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 281–308, 2020.

[132] Sheng Wei and Miodrag Potkonjak. Self-consistency and consistency-based detection and diagnosis of malicious circuitry. *IEEE Trans. VLSI Syst.*, 22(9):1845–1853, 2014.

[133] S. Wei and M. Potkonjak. Scalable hardware trojan diagnosis. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 20(6):1049–1057, 2012.

[134] Oliver Soll, Thomas Korak, Michael Muehlberghuber, and Michael Hutter. Em-based detection of hardware trojans on fpgas. In *Hardware-Oriented Security and Trust (HOST), 2014 IEEE International Symposium on*, pages 84–87. IEEE, 2014.

[135] Dakshi Agrawal, Selcuk Baktir, Deniz Karakoyunlu, Pankaj Rohatgi, and Berk Sunar. Trojan detection using ic fingerprinting. In *Security and Privacy, 2007. SP'07. IEEE Symposium on*, pages 296–310. IEEE, 2007.

[136] Rajat Subhra Chakraborty, Francis G. Wolff, Somnath Paul, Christos A. Papachristou, and Swarup Bhunia. MERO: A statistical approach for hardware trojan detection. In *Cryptographic Hardware and Embedded Systems - CHES*

2009, 11th International Workshop, Lausanne, Switzerland, September 6-9, 2009, Proceedings, pages 396–410, 2009.

[137] Jeyavijayan JV Rajendran and Siddharth Garg. Logic encryption. In *Hardware Protection through Obfuscation*, pages 71–88. Springer, 2017.

[138] Chongxi Bao, Yang Xie, and Ankur Srivastava. A security-aware design scheme for better hardware trojan detection sensitivity. In *IEEE International Symposium on Hardware Oriented Security and Trust, HOST 2015, Washington, DC, USA, 5-7 May, 2015*, pages 52–55, 2015.

[139] Adam Waksman, Matthew Suozzo, and Simha Sethumadhavan. FANCI: identification of stealthy malicious logic using boolean functional analysis. In *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013*, pages 697–708, 2013.

[140] Jie Zhang, Feng Yuan, Lingxiao Wei, Yannan Liu, and Qiang Xu. Veritrust: Verification for hardware trust. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 34(7):1148–1161, 2015.

[141] Zhang Chen, Pingqiang Zhou, T. Y. Ho, and Y. Jin. How secure is split manufacturing in preventing hardware trojan? In *2016 IEEE Asian Hardware-Oriented Security and Trust (AsianHOST)*, pages 1–6, 2016.

[142] Yujie Wang, Pu Chen, Jiang Hu, and Jeyavijayan Rajendran. The cat and mouse in split manufacturing. In *Proceedings of the 53rd Annual Design Automation Conference, DAC 2016, Austin, TX, USA, June 5-9, 2016*, pages 165:1–165:6, 2016.

[143] Jeyavijayan JV Rajendran, Ozgur Sinanoglu, and Ramesh Karri. Is split manufacturing secure? In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 1259–1264. EDA Consortium, 2013.

[144] Rajat Subhra Chakraborty and Swarup Bhunia. Security against hardware trojan through a novel application of design obfuscation. In *Proceedings of the*

2009 International Conference on Computer-Aided Design, pages 113–116. ACM, 2009.

[145] Matthew Hicks, Murph Finnicum, Samuel T. King, Milo M. K. Martin, and Jonathan M. Smith. Overcoming an untrusted computing base: Detecting and removing malicious hardware automatically. In *31st IEEE Symposium on Security and Privacy, S&P 2010, Oakland, California, USA*, pages 159–172, 2010.

[146] Adam Waksman and Simha Sethumadhavan. Tamper evident microprocessors. In *Security and Privacy (SP), 2010 IEEE Symposium on*, pages 173–188. IEEE, 2010.

[147] Riad S. Wahby, Max Howald, Siddharth Garg, abhi shelat, and Michael Walfish. Verifiable asics. In *IEEE Security and Privacy (Oakland) 2016, eprint/2016/1243*, 2016.

[148] Stefan Dziembowski, Sebastian Faust, and François-Xavier Standaert. Private circuits iii: Hardware trojan-resilience via testing amplification. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 142–153. ACM, 2016.

[149] Giuseppe Ateniese, Aggelos Kiayias, Bernardo Magri, Yiannis Tselekounis, and Daniele Venturi. Secure outsourcing of circuit manufacturing. Cryptology ePrint Archive, Report 2016/527, 2016. http://eprint.iacr.org/2016/527.

[150] Liming Chen and Algirdas Avizienis. N-version programming: A fault-tolerance approach to reliability of software operation. In *Digest of Papers FTCS-8: Eighth Annual International Conference on Fault Tolerant Computing*, pages 3–9, 1978.

[151] Ying C Yeh. Triple-triple redundant 777 primary flight computer. In *Aerospace Applications Conference, 1996. Proceedings., 1996 IEEE*, volume 1, pages 293–307. IEEE, 1996.

[152] Ying C Yeh. Design considerations in boeing 777 fly-by-wire computers. In *High-Assurance Systems Engineering Symposium, 1998. Proceedings. Third IEEE International*, pages 64–72. IEEE, 1998.

[153] Benjamin Cox and David Evans. N-variant systems: A secretless framework for security through diversity. In *Proceedings of the 15th USENIX Security Symposium, Vancouver, BC, Canada, July 31 - August 4, 2006*, 2006.

[154] Yousra Alkabani and Farinaz Koushanfar. N-variant IC design: methodology and applications. In *Proceedings of the 45th Design Automation Conference, DAC 2008, Anaheim, CA, USA, June 8-13, 2008*, pages 546–551, 2008.

[155] Mark Beaumont, Bradley Hopkins, and Tristan Newby. Safer path: Security architecture using fragmented execution and replication for protection against trojaned hardware. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 1000–1005. EDA Consortium, 2012.

[156] Mark Beaumont, Bradley Hopkins, and Tristan Newby. Hardware trojan resistant computation using heterogeneous cots processors. In *Proceedings of the Thirty-Sixth Australasian Computer Science Conference-Volume 135*, pages 97–106. Australian Computer Society, Inc., 2013.

[157] Fred B Schneider. Implementing fault-tolerant services using the state machine approach: A tutorial. *ACM Computing Surveys (CSUR)*, 22(4):299–319, 1990.

[158] Fred B Schneider. The state machine approach: A tutorial. *Fault-tolerant distributed computing*, pages 18–41, 1990.

[159] Alysson Bessani, Joao Sousa, and Eduardo EP Alchieri. State machine replication for the masses with bft-smart. In *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pages 355–362. IEEE, 2014.

[160] Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. In *Concurrency: the Works of Leslie Lamport*, pages 179–196. 2019.

[161] Miguel Castro, Barbara Liskov, et al. Practical byzantine fault tolerance. In *OSDI*, volume 99, pages 173–186, 1999.

[162] Marshall Pease, Robert Shostak, and Leslie Lamport. Reaching agreement in the presence of faults. *Journal of the ACM (JACM)*, 27(2):228–234, 1980.

[163] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. In *Concurrency: the Works of Leslie Lamport*, pages 203–226. 2019.

[164] Karolos Antoniadis, Rachid Guerraoui, Dahlia Malkhi, and Dragos-Adrian Seredinschi. State machine replication is more expensive than consensus. Technical report, 2018.

[165] Cynthia Dwork, Nancy Lynch, and Larry Stockmeyer. Consensus in the presence of partial synchrony. *Journal of the ACM (JACM)*, 35(2):288–323, 1988.

[166] Gabriel Bracha and Sam Toueg. Asynchronous consensus and broadcast protocols. *Journal of the ACM (JACM)*, 32(4):824–840, 1985.

[167] Michael J Fischer, Nancy A Lynch, and Michael S Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM (JACM)*, 32(2):374–382, 1985.

[168] Elizabeth Borowsky and Eli Gafni. Generalized flp impossibility result for t-resilient asynchronous computations. In *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pages 91–100, 1993.

[169] NetCraft. August 2020 web server survey. https://news.netcraft.com/archives/category/web-server-survey/, 2020.

[170] Cisco and affiliates. Global - 2020 Forecast Highlights. https://www.cisco.com/c/dam/m/en_us/solutions/service-provider/vni-forecast-highlights/pdf/Global_2020_Forecast_Highlights.pdf, 2016.

[171] Cisco and affiliates. Global - 2022 Forecast Highlights. https://www.cisco.com/c/dam/m/en_us/solutions/service-provider/vni-forecast-highlights/pdf/Global_2022_Forecast_Highlights.pdf, 2018.

[172] Marc Juarez, Sadia Afroz, Gunes Acar, Claudia Diaz, and Rachel Greenstadt. A critical evaluation of website fingerprinting attacks. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 263–274. ACM, 2014.

[173] Tao Wang and Ian Goldberg. On realistically attacking tor with website fingerprinting. *Proceedings on Privacy Enhancing Technologies*, 2016(4):21–36, 2016.

[174] WolfSSL. A Comparison of Differences in TLS 1.1 and TLS 1.2. https://www.wolfssl.com/a-comparison-of-differences-in-tls-1-1-and-tls-1-2/, 2015.

[175] Nick Sullivan. A Detailed Look at RFC 8446 (a.k.a. TLS 1.3). https://blog.cloudflare.com/rfc-8446-aka-tls-1-3/, 2018.

[176] Cloudflare. What is Transport Layer Security (TLS)? https://www.cloudflare.com/learning/ssl/transport-layer-security-tls/, 2018.

[177] Babur Khan. TLS 1.3 - Status, Concerns & Impact. https://www.a10networks.com/blog/tls-13-status-concerns-impact/, 2019.

[178] Christopher Wood Joseph Salowey, Sean Turner. IETF News: TLS 1.3. https://ietf.org/blog/tls13/, 2018.

[179] Nick Naziridis. TLS 1.3 is here to stay. https://www.ssl.com/article/tls-1-3-is-here-to-stay/, 2018.

[180] Dimitrios Schoinianakis, Norbert Goetze, and Gerald Lehmann. Mdiet: Malware detection in encrypted traffic. In *6th International Symposium for ICS & SCADA Cyber Security Research 2019 6*, pages 31–37, 2019.

[181] Blake Harrell Anderson, David McGrew, Subharthi Paul, Ivan Nikolaev, and Martin Grill. Malware classification and attribution through server fingerprinting using server certificate data, May 17 2018. US Patent App. 15/353,160.

[182] Blake Anderson, Subharthi Paul, and David McGrew. Deciphering malware's use of tls (without decryption). *Journal of Computer Virology and Hacking Techniques*, 14(3):195–211, 2018.

[183] Nick Pantic and Mohammad I Husain. Covert botnet command and control using twitter. In *Proceedings of the 31st annual computer security applications conference*, pages 171–180. ACM, 2015.

[184] Yue Li, Lidong Zhai, Zhilei Wang, and Yunlong Ren. Control method of twitter-and sms-based mobile botnet. In *International Conference on Trustworthy Computing and Services*, pages 644–650. Springer, 2012.

[185] Pieter Burghouwt, Marcel Spruit, and Henk Sips. Detection of covert botnet command and control channels by causal analysis of traffic flows. In *Cyberspace Safety and Security*, pages 117–131. Springer, 2013.

[186] Pieter Burghouwt, Marcel Spruit, and Henk Sips. Towards detection of botnet communication through social media by monitoring user activity. In *International Conference on Information Systems Security*, pages 131–143. Springer, 2011.

[187] Chao-Yuan Wu, R Manmatha, Alexander J Smola, and Philipp Krahenbuhl. Sampling matters in deep embedding learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2840–2848, 2017.

[188] Ben Harwood, BG Kumar, Gustavo Carneiro, Ian Reid, Tom Drummond, et al. Smart mining for deep metric learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2821–2829, 2017.

[189] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[190] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742. IEEE, 2006.

[191] Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 539–546. IEEE, 2005.

[192] Qingzhao Tan, Ziming Zhuang, Prasenjit Mitra, and C Lee Giles. Efficiently detecting webpage updates using samples. In *International Conference on Web Engineering*, pages 285–300. Springer, 2007.

[193] Marc Spaniol, Arturas Mazeika, Dimitar Denev, and Gerhard Weikum. "catch me if you can": Visual analysis of coherence defects in web archiving. In *The 9 th International Web Archiving Workshop (IWAW 2009) Corfu, Greece, September/October, 2009 Workshop Proceedings*, page 1, 2009.

[194] Konstantinos Solomos, John Kristoff, Chris Kanich, and Jason Polakis. Persistent tracking in modern browsers. In *28th Annual Network and Distributed System Security Symposium, NDSS 2021*. The Internet Society, 2021.

[195] Felix Fuentes and Dulal C Kar. Ethereal vs. tcpdump: a comparative study on packet sniffing tools for educational purpose. *Journal of Computing Sciences in Colleges*, 20(4):169–176, 2005.

[196] Antonio Gulli and Sujit Pal. *Deep learning with Keras*. Packt Publishing Ltd, 2017.

[197] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283, Savannah, GA, November 2016. USENIX Association.

[198] Travis E Oliphant. *A guide to NumPy*, volume 1. Trelgol Publishing USA, 2006.

[199] Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature methods*, 17(3):261–272, 2020.

[200] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(Feb):281–305, 2012.

[201] James S Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Advances in neural information processing systems*, pages 2546–2554, 2011.

[202] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[203] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models.

[204] Alfredo Pironti and Nikos Mavrogiannopoulos. Length hiding padding for the transport layer security protocol. *tech. rep., Internet-Draft draft-pironti-tls-length-hiding-00, IETF Secretariat*, 2013.

[205] Alfredo Pironti, Pierre-Yves Strub, and Karthikeyan Bhargavan. Identifying website users by tls traffic analysis: New attacks and effective countermeasures. 2012.

[206] Global Web Index. Digital consumers own 3.2 connected devices. https://blog.globalwebindex.com/chart-of-the-day/digital-consumers-own-3-point-2-connected-devices/, November 2017.

[207] Sophos. Users weighed down by multiple gadgets and mobile devices, new sophos survey reveals. https://www.sophos.com/en-us/press-office/press-releases/2013/03/mobile-security-survey.aspx, March 2013.

[208] Google Cast 1.16.7 app on the Play Store. https://play.google.com/store/apps/details?id=com.google.android.apps.chromecast.app, 2016.

[209] Shopkick. https://www.shopkick.com/, June 2016.

[210] Silverpush. https://www.silverpush.co/, 2015.

[211] Google Nearby Messages API. https://developers.google.com/nearby/messages/android/get-beacon-messages, 2016.

[212] Google Proximity Beacon API. https://developers.google.com/beacons/proximity/guides, 2016.

[213] Bluetooth Low Energy API Level 18. https://developer.android.com/guide/topics/connectivity/bluetooth-le.html, 2016.

[214] Cueaudio. https://cueaudio.com/data-over-sound/, 2021.

[215] Google. Chromecast guest mode - guest mode faqs. https://support.google.com/chromecast/answer/6109297?hl=en, August 2016.

[216] Indianapolis Colts Mobile 3.1.1 app on the Play Store. https://play.google.com/store/apps/details?id=com.yinzcam.nfl.colts, 2016.

[217] Alex Silverman. Colts to begin using lisnr technology to reach fans' mobile devices at games, events. http://www.sportsbusinessdaily.com/Daily/Issues/2016/07/19/Franchises/Colts.aspx, July 2016.

[218] Made in America Festival 1.0.8 app on the Play Store. https://play.google.com/store/apps/details?id=com.lisnr.festival.madeinamericaandroid, 2015.

[219] C. Roeding and A.T. Emigh. Method and system for location-triggered rewards. https://www.google.com/patents/US8489112, 2013. US Patent 8,489,112.

[220] Dan Goodin. Beware of ads that use inaudible sound to link your phone, TV, tablet, and PC. http://arstechnica.com/tech-policy/2015/11/beware-of-ads-that-use-inaudible-sound-to-link-your-phone-tv-tablet-and-pc/, 2015.

[221] Chris Calabrese. Comments for November 2015 Workshop on Cross-Device Tracking. https://cdt.org/files/2015/10/10.16.15-CDT-Cross-Device-Comments.pdf, 2015-12-09T18:54:04Z.

[222] History GK 5.0 app on the Play Store. https://play.google.com/store/apps/details?id=com.gktalk.history, 2016.

[223] Philipp Winter, Richard Köwer, Martin Mulazzani, Markus Huber, Sebastian Schrittwieser, Stefan Lindskog, and Edgar Weippl. Spoiled Onions: Exposing Malicious Tor Exit Relays. In *Privacy Enhancing Technologies Symposium*. Springer, 2014.

[224] Moxie Marlinspike. New tricks for defeating ssl in practice. In *Black Hat USA 2009*, 2009.

[225] Danny Palmer. Android security: Six more apps containing Joker malware removed from the Google Play Store. https://www.zdnet.com/article/android-security-six-more-apps-containing-joker-malware-removed-from-the-google-play-store/, 2020.

[226] Andy Greenberg. How Spies Snuck Malware Into the Google Play Store—Again and Again. https://www.wired.com/story/phantomlance-google-play-malware-apt32/, 2020.

[227] Kopo M Ramokapane, Anthony C Mazeli, and Awais Rashid. Skip, skip, skip, accept!!!: A study on the usability of smartphone manufacturer provided default features and user privacy. *Proceedings on Privacy Enhancing Technologies*, 2019(2):209–227, 2019.

[228] Alessandro Andreadis and Giovanni Giambene. The global system for mobile communications. *Protocols for High-Efficiency Wireless Networks*, pages 17–44, 2002.

[229] 3GPP. 3rd generation partnership project, technical specification of international mobile station equipment identities (imei). http://www.3gpp.org/DynaReport/22016.htm, 2015.

[230] Guolei Yang, Neil Zhenqiang Gong, and Ying Cai. Fake co-visitation injection attacks to recommender systems. In *NDSS*, 2017.

[231] Joseph A Calandrino, Ann Kilzer, Arvind Narayanan, Edward W Felten, and Vitaly Shmatikov. " you might also like:" privacy risks of collaborative filtering. In *2011 IEEE symposium on security and privacy*, pages 231–246. IEEE, 2011.

[232] Xiaoyuan Su and Taghi M Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in artificial intelligence*, 2009:4, 2009.

[233] Jun Yan, Ning Liu, Gang Wang, Wen Zhang, Yun Jiang, and Zheng Chen. How much can behavioral targeting help online advertising? In *Proceedings of the 18th international conference on World wide web*, pages 261–270. ACM, 2009.

[234] Weinan Zhang, Lingxi Chen, and Jun Wang. Implicit look-alike modelling in display ads: Transfer collaborative filtering to ctr estimation. *arXiv preprint arXiv:1601.02377*, 2016.

[235] Arik Friedman, Bart P Knijnenburg, Kris Vanhecke, Luc Martens, and Shlomo Berkovsky. Privacy aspects of recommender systems. In *Recommender Systems Handbook*, pages 649–688. Springer, 2015.

[236] Rui Chen, Min Xie, and Laks VS Lakshmanan. Thwarting passive privacy attacks in collaborative filtering. In *International Conference on Database Systems for Advanced Applications*, pages 218–233. Springer, 2014.

[237] Wei Meng, Xinyu Xing, Anmol Sheth, Udi Weinsberg, and Wenke Lee. Your online interests: Pwned! a pollution attack against targeted advertising. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 129–140. ACM, 2014.

[238] Xinyu Xing, Wei Meng, Dan Doozan, Alex C Snoeren, Nick Feamster, and Wenke Lee. Take this personally: Pollution attacks on personalized services. In *Proceedings of the 22nd USENIX Security Symposium*, 2013.

[239] Real-time bidding and malvertising: A case study. https://blog.malwarebytes.org/cybercrime/2015/04/real-time-bidding-and-malvertising-a-case-study/, April 2015.

[240] Aditya K Sood and Richard J Enbody. Targeted cyberattacks: a superset of advanced persistent threats. *IEEE security & privacy*, (1):54–61, 2013.

[241] Addons Detector. Silverpush android apps. https://public.addonsdetector.com/silverpush-android-apps/, November 2015.

[242] McDo Philippines app on the Play Store. https://play.google.com/store/apps/details?id=ph.mobext.mcdelivery, 2016.

[243] Ftc public discussion on cross-device tracking. https://www.ftc.gov/news-events/audio-video/video/cross-device-tracking-part-1, November 2015.

[244] Edith Ramirez. Transcript - Part 1. In *FTC Cross-Device Tracking Workshop*, 2015.

[245] Edith Ramirez. Transcript - Part 2. In *FTC Cross-Device Tracking Workshop*, 2015.

[246] Android Help. Change app permissions on your Android phone. https://support.google.com/android/answer/9431959?hl=en-GB, 2020.

[247] City Frequencies. PilferShush. https://www.cityfreqs.com.au/pilfer.php, 2021.

[248] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, pages 388–397, 1999.

[249] Karine Gandolfi, Christophe Mourtel, and Francis Olivier. Electromagnetic analysis: Concrete results. In Çetin K. Koç, David Naccache, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems — CHES 2001*, pages 251–261, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[250] Julien Doget, Emmanuel Prouff, Matthieu Rivain, and François-Xavier Standaert. Univariate side channel attacks and leakage modeling. *J. Cryptographic Engineering*, 1(2):123–144, 2011.

[251] François-Xavier Standaert, Tal Malkin, and Moti Yung. A unified framework for the analysis of side-channel key recovery attacks. In *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings*, pages 443–461, 2009.

[252] Matthieu Rivain and Emmanuel Prouff. Provably secure higher-order masking of aes. In Stefan Mangard and François-Xavier Standaert, editors, *Cryptographic Hardware and Embedded Systems, CHES 2010*, pages 413–427, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[253] Svetla Nikova, Christian Rechberger, and Vincent Rijmen. Threshold implementations against side-channel attacks and glitches. In *Information and Communications Security, 8th International Conference, ICICS 2006, Raleigh, NC, USA, December 4-7, 2006, Proceedings*, pages 529–545, 2006.

[254] Tetsuya Ichikawa, Tomomi Kasuya, and Mitsuru Matsui. Hardware evaluation of the aes finalists. In *AES Candidate Conference*, volume 2000, pages 279–285. Citeseer, 2000.

[255] Emilia Käsper and Peter Schwabe. Faster and timing-attack resistant aes-gcm. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 1–17. Springer, 2009.

[256] Takeshi Sugawara, Daisuke Suzuki, Minoru Saeki, Mitsuru Shiozaki, and Takeshi Fujino. On measurable side-channel leaks inside asic design primitives. In *International Conference on Cryptographic Hardware and Embedded Systems*, pages 159–178. Springer, 2013.

[257] Assia Tria and Hamid Choukri. Invasive attacks. In Henk C. A. van Tilborg and Sushil Jajodia, editors, *Encyclopedia of Cryptography and Security*, pages 623–629, Boston, MA, 2011. Springer US.

[258] Jochen Hoenicke. Extracting the private key from a TREZOR. https://jochen-hoenicke.de/crypto/trezor-power-analysis/, 2015.

[259] Tim Güneysu and Tobias Oder. Towards lightweight identity-based encryption for the post-quantum-secure internet of things. In *2017 18th International Symposium on Quality Electronic Design (ISQED)*, pages 319–324, 2017.

[260] Hwajeong Seo and Reza Azarderakhsh. Curve448 on 32-bit arm cortex-m4. In *International Conference on Information Security and Cryptology*, pages 125–139. Springer, 2020.

[261] Niels Roelofs, Niels Samwel, Lejla Batina, and Joan Daemen. Online template attack on ecdsa:. In Abderrahmane Nitaj and Amr Youssef, editors, *Progress*

in Cryptology - AFRICACRYPT 2020, pages 323–336, Cham, 2020. Springer International Publishing.

[262] Marios O Choudary and Markus G Kuhn. Efficient, portable template attacks. *IEEE Transactions on Information Forensics and Security*, 13(2):490–501, 2017.

[263] Aymeric Genêt, Natacha Linard de Guertechin, and Novak Kaluderović. Full key recovery side-channel attack against ephemeral sike on the cortex-m4. In *Constructive Side-Channel Analysis and Secure Design*, pages 228–254, Cham, 2021. Springer International Publishing.

[264] Melissa Azouaoui, Kostas Papagiannopoulos, and Dominik Zürner. Blind side-channel sifa. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 555–560. IEEE, 2021.

[265] Thomas Schamberger, Julian Renner, Georg Sigl, and Antonia Wachter-Zeh. A power side-channel attack on the cca2-secure hqc kem. In *International Conference on Smart Card Research and Advanced Applications*, pages 119–134. Springer, 2020.

[266] Philip Sperl and Konstantin Böttinger. Side-channel aware fuzzing. In *European Symposium on Research in Computer Security*, pages 259–278. Springer, 2019.

[267] Hwajeong Seo, Mila Anastasova, Amir Jalali, and Reza Azarderakhsh. Supersingular isogeny key encapsulation (sike) round 2 on arm cortex-m4. *IEEE Transactions on Computers*, 2020.

[268] Reza Azarderakhsh, Matthew Campagna, Craig Costello, LD Feo, Basil Hess, Amir Jalali, David Jao, Brian Koziel, Brian LaMacchia, Patrick Longa, et al. Supersingular isogeny key encapsulation. *Submission to the NIST Post-Quantum Standardization project*, 152:154–155, 2017.

[269] Matthias J. Kannwischer, Joost Rijneveld, Peter Schwabe, and Ko Stoffelen. PQM4: Post-quantum crypto library for the ARM Cortex-M4. https://github.com/mupq/pqm4.

[270] Ellen Gunnarsdóttir, Lejla Batina, and Lukasz Chmielewski. Trezor one side-channel analysis setup. 2020.

[271] Lejla Batina, Lukasz Chmielewski, Björn Haase, Niels Samwel, and Peter Schwabe. Sca-secure ECC in software - mission impossible? *IACR Cryptol. ePrint Arch.*, page 1003, 2021.

[272] Neil Weste and David Harris. *CMOS VLSI Design: A Circuits and Systems Perspective*. Addison-Wesley Publishing Company, USA, 4th edition, 2010.

[273] Liran Lerman, Romain Poussier, Olivier Markowitch, and François-Xavier Standaert. Template attacks versus machine learning revisited and the curse of dimensionality in side-channel analysis: extended version. *Journal of Cryptographic Engineering*, 8(4):301–313, Nov 2018.

[274] Eleonora Cagli, Cécile Dumas, and Emmanuel Prouff. Convolutional neural networks with data augmentation against jitter-based countermeasures. In *International Conference on Cryptographic Hardware and Embedded Systems*, pages 45–68. Springer, 2017.

[275] Zdenek Martinasek and Vaclav Zeman. Innovative method of the power analysis. *Radioengineering*, 22(2):586–594, 2013.

[276] Zdenek Martinasek, Jan Hajny, and Lukas Malina. Optimization of power analysis using neural network. In *International Conference on Smart Card Research and Advanced Applications*, pages 94–107. Springer, 2013.

[277] Houssem Maghrebi, Thibault Portigliatti, and Emmanuel Prouff. Breaking cryptographic implementations using deep learning techniques. In *International Conference on Security, Privacy, and Applied Cryptography Engineering*, pages 3–26. Springer, 2016.

[278] Shuguo Yang, Yongbin Zhou, Jiye Liu, and Danyang Chen. Back propagation neural network based leakage characterization for practical security analysis of cryptographic implementations. In *International Conference on Information Security and Cryptology*, pages 169–185. Springer, 2011.

[279] Emmanuel Prouff, Remi Strullu, Ryad Benadjila, Eleonora Cagli, and Cécile Dumas. Study of deep learning techniques for side-channel analysis and introduction to ASCAD database. *IACR Cryptology ePrint Archive*, 2018:53, 2018.

[280] Shaine A Morris and Timothy C Slesnick. Magnetic resonance imaging. *Visual Guide to Neonatal Cardiology*, pages 104–108, 2018.

[281] Sinno Jialin Pan, Qiang Yang, et al. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

[282] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[283] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[284] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[285] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, volume 4, page 12, 2017.

[286] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[287] United States. Defense Science Board. *Defense science board task force on high performance microchip supply*. Office of the Under Secretary of Defense for Acquisition, Technology and Logistics, 2005.

[288] Inez Miyamoto, Thomas H Holzer, and Shahryar Sarkani. Why a counterfeit risk avoidance strategy fails. *Computers & Security*, 2017.

[289] Samuel T. King, Joseph Tucek, Anthony Cozzie, Chris Grier, Weihang Jiang, and Yuanyuan Zhou. Designing and implementing malicious hardware. In *First USENIX Workshop on Large-Scale Exploits and Emergent Threats, LEET '08, San Francisco, CA, USA, April 15, 2008, Proceedings*, 2008.

[290] Don Edenfeld, Andrew B. Kahng, Mike Rodgers, and Yervant Zorian. 2003 technology roadmap for semiconductors. *IEEE Computer*, 37(1):47–56, 2004.

[291] Stefan Heck, Sri Kaza, and Dickon Pinner. Creating value in the semiconductor industry. *McKinsey & Company*, 2011.

[292] Age Yeh. Trends in the global IC design service market. *DIGITIMES research*, 2012.

[293] Bastian Fredriksson. A case study in smartcard security analysing mifare classic rev. 2016.

[294] Nicolas T Courtois. The dark side of security by obscurity and cloning mifare classic rail and building passes, anywhere, anytime. 2009.

[295] Paul Kocher, Jann Horn, Anders Fogh, Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, et al. Spectre attacks: Exploiting speculative execution. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 1–19. IEEE, 2019.

[296] Moritz Lipp, Michael Schwarz, Daniel Gruss, Thomas Prescher, Werner Haas, Anders Fogh, Jann Horn, Stefan Mangard, Paul Kocher, Daniel Genkin, Yuval Yarom, and Mike Hamburg. Meltdown: Reading Kernel Memory from User

Space. In William Enck and Adrienne Porter Felt, editors, *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018*, pages 973–990. USENIX Association, 2018.

[297] Claudio Canella, Jo Van Bulck, Michael Schwarz, Moritz Lipp, Benjamin Von Berg, Philipp Ortner, Frank Piessens, Dmitry Evtyushkin, and Daniel Gruss. A systematic evaluation of transient execution attacks and defenses. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 249–266, 2019.

[298] Intel Inc. About the intel manageability firmware critical vulnerability. http://www.intel.com/content/www/us/en/architecture-and-technology/intel-amt-vulnerability-announcement.html, 2017.

[299] Guoxing Chen, Sanchuan Chen, Yuan Xiao, Yinqian Zhang, Zhiqiang Lin, and Ten H Lai. SgxPectre: Stealing Intel secrets from SGX enclaves via speculative execution. In *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 142–157. IEEE, 2019.

[300] Jo Van Bulck, Marina Minkin, Ofir Weisse, Daniel Genkin, Baris Kasikci, Frank Piessens, Mark Silberstein, Thomas F Wenisch, Yuval Yarom, and Raoul Strackx. Foreshadow: Extracting the keys to the Intel SGX kingdom with transient out-of-order execution. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 991–1008, 2018.

[301] Stephan van Schaik, Andrew Kwong, Daniel Genkin, and Yuval Yarom. SGAxe: How sgx fails in practice, 2020.

[302] Mohammad Tehranipoor and Cliff Wang. *Introduction to hardware security and trust*. Springer Science & Business Media, 2011.

[303] Miodrag Potkonjak, Ani Nahapetian, Michael Nelson, and Tammara Massey. Hardware trojan horse detection using gate-level characterization. In *Proceedings of the 46th Design Automation Conference, DAC 2009, San Francisco, CA, USA, July 26-31, 2009*, pages 688–693, 2009.

[304] Georg T. Becker, Francesco Regazzoni, Christof Paar, and Wayne P. Burleson. Stealthy dopant-level hardware trojans: extended version. *J. Cryptographic Engineering*, 4(1):19–31, 2014.

[305] Adam Waksman and Simha Sethumadhavan. Silencing hardware backdoors. In *32nd IEEE Symposium on Security and Privacy, S&P 2011, Oakland, California, USA*, pages 49–63. IEEE, 2011.

[306] Swarup Bhunia, Michael S Hsiao, Mainak Banga, and Seetharam Narasimhan. Hardware trojan attacks: threat analysis and countermeasures. *Proceedings of the IEEE*, 102(8):1229–1247, 2014.

[307] Sheng Wei and Miodrag Potkonjak. The undetectable and unprovable hardware trojan horse. In *Proceedings of the 50th Annual Design Automation Conference*, page 144. ACM, 2013.

[308] Petr Svenda. Resilience via practical interoperability of multiparty schnorr signature schemes. In *43rd IEEE Symposium on Security and Privacy, S&P 2022, Oakland, California, USA*. IEEE, 2022.

[309] Jacob Appelbaum, Judith Horchert, and Christian Stöcker. Shopping for spy gear: Catalog advertises nsa toolbox. *Der Spiegel*, 29, 2013.

[310] Daniel Genkin, Adi Shamir, and Eran Tromer. Rsa key extraction via low-bandwidth acoustic cryptanalysis. In *International Cryptology Conference*, pages 444–461. Springer, 2014.

[311] Michael Backes, Markus Dürmuth, and Dominique Unruh. Compromising reflections-or-how to read lcd monitors around the corner. In *Security and Privacy, 2008. SP 2008. IEEE Symposium on*, pages 158–169. IEEE, 2008.

[312] Lorenzo Strigini. Fault tolerance against design faults. 2005.

[313] Frank Imeson, Ariq Emtenan, Siddharth Garg, and Mahesh V Tripunitara. Securing computer hardware using 3d integrated circuit (ic) technology and split manufacturing for obfuscation.

[314] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on information theory*, 31(4):469–472, 1985.

[315] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Conference on the Theory and Application of Cryptographic Techniques*, pages 186–194. Springer, 1986.

[316] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings*, pages 129–140, 1991.

[317] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Secure distributed key generation for discrete-log based cryptosystems. *J. Cryptology*, 20(1):51–83, 2007.

[318] Douglas R. Stinson and Reto Strobl. Provably secure distributed schnorr signatures and a (t, n) threshold scheme for implicit certificates. In *Information Security and Privacy, 6th Australasian Conference, ACISP 2001, Sydney, Australia, July 11-13, 2001, Proceedings*, pages 417–434, 2001.

[319] Thomas Ristenpart and Scott Yilek. The power of proofs-of-possession: Securing multiparty signatures against rogue-key attacks. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 228–245. Springer, 2007.

[320] Felix Brandt. Efficient cryptographic protocol design based on distributed el gamal encryption. In *Information Security and Cryptology - ICISC 2005, 8th International Conference, Seoul, Korea, December 1-2, 2005, Revised Selected Papers*, pages 32–47, 2005.

[321] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 103–112. ACM, 1988.

[322] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Keccak sponge function family main document. *Submission to NIST (Round 2)*, 3:30, 2009.

[323] Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of cryptology*, 4(3):161–174, 1991.

[324] Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, Alexandria, VA, USA, Ioctober 30 - November 3, 2006*, pages 390–399. ACM, 2006.

[325] Kazuo Ohta and Tatsuaki Okamoto. A digital multisignature scheme based on the fiat-shamir scheme. In *Advances in Cryptology - ASIACRYPT '91*, pages 139–148, 1991.

[326] Markus Michels and Patrick Horster. On the risk of disruption in several multiparty signature schemes. In *Advances in Cryptology - ASIACRYPT '96, International Conference on the Theory and Applications of Cryptology and Information Security, Kyongju, Korea, November 3-7, 1996, Proceedings*, pages 334–345, 1996.

[327] Silvio Micali, Kazuo Ohta, and Leonid Reyzin. Accountable-subgroup multisignatures: extended abstract. In *CCS 2001, Proceedings of the 8th ACM Conference on Computer and Communications Security*, pages 245–254, 2001.

[328] Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme. In *Public Key Cryptography - PKC 2003*, pages 31–46, 2003.

[329] Steve Lu, Rafail Ostrovsky, Amit Sahai, Hovav Shacham, and Brent Waters. Sequential aggregate signatures, multisignatures, and verifiably encrypted

signatures without random oracles. *Journal of cryptology*, 26(2):340–373, 2013.

[330] George Robert Blakley. Safeguarding cryptographic keys. *Proc. of the National Computer Conference1979*, 48:313–317, 1979.

[331] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[332] Wolfgang Rankl and Wolfgang Effing. *Smart card handbook*. John Wiley & Sons, 2004.

[333] Sean Turner, Russ Housley, Tim Polk, Daniel RL Brown, and Kelvin Yiu. Elliptic curve cryptography subject public key information. 2009.

[334] Mehmet Adalier. Efficient and secure elliptic curve cryptography implementation of curve p-256. 2015.

[335] Petr Svenda. Nuances of the javacard api on the cryptographic smart cards–jcalgtest project. 2014.

[336] Rod Johnson et al. Introduction to the spring framework. *TheServerSide. com*, 21:22, 2005.

[337] George Danezis, Claudia Diaz, and Paul Syverson. Systems for anonymous communication. *Handbook of Financial Cryptography and Security, Cryptography and Network Security Series*, pages 341–389, 2009.

[338] Hua Hong Semiconductor. Hua hong semiconductor limited. http://www.huahonggrace.com/html/about.php, 2017.

[339] Semiconductor Manufacturing International Corporation. Embedded non-volatile memory for smart card & mcu. http://www.smics.com/eng/foundry/technology/tec_envm.php, 2017.

[340] Taiwan Semiconductor Manufacturing Company Limited TSMC. Value chain aggregator - km211. http://www.tsmc.com/english/dedicatedFoundry/services/value_chain_aggregator_km211.htm, 2017.

[341] StarChip. Smart card ics. http://www.starchip-ic.com/en/smart-card-chips/, 2017.

[342] City Frequencies. EnigmaBridge HSM. https://enigmabridge.com/hsm.html, 2021.

[343] City Frequencies. EnigmaBridge MPC. https://enigmabridge.com/mpc.html, 2021.

[344] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Decentralized Business Review*, page 21260, 2008.

[345] Pieter Wuille, Jonas Nick, and Anthony Towns. Taproot: SegWit Version 1 Spending Rules. https://github.com/bitcoin/bips/blob/master/bip-0341.mediawiki, 2020.

[346] Pieter Wuille, Jonas Nick, and Anthony Towns. Validation of Taproot scripts. https://github.com/bitcoin/bips/blob/master/bip-0342.mediawiki, 2020.

[347] G. He, M. Yang, X. Gu, J. Luo, and Y. Ma. A novel active website fingerprinting attack against tor anonymous system. In *Proceedings of the 2014 IEEE 18th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pages 112–117, 2014.

[348] M. Yang, X. Gu, Z. Ling, C. Yin, and J. Luo. An active de-anonymizing attack against tor web traffic. *Tsinghua Science and Technology*, 22(6):702–713, 2017.