Systematic Classification of Side-Channel Attacks: A Case Study for Mobile Devices

Raphael Spreitzer, Veelasha Moonsamy, Thomas Korak, and Stefan Mangard

***Abstract*—Side-channel attacks on mobile devices have gained increasing attention since their introduction in 2007. While traditional side-channel attacks, such as power analysis attacks and electromagnetic analysis attacks, required physical presence of the attacker as well as expensive equipment, an (unprivileged) application is all it takes to exploit the leaking information on modern mobile devices. Given the vast amount of sensitive information that are stored on smartphones, the ramifications of side-channel attacks affect both the security and privacy of users and their devices.**

In this paper, we propose a new categorization system for side-channel attacks, which is necessary as side-channel attacks have evolved significantly since their scientific investigations during the smart card era in the 1990s. Our proposed classification system allows to analyze side-channel attacks systematically, and facilitates the development of novel countermeasures. Besides this new categorization system, the extensive survey of existing attacks and attack strategies provides valuable insights into the evolving field of side-channel attacks, especially when focusing on mobile devices. We conclude by discussing open issues and challenges in this context and outline possible future research directions.

***Index Terms*—Side-channel attacks, information leakage, classification, smartphones, mobile devices, survey, Android.**

I. Introduction

SIDE-channel attacks exploit (unintended) information leakage of computing devices or implementations to infer sensitive information. Starting with the seminal works of Kocher [1], Kocher et al. [2], Quisquater and Samyde [3], as well as Mangard et al. [4], many follow-up papers considered attacks against cryptographic implementations to exfiltrate key material from smart cards by means of timing information, power consumption, or electromagnetic (EM) emanation. These "traditional" side-channel attacks required the attacker to be in physical possession of the device to be able to observe and learn the leaking information, yet different attacks assumed different types of attackers and different levels of invasiveness. More specifically, in order to systematically analyze side-channel attacks, they have been categorized along the following two orthogonal axes:

1) *Active* vs *passive*: Depending on whether the attacker actively influences the behavior of the device or only passively observes leaking information.
2) *Invasive* vs *semi-invasive* vs *non-invasive*: Depending on whether or not the attacker removes the passivation layer of the chip, depackages the chip, or does not manipulate the packaging at all.

However, with the era of cloud computing, the scope and the scale of side-channel attacks have changed significantly in the early 2000s. While early attacks required attackers to be in physical possession of the device, newer side-channel attacks such as cache-timing attacks [5]–[7] or DRAM row buffer attacks [8] are conducted remotely by executing malicious software in the targeted cloud environment. With the advent of mobile devices, and in particular the plethora of embedded features and sensors, even more sophisticated side-channel attacks targeting smartphones have been proposed since around the year 2010. For example, attacks allow to infer keyboard input on touchscreens via sensor readings from native apps [9]–[11] and websites [12], to deduce a user's location via the power consumption available from the proc filesystem (procfs) [13], and also to infer a user's identity, location, and diseases [14] via the procfs.

Clearly, side-channel attacks have a long history and have evolved significantly from attacks on specialized computing devices in the smart card era, to attacks on general-purpose computing platforms in desktop computers and cloud computing infrastructures, and finally to attacks on mobile devices. Although side-channel attacks and platform security are already well-studied topics, it must be noted that smartphone security and associated privacy aspects differ from platform security in the context of smart cards, desktop computers, and cloud computing. Especially the following *key enablers* enable more devastating attacks on mobile devices.

1) *Always-on and portability*: First and foremost, mobile devices are always turned on and due to their mobility they are carried around at all times. Thus, they are tightly integrated into our everyday lives.
2) *Bring your own device (BYOD)*: To decrease the number of devices carried around, employees use personal devices to process corporate data and to access corporate infrastructure, which clearly indicates the importance of secure mobile devices.
3) *Ease of software installation*: Due to the appification [15] of mobile devices, *i.e.*, where there is an app for almost everything, additional software can be installed easily by means of established app markets. Hence, malicious apps can also be spread at a fast pace.
4) *OS based on Linux kernel*: Modern mobile operating systems (OS), for example, Android, are based on the Linux kernel. The Linux kernel, however, has initially been designed for desktop machines and information or

R. Spreitzer, and S. Mangard are with Graz University of Technology, IAIK, Graz, Austria. (e-mail: raphael.spreitzer@iaik.tugraz.at, stefan.mangard@iaik.tugraz.at).
T. Korak was with Graz University of Technology, IAIK, Graz, Austria.
V. Moonsamy is with Radboud University, Digital Security Group, Nijmegen, The Netherlands. (e-mail: email@veelasha.org).

features that are considered harmless on these platforms turn out to be an immense security and/or privacy threat on mobile devices (cf. [16]).

5) *Features and sensors*: Last but not least, these devices include many features and sensors, which are not present on traditional platforms. Due to the inherent nature of mobile devices (always-on and carried around, connectivity, inherent input methods, etc.), such features often enable devastating side-channel attacks. Besides, these sensors have also been used to attack external hardware, such as keyboards and computer hard drives [17]–[19], to infer videos played on TVs [20], and even to attack 3D printers [21], [22], which clearly demonstrates the immense power of mobile devices.

Due to the above mentioned *key enablers*, a new area of side-channel attacks has evolved and the majority of more recent side-channel attacks are strictly non-invasive and rely on the execution of malicious software in the targeted environment. Considering these developments, we observe that the classification system that has been established to analyze side-channel attacks on smart cards does not meet these new attack settings and strategies anymore. Hence, the existing classification system does not allow a systematic categorization of modern side-channel attacks, including side-channel attacks on mobile devices.

In this work, we close this gap by establishing a new categorization system for modern side-channel attacks on mobile devices. Therefore, we survey existing side-channel attacks and identify commonalities between them. The gained insights allow researchers to identify future research directions and to cope with these attacks on a larger scale.

A. Motivation and High-Level Categorization

It is important to note that side-channel attacks against smartphones can be launched by attackers who are in physical possession of the devices and also by remote attackers who managed to spread a seemingly innocuous application via an existing app store. In some cases such side-channel attacks can even be launched via websites and, thus, without relying on the user to install an app. Nevertheless, in today's appified software platforms where apps are distributed easily via available app markets, an attack scenario requiring the user to install a seemingly harmless game is entirely practical.

Interestingly, side-channel attacks on smartphones exploit physical properties as well as software properties. A malicious application can exploit the accelerometer sensor [9], [10] (a physical property) in order to attack the user input, which is due to the inherent input method relying on touchscreens. In addition, attacks can also be conducted by exploiting software features (a logical property) provided by the Android API or the mobile OS itself (cf. [13], [14]). This clearly indicates that smartphones significantly broaden the scope as well as the scale of attacks. Especially the appification [15] of mobile platforms—*i.e.*, where there is an app for everything—allows to easily target devices and users at an unprecedented scale compared to the smart card and the cloud setting.

Figure 1 illustrates a high-level categorization system for side-channel attacks. We indicate the type of information that is exploited (WHAT?) and how the adversary learns the leaking information (HOW?) on the y-axis and x-axis, respectively. Furthermore, we indicate how *existing* side-channel attacks against smart cards, cloud computing infrastructures, and smartphones relate to it, *i.e.*, where existing attacks on the respective platforms are located in this new categorization system. For example, attackers exploit hardware-based information leakage (physical properties) [4] of smart cards by measuring the power consumption with an oscilloscope. In this case, the attacker must be in possession of the device under attack, which is indicated by the red cross-hatched area.

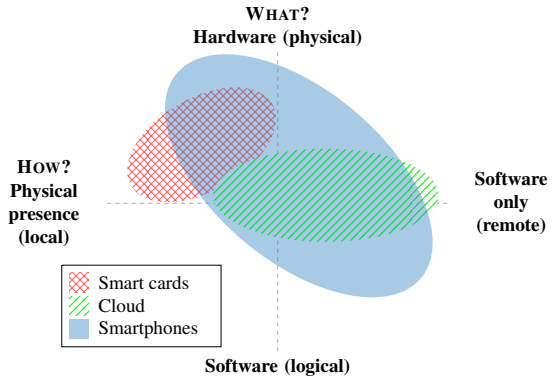


Fig. 1. Scope of attacks for smart cards, cloud infrastructures, and smartphones.

In contrast, side-channel attacks against cloud-computing infrastructures do not (necessarily) require the attacker to be physically present—unless we consider a malicious cloud provider—as the attacker is able to remotely execute software. Usually, these attacks exploit microarchitectural behavior (such as cache attacks [5]–[7], [23]) or software features (such as page deduplication [24]) in order to infer secret information from co-located processes. Hence, the green dashed area in Figure 1 is shifted to the right as these attacks mostly rely on software execution, and it is also shifted to the area below the x-axis as these attacks also target software features.

Even more manifold and diverse side-channel attacks have been proposed for smartphones, which is indicated by the larger area in Figure 1. These manifold side-channel attacks mainly result from the five aforementioned *key enablers*. More specifically, this area indicates that on smartphones we have to deal with local attackers that exploit physical properties, but also with attackers that execute software on the smartphone in order to exploit both physical properties as well as software features (logical properties, such as the memory footprint [25] or the data-usage statistics [14], [26]). In the remainder of this paper we will refine this high-level categorization system in order to systematically analyze modern side-channel attacks.

Although we do not explicitly focus on Android in this paper, the majority of the existing papers deal with the Android operating system. This reflects the trend that the research community focuses mostly on Android because of its openness and also because it has the biggest market share among all mobile operating systems. Gartner [27] reports that Android sales (86% in Q1 2017) clearly outperform Apple iOS sales (14% in Q1 2017).

B. Outline

The remainder of this paper is organized as follows. Section II introduces background information in terms of mobile operating systems, the basic notion of side-channel attacks, and related work. In Section III, we discuss different types of information leaks and provide a definition for software-only side-channel attacks. Furthermore, we introduce our new categorization system for modern side-channel attacks. We survey existing attacks in Sections IV, V, and VI, and we classify existing attacks according to our newly introduced classification system in Section VII. We discuss existing countermeasures in Section VIII. Finally, we discuss open issues, challenges, and future research directions in Section IX and conclude in Section X.

II. Background

In this section, we introduce the basics of mobile security, define the general notion of side-channel attacks, and we establish the boundaries between side-channel attacks and other attacks on mobile devices. We stress that side-channel attacks do not exploit specific software vulnerabilities of the OS or any specific library, but instead exploit available information that either leaks unintentionally or that is (in some cases) published for benign reasons in order to infer sensitive information indirectly. Finally, we also discuss related work.

A. A Primer on Smartphone Security

Mobile devices, such as tablet computers and smartphones, are powerful multi-purpose computing platforms that enable many different application scenarios. Third-party applications can be easily installed in order to extend the basic functionality of these devices. Examples include gaming applications that make use of the many different sensors, office applications, banking applications, and many more. These examples clearly demonstrate that mobile devices are already tightly integrated into our everyday lives, which leads to sensitive data and information being stored and processed on these devices.

In order to protect this information properly, modern mobile operating systems rely on two fundamental security concepts, *i.e.*, the concept of application sandboxing and the concept of permission systems. For instance, on Android the underlying Linux kernel ensures the concept of sandboxed applications. Each application is assigned a user ID (UID), which allows the kernel to prevent applications from accessing resources of other applications. The permission system on the other hand allows applications to request access to specific resources outside of its sandbox, which typically includes resources that are considered as being sensitive or privacy relevant. Android also categorizes permissions depending on so-called protection levels. The two important categories of Android permissions are *normal* permissions and *dangerous* permissions, respectively. While *normal* permissions are granted automatically during the installation procedure, *dangerous* permissions must be explicitly granted by the user. Other mobile operating systems such as Apple's iOS rely on similar protection mechanisms.

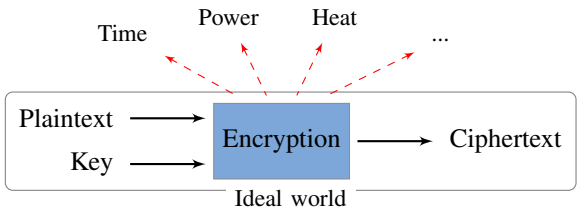


Fig. 2. An implementation produces unintended output as a byproduct.

Besides these basic security concepts on the OS level, applications themselves rely on cryptographic primitives, cryptographic protocols, and dedicated security mechanisms to protect sensitive resources. For instance, applications rely on encryption primitives to protect sensitive information being stored on the device or when transmitting data over the Internet. Another example of a dedicated security mechanism is a personal identification number (PIN) required to access a specific service such as a banking application.

B. Side-Channel Attacks

Although the above mentioned concepts are secure (or are typically considered as being secure) in theory, a specific implementation of such a mechanism is not necessarily secure in practice. Since side-channel attacks have been extensively used to attack cryptographic implementations, let us consider the following illustrative example. In an ideal world, an implementation of a cryptographic algorithm takes a specific input and produces a specific (intended) output. For example, an encryption algorithm takes the plaintext as well as cryptographic key material to produce the ciphertext. However, in practice, an implementation of an encryption algorithm usually also "outputs" unintended information as a byproduct of the actual computations. Such unintended information leakage might be a different power consumption or a different execution time due to instructions being conditionally executed depending on the processed data (cf. Figure 2). Attacks exploiting such unintended information leaks are denoted as side-channel attacks and have been impressively used to bypass or break protection mechanisms such as encryption algorithms.

Subsequently, we discuss the general notion of side-channel attacks. We distinguish between passive side-channel attacks, as in the example above, and active side-channel attacks.

Passive Side-Channel Attacks. The general notion of a passive side-channel attack can be described by means of three main components, *i.e.*, *target*, *side channel*, and *attacker*. A *target* represents anything of interest to possible attackers. During the computation or operation of the target, it influences a *side channel* (physical or logical properties) and thereby emits potential sensitive information. An *attacker* who is able to observe these side channels potentially learns useful information related to the actual computations or operations performed by the target. Therefore, an attacker models possible effects of specific causes. Later on, careful investigations of observed effects can then be used to learn information about possible causes.

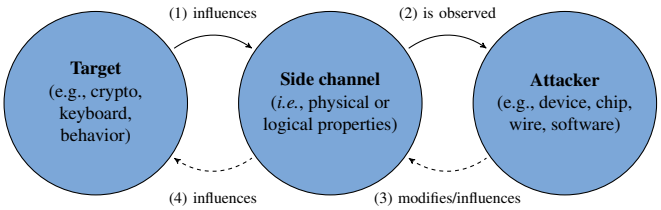


Fig. 3. General notion of passive (⟶) and active (⇠⇢) side-channel attacks.

Active Side-Channel Attacks. An active attacker tampers with the device or modifies/influences the targeted device via a side channel, e.g., via an external interface or environmental conditions. Thereby, the attacker influences the computation/operation performed by the device in a way that allows to bypass specific security mechanisms directly or that leads to malfunctioning, which in turn enables possible attacks, e.g., indirectly via the leaking side-channel information or directly via the (erroneous) output of the targeted device.

Figure 3 depicts the general notion of side-channel attacks. A target emits sensitive information as it influences specific side channels. For example, physically operating a smartphone via the touchscreen, *i.e.*, the touchscreen input represents the target, causes the smartphone to undergo specific movements and accelerations in all three dimensions. In this case, one possible side channel is the acceleration of the device (a physical property), which can be observed via the embedded accelerometer sensor and accessed by an app via the official Sensor API.

The relations defined via the solid arrows, *i.e.*, *target* ⟶ *side channel* ⟶ *attacker*, represent passive side-channel attacks. The relations defined via the dashed arrows, *i.e.*, *target* ⇠⇢ *side channel* ⇠⇢ *attacker*, represent active side-channel attacks where the attacker actively influences/manipulates the target via a side channel. Thereby, the attacker either tries (i) to enforce behavior that allows to bypass security mechanisms directly, or (ii) to observe leaking side-channel information or the (sometimes erroneous) output of the targeted device. Hence, a passive side-channel attack consists of steps (1) and (2), whereas an active side-channel attack also includes steps (3) and (4).

Differentiation From Other Attacks. Irrespective of whether an attacker is passive or active, we only consider side-channel attacks. Side-channel attacks *do not* exploit software bugs or anomalies within the OS or apps that, for example, allow to access the main communication channel directly. For example, buffer overflow attacks allow to access the main communication channel directly (*i.e.*, the main memory) and, thus, do not represent side-channel attacks.

Similarly, we also do not consider other attacks that learn information that is available from the main channel. For example, Luzio et al. [28] exploited Wi-Fi probe-requests, which contain the service set identifier (SSID) of preferred Wi-Fi hotspots in clear. These probe-requests allow mobile devices to determine nearby Wi-Fi hotspots in order to preferably connect to already known hotspots. These attacks do not represent side-channel attacks as the learned information is directly available from the main channel.

Furthermore, we also do not survey covert channels where two entities (e.g., processes) communicate over a channel that is not explicitly provided by the platform or the operating system. Although identified side channels can in general also be used as a covert channel, *i.e.*, as a means to stealthily communicate between two processes whereby one process influences the side channel and the other one observes it, we do not explicitly survey covert channels such as [29] in this paper. Nevertheless, our newly introduced classification system can also be used to classify covert channels.

C. Related Surveys

In this section, we discuss surveys on mobile security, as well as side-channel attacks on smart cards, PCs, cloud infrastructures, and smartphones.

Surveys on Mobile Security. Most surveys on mobile security primarily focused on malware in general, and many of these surveys only mention side-channel attacks as a side node. Enck [30] surveyed possible protection mechanisms beyond the standard protection mechanisms provided by Android. These include tools that analyze permissions and action strings (within the Android Manifest) to assess the risk of Android apps, policy-based approaches that allow a more fine-grained protection of Android apps, as well as static and dynamic code analysis tools to perform application analysis, which in turn allows to detect malware.

La Polla et al. [31] surveyed threats and vulnerabilities (*i.e.*, botnets, Trojans, viruses, and worms) with a focus on work published from 2004 until 2011. Suarez-Tangil et al. [32] and Faruki et al. [33] continued this line of research for the period from 2010 until 2013, and from 2010 until 2014, respectively.

Rashidi and Fung [34] surveyed techniques (e.g., based on static and dynamic code analysis) to cope with malware on mobile devices and Sadeghi et al. [35] surveyed tools and analysis techniques to identify malware. In addition, Sadeghi et al. provided a "survey of surveys" discussing surveys and their main contributions in more detail. We refer to their work for a more detailed investigation of malware analysis techniques and further literature on this topic. Tam et al. [36] surveyed mobile malware analysis techniques (static, dynamic, hybrid) as well as malware tactics to hinder analysis (obfuscation).

Surveys on Side-Channel Attacks. The survey of Tunstall [37] focused on smart card security, in particular side-channel attacks against cryptographic algorithms.

Zander et al. [38] surveyed covert channels via computer network protocols, and Biswas et al. [39] conducted an in-depth study on network timing channels (remote timing side channels) as well as in-system timing channels (focusing on hardware-based timing channels such as cache attacks) on commodity PCs. They surveyed timing channels according to their suitability for covert channels, timing side channels, and network flow watermarking (e.g., to de-anonymize Tor).

Regarding cloud computing platforms, Ge et al. [7] and Szefer [40] surveyed microarchitectural attacks with a focus on cache attacks. Ullrich et al. [41] focused on network-based covert channels and network-based side channels in cloud settings. Betz et al. [42] focused on covert channels and mentioned a few side-channel attacks in the cloud setting.

TABLE I
EXISTING SURVEYS AND WHAT THEY FOCUS ON. UPPER PART: SURVEYS ON MOBILE SECURITY. LOWER PART: SURVEYS ON SIDE-CHANNEL ATTACKS.

Year	Survey	Platform	Topic
2011	Enck [30]	Smartphone	Malware/app analysis and protection mechanisms
2013	La Polla et al. [31]	Smartphone	Threats and vulnerabilities, focusing on the period 2004–2011
2014	Suarez-Tangil et al. [32]	Smartphone	Threats and vulnerabilities, focusing on the period 2010–2013
2015	Faruki et al. [33]	Smartphone	Threats and vulnerabilities, focusing on the period 2010–2014
2015	Rashidi and Fung [34]	Smartphone	Analysis techniques to cope with malware
2016	Sadeghi et al. [35]	Smartphone	Tools and techniques to identify malware
2017	Tam et al. [36]	Smartphone	Analysis techniques to identify malware
2014	Tunstall [37]	Smart card	Side-channel attacks on cryptographic algorithms
2007	Zander et al. [38]	PC	Covert channels via computer network protocols
2017	Biswas et al. [39]	PC	Timing channels, focusing on microarchitectural attacks
2016	Ge et al. [7]	Cloud	Microarchitectural attacks
2016	Szefer [40]	Cloud	Microarchitectural attacks
2017	Ullrich et al. [41]	Cloud	Network-based side channels (and communication channels)
2017	Betz et al. [42]	Cloud	Communication channels
2016	Xu et al. [43]	Smartphone	Attacks & defense measures
2016	Hussain et al. [44]	Smartphone	Sensor-based keylogging attacks
2016	Nahapetian [45]	Smartphone	Sensor-based keylogging attacks

The focus of our paper is on side-channel attacks against mobile devices. Surveys about this topic are quite scarce and consider specific types of side-channel attacks only. Xu et al. [43] surveyed attacks and defenses on Android at a broader scale and thereby provide a comprehensive overview of the research landscape. They considered system privilege escalation, issues in the permission model, side channels and covert channels (a high-level overview of exploits considering the accelerometer, the CPU cache, and the procfs), feature abuses, malware detection, and app repackaging. Hussain et al. [44] and Nahapetian [45] surveyed sensor-based keylogging attacks. However, a systematic survey and classification of all existing categories of side-channel attacks on mobile devices does not exist so far. Hence, we close this gap in this paper.

Table I summarizes the main focus of the above discussed surveys and provides references for the interested reader.

III. TAXONOMY OF SIDE CHANNELS

In this section, we discuss the different types of information leaks, how the key enablers presented in Section I enable so-called *software-only attacks* on today's smartphones, and the generic adversary model followed by software-only attacks. Finally, we present our new categorization system.

A. Types of Information Leaks

Considering side-channel attacks on mobile devices, we identify two types of information leaks, namely *unintended information leaks* and *information published on purpose*. Figure 4 depicts these two types of information leaks. Informally, side-channel attacks exploiting unintended information leaks can be considered as "traditional" side-channel attacks since this category has already been extensively analyzed during the smart card era [4]. For example, unintended information leaks include the execution time, the power consumption, or the electromagnetic emanation of a computing device. This type of information leak is considered as unintended because smart card designers and developers did not plan to leak the timing information or power consumption of computing devices on purpose.

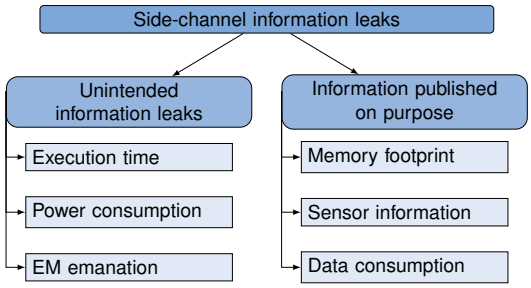


Fig. 4. Types of side-channel information leaks.

The second category of information leaks (referred to as *information published on purpose*) is mainly a result of the ever-increasing number of features provided by today's smartphones. In contrast to unintended information leaks, the exploited information is published on purpose and for benign reasons. For instance, specific features require the device to share (seemingly harmless) information and resources with apps running in parallel on the system. This information is either shared by the OS directly (e.g., via the procfs) or through the official Android API.[1] Although this information is extensively used by many legitimate applications for benign purposes, it sometimes turns out to leak sensitive information and, thus, leads to devastating side-channel attacks.

Many investigations impressively demonstrated that seemingly harmless information allows to infer sensitive information that is protected by dedicated security mechanisms, such as permissions. Examples of such seemingly harmless information are the memory footprint of an application as well as the data-usage statistics that keep track of the amount of incoming and outgoing network traffic. Both, the memory footprint [25] as well as the data-usage statistics [26], allow to infer a user's visited websites. The fundamental design weakness of assuming information as being innocuous (e.g., the memory footprint or the data-usage statistics) means that it is not protected by dedicated permissions.

[1]In the literature, some of the information leaks through the procfs are also denoted as *storage side channels* [46].

Furthermore, the second category seems to be more dangerous in the context of smartphones as new features are frequently added and new software interfaces allow to access an unlimited number of unprotected resources. Even developers taking care of secure implementations in the sense of unintended information leaks, e.g., by providing constant-time crypto implementations and taking care of possible software vulnerabilities such as buffer overflow attacks, inevitably leak sensitive information due to shared resources, the OS, or the Android API. Additionally, the provided software interfaces to access information and shared resources enable so-called *software-only attacks*, *i.e.*, side-channel attacks that only require the execution of software. This clearly represents an immense threat as these attacks (1) do not exploit any obvious software vulnerabilities, (2) do not rely on specific privileges or permissions, and (3) can be conducted remotely via seemingly harmless apps or even websites.

B. Software-only Side-Channel Attacks

Irrespective of whether a physical property (e.g., execution time [6] and power consumption [13]) or a logical property (e.g., memory footprint [25] and data-usage statistics [14], [26]) are exploited, many of these information leaks can be exploited by means of *software-only attacks*. More specifically, software-only attacks exploit leaking information without additional equipment, which was required for traditional side-channel attacks. For example, an oscilloscope is necessary to measure the power consumption of a smart card during its execution, or an EM probe is necessary to measure the EM emanation. In contrast, today's smartphones allow an impressive number of side-channel leaks to be exploited via software-only attacks. Besides, an attack scenario that requires the user to install an (unprivileged) application—*i.e.*, an addictive game—is entirely reasonable in an appified ecosystem.

For side-channel attacks in general, it does not matter whether the leaking information is collected via dedicated equipment or whether an unprivileged app collects the leaking information directly on the device under attack (software-only attacks). Interestingly, however, the immense amount of information published on purpose also allows to observe physical properties of the device as well as physical interactions with the device. Consequently, *software-only side channel attacks* have gained increasing attention in the last few years and impressive attacks are being continuously published.

Runtime-Information Gathering Attacks. Zhang et al. [16] coined the term runtime-information gathering (RIG) attack, which refers to attacks that require a malicious app to run side-by-side with a victim app on the same device in order to collect runtime information of the victim. According to Zhang et al. [16, p. 1] "(RIG) here refers to any malicious activities that involve collecting the data produced or received by an app during its execution, in an attempt to directly steal or indirectly infer sensitive user information". The crucial point in their definition is the distinction between *directly stealing* and *indirectly inferring* sensitive information. Inferring sensitive information indirectly is done by means of side-channel attacks. Hence, this generic class of attacks also includes a subset of side-channel attacks, especially side-channel attacks that can be launched via software-only attacks. However, RIG attacks also include attacks that we do not consider as side-channel attacks, *i.e.*, attacks that *directly* steal sensitive information. For example, RIG attacks also include attacks where apps request permissions which are exploited for (more obvious) attacks such as requesting the permission to access the microphone in order to eavesdrop on phone conversations.

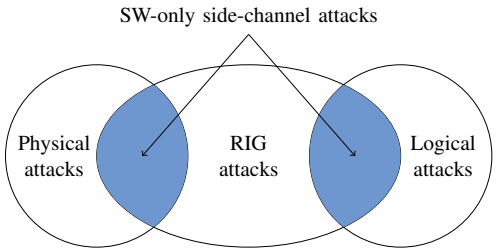


Fig. 5. SW-only side-channel attacks allow to exploit physical as well as logical properties.

Screenmilker [47]—an attack exploiting ADB[2] capabilities to take screenshots programmatically—is also considered being a RIG attack. We do not consider such attacks as side-channel attacks because these attacks exploit implementation flaws, *i.e.*, the exploited screenshot tool does not implement any authentication mechanism and hence any application can take screenshots programmatically. Similarly, we do not consider buffer overflow attacks as side-channel attacks because buffer overflow attacks represent a software vulnerability and allow to access the main channel directly, for example, by reading the main memory directly. Side-channel attacks, however, attack targets that are secure from a software perspective and still leak information unintentionally.

Figure 5 illustrates the new type of software-only side-channel attacks that allow to exploit both, physical properties as well as software features (logical properties), without additional equipment. Attacks exploiting information leaks resulting from hardware components, e.g., the power consumption, are classified as (physical) attacks exploiting physical properties. Attacks exploiting information leaks resulting from software components, e.g., statistics about network traffic, are classified as (logical) attacks exploiting logical properties.

As software-only attacks also rely on software being executed side-by-side with the victim application, software-only attacks are a sub-category of RIG attacks. It should be noted that physical attacks on smartphones might still rely on dedicated hardware and some logical attacks can also be conducted without running software on the device under attack. Such attacks are covered by the non-overlapping areas of "physical attacks" and "logical attacks" in Figure 5. However, physical attacks that cannot be conducted by running software on the device are more targeted attacks as they require attackers to be in physical presence of the device.

[2]The Android Debug Bridge (ADB) is a command line tool that allows to execute privileged commands on devices where USB debugging is activated.

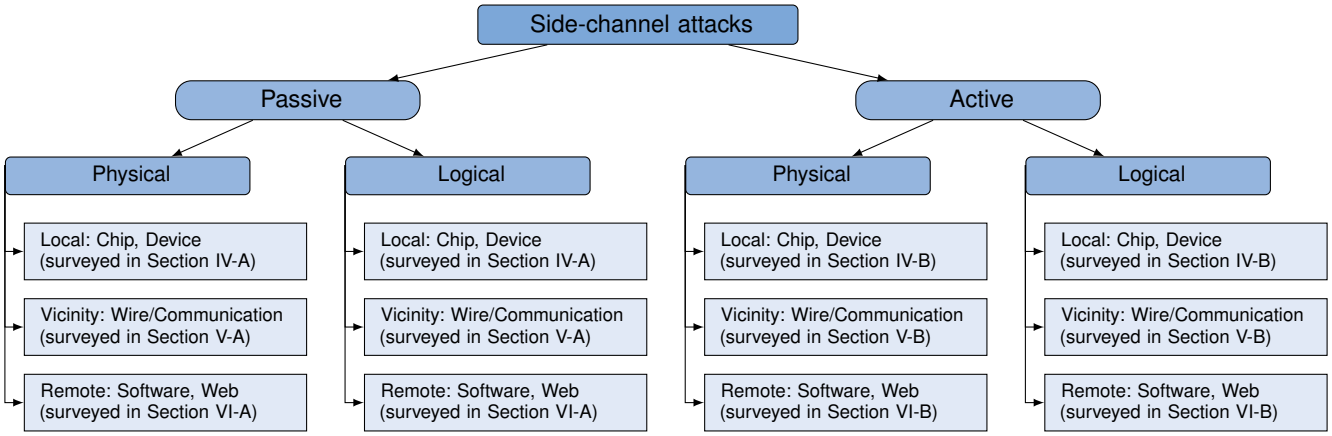


Fig. 6. Proposed classification system for side-channel attacks: (1) passive vs active, (2) physical properties vs logical properties, (3) local attackers vs vicinity attackers vs remote attackers.

C. Adversary Model and Attack Scenario

In contrast to traditional attacks that require an attacker to have the device under physical control or to be physically present with the victim, the adversary model for most (existing) side-channel attacks on smartphones shifted the scope to remote software execution by means of apps or websites. This also increases the scale of these attacks. While traditional side-channel attacks targeted only a few devices, modern side-channel attacks target possibly millions of devices or users at the same time. With this general overview of the adversary model in mind, most software-only attacks usually consider the following two-phase attack scenario for passive attacks.

Training Phase. In the training phase, the attacker "profiles" actions or events of interest, either during an online phase on the attacked device or during an offline phase in dedicated environments. Sometimes the training phase includes the training of a machine-learning model such as a supervised classifier. More abstractly, the attacker builds "templates" based on events of interest. In addition, the attacker crafts an app (or website) that ideally does not require any permissions or privileges in order to avoid raising the user's suspicion. This app is used in the attack phase to gather leaking information.

Attack Phase. The attack phase usually consists of three steps. (1) A malicious application—that is hidden inside a popular app—is spread via existing app markets. After installation, this malicious app waits in the background until the targeted app/action/event starts and then (2) it observes the leaking side-channel information. Based on the gathered information, (3) it employs the previously established model or templates to infer secret information. Depending on the complexity of the inference mechanism, e.g., the complexity of the machine-learning classifier, the gathered side-channel information could also be sent to a remote server, which then performs the heavy computations to infer the secret information.

D. A New Categorization System

Based on our observations we propose a new categorization system as depicted in Figure 6. More specifically, we classify side-channel attacks along three axes.

1) *Passive* vs *active*: This category distinguishes between attackers who passively observe leaking side-channel information and attackers who also actively influence the target via any side channel. For instance, an attacker can manipulate the target, its input, or its environment via any side channel in order to subsequently observe leaking information via abnormal behavior of the target (cf. [4]) or to bypass security mechanisms.
2) *Physical properties* vs *logical properties*: This category classifies side-channel attacks according to the exploited information, *i.e.*, depending on whether the attack exploits physical properties (hardware) or logical properties (software features). Physical properties include the power consumption, the electromagnetic emanation, or the physical movements of a smartphone during the operation. Logical properties include usage statistics provided by the operating system, such as the data-usage statistics or the memory footprint of an application.
3) *Local attackers* vs *vicinity attackers* vs *remote attackers*: Side-channel attacks are classified depending on whether or not the attacker must be in physical proximity/vicinity of the target. *Local attackers* clearly must be in (temporary) possession of the device or at least in close proximity. Depending on whether the adversary also needs to remove the package in order to access the chip, we classify local attackers into attackers that need access to the chip or only the device itself. *Vicinity attackers* are able to wiretap or eavesdrop the network communication of the target or to be somewhere in the vicinity of the target. *Remote attackers* only rely on software execution on the targeted device, e.g., either by means of executing software on the targeted device or by means of websites. Clearly, the scale increases significantly for these three attackers as a local attacker relies on stronger assumptions than a remote attacker. Especially the immense number of software-only attacks (that allow to conduct side-channel attacks remotely) stress the need for this category.

Subsequently, we briefly survey existing attacks according to our new classification system. Although the focus of this paper is on side-channel attacks against mobile devices, we

also discuss attacks that have been applied in the smart card or desktop/cloud setting, as today's smartphones are vulnerable to (all or most of the) existing side-channel attacks against these platforms as well. As mentioned before, we do not explicitly focus on Android devices, but the majority of existing papers investigate side-channel attacks on Android.

We start with *local* side-channel attacks in Section IV, continue with *vicinity* side-channel attacks in Section V, and finally we discuss *remote* side-channel attacks in Section VI. Each of these sections is further divided into *passive attacks* and *active attacks*. Note that this structure reflects our proposed classification system. However, for the sake of readability the structure of the subsections does not reflect the categorization of physical properties and logical properties.

IV. Local Side-Channel Attacks

In this section, we survey side-channel attacks that require a local adversary. Some of these attacks will show that the transition between local attacks and vicinity attacks is seamless as the distance between the victim (device) and the attacker can be increased, especially in case of some passive attacks.

A. Passive Attacks

We start with traditional side-channel attacks that aim to break insecure cryptographic implementations (of mathematically secure primitives). Besides, we discuss attacks that target the user's interaction with the device as well as the user's input on the touchscreen, *i.e.*, attacks that result from the inherent nature of mobile devices.

Power Analysis Attacks. The actual power consumption of a computing device or implementation depends on the processed data and the executed instructions. Power analysis attacks exploit this information leak to infer sensitive information. As the name suggests, the power consumption, typically measured as the voltage drop across a resistor inserted in the supply line, serves as the side channel. State-of-the-art printed circuit board designs (PCB-designs), including multilayer routing as well as surface mounted devices (SMD), and packaging techniques (e.g., ball-grid array) make it hard to access the appropriate power supply lines in modern smartphones without permanent modifications. Therefore, in contrast to smart cards, measuring the power consumption became less relevant for side-channel attacks targeting smartphones.

Depending on whether a single measurement trace or multiple traces are required, we distinguish between simple power analysis (SPA) attacks and differential power analysis (DPA) attacks, as defined by Kocher et al. [2]. SPA attacks rely on the interpretation of power traces in order to reveal, for example, the sequence of executed instructions, which allows to break implementations where the executed instructions depend on secret data. However, the power consumption also depends on the processed data, although the variations are smaller. Therefore, DPA attacks rely on statistical investigations of multiple traces in order to infer information about the processed data.

Attacks. Messerges and Dabbish [48] exploited the power consumption of a smart card to attack the Data Encryption Standard (DES) algorithm. Hardly any side-channel attacks using a similar setup for measuring the power consumption targeting smartphones are published. Nevertheless, a coarse-grained power-consumption monitoring of smartphones allows to identify running apps, as demonstrated by Yan et al. [49].

Electromagnetic Analysis Attacks. Another way to attack the leaking power consumption of computing devices is to exploit electromagnetic emanations, which are usually easier to obtain since the power line cannot be accessed directly in general. Irrespective of whether the power trace is obtained directly via the power line or via electromagnetic emanations, these attacks are usually denoted as differential power analysis attacks. In this context it is also worth to mention that depending on the used equipment (EM probes for capturing the electromagnetic emanation), targeting a specific location above the chip can improve the signal-to-noise ratio of the measurements. As a result of taking advantage of spatial information, the number of required measurements for a successful attack can be decreased.

Attacks. Traditional side-channel attacks exploiting the electromagnetic emanations of smart cards have also been applied on mobile devices. Gebotys et al. [50] demonstrated attacks on software implementations of the Advanced Encryption Standard (AES) and Elliptic Curve Cryptography (ECC) on Java-based PDAs. Later on, Nakano et al. [51] attacked ECC and RSA implementations of the default crypto provider (JCE) on Android smartphones, Goller and Sigl [52] attacked RSA implementations on Android, and Belgarric et al. [53] attacked the Elliptic Curve Digital Signature Algorithm (ECDSA) implementation of Android's Bouncy Castle. In a similar manner, Genkin et al. [54] attacked the OpenSSL implementation of ECDSA on Android and the CommonCrypto implementation of ECDSA on iOS, respectively.

Differential Computation Analysis. The basic idea of white-box crypto implementations is to embed the secret key into the software implementation in a way that prevents an attacker from extracting the key, even in case the adversary has access to the source code itself. Therefore, the key and the algorithm itself are merged such that the key is hidden inside the code and cannot be easily separated. The white-box attack model assumes that the adversary has full control over the device and the execution environment.

Attacks. Bos et al. [55] showed that binary instrumentation can be used to observe and control the intermediate state of white-box crypto implementations. Thereby, the instrumentation allows to precisely monitor the execution of the program and the observation of, e.g., the intermediate state and read/write accesses to memory, allow to profile program behavior. Based on the similarity to DPA attacks, Bos et al. denoted these attacks as differential computation analysis (DCA) attacks. Nevertheless, in contrast to DPA attacks, DCA attacks do not need to deal with any measurement noise.

Although attacks against white-box crypto implementations have not been applied on mobile devices so far, such an attack scenario works for these devices as well.

Smudge Attacks. The most common input method on mobile devices is the touchscreen, *i.e.*, users tap and swipe

on the screen with their fingers. Due to the inherent nature of touchscreens, users always leave residues in the form of fingerprints and smudges on the screen.

Attacks. Aviv et al. [56] pointed out that side-channel attacks can be launched due to specific interactions with the smartphone or touchscreen-based devices in general. More specifically, forensic investigations of smudges (oily residues from the user's fingers) on the touchscreen allow to infer unlock patterns. Even after cleaning the phone or placing the phone into the pocket, smudges seem to remain most of the time. Hence, smudges are quite persistent which increases the threat of smudge attacks. Follow-up work considering an attacker who employs fingerprint powder to infer keypad inputs has been presented by Zhang et al. [57] and also an investigation of the heat traces—left on the screen due to finger touches—by means of thermal cameras has been performed [58].

Shoulder Surfing and Reflections. Touchscreens of mobile devices optically/visually emanate the displayed content. Often these visual emanations are reflected by objects in the environment, such as sunglasses and tea pots [59], [60].

Attacks. Maggi et al. [61] observed that touchscreen input can be recovered by monitoring the visual feedback (pop-up characters) on soft keyboards during the user input. Therefore, they rely on cameras that are pointed directly on the targeted screen. Raguram et al. [62], [63] observed that reflections, e.g., on the user's sunglasses, can also be used to recover input typed on touchscreens. However, the attacker needs to point the camera, used to capture the reflections, directly on the targeted user. Subsequently, they rely on computer vision techniques and machine learning techniques to infer the user input from the captured video stream. Xu et al. [64] extended the range of reflection-based attacks by considering reflections of reflections. Although, they do not rely on the visual feedback of the soft keyboard but instead track the user's fingers on the smartphone while interacting with the device.

By increasing the distance between the attacker and the victim, e.g., by relying on more expensive and sophisticated cameras, some of these attacks might as well be considered as vicinity attacks.

Hand/Device Movements. Many input methods on various devices rely on the user operating the device with her hands and fingers. For instance, users tend to hold the device in their hands while operating it with their fingers.

Attacks. Similar to reflections, Shukla et al. [65] proposed to monitor hand movements as well as finger movements—without directly pointing the camera at the targeted screen—in order to infer entered PIN inputs. Sun et al. [66] monitored the backside of tablets during user input and detected subtle motions that can be used to infer keystrokes, while Yue et al. [67] proposed an attack where the input on touch-enabled devices can be estimated from a video of a victim tapping on a touch screen.

Again, by increasing the distance between the attacker and the victim, these attacks might also be considered as vicinity attacks, which demonstrates the seamless transition from local attacks to vicinity attacks for these types of attacks.

B. Active Attacks

An active attacker also manipulates the target, its input, or its environment in order to subsequently observe leaking information via abnormal behavior of the target or to bypass security mechanisms directly. While the transition between local and vicinity attackers is seamless in case of passive attacks, active attacks always assume that the attacker is in possession of the device (at least temporary).

Active attacks against cryptographic implementations date back to the works of Boneh et al. [68] (a.k.a. Bellcore attack) who attacked RSA crypto systems, especially implementations based on the Chinese Remainder Theorem (CRT), by relying on random hardware faults that result in the output of an erroneous signature. Later, Biham and Shamir [69] coined the term differential fault analysis (DFA) attacks and demonstrated that the introduction of faults and observing differences in the output ciphertext allow to recover the secret key of symmetric primitives. The basic idea of these attacks is to solve algebraic equations based on erroneous outputs (and valid outputs).

Clock/Power Glitching. Variations of the clock signal, e.g., overclocking, have been shown to be an effective method for fault injection on embedded devices in the past. One prerequisite for this attack is an external clock source. Microcontrollers applied in smartphones typically have an internal clock generator, making clock tampering difficult. Besides clock tampering, intended variations of the power supply represent an additional method for fault injection. With minor hardware modifications, power-supply tampering can be applied on most microcontroller platforms.

Attacks. In [70] it is shown how to disturb the program execution of an ARM CPU on a Raspberry PI by underpowering, *i.e.*, the supply voltage is set to ground (GND) for a short time. Due to the relatively easy application on modern microcontrollers, voltage-glitching attacks pose a serious threat for smartphones if attackers have physical access to the device. For instance, O'Flynn [71] demonstrated that by shorting the power supply of an off-the-shelf Android smartphone, a fault can be introduced that leads to an incorrect loop count.

Electromagnetic Fault Injection (EMFI). Transistors placed on microchips can be influenced by electromagnetic emanation. EMFI attacks take advantage of this fact. These attacks use short (in the range of nanoseconds), high-energy EM pulses to, e.g., change the state of memory cells, resulting in erroneous calculations. In contrast to voltage glitching, where the injected fault is typically global, EMFI allows to target specific regions of a microchip by precisely placing the EM probe, e.g., on the instruction memory, the data memory, or CPU registers. Compared to optical fault injection, EMFI attacks do not necessarily require a decapsulation of the chip, which makes them more practical.

Attacks. Ordas et al. [72] reported successful EMFI attacks targeting the AES hardware module of a 32 bit ARM processor. Rivière et al. [73] used EMFI attacks to force instruction skips and instruction replacements on modern ARM microcontollers. Considering the fact that ARM processors are applied in modern smartphones, EMFI attacks represent a serious threat for such devices.

TABLE II
OVERVIEW OF LOCAL SIDE-CHANNEL ATTACKS AND CORRESPONDING TARGETS. ✓ AND ✗ INDICATE WHETHER OR NOT A SPECIFIC ATTACK HAS BEEN PERFORMED ON THE CORRESPONDING TARGET.

Attack	Active/passive	Property	Targets		
			Crypto, program flow	Application inference	User input
Power analysis attacks	Passive	Physical	✓ [48]	✓ [49]	✗
Electromagnetic analysis attacks	Passive	Physical	✓ [50]–[54]	✗	✗
Differential computation analysis	Active/passive	Logical	✓ [55], [74]	✗	✗
Smudge attacks	Passive	Physical	✗	✗	✓ [56]–[58]
Shoulder surfing and reflections	Passive	Physical	✗	✗	✓ [61]–[64]
Hand/device movements	Passive	Physical	✗	✗	✓ [65]–[67]
Clock/power glitching	Active	Physical	✓ [70], [71]	✗	✗
Electromagnetic fault injection	Active	Physical	✓ [72], [73]	✗	✗
Laser/optical faults	Active	Physical	✓ [75]–[77]	✗	✗
Temperature variation	Active	Physical	✓ [78], [79]	✗	✗
NAND mirroring	Active	Physical	✓ [80]	✗	✗

Laser/Optical Faults. Optical fault attacks using a laser beam are among the most-effective fault-injection techniques. These attacks take advantage of the fact that a focused laser beam can change the state of a transistor on a microcontroller, resulting in, e.g., bit flips in memory cells. Compared to other fault-injection techniques (voltage glitching, EMFI), the effort for optical fault injection is high. First, decapsulation of the chip is a prerequisite in order to access the silicon with the laser beam. Second, finding the correct location for the laser beam to produce exploitable faults is also not a trivial task.

Attacks. First optical fault-injection attacks targeting an 8-bit microcontroller have been published by Skorobogatov and Anderson [75] in 2002. Inspired by their work, several optical fault-injection attacks have been published in the following years, most of them targeting smart cards or low-resource embedded devices (e.g., [76], [77]). The increasing number of metal layers on top of the silicon, decreasing feature size (small process technology), and the high decapsulation effort make optical fault injection difficult to apply on modern microprocessors used in smartphones.

Temperature Variation. Operating a device outside of its specified temperature range allows to cause faulty behavior. Heating up a device above the maximum specified temperature can cause faults in memory cells. Cooling down the device has an effect on the speed RAM content fades away after power off (*remanence effect* of RAM).

Attacks. Hutter and Schmidt [78] presented heating fault attacks targeting an AVR microcontroller. They prove the practicability of this approach by successfully attacking an RSA implementation on named microcontroller. FROST [79], on the other hand, is a tool to recover disc encryption keys from RAM on Android devices by means of cold-boot attacks. Here the authors take advantage of the increased time data in RAM remains valid after power off due to low temperature.

Differential Computation Analysis. As already mentioned above, the white-box model assumes that the attacker has full control over the execution environment. This also means that the attacker can produce erroneous or faulty outputs by manipulating intermediate values during the computation.

Attacks. Sanfelix et al. [74] demonstrated that attackers in the white-box model can also perform fault injection attacks. As the attacker has full control over the execution environment and the executed binary, she can also manipulate data during the program execution or manipulate the control flow of the execution. Similar to other fault attacks, the idea is to observe differences between normal outputs and erroneous outputs of the binary in order to break the cryptographic implementations.

NAND Mirroring. Data mirroring refers to the replication of data storage between different locations. Such techniques are used to recover critical data after disasters but also allow to restore a previous system state.

Attacks. The Apple iPhone protects a user's privacy by encrypting the data. Therefore, a passcode and a hardware-based key are used to derive various keys that can be used to protect the data on the device. As a dedicated hardware-based key is used to derive these keys, brute-force attempts must be done on the attacked device. Furthermore, brute-force attempts are discouraged by gradually increasing the waiting time between wrongly entered passcodes up to the point where the phone is wiped. In response to the Apple vs FBI case, Skorobogatov [80] demonstrated that NAND mirroring can be used to reset the phone state and, thus, can be used to brute-force the passcode. Clearly, this approach also represents an active attack as the attacker actively influences (resets) the state of the device.

C. Overview

Table II summarizes the discussed attack categories and the targeted information. In terms of targets, we identified cryptographic implementations (crypto), the program flow of applications (which sometimes also allows to attack crypto because different branches might be executed depending on specific key bits), application inference (inference of the executed application), and user input. An attack category not targeting specific information (yet), which is indicated by an ✗, represents a possible gap that might be investigated in future research. For example, power analysis attacks might allow to target user input, such as keystrokes or even actual characters, and shoulder surfing and reflection attacks might well allow to infer running applications. However, for some attacks it is (highly) unlikely that they will work against specific targets. For example, attacking cryptographic algorithms by means of smudge attacks is unlikely to work.

TABLE III
OVERVIEW OF VICINITY SIDE-CHANNEL ATTACKS AND CORRESPONDING TARGETS. ✓ AND ✗ INDICATE WHETHER OR NOT A SPECIFIC ATTACK HAS BEEN PERFORMED ON THE CORRESPONDING TARGET.

Attack	Active/passive	Property	Targets			
			Visited websites	Application/action inference	Identify users/devices	User input
Network traffic analysis	Active/passive	Physical/logical	✓ [81]–[86]	✓ [87]	✓ [88]	✗
USB power analysis	Passive	Physical	✓ [89]	✗	✓ [90]	✗
Wi-Fi signal monitoring	Passive	Physical	✗	✗	✗	✓ [91], [92]

V. VICINITY SIDE-CHANNEL ATTACKS

In this section, we survey attacks where the attacker must be in the vicinity of the targeted user/device, *i.e.*, attacks where the attacker compromises, for example, any infrastructure facility within the user's environment.

A. Passive Attacks

Network Traffic Analysis. In general, the encryption of messages transmitted between two parties only hides the actual content, while specific meta data such as the overall amount of data is not protected. This meta data allows to infer sensitive information about the content and the communicating parties.

Attacks. Network traffic analysis has been extensively studied in the context of website fingerprinting attacks. These attacks [82]–[86] wiretap network connections and observe traffic signatures, e.g., unique packet lengths, inter-packet timings, etc., to infer visited websites and even work in case the traffic is routed through Tor. While most of these attacks target the network communication in general, attacks explicitly targeting mobile devices also exist. For instance, Stöber et al. [88] assumed that an adversary eavesdrops on the UMTS transmission and showed that smartphones can be fingerprinted based on the background traffic of installed apps. Conti et al. [87] considered an adversary who controls Wi-Fi access points near the targeted device, which allows to infer specific app actions such as posting Facebook status messages. In similar settings, traffic analysis techniques allow to fingerprint specific apps as well as actions performed in specific apps [93]–[98].

While the above presented attacks exploit logical properties, *i.e.*, the fact that encrypted packets do not hide meta data, Schulz et al. [99] exploited the EM emanation of Ethernet cables (hardware properties), which allowed them to observe parts of the transmitted Ethernet frames.

USB Power Analysis. Due to the inherent usage patterns of mobile devices, users are constantly in the need to charge their devices, which is why public USB charging stations have been set up. Similar to power analysis attacks, modified charging stations can be used to collect power traces that allow to infer sensitive information about users and mobile devices.

Attacks. The identification (or localization) of specific users is considered a privacy risk due to the possibility of tracking individuals. Conti et al. [90] demonstrated that wall-socket smart meters that capture the power consumption of plugged devices can be used to identify users/notebooks. Although they demonstrated their attack on notebooks, it is likely that the same attack works for smartphones as well. In a similar setting, Yang et al. [89] demonstrated that visited websites can be inferred by power traces collected via USB charging stations. Such attacks even work if dedicated protection mechanisms, e.g., adapters that block data pins on USB cables, are in place.

Wi-Fi Signal Monitoring. Wi-Fi devices continuously monitor the wireless channel (channel state information (CSI)) to effectively transmit data. This is necessary as environmental changes cause the CSI values to change.

Attacks. Ali et al. [100] observed that even finger motions impact wireless signals and cause unique patterns in the time-series of CSI values. In a setting with a sender (notebook) and a receiver (Wi-Fi router), they showed that keystrokes on an external keyboard cause distortions in the Wi-Fi signal. They infer entered keys by monitoring these changes of the CSI values. Later on, Zhang et al. [91] inferred unlock patterns on smartphones via a notebook that is connected to the wireless hotspot provided by the smartphone. Li et al. [92] further improved these attacks by considering an attacker controlling only a Wi-Fi access point. They infer the PIN input on smartphones and also analyze network packets to determine when the sensitive input starts.

B. Active Attacks

Besides passively observing leaking information, vicinity attacks can be improved by considering active attackers as demonstrated by the following example.

Network Traffic Analysis. Network traffic analysis has already been discussed in the context of passive side-channel attacks. Active attackers learn additional information by actively influencing transmitted packets, e.g., by delaying packets.

Attacks. He et al. [81] demonstrated that an active attacker, e.g., represented by an Internet Service Provider (ISP), could delay HTTP requests from Tor users in order to increase the performance of website fingerprinting attacks. The idea is that instead of observing the generated traffic for all resources on a webpage in parallel, *i.e.*, the response packets from multiple requests in parallel overlap, an attacker delays the packet requesting a resource until the response from the previous request has been fully retrieved.

C. Overview

Table III summarizes the discussed attack categories and the targeted information. The identified targets are the inference of visited websites, application inference (or specific actions within applications), identification of users and devices, and user input. Again an attack category not targeting specific information (indicated by an ✗) represents a possible gap that might be closed in future research. For example, USB power analysis attacks might allow to target user input.

VI. Remote Side-Channel Attacks

The attacks presented in this section can be categorized as software-only attacks. In contrast to the local side-channel attacks as well as the vicinity side-channel attacks presented in the previous sections, these attacks neither require the attacker to be in the proximity nor in the vicinity of the targeted user. Hence, these attacks can be executed remotely and target a much larger scale since the victim user installed a malicious application on her device.

A. Passive Attacks

Linux-inherited *procfs* Leaks. Linux releases "accounting" information that is considered as being harmless via the procfs. This includes, for example, the memory footprint (total virtual memory size and total physical memory size) of each application via `/proc/[pid]/statm`, the CPU utilization times via `/proc/[pid]/stat`, the number of context switches via `/proc/[pid]/status`, but also system-wide information such as interrupt counters via `/proc/interrupts` and context switches via `/proc/stat`.

Attacks. Jana and Shmatikov [25] observed that the memory footprint of the browser correlates with the rendered website. Thus, by monitoring the memory footprint they inferred a user's browsing behavior (browser history), which represents sensitive information and is normally protected by a dedicated permission. Later on, Chen et al. [101] exploited this information to detect *Activity* transitions within Android apps. They observed that the shared memory size increases by the size of the graphics buffer in both processes, *i.e.*, the app process and the window compositor process (*SurfaceFlinger*). These increases occur due to the inter-process communication (IPC) between the app and the window manager. Besides, they also considered CPU utilization and network activity in order to infer the exact activity later on.

Similar to the memory footprint of applications, the procfs also provides system-wide information about the number of interrupts and context switches. Again, this information is considered as being innocuous and is, thus, published on purpose and is accessible without any permission. Simon et al. [11] exploited this information to infer text entered via swipe input methods. More specifically, they observed that the number of interrupts and context switches correlates with the user's finger movements across the keyboard when transitioning from letter to letter. Diao et al. [102] presented two attacks to infer unlock patterns and the app running in the foreground. The information leaks exploited were gathered from interrupt time series of the device's touchscreen controller. Besides, also the power consumption is released via the procfs. Yan et al. [49] showed that the power consumption allows to infer the number of entered characters on the soft keyboard.

Data-Usage Statistics. Android keeps track of the amount of incoming and outgoing network traffic on a per-application basis. These statistics allow users to keep an eye on the data consumption of any app and can be accessed without any permission.

Attacks. Data-usage statistics are captured with a fine-grained granularity, *i.e.*, packet lengths of single TCP packets can be observed, and have already been successfully exploited. Zhou et al. [14] demonstrated that by monitoring the data-usage statistics an adversary can infer sensitive information of specific apps. They were able to infer disease conditions accessed via *WebMD*, and the financial portfolio via *Yahoo! Finance*. In addition, they also showed how to infer a user's identity by observing the data-usage statistics of the *Twitter* app and exploiting the publicly available Twitter API.

Later, it has been shown that the data-usage statistics can also be exploited to infer a user's browsing behavior [26]. The fine-grained statistics of incoming and outgoing network packets allow to fingerprint websites, which even works in case the traffic is routed through the anonymity network Tor.

Page Deduplication. To reduce the overall memory footprint of a system, (some) operating systems[3] search for identical pages within the physical memory and merge them—even across different processes—which is called page deduplication. As soon as one process intends to write onto such a deduplicated page, a copy-on-write fault occurs and the process gets its own copy of this memory region again.

Attacks. Such copy-on-write faults have been exploited by Suzaki et al. [103] to detected applications on Linux and Windows as well as file downloads. Recently, Gruss et al. [24] demonstrated the possibility to measure the timing differences between normal write accesses and copy-on-write faults from within JavaScript code. Based on these precise timings they suggest to fingerprint visited websites by allocating memory that stores images found on popular websites. If the user browses the website with the corresponding image, then at some point the OS detects the identical content in the pages and deduplicates these pages. By continuously writing to the allocated memory, the attacker might observe a copy-on-write fault in which case the attacker knows that the user currently browses the corresponding website.

Microarchitectural Attacks. Modern computer architectures include many components to improve the overall effectiveness and performance. For instance, CPU caches represent an important component within the memory hierarchy of modern computer architectures. Multiple cache levels bridge the gap between the latency of main memory accesses and the fast CPU clock frequencies. Microarchitectural attacks exploit specific effects like the timing behavior of these components, e.g., branch prediction units and CPU caches, in order to learn sensitive information about executed instructions, code paths, etc. More specifically, by measuring execution times and memory accesses, an attacker can infer sensitive information from processes running in parallel on the same device. As CPU caches have been shown to represent a powerful source of information leaks, we focus on cache attacks.

Attacks. Cache-timing attacks against AES have already been investigated on Android-based mobile devices. For instance, Bernstein's cache-timing attack [104] has been launched on development boards [105]–[107] and on Android smartphones [108], [109] in order to reduce the effective key size of AES. Besides, similar cache attacks have been

[3]For example, CyanogenMod OS allows to enable page deduplication.

launched on embedded devices [110] and more fine-grained attacks [5] against AES have also been applied on smartphones [111]. These attacks relied on privileged access to precise timing measurements, but as stated by Oren et al. [112] cache attacks can also be exploited via JavaScript and, thus, do not require native code execution anymore. They even demonstrated the possibility to track user behavior including mouse movements as well as browsed websites via JavaScript-based cache attacks. A recent paper by Lipp et al. [113] demonstrates that all existing cache attacks, including the effective Flush+Reload attack [6], can be applied on modern Android smartphones without any privileges. While early attacks on smartphones exclusively targeted cryptographic implementations, their work also shows that user interactions (touch actions and swipe actions) can be inferred through this side channel. Similar investigations of Flush+Reload on ARM have also been conducted by Zhang et al. [114].

As some of these attacks actively influence the behavior of the victim, e.g., the execution time, some microarchitectural attacks can also be considered as active attacks. For a more detailed survey about microarchitectural attacks in general, we refer to the survey papers by Ge et al. [7] and Szefer [40].

Sensor-based Keyloggers. Cai et al. [115] and Raij et al. [116] were one of the first to discuss privacy implications resulting from mobile devices equipped with cameras, microphones, GPS sensors, and motion sensors in general. Nevertheless, a category of attacks that received significant attention are sensor-based keyloggers. These attacks are based on two observations. First, smartphones are equipped with lots of sensors—both motion sensors as well as ambient sensors—that can be accessed without any permission, and second, these devices are operated with fingers while being held in the users' hands. Hence, the following attacks are all based on the observation that users tap/touch/swipe the touchscreen and that the device is slightly tilt and turned during the operation.

Attacks. In 2011, Cai and Chen [9] were the first to observe a correlation between entered digits on touchscreens and the readings from the accelerometer sensor that can be exploited for motion-based keylogging attacks. Following this work, Owusu et al. [117] extended the attack to infer single characters, and Aviv [118] and Aviv et al. [10] investigated the accelerometer to attack PIN and pattern inputs. Subsequent publications [119]–[121] also considered the combination of the accelerometer and the gyroscope in order to improve the performance as well as to infer even longer text inputs [122].

Since the W3C specifications allow access to the motion and orientation sensors from JavaScript, motion-based keylogging attacks have even been performed via websites [12], [123]. Even worse, some browsers continue to execute JavaScript, although the user closed the browser or turned off the screen.

While the above summarized attacks exploit different motion sensors, e.g., accelerometer and gyroscope, ambient sensors can also be used for keylogging attacks. Spreitzer [124] presented an attack that exploits an ambient sensor, namely the ambient-light sensor, in order to infer a user's PIN input on touchscreens. Minor tilts and turns during keyboard input lead to variations of the ambient-light sensor readings, which are then correlated with keyboard input on the touchscreen.

As demonstrated by Simon and Anderson [125], PIN inputs on smartphones can also be inferred by continuously taking pictures via the front camera. Afterwards, PIN digits can be inferred by image analysis and by investigating the relative changes of objects in subsequent pictures that correlate with the entered digits. Fiebig et al. [126] demonstrated that the front camera can be used to capture the screen reflections in the user's eyeballs, which allows to infer user input. In a similar manner, Narain et al. [127] and Gupta et al. [128] showed that tap sounds (inaudible to the human ear) recorded via smartphone stereo-microphones can be used to infer typed text on the touchscreen. However, these attacks require dedicated permissions to access the camera and the microphone, which might raise the user's suspicion. In contrast, the motion and ambient sensors can be accessed without any permission.

For a more complete overview of sensor-based keylogging attacks, we refer to the survey papers by Hussain et al. [44] and Nahapetian [45]. Considering the significant number of papers that have been published in this context, user awareness about such attacks should be raised. Especially since Mehrnezhad et al. [123] found that the perceived risk of motion sensors, especially ambient sensors, among users is very low.

Fingerprinting Devices/Users. The identification of smartphones (and users) without a user's awareness is considered a privacy risk. While obvious identification mechanisms such as device IDs and web cookies can be thwarted, imperfections of hardware components, e.g., sensors, as well as specific software features can also be employed to stealthily fingerprint and identify devices and users, respectively.

Attacks. Bojinov et al. [129] and Dey et al. [130] observed that unique variations of sensor readings (e.g., of the accelerometer) can be used to fingerprint devices. These variations are a result of the manufacturing process and are persistent throughout the life of the sensor/device. As these sensors can be accessed via JavaScript, it is possible to fingerprint devices via websites [131]. Similarly, such imperfections also affect the microphones and speakers [132], [133], which also allow to fingerprint devices. In addition, by combining multiple sensors, even higher accuracies can be achieved [134].

Kurtz et al. [135] demonstrated how to fingerprint mobile device configurations, e.g., device names, language settings, installed apps, etc. Hence, their fingerprinting approach exploits software properties (*i.e.*, software configurations) only. Hupperich et al. [136] proposed to combine hardware features as well as software features to fingerprint mobile devices.

Location Inference. As smartphones are always carried around, information about a phone's location inevitably reveals the user's location. Hence, resources that obviously can be used to determine a user's location, e.g., the GPS sensor, are considered as privacy relevant and, thus, require a dedicated permission. Yet, even without permissions, side-channel attacks can be used to infer precise location information about users.

Attacks. Han et al. [141], Nawaz et al. [142], and Narain et al. [143] demonstrated that the accelerometer and the gyroscope can be used to infer car driving routes. Similarly,

TABLE IV
OVERVIEW OF REMOTE SIDE-CHANNEL ATTACKS AND CORRESPONDING TARGETS. ✓ AND ✗ INDICATE WHETHER OR NOT A SPECIFIC ATTACK HAS BEEN PERFORMED ON THE CORRESPONDING TARGET.

Attack	Active/passive	Property	Targets						
			Visited websites	Application/action inference	Identify users/devices	User input	Crypto	Location inference	Privilege escalation
procfs leaks	Passive	Physical/logical	✓ [25], [26]	✓ [101], [102]	✓ [14]	✓ [11], [49], [102]	✗	✓ [13]	✗
Data-usage statistics	Passive	Logical	✓ [26]	✓ [14]	✓ [14]	✗	✗	✗	✗
Page deduplication	Passive	Logical	✓ [24]	✓ [103]	✗	✗	✗	✗	✗
Microarchitectural attacks	Active/passive	Physical	✓ [112]	✓ [112], [113]	✗	✓ [113]	✓ [108]–[111], [113]	✗	✗
Sensors	Passive	Physical	✗	✓	✓ [129]–[131]	✓ [9], [10], [12], [117]–[124], [137]	✗	✓ [138]	✗
Microphone	Passive	Physical	✗	✗	✓ [132], [133]	✓ [127], [128], [139]	✗	✗	✗
Speakers	Passive	Physical	✗	✗	✓ [132], [133]	✗	✗	✗	✗
Camera	Passive	Physical	✗	✗	✗	✓ [125], [126]	✗	✗	✗
Device configurations	Passive	Logical	✗	✗	✓ [135]	✗	✗	✗	✗
Rowhammer	Active	Physical	✗	✗	✗	✗	✗	✗	✓ [140]

Hemminki et al. [144] inferred the transportation mode, e.g., train, bus, metro, etc., via the accelerometer readings. Besides the accelerometer and the gyroscope, ambient sensors can also be used to infer driving routes. Ho et al. [138] exploited the correlation between sensor readings of the barometer sensor and the geographic elevation to infer driving routes.

Even less obvious side-channels that allow to infer driving routes and locations are the speaker status information (e.g., speaker on/off) and the power consumption (available via the procfs). More specifically, Zhou et al. [14] observed that the Android API allows to query whether or not the speaker is currently active, *i.e.*, boolean information that indicates whether or not any app is playing sound on the speakers. They exploit this information to attack the turn-by-turn voice guidance of navigation systems. By continuously querying this API, they determine how long the speaker is active. This information allows to infer the speech length of voice direction elements, e.g., the length of "Turn right onto East Main Street". As driving routes consist of many such turn-by-turn voice guidances, fingerprinting driving routes is possible.

Michalevsky et al. [13] observed that the power consumption (available in the procfs) is related to the strength of the cellular signal, which depends on the distance to the base station. Given this information, they inferred a user's location.

Speech Recognition. Eavesdropping conversations represents a severe privacy threat. Thus, a dedicated permission protects the access to the microphone. However, acoustic signals, such as human speech, in the vicinity of a mobile device also influence the gyroscope measurements.

Attacks. Michalevsky et al. [137] exploited the gyroscope sensor to measure acoustic signals in the vicinity of the phone and to recover speech information. Although they only consider a small set of vocabulary, *i.e.*, digits only, their work demonstrates the immense power of gyroscope sensors in today's smartphones. By exploiting the gyroscope sensor to eavesdrop on a user's conversations they are able to bypass the permission required to access the microphone.

Soundcomber. Customer service departments often rely on automated menu services to interact with customers over the phone. A well-known example are the interactive voice response systems supported by telephone services that use dual-tone multi-frequency (DTMF) signaling to transmit entered numbers, *i.e.*, an audio signal is transmitted for each key.

Attacks. As DTMF tones are also played locally, Schlegel et al. [139] showed that by requesting permission to access the microphone, these tones can be recorded and used to infer sensitive input provided to these automated menu services. More specifically, they exploit this information to infer credit card numbers entered while interacting with such interactive voice response systems of credit card companies.

B. Active Attacks

An area of research that gains increasing attention among the scientific community are active side-channel attacks that can be exploited via software execution only. The most prominent example is the so-called Rowhammer attack that exploits DRAM disturbance errors to conduct software-induced fault attacks.

Rowhammer. The increasing density of memory cells within the DRAM requires the size of these cells to decrease, which in turn decreases the charging of single cells but also causes electromagnetic coupling effects between cells.

Attacks. Kim et al. [145] demonstrated that these observations can be used to induce hardware faults, *i.e.*, bit flips in neighboring cells, via frequent memory accesses to the main memory. Thereby, they showed that frequent memory accesses in the attacker's memory allow to induce faults (bit flips) in the victim's memory. Seaborn and Dullien [146] demonstrated how to possibly exploit these bit flips from native code and Gruss et al. [147] showed that such bit flips can even be induced via JavaScript code. A recent paper [140] demonstrates the exploitation of the Rowhammer bug to gain root privileges on Android smartphones by inducing bit flips from an unprivileged application.

C. Overview

Table IV summarizes the discussed attack categories and the targeted information. The target "application/action inference" also refers to sensitive information that can be inferred from specific actions. For example, diseases conditions, stock portfolios, etc. can be inferred from data-usage statistics (cf. [14]). The target "user input" refers to PIN and pattern inputs on the screen, inter-keystroke timing information, and also the DTMF tone exploitation [139]. Again an attack category not targeting specific information (yet), which is indicated by an ✗, represents a possible gap that might be closed in future research.

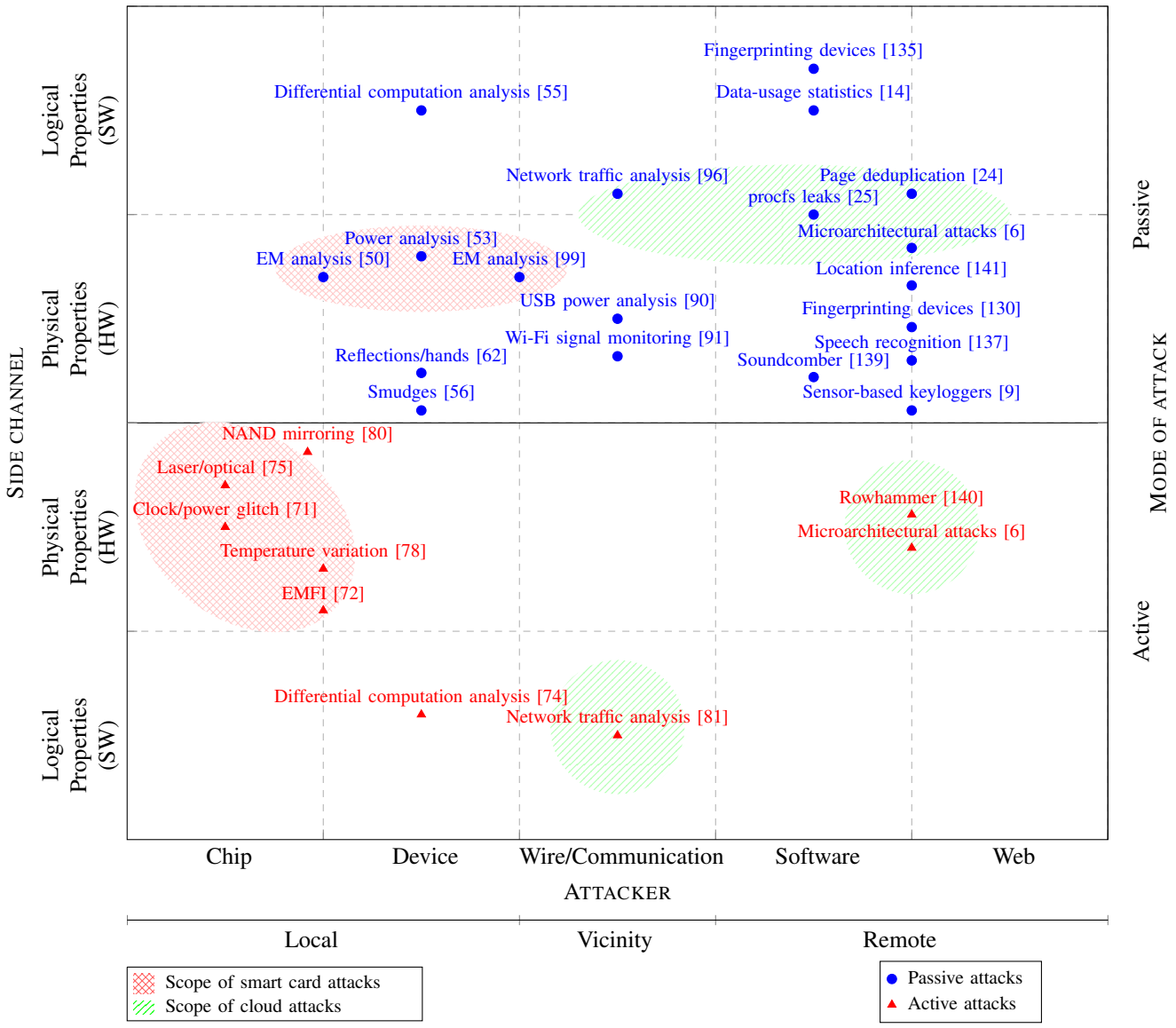


Fig. 7. Overview of side-channel attacks: (1) active vs passive, (2) logical properties vs physical properties, (3) local vs vicinity vs remote.

VII. Trend Analysis

In Figure 7 we classify the attacks surveyed in Sections IV–VI according to our new classification system. We distinguish between *active* and *passive* attackers along the (right) y-axis. Passive attacks are classified above the x-axis and active attacks are classified below the x-axis. The (left) y-axis distinguishes between the exploitation of *physical properties* and *logical properties*. As both of these categories can be exploited by passive as well as active attackers, these two categories are mirrored along the x-axis. The x-axis categorizes side-channel attacks according to the attacker's proximity to the targeted device. For instance, some attacks require an attacker to have access to the targeted device or even to have access to components within the device, e.g., the attacker might remove the back cover in order to measure the EM emanation of the chip. Stronger adversaries (with weaker assumptions) might rely on wiretapping techniques. The strongest adversaries rely on unprivileged applications being executed on the targeted device or even only that the victim visits a malicious website.

Based on this classification system we observe specific trends in modern side-channel attacks that will be discussed within the following paragraphs. This trend analysis also includes pointers for possible research directions.

From Local to Remote Attacks. The first trend that can be observed is that, in contrast to the smart card era, the smartphone era faces a shift towards remote side-channel attacks that focus on both hardware properties and software features. The shift from local attacks (during the smart card era) towards remote attacks (on mobile devices) can be addressed to the fact that the attack scenario as well as the attacker have changed significantly. More specifically, side-channel attacks against smart cards have been conducted to reveal sensitive information that should be protected from being accessed by benign users. For example, in case of pay-TV cards the secret keys must be protected against benign users, *i.e.*, users who bought these pay-TV cards in the first place. The attacker in

this case might be willing to invest in equipment in order to reveal the secret key as this key could be sold later on.

In contrast, today's smartphones are used to store and process sensitive information, and attackers interested in this information are usually not the users themselves but rather criminals, imposters, and other malicious entities that aim to steal this sensitive information from users. Especially the appification of the mobile ecosystem provides tremendous opportunities for attackers to exploit identified side-channel leaks via software-only attacks. Hence, this shift also significantly increases the scale at which attacks are conducted. While local attacks only target a few devices, remote attacks can be conducted on millions of devices at the same time by distributing software via available app markets.

From Active to Passive Attacks. The second trend that can be observed is that fault injection attacks have been quite popular on smart cards, whereas such (local) fault attacks are not that widely investigated on smartphones, at least at the moment. Consequently, we also observe that the variety of fault attacks conducted in the smart card era has decreased significantly in the smartphone era, which can be addressed to the following observations. First, the targeted device itself, e.g., a smartphone, is far more expensive than a smart card and, hence, fault attacks that potentially permanently break the device are only acceptable for very targeted attacks. Even in case of highly targeted attacks (cf. Apple vs FBI dispute), zero-day vulnerabilities might be chosen instead of local fault attacks.[4] Second, remote fault attacks seem to be harder to conduct as such faults are harder to induce via software execution. Currently, the only remote fault attack (also known as software-induced fault attack) is the Rowhammer attack, which however gets increasing attention among the scientific community and has already been exploited to gain root access on Android devices [140]. Although software-induced fault attacks have not been investigated extensively in the past, we expect further research to be conducted in this context.

Some microarchitectural attacks can also be considered as active attacks because the attacker influences the behavior of the targeted program (victim). For example, cache attacks can be used to slow down the execution of the victim due to cache contention. However, this does not introduce a fault in the computation and, hence, Rowhammer currently represents the only software-induced fault attack.

Exploiting Physical and Logical Properties. In contrast to physical properties, logical properties (software features) do not result from any physical interaction with the device, but due to dedicated features provided via software. While traditional side-channel attacks mostly exploited physical properties and required dedicated equipment, more recent side-channel attacks exploit physical properties as well as logical properties. Interestingly, the immense number of sensors in smartphones also allows to exploit physical properties by means of software-only attacks, which was not possible on smart cards. Although the majority of attacks on mobile devices still exploits physical properties, the exploitation of logical properties also receives increasing attention. Especially the procfs seems to provide an almost inexhaustible source for possible information leaks. For example, the memory footprint released via the procfs has been used to infer visited websites [25], or the number of context switches has been used to infer swipe input [11]. Besides, information that is available via official APIs is in some cases also available via the procfs such as the data-usage statistics that have been exploited to infer a user's identity [14] and to infer visited websites [26].

Empty Areas. As can be observed, a few areas in this categorization system (cf. Figure 7) are not (yet) covered or are not covered that densely. For instance, there is currently no active side-channel attack that can be executed remotely and that exploits logical properties (software features) to induce faults or to actively influence the targeted program (victim). However, by considering existing passive attacks, one could come up with more advanced attacks by introducing an active attacker. Such an active attacker might, for example, block/influence a shared resource in order to cause malfunctioning of the target. For instance, considering the passive attack exploiting the speaker status (on/off) to infer a user's driving routes [14], one could easily influence the victim application by playing inaudible sounds in the right moment in order to prevent the turn-by-turn voice guidance from accessing the speaker. Thereby, the active attacker prevents the target (victim) from accessing the shared resource, *i.e.*, the speaker, and based on this induced behavior an active attacker might gain an advantage compared to a passive attacker. We expect advances in this (yet) uncovered area of active side-channel attacks that target software features.

Tabular Summary of Surveyed Attacks. Table V provides a tabular summary for the categorization of the surveyed attacks. For some attacks we observe that active as well as passive modes of attack have already been considered, e.g., differential computation analysis and network traffic analysis attacks. Some attacks can also be conducted by exploiting physical properties as well as logical properties, e.g., the fingerprinting of devices and network traffic analysis attacks.

VIII. Discussion of Countermeasures

In this section, we discuss existing countermeasures against the most prominent attacks. Overall we aim to shed light onto possible pitfalls of existing countermeasures and to stimulate future research to come up with more generic countermeasures against side-channel attacks.

A. Local Side-Channel Attacks

Protecting Cryptographic Implementations. Cryptographic implementations represent a prominent target of side-channel attacks as a successful attack allows to recover sensitive data and to break mechanisms building upon these primitives. Therefore, countermeasures to protect cryptographic implementations have already been proposed for the smart card world. These countermeasures can be applied to protect cryptographic implementations on smartphones as well. For

[4]However, in September 2016 Skorobogatov [80] demonstrated that NAND mirroring allows to bypass the PIN entry limit on the iPhone 5c.

TABLE V
SUMMARY OF SURVEYED ATTACKS.

Attack	Mode of attack		Exploited information		Location of attacker		
	Active	Passive	Physical properties	Logical properties	Local	Vicinity	Remote
Power analysis		✓	✓		✓		
EM analysis		✓	✓		✓	✓	
NAND mirroring	✓		✓		✓		
Laser/optical	✓		✓		✓		
Clock/power glitch	✓		✓		✓		
Temperature variation	✓		✓		✓		
EMFI	✓		✓		✓		
Differential computation analysis	✓	✓		✓	✓		
Reflections/hands		✓	✓		✓	✓	
Smudges		✓	✓		✓		
Network traffic analysis	✓	✓	✓	✓		✓	
USB power analysis		✓	✓			✓	
Wi-Fi signal monitoring		✓	✓			✓	
Fingerprinting devices		✓	✓	✓			✓
Data-usage statistics		✓		✓			✓
Page deduplication		✓		✓			✓
procfs leaks		✓	✓	✓			✓
Microarchitectural attacks	✓	✓	✓				✓
Location inference		✓	✓				✓
Speech recognition		✓	✓				✓
Soundcomber		✓	✓				✓
Sensor-based keyloggers		✓	✓				✓
Rowhammer	✓		✓				✓

example, masking of sensitive values such as the randomization of key-dependent values during cryptographic operations, or execution randomization are countermeasures for hardening the implementation against passive attacks such as power analysis or EM analysis [4]. Executing critical calculations twice allows to detect faults that are injected during an active side-channel attack [148].

Protecting User Input. Mitigation techniques to prevent attackers from inferring user input on touchscreens by means of smudge attacks or shoulder surfing attacks are not that thoroughly investigated yet. Nevertheless, proposed countermeasures include, for example, randomly starting the vibrator to prevent attacks that monitor the backside of the device [66], or to randomize the layout of the soft keyboard each time the user provides input to prevent smudge attacks [118] as well as attacks that monitor the hand movement [65]. Aviv [118] also proposed to align PIN digits in the middle of the screen and after each authentication the user needs to swipe down across all digits in order to hide smudges. Besides, Kwon and Na [149] introduced a new authentication mechanism denoted as *TinyLock* that should prevent smudge attacks against pattern unlock mechanisms. Krombholz et al. [150] proposed an authentication mechanism for devices with pressure-sensitive screens that should prevent smudge attacks and shoulder surfing attacks. Raguram et al. [62], [63] suggested to decrease the screen brightness, to disable visual feedback (e.g., pop-up characters) on soft keyboards, and to use anti-reflective coating in eyeglasses to prevent attackers from exploiting reflections.

B. Vicinity Side-Channel Attacks

Preventing Network Traffic Analysis. Countermeasures to prevent attackers from applying traffic analysis techniques on wiretapped network connections have been extensively considered in the context of website fingerprinting attacks. The main idea of these obfuscation techniques is to hide information that allows attackers to uniquely identify communication partners or transmitted content such as visited websites. Proposed countermeasures [151]–[155], however, require the application, e.g., the browser application, as well as the remote server to cooperate. Furthermore, it has already been pointed out in [26] that these countermeasures add overhead in terms of bandwidth and data consumption which might not be acceptable in case of mobile devices with limited data plans.

C. Remote Side-Channel Attacks

Permissions. The most straight-forward approach always discussed as a viable means to prevent specific types of software-only side-channel attacks is to protect the exploited information or resource by means of dedicated permissions. However, there is a study [156] that showed that permission-based approaches are not quite convincing. Some users do not understand the exact meaning of specific permissions, and others do not care about requested permissions. Acar et al. [15] even attested that the Android permission system "has failed in practice". Despite these problems it seems to be nearly impossible to add dedicated permissions for every exploited information.

Keyboard Layout Randomization. In order to prevent sensor-based keylogging attacks that exploit the correlation between user input and the device movements observed via sensor readings, the keyboard layout of soft keyboards could be randomized [117]. For instance, the Android-based CyanogenMod OS allows to enable such a feature for PIN inputs optionally. However, it remains an open question how this would affect usability in case of QWERTY keyboards and, intuitively, it might make keyboard input nearly impossible.

Limiting Access or Sampling Frequency. It has also been suggested to disable access to sensor readings during sensitive input or to reduce the sampling frequency of sensors. This, however, would hinder applications that heavily rely on sensor readings such as pedometers.

Side-channel attacks like *Soundcomber* might be prevented by *AuDroid* [157], which is an extension to the SELinux reference monitor that has been integrated into Android to control access to system audio resources. As pointed out by the authors, there is no security mechanism in place for the host OS to control access to mobile device speakers, thus allowing untrusted apps to exploit this communication channel. *AuDroid* enforces security policies that prevent data in system apps and services from being leaked to (or used by) untrusted parties.

Noise Injection. Randomly starting the phone vibrator has been suggested by Owusu et al. [117] to prevent sensor-based keyloggers that exploit the accelerometer sensor. However, Shrestha et al. [158] showed that random vibrations do not provide protection. As an alternative, Shrestha et al. proposed a tool named *Slogger* that introduces noise into sensor readings as soon as the soft keyboard is running. In order to do so, Slogger relies on a tool that needs to be started via the ADB shell (in order to be executed with ADB capabilities). Slogger injects events into the files corresponding to the accelerometer and the gyroscope located in `/dev/input/`, which is why ADB privileges are required for this defense mechanism. The authors even evaluated the effectiveness of Slogger against two sensor-based keyloggers and found that the accuracy of sensor-based keyloggers can be reduced significantly. Das et al. [131] also suggested to add noise to sensor readings in order to prevent device fingerprinting via hardware imperfections of sensors. A more general approach that targets the injection of noise into the information provided via the procfs has been proposed by Xiao et al. [46].

Preventing Microarchitectural Attacks. The inherent nature of modern computer architectures enables sophisticated attacks due to shared resources and especially due to dedicated performance optimization techniques. A famous and popular example is the memory hierarchy that introduces significant performance gains but also enables microarchitectural attacks such as cache attacks. Although specific cryptographic implementations can be protected against such attacks, e.g., bitsliced implementations [159], [160] or dedicated hardware instructions can be used to protect AES implementations, generic countermeasures against cache attacks represent a non-trivial challenge. However, we consider it of utmost importance to spur further research in the context of countermeasures, especially since cache attacks do not only pose a risk for cryptographic algorithms, but also for other sensitive information such as keystrokes [23], [113].

App Guardian. Most of the above presented countermeasures aim to prevent very specific attacks only, but cannot be applied to prevent attacks within a specific category of our classification system, e.g., software-only attacks located in the upper right of our new classification system (cf. Figure 7). At least some of these attacks, however, have been addressed by App Guardian [16], which represents a more general approach to defend against software-only attacks. App Guardian is a third-party application that runs in user mode and employs side-channel information to detect RIG attacks (including software-only side-channel attacks). The basic idea of App Guardian is to stop the malicious application while the principal (the app to be protected) is being executed and to resume the (potentially malicious) application later on. Although App Guardian still faces challenges, it is a novel idea to cope with such side-channel attacks in general. More specifically, it tries to cope with all passive attacks that require the attacker to execute software on the targeted device.

App Guardian seems to be a promising research project to cope with side-channel attacks on smartphones at a larger scale. However, an unsolved issue of App Guardian is the problem that it still struggles with the proper identification of applications to be protected. Furthermore, App Guardian relies on side-channel information—to detect ongoing side-channel attacks—that has been removed in Android 7. Hence, App Guardian needs to be updated in order to also work on recent Android versions and its effectiveness should be further evaluated against existing side-channel attacks. Furthermore, it might be interesting to extend its current framework to cope with side-channel attacks conducted from within the browser, *i.e.*, to mitigate side-channel attacks via JavaScript.

D. Summary

Although local attacks target only a few devices or users, we also observe that we require a much broader range of countermeasures because also the attack methodologies of local attacks are much broader. For instance, we have to deal with attackers that measure the power consumption of the device in order to break cryptographic implementations, we have to deal with fault attacks such as clock/power glitching and temperature variations, and at the same time we have to deal with attackers that exploit smudges left on the touchscreen.

In contrast, the commonality of all remote attacks is that they require software execution on the targeted device. Although this means that remote attacks target devices and users at a much broader scale, more generic countermeasures such as App Guardian seem to be the most promising approach to cope with these attacks in the future.

IX. Issues, Challenges, and Future Research

In this section we discuss open issues and challenges that need to be addressed in future research. Hence, this section is not meant to provide solutions to existing problems. Instead, with the presented classification system for modern side-channel attacks we aim to shed light onto this vivid research area and, thereby, to point out high-level research directions. Overall, the ultimate goal is to spur further research in the context of side-channel attacks and countermeasures and, as a result, to pave the way for a more secure computing platform for smart and mobile devices.

Countermeasures. Side-channel attacks are published at an unprecedented pace and appropriate defense mechanisms

are often either not (yet) available or cannot be deployed easily. Especially the five *key enablers* identified in this paper enable devastating side-channel attacks that can be conducted remotely and, thus, target an immense number of devices and users at the same time. Although countermeasures are being researched, we observe a cat and mouse game between attackers and system engineers trying to make systems secure from a side-channel perspective. Besides, even if effective countermeasures were readily available, the mobile ecosystem of Android would impede a large-scale deployment of many of these defense mechanisms. The main problem is that even in case Google would apply defense mechanisms and patch these information leaks, multiple device manufacturers as well as carriers also need to apply these patches to deploy countermeasures in practice. Hence, chances are that such countermeasures will never be deployed, especially not in case of outdated operating systems. We hope to stimulate research to come up with viable countermeasures in order to prevent such attacks at a larger scale, *i.e.*, by considering larger areas within the new categorization system, while also considering the challenges faced by the mobile ecosystem. For instance, App Guardian [16] follows the right direction by trying to cope with attacks at a larger scale, while at the same time it can be deployed as a third-party application.

Reproducibility and Responsible Disclosure. In order to foster research in the context of countermeasures, it would be helpful to publish the corresponding frameworks used to conduct side-channel attacks. While this might also address the long-standing problem of reproducibility of experiments in computer science in general, this would enable a more efficient evaluation of developed countermeasures. At the same time, however, responsible disclosure must be upheld, which sometimes turns out to be a difficult balancing act. On the one hand, researchers want to publish their findings as soon as possible and on the other hand, putting countermeasures to practice might take some time.

Different Mobile Operating Systems and Cross-Platform Development. Research should not only focus on one particular OS exclusively, *i.e.*, especially Android seems to attract the most attention. Instead, the applicability of side-channel attacks should be investigated on multiple platforms, as many (or most) of the existing attacks work on other platforms as well. This is due to the fact that different platforms and devices from different vendors aim to provide the same features such as sensors and software interfaces, and rely on similar security concepts like permission systems and application sandboxing.

In addition, the increasing trend to develop applications for multiple platforms (cross-platform development) also increases the possibility to target multiple platforms at the same time. For example, the increasing popularity of HTML5 apps and the increasing availability of web APIs to access native resources from JavaScript significantly increases the scale of side-channel attacks as specific attacks possibly target multiple platforms at the same time.

Wearables. Although we put a strong focus on smartphones in this paper, we stress that wearables in general must be considered in future research. For example, smartwatches have already been employed to attack user input on POS terminals and hardware QWERTY keyboards [161]–[164]. Besides, it has also been demonstrated that smartwatches can be used to infer input on smartphones [165], [166] as well as text written on whiteboards [167]. With the ever increasing number of smart devices connected to our everyday lives, the threat of side-channel attacks increases. We are likely to see higher accuracies when these attacks are performed across multiple devices, e.g., when combining data from smartwatches and smartphones. Furthermore, Farshteindiker et al. [168] also demonstrated how hardware implants (bugs)—possibly used by intelligence agencies—can be used to exfiltrate data by communicating with a smartphone. The communication channel is based on inaudible sounds emitted from the implant which are captured by the gyroscope of the smartphone. This interconnection clearly demonstrates the potential of attacks when multiple wearable devices are combined.

Internet of Things. Another area of research which is rapidly growing is the Internet of Things (IoT). As all devices in the IoT network are inter-connected and accessible via the Internet, we foresee that attackers will exploit side-channel leaks to target different kinds of IoT appliances. In fact such an attack has already been carried out by Zhang et al. [16]. They investigated an Android-based Wi-Fi camera and observed that a particular side-channel leak on Android can be exploited to infer whether or not the user is at home. This example demonstrates that side-channel leaks do not only pose a threat to a user's privacy and security from a system security point of view, but also pose a threat to smart home appliances and security systems, such as smart thermostats, cameras, and alarm systems. Although this sounds utopian at first, the above example clearly demonstrates that side-channel leaks (on smartphones) also pose a threat to these IoT appliances and puts even users' physical possessions at risk.

Combination of Multiple Information Leaks. In order to improve the accuracy of existing attacks or to come up with more sophisticated attack scenarios, multiple side-channel leaks can also be combined. For instance, the combination of cache attacks and sensor-based keyloggers as mentioned in [113] could be used to improve keylogging attacks. First, cache attacks can be used to determine the exact time when a key is entered and, second, sensor-based keyloggers can be used to infer the actual key. Furthermore, website fingerprinting attacks could be combined with sensor-based keyloggers as mentioned in [26], which would allow to steal login credentials for specific websites.

In addition, side-channel attacks can also be used to improve attacks that exploit software vulnerabilities. For example, although Screenmilker [47] does not represent a side-channel attack—because a software vulnerability is exploited—it relies on side-channel information in order to exploit this vulnerability in the right moment. Lin et al. [47] suggested to rely on CPU utilization, memory consumption, and network activities in order to determine whether the targeted app is executed and, thus, were able to take screenshots in the right moment.

Code Analysis Tools. The appification of mobile devices

enables an easy installation of apps from the app markets. However, these apps can be implemented by anyone who has a developer account and, thus, the code needs to be checked and verified appropriately, *i.e.*, for presence of malicious behavior and side channels. While the app vetting processes of app stores, e.g., Google Play, already check for presence of malicious behavior, dedicated technologies, such as static and dynamic code analysis, should also be employed in order to prevent apps prone to side-channel attacks and apps exploiting side-channel information leaks from being distributed via app markets. This, however, does not seem to be a trivial task since most side-channel attacks exploit information or resources that can be accessed without any specific privileges or permissions.

Static and dynamic code analysis tools could also help to fix implementation flaws that lead to side-channel attacks. Some implementation flaws exist for many years without being noticed as has been demonstrated in [169] for the OpenSSL implementation of the digital signature algorithm. Fostering the development and application of tools to find and detect such flaws during the software development process could help to prevent vulnerable code from being deployed.

A possible starting point for the investigation and extension of code analysis tools that might allow to scan applications for possible side-channel attacks would be one of the survey papers discussed in Section II-C.

X. Conclusion

Considering the immense threat arising from side-channel attacks on mobile devices, a thorough understanding of information leaks and possible exploitation techniques is necessary. Based on this open issue, we surveyed existing side-channel attacks and identified commonalities between these attacks in order to systematically categorize all existing attacks. With the presented classification system we aim to provide a thorough understanding of information leaks and hope to spur further research in the context of side-channel attacks as well as countermeasures and, thereby, to pave the way for secure computing platforms.

Acknowledgment

The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 644052 (HECTOR), and the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 681402). Veelasha Moonsamy has been supported by the Technology Foundation STW (project 13499 - TYPHOON & ASPASIA) from the Dutch government. Further, we would like to thank Florian Mendel for helpful discussions about active side-channel attacks as well as Cristofaro Mune and Nikita Abdullin for pointing out a missing attack category.

References

[1] P. C. Kocher, "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems," in *Advances in Cryptology – CRYPTO 1996*, ser. LNCS, vol. 1109. Springer, 1996, pp. 104–113.

[2] P. C. Kocher, J. Jaffe, and B. Jun, "Differential Power Analysis," in *Advances in Cryptology – CRYPTO 1999*, ser. LNCS, vol. 1666. Springer, 1999, pp. 388–397.

[3] J. Quisquater and D. Samyde, "ElectroMagnetic Analysis (EMA): Measures and Counter-Measures for Smart Cards," in *Smart Card Programming and Security – E-smart 2001*, ser. LNCS, vol. 2140. Springer, 2001, pp. 200–210.

[4] S. Mangard, E. Oswald, and T. Popp, *Power Analysis Attacks - Revealing the Secrets of Smart Cards*. Springer, 2007.

[5] E. Tromer, D. A. Osvik, and A. Shamir, "Efficient Cache Attacks on AES, and Countermeasures," *J. Cryptology*, vol. 23, pp. 37–71, 2010.

[6] Y. Yarom and K. Falkner, "FLUSH+RELOAD: A High Resolution, Low Noise, L3 Cache Side-Channel Attack," in *USENIX Security Symposium 2014*. USENIX Association, 2014, pp. 719–732.

[7] Q. Ge, Y. Yarom, D. Cock, and G. Heiser, "A Survey of Microarchitectural Timing Attacks and Countermeasures on Contemporary Hardware," *Journal of Cryptographic Engineering*, pp. 1–27, 2016.

[8] P. Pessl, D. Gruss, C. Maurice, M. Schwarz, and S. Mangard, "DRAMA: Exploiting DRAM Addressing for Cross-CPU Attacks," in *USENIX Security Symposium 2016*. USENIX Association, 2016, pp. 565–581.

[9] L. Cai and H. Chen, "TouchLogger: Inferring Keystrokes on Touch Screen from Smartphone Motion," in *USENIX Workshop on Hot Topics in Security – HotSec*. USENIX Association, 2011.

[10] A. J. Aviv, B. Sapp, M. Blaze, and J. M. Smith, "Practicality of Accelerometer Side Channels on Smartphones," in *Annual Computer Security Applications Conference – ACSAC 2012*. ACM, 2012, pp. 41–50.

[11] L. Simon, W. Xu, and R. Anderson, "Don't Interrupt Me While I Type: Inferring Text Entered Through Gesture Typing on Android Keyboards," *PoPETs*, vol. 2016, pp. 136–154, 2016.

[12] M. Mehrnezhad, E. Toreini, S. F. Shahandashti, and F. Hao, "TouchSignatures: Identification of User Touch Actions and PINs Based on Mobile Sensor Data via JavaScript," *J. Inf. Sec. Appl.*, vol. 26, pp. 23–38, 2016.

[13] Y. Michalevsky, A. Schulman, G. A. Veerapandian, D. Boneh, and G. Nakibly, "PowerSpy: Location Tracking Using Mobile Device Power Analysis," in *USENIX Security Symposium 2015*. USENIX Association, 2015, pp. 785–800.

[14] X. Zhou, S. Demetriou, D. He, M. Naveed, X. Pan, X. Wang, C. A. Gunter, and K. Nahrstedt, "Identity, Location, Disease and More: Inferring Your Secrets From Android Public Resources," in *Conference on Computer and Communications Security – CCS 2013*. ACM, 2013, pp. 1017–1028.

[15] Y. Acar, M. Backes, S. Bugiel, S. Fahl, P. D. McDaniel, and M. Smith, "SoK: Lessons Learned from Android Security Research for Appified Software Platforms," in *IEEE Symposium on Security and Privacy – S&P 2016*. IEEE, 2016, pp. 433–451.

[16] N. Zhang, K. Yuan, M. Naveed, X. Zhou, and X. Wang, "Leave Me Alone: App-Level Protection against Runtime Information Gathering on Android," in *IEEE Symposium on Security and Privacy – S&P 2015*. IEEE Computer Society, 2015, pp. 915–930.

[17] P. Marquardt, A. Verma, H. Carter, and P. Traynor, "(sp)iPhone: Decoding Vibrations From Nearby Keyboards Using Mobile Phone Accelerometers," in *Conference on Computer and Communications Security – CCS 2011*. ACM, 2011, pp. 551–562.

[18] T. Zhu, Q. Ma, S. Zhang, and Y. Liu, "Context-free Attacks Using Keyboard Acoustic Emanations," in *Conference on Computer and Communications Security – CCS 2014*. ACM, 2014, pp. 453–464.

[19] S. Biedermann, S. Katzenbeisser, and J. Szefer, "Hard Drive Side-Channel Attacks Using Smartphone Magnetic Field Sensors," in *Financial Cryptography – FC 2015*, ser. LNCS, vol. 8975. Springer, 2015, pp. 489–496.

[20] L. Schwittmann, V. Matkovic, M. Wander, and T. Weis, "Video Recognition Using Ambient Light Sensors," in *Pervasive Computing and Communication Workshops – PerCom 2016*. IEEE Computer Society, 2016, pp. 1–9.

[21] C. Song, F. Lin, Z. Ba, K. Ren, C. Zhou, and W. Xu, "My Smartphone Knows What You Print: Exploring Smartphone-based Side-channel Attacks Against 3D Printers," in *Conference on Computer and Communications Security – CCS 2016*. ACM, 2016, pp. 895–907.

[22] A. Hojjati, A. Adhikari, K. Struckmann, E. Chou, T. N. T. Nguyen, K. Madan, M. S. Winslett, C. A. Gunter, and W. P. King, "Leave Your Phone at the Door: Side Channels that Reveal Factory Floor Secrets," in *Conference on Computer and Communications Security – CCS 2016*. ACM, 2016, pp. 883–894.

[23] D. Gruss, R. Spreitzer, and S. Mangard, "Cache Template Attacks: Automating Attacks on Inclusive Last-Level Caches," in *USENIX Security Symposium 2015*. USENIX Association, 2015, pp. 897–912.

[24] D. Gruss, D. Bidner, and S. Mangard, "Practical Memory Deduplication Attacks in Sandboxed Javascript," in *European Symposium on Research in Computer Security – ESORICS 2015*, ser. LNCS, vol. 9327. Springer, 2015, pp. 108–122.

[25] S. Jana and V. Shmatikov, "Memento: Learning Secrets from Process Footprints," in *IEEE Symposium on Security and Privacy – S&P 2012*. IEEE Computer Society, 2012, pp. 143–157.

[26] R. Spreitzer, S. Griesmayr, T. Korak, and S. Mangard, "Exploiting Data-Usage Statistics for Website Fingerprinting Attacks on Android," in *Security and Privacy in Wireless and Mobile Networks – WISEC 2016*. ACM, 2016, pp. 49–60.

[27] Gartner, "Global Market Share Held by the Leading Smartphone Operating Systems in Sales to End Users From 1st Quarter 2009 to 1st Quarter 2017," https://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/, accessed: 2017-06-13.

[28] A. D. Luzio, A. Mei, and J. Stefa, "Mind Your Probes: De-Anonymization of Large Crowds Through Smartphone WiFi Probe Requests," in *IEEE INFOCOM 2016*. IEEE, 2016, pp. 1–9.

[29] R. Spolaor, L. Abudahi, V. Moonsamy, M. Conti, and R. Poovendran, "No Free Charge Theorem: A Covert Channel via USB Charging Cable on Mobile Devices," in *Applied Cryptography and Network Security – ACNS 2017*. Springer, 2017, in press.

[30] W. Enck, "Defending Users Against Smartphone Apps: Techniques and Future Directions," in *Information Systems Security – ICISS 2011*, ser. LNCS, vol. 7093. Springer, 2011, pp. 49–70.

[31] M. L. Polla, F. Martinelli, and D. Sgandurra, "A Survey on Security for Mobile Devices," *IEEE Communications Surveys and Tutorials*, vol. 15, pp. 446–471, 2013.

[32] G. Suarez-Tangil, J. E. Tapiador, P. Peris-Lopez, and A. Ribagorda, "Evolution, Detection and Analysis of Malware for Smart Devices," *IEEE Communications Surveys and Tutorials*, vol. 16, pp. 961–987, 2014.

[33] P. Faruki, A. Bharmal, V. Laxmi, V. Ganmoor, M. S. Gaur, M. Conti, and M. Rajarajan, "Android Security: A Survey of Issues, Malware Penetration, and Defenses," *IEEE Communications Surveys and Tutorials*, vol. 17, pp. 998–1022, 2015.

[34] B. Rashidi and C. J. Fung, "A Survey of Android Security Threats and Defenses," *JoWUA*, vol. 6, pp. 3–35, 2015.

[35] A. Sadeghi, H. Bagheri, J. Garcia, and S. Malek, "A Taxonomy and Qualitative Comparison of Program Analysis Techniques for Security Assessment of Android Software," *IEEE Transactions on Software Engineering*, vol. PP, no. 99, pp. 1–48, 2017.

[36] K. Tam, A. Feizollah, N. B. Anuar, R. Salleh, and L. Cavallaro, "The Evolution of Android Malware and Android Analysis Techniques," *ACM Comput. Surv.*, vol. 49, pp. 76:1–76:41, 2017.

[37] M. Tunstall, *Smart Card Security*. Cham: Springer International Publishing, 2017, pp. 217–251.

[38] S. Zander, G. J. Armitage, and P. Branch, "A Survey of Covert Channels and Countermeasures in Computer Network Protocols," *IEEE Communications Surveys and Tutorials*, vol. 9, pp. 44–57, 2007.

[39] A. K. Biswas, D. Ghosal, and S. Nagaraja, "A Survey of Timing Channels and Countermeasures," *ACM Comput. Surv.*, vol. 50, pp. 6:1–6:39, 2017.

[40] J. Szefer, "Survey of Microarchitectural Side and Covert Channels, Attacks, and Defenses," *IACR Cryptology ePrint Archive, Report 2016/479*, 2016.

[41] J. Ullrich, T. Zseby, J. Fabini, and E. R. Weippl, "Network-Based Secret Communication in Clouds: A Survey," *IEEE Communications Surveys and Tutorials*, vol. 19, pp. 1112–1144, 2017.

[42] J. Betz, D. Westhoff, and G. Müller, "Survey on Covert Channels in Virtual Machines and Cloud Computing," *Trans. Emerging Telecommunications Technologies*, vol. 28, 2017.

[43] M. Xu, C. Song, Y. Ji, M. Shih, K. Lu, C. Zheng, R. Duan, Y. Jang, B. Lee, C. Qian, S. Lee, and T. Kim, "Toward Engineering a Secure Android Ecosystem: A Survey of Existing Techniques," *ACM Comput. Surv.*, vol. 49, pp. 38:1–38:47, 2016.

[44] M. Hussain, A. Al-Haiqi, A. A. Zaidan, B. B. Zaidan, M. L. M. Kiah, N. B. Anuar, and M. Abdulnabi, "The Rise of Keyloggers on Smartphones: A Survey and Insight Into Motion-Based Tap Inference Attacks," *Pervasive and Mobile Computing*, vol. 25, pp. 1–25, 2016.

[45] A. Nahapetian, "Side-Channel Attacks on Mobile and Wearable Systems," in *Consumer Communications & Networking Conference – CCNC 2016*. IEEE, 2016, pp. 243–247.

[46] Q. Xiao, M. K. Reiter, and Y. Zhang, "Mitigating Storage Side Channels Using Statistical Privacy Mechanisms," in *Conference on Computer and Communications Security – CCS 2015*. ACM, 2015, pp. 1582–1594.

[47] C. Lin, H. Li, X. Zhou, and X. Wang, "Screenmilker: How to Milk Your Android Screen for Secrets," in *Network and Distributed System Security Symposium – NDSS 2014*. The Internet Society, 2014.

[48] T. S. Messerges and E. A. Dabbish, "Investigations of Power Analysis Attacks on Smartcards," in *Workshop on Smartcard Technology – Smartcard 1999*. USENIX Association, 1999.

[49] L. Yan, Y. Guo, X. Chen, and H. Mei, "A Study on Power Side Channels on Mobile Devices," in *Symposium of Internetware – Internetware 2015*. ACM, 2015, pp. 30–38.

[50] C. H. Gebotys, S. Ho, and C. C. Tiu, "EM Analysis of Rijndael and ECC on a Wireless Java-Based PDA," in *Cryptographic Hardware and Embedded Systems – CHES 2005*, ser. LNCS, vol. 3659. Springer, 2005, pp. 250–264.

[51] Y. Nakano, Y. Souissi, R. Nguyen, L. Sauvage, J. Danger, S. Guilley, S. Kiyomoto, and Y. Miyake, "A Pre-processing Composition for Secret Key Recovery on Android Smartphone," in *Information Security Theory and Practice – WISTP 2014*, ser. LNCS, vol. 8501. Springer, 2014, pp. 76–91.

[52] G. Goller and G. Sigl, "Side Channel Attacks on Smartphones and Embedded Devices Using Standard Radio Equipment," in *Constructive Side-Channel Analysis and Secure Design – COSADE 2015*, ser. LNCS, vol. 9064. Springer, 2015, pp. 255–270.

[53] P. Belgarric, P. Fouque, G. Macario-Rat, and M. Tibouchi, "Side-Channel Analysis of Weierstrass and Koblitz Curve ECDSA on Android Smartphones," in *Topics in Cryptology – CT-RSA 2016*, ser. LNCS, vol. 9610. Springer, 2016, pp. 236–252.

[54] D. Genkin, L. Pachmanov, I. Pipman, E. Tromer, and Y. Yarom, "ECDSA Key Extraction from Mobile Devices via Nonintrusive Physical Side Channels," in *Conference on Computer and Communications Security – CCS 2016*. ACM, 2016, pp. 1626–1638.

[55] J. W. Bos, C. Hubain, W. Michiels, and P. Teuwen, "Differential Computation Analysis: Hiding Your White-Box Designs is Not Enough," in *Cryptographic Hardware and Embedded Systems – CHES 2016*, ser. LNCS, vol. 9813. Springer, 2016, pp. 215–236.

[56] A. J. Aviv, K. L. Gibson, E. Mossop, M. Blaze, and J. M. Smith, "Smudge Attacks on Smartphone Touch Screens," in *Workshop on Offensive Technologies – WOOT 2010*. USENIX Association, 2010.

[57] Y. Zhang, P. Xia, J. Luo, Z. Ling, B. Liu, and X. Fu, "Fingerprint Attack Against Touch-Enabled Devices," in *Security and Privacy in Smartphones & Mobile Devices – SPSM@CCS*. ACM, 2012, pp. 57–68.

[58] P. Andriotis, T. Tryfonas, G. C. Oikonomou, and C. Yildiz, "A Pilot Study on the Security of Pattern Screen-Lock Methods and Soft Side Channel Attacks," in *Security and Privacy in Wireless and Mobile Networks – WISEC 2013*. ACM, 2013, pp. 1–6.

[59] M. Backes, M. Dürmuth, and D. Unruh, "Compromising Reflections-or-How to Read LCD Monitors around the Corner," in *IEEE Symposium on Security and Privacy – S&P 2008*. IEEE Computer Society, 2008, pp. 158–169.

[60] M. Backes, T. Chen, M. Dürmuth, H. P. A. Lensch, and M. Welk, "Tempest in a Teapot: Compromising Reflections Revisited," in *IEEE Symposium on Security and Privacy – S&P 2009*. IEEE Computer Society, 2009, pp. 315–327.

[61] F. Maggi, S. Gasparini, and G. Boracchi, "A Fast Eavesdropping Attack Against Touchscreens," in *Information Assurance and Security – IAS 2011*. IEEE, 2011, pp. 320–325.

[62] R. Raguram, A. M. White, D. Goswami, F. Monrose, and J. Frahm, "iSpy: Automatic Reconstruction of Typed Input from Compromising Reflections," in *Conference on Computer and Communications Security – CCS 2011*. ACM, 2011, pp. 527–536.

[63] R. Raguram, A. M. White, Y. Xu, J. Frahm, P. Georgel, and F. Monrose, "On the Privacy Risks of Virtual Keyboards: Automatic Reconstruction of Typed Input from Compromising Reflections," *IEEE Trans. Dependable Sec. Comput.*, vol. 10, pp. 154–167, 2013.

[64] Y. Xu, J. Heinly, A. M. White, F. Monrose, and J. Frahm, "Seeing Double: Reconstructing Obscured Typed Input from Repeated Compromising Reflections," in *Conference on Computer and Communications Security – CCS 2013*. ACM, 2013, pp. 1063–1074.

[65] D. Shukla, R. Kumar, A. Serwadda, and V. V. Phoha, "Beware, Your Hands Reveal Your Secrets!" in *Conference on Computer and Communications Security – CCS 2014*. ACM, 2014, pp. 904–917.

[66] J. Sun, X. Jin, Y. Chen, J. Zhang, Y. Zhang, and R. Zhang, "VISIBLE: Video-Assisted Keystroke Inference from Tablet Backside Motion," in *Network and Distributed System Security Symposium – NDSS 2016*. The Internet Society, 2016.

[67] Q. Yue, Z. Ling, X. Fu, B. Liu, K. Ren, and W. Zhao, "Blind Recognition of Touched Keys on Mobile Devices," in *Conference on Computer and Communications Security – CCS 2014*. ACM, 2014, pp. 1403–1414.

[68] D. Boneh, R. A. DeMillo, and R. J. Lipton, "On the Importance of Checking Cryptographic Protocols for Faults (Extended Abstract)," in *Advances in Cryptology – EUROCRYPT 1997*, ser. LNCS, vol. 1233. Springer, 1997, pp. 37–51.

[69] E. Biham and A. Shamir, "Differential Fault Analysis of Secret Key Cryptosystems," in *Advances in Cryptology – CRYPTO 1997*, ser. LNCS, vol. 1294. Springer, 1997, pp. 513–525.

[70] NewAE Technology Inc., "Fault Injection Raspberry PI," https://wiki.newae.com, accessed: 2016-08-03.

[71] C. O'Flynn, "Fault Injection using Crowbars on Embedded Systems," *IACR Cryptology ePrint Archive, Report 2016/810*, 2016.

[72] S. Ordas, L. Guillaume-Sage, and P. Maurine, "Electromagnetic Fault Injection: The Curse of Flip-Flops," *Journal of Cryptographic Engineering*, pp. 1–15, 2016.

[73] L. Rivière, Z. Najm, P. Rauzy, J. Danger, J. Bringer, and L. Sauvage, "High Precision Fault Injections on the Instruction Cache of ARMv7-M Architectures," in *Hardware Oriented Security and Trust – HOST 2015*. IEEE Computer Society, 2015, pp. 62–67.

[74] E. Sanfelix, C. Mune, and J. de Haas, "Unboxing the White-Box: Practical Attacks Against Obfuscated Ciphers," Blackhat 2015.

[75] S. P. Skorobogatov and R. J. Anderson, "Optical Fault Induction Attacks," in *Cryptographic Hardware and Embedded Systems – CHES 2002*, ser. LNCS, vol. 2523. Springer, 2002, pp. 2–12.

[76] J. G. J. van Woudenberg, M. F. Witteman, and F. Menarini, "Practical Optical Fault Injection on Secure Microcontrollers," in *Fault Diagnosis and Tolerance in Cryptography – FDTC 2011*. IEEE Computer Society, 2011, pp. 91–99.

[77] C. Roscian, A. Sarafianos, J. Dutertre, and A. Tria, "Fault Model Analysis of Laser-Induced Faults in SRAM Memory Cells," in *Fault Diagnosis and Tolerance in Cryptography – FDTC 2013*. IEEE Computer Society, 2013, pp. 89–98.

[78] M. Hutter and J. Schmidt, "The Temperature Side Channel and Heating Fault Attacks," in *Smart Card Research and Advanced Applications – CARDIS 2013*, ser. LNCS, vol. 8419. Springer, 2013, pp. 219–235.

[79] T. Müller and M. Spreitzenbarth, "FROST - Forensic Recovery of Scrambled Telephones," in *Applied Cryptography and Network Security – ACNS 2013*, ser. LNCS, vol. 7954. Springer, 2013, pp. 373–388.

[80] S. Skorobogatov, "The Bumpy Road Towards iPhone 5c NAND Mirroring," *arXiv ePrint Archive, Report 1609.04327*, 2016.

[81] G. He, M. Yang, X. Gu, J. Luo, and Y. Ma, "A Novel Active Website Fingerprinting Attack Against Tor Anonymous System," in *Computer Supported Cooperative Work in Design – CSCWD 2014*. IEEE, 2014, pp. 112–117.

[82] X. Cai, X. C. Zhang, B. Joshi, and R. Johnson, "Touching From a Distance: Website Fingerprinting Attacks and Defenses," in *Conference on Computer and Communications Security – CCS 2012*. ACM, 2012, pp. 605–616.

[83] A. Panchenko, L. Niessen, A. Zinnen, and T. Engel, "Website Fingerprinting in Onion Routing Based Anonymization Networks," in *Workshop on Privacy in the Electronic Society – WPES 2011*. ACM, 2011, pp. 103–114.

[84] T. Wang and I. Goldberg, "Improved Website Fingerprinting on Tor," in *Workshop on Privacy in the Electronic Society – WPES 2013*. ACM, 2013, pp. 201–212.

[85] M. Juárez, S. Afroz, G. Acar, C. Díaz, and R. Greenstadt, "A Critical Evaluation of Website Fingerprinting Attacks," in *Conference on Computer and Communications Security – CCS 2014*. ACM, 2014, pp. 263–274.

[86] A. Panchenko, F. Lanze, J. Pennekamp, T. Engel, A. Zinnen, M. Henze, and K. Wehrle, "Website Fingerprinting at Internet Scale," in *Network and Distributed System Security Symposium – NDSS 2016*. The Internet Society, 2016.

[87] M. Conti, L. V. Mancini, R. Spolaor, and N. V. Verde, "Analyzing Android Encrypted Network Traffic to Identify User Actions," *IEEE Trans. Information Forensics and Security*, vol. 11, pp. 114–125, 2016.

[88] T. Stöber, M. Frank, J. B. Schmitt, and I. Martinovic, "Who Do You Sync You Are?: Smartphone Fingerprinting via Application Behaviour," in *Security and Privacy in Wireless and Mobile Networks – WISEC 2013*. ACM, 2013, pp. 7–12.

[89] Q. Yang, P. Gasti, G. Zhou, A. Farajidavar, and K. S. Balagani, "On Inferring Browsing Activity on Smartphones via USB Power Analysis Side-Channel," *IEEE Trans. Information Forensics and Security*, vol. 14, 2016, in press.

[90] M. Conti, M. Nati, E. Rotundo, and R. Spolaor, "Mind The Plug! Laptop-User Recognition Through Power Consumption," in *Workshop on IoT Privacy, Trust, and Security – IoTPTS@AsiaCCS*. ACM, 2016, pp. 37–44.

[91] J. Zhang, X. Zheng, Z. Tang, T. Xing, X. Chen, D. Fang, R. Li, X. Gong, and F. Chen, "Privacy Leakage in Mobile Sensing: Your Unlock Passwords Can Be Leaked through Wireless Hotspot Functionality," *Mobile Information Systems*, vol. 2016, pp. 8 793 025:1–8 793 025:14, 2016.

[92] M. Li, Y. Meng, J. Liu, H. Zhu, X. Liang, Y. Liu, and N. Ruan, "When CSI Meets Public WiFi: Inferring Your Mobile Phone Password via WiFi Signals," in *Conference on Computer and Communications Security – CCS 2016*. ACM, 2016, pp. 1068–1079.

[93] S. Dai, A. Tongaonkar, X. Wang, A. Nucci, and D. Song, "NetworkProfiler: Towards Automatic Fingerprinting of Android Apps," in *IEEE INFOCOM 2013*. IEEE, 2013, pp. 809–817.

[94] Q. Wang, A. Yahyavi, B. Kemme, and W. He, "I Know What You Did on Your Smartphone: Inferring App Usage Over Encrypted Data Traffic," in *Communications and Network Security – CNS 2015*. IEEE, 2015, pp. 433–441.

[95] S. Miskovic, G. M. Lee, Y. Liao, and M. Baldi, "AppPrint: Automatic Fingerprinting of Mobile Applications in Network Traffic," in *Passive and Active Measurement – PAM 2015*, ser. LNCS, vol. 8995. Springer, 2015, pp. 57–69.

[96] V. F. Taylor, R. Spolaor, M. Conti, and I. Martinovic, "AppScanner: Automatic Fingerprinting of Smartphone Apps from Encrypted Network Traffic," in *IEEE European Symposium on Security and Privacy – EURO S&P 2016*. IEEE, 2016, pp. 439–454.

[97] H. F. Alan and J. Kaur, "Can Android Applications Be Identified Using Only TCP/IP Headers of Their Launch Time Traffic?" in *Security and Privacy in Wireless and Mobile Networks – WISEC 2016*. ACM, 2016, pp. 61–66.

[98] B. Saltaformaggio, H. Choi, K. Johnson, Y. Kwon, Q. Zhang, X. Zhang, D. Xu, and J. Qian, "Eavesdropping on Fine-Grained User Activities Within Smartphone Apps Over Encrypted Network Traffic," in *Workshop on Offensive Technologies – WOOT 2016*. USENIX Association, 2016.

[99] M. Schulz, P. Klapper, M. Hollick, E. Tews, and S. Katzenbeisser, "Trust The Wire, They Always Told Me!: On Practical Non-Destructive Wire-Tap Attacks Against Ethernet," in *Security and Privacy in Wireless and Mobile Networks – WISEC 2016*. ACM, 2016, pp. 43–48.

[100] K. Ali, A. X. Liu, W. Wang, and M. Shahzad, "Keystroke Recognition Using WiFi Signals," in *Mobile Computing and Networking – MOBICOM 2015*. ACM, 2015, pp. 90–102.

[101] Q. A. Chen, Z. Qian, and Z. M. Mao, "Peeking into Your App without Actually Seeing It: UI State Inference and Novel Android Attacks," in *USENIX Security Symposium 2014*. USENIX Association, 2014, pp. 1037–1052.

[102] W. Diao, X. Liu, Z. Li, and K. Zhang, "No Pardon for the Interruption: New Inference Attacks on Android Through Interrupt Timing Analysis," in *IEEE Symposium on Security and Privacy – S&P 2016*. IEEE, 2016, pp. 414–432.

[103] K. Suzaki, K. Iijima, T. Yagi, and C. Artho, "Memory Deduplication as a Threat to the Guest OS," in *European Workshop on System Security – EUROSEC 2011*. ACM, 2011, p. 1.

[104] D. J. Bernstein, "Cache-Timing Attacks on AES," 2004, URL: `http://cr.yp.to/papers.html#cachetiming`.

[105] M. Weiß, B. Heinz, and F. Stumpf, "A Cache Timing Attack on AES in Virtualization Environments," in *Financial Cryptography – FC 2012*, ser. LNCS, vol. 7397. Springer, 2012, pp. 314–328.

[106] M. Weiß, B. Weggenmann, M. August, and G. Sigl, "On Cache Timing Attacks Considering Multi-core Aspects in Virtualized Embedded Systems," in *Conference on Trusted Systems – INTRUST 2014*, ser. LNCS, vol. 9473. Springer, 2014, pp. 151–167.

[107] A. Zankl, K. Miller, J. Heyszl, and G. Sigl, "Towards Efficient Evaluation of a Time-Driven Cache Attack on Modern Processors," in *European Symposium on Research in Computer Security – ESORICS 2016*, ser. LNCS, vol. 9879. Springer, 2016, pp. 3–19.

[108] R. Spreitzer and T. Plos, "On the Applicability of Time-Driven Cache Attacks on Mobile Devices," in *Network and System Security – NSS 2013*, ser. LNCS, vol. 7873. Springer, 2013, pp. 656–662.

[109] R. Spreitzer and B. Gérard, "Towards More Practical Time-Driven Cache Attacks," in *Information Security Theory and Practice – WISTP 2014*, ser. LNCS, vol. 8501. Springer, 2014, pp. 24–39.

[110] A. Bogdanov, T. Eisenbarth, C. Paar, and M. Wienecke, "Differential Cache-Collision Timing Attacks on AES with Applications to Embedded CPUs," in *Topics in Cryptology – CT-RSA 2010*, ser. LNCS, vol. 5985. Springer, 2010, pp. 235–251.

[111] R. Spreitzer and T. Plos, "Cache-Access Pattern Attack on Disaligned AES T-Tables," in *Constructive Side-Channel Analysis and Secure Design – COSADE 2013*, ser. LNCS, vol. 7864. Springer, 2013, pp. 200–214.

[112] Y. Oren, V. P. Kemerlis, S. Sethumadhavan, and A. D. Keromytis, "The Spy in the Sandbox: Practical Cache Attacks in JavaScript and their Implications," in *Conference on Computer and Communications Security – CCS 2015*. ACM, 2015, pp. 1406–1418.

[113] M. Lipp, D. Gruss, R. Spreitzer, C. Maurice, and S. Mangard, "ARMageddon: Cache Attacks on Mobile Devices," in *USENIX Security Symposium 2016*. USENIX Association, 2016, pp. 549–564.

[114] X. Zhang, Y. Xiao, and Y. Zhang, "Return-Oriented Flush-Reload Side Channels on ARM and Their Implications for Android Devices," in *Conference on Computer and Communications Security – CCS 2016*. ACM, 2016, pp. 858–870.

[115] L. Cai, S. Machiraju, and H. Chen, "Defending Against Sensor-Sniffing Attacks on Mobile Phones," in *Workshop on Networking, Systems, and Applications for Mobile Handhelds – MobiHeld*. ACM, 2009, pp. 31–36.

[116] A. Raij, A. Ghosh, S. Kumar, and M. B. Srivastava, "Privacy Risks Emerging from the Adoption of Innocuous Wearable Sensors in the Mobile Environment," in *Conference on Human Factors in Computing Systems – CHI 2011*. ACM, 2011, pp. 11–20.

[117] E. Owusu, J. Han, S. Das, A. Perrig, and J. Zhang, "ACCessory: Password Inference Using Accelerometers on Smartphones," in *Mobile Computing Systems and Applications – HotMobile 2012*. ACM, 2012, p. 9.

[118] A. J. Aviv, "Side Channels Enable By Smartphone Interaction," Ph.D. dissertation, University of Pennsylvania, 2012.

[119] Z. Xu, K. Bai, and S. Zhu, "TapLogger: Inferring User Inputs on Smartphone Touchscreens Using On-Board Motion Sensors," in *Security and Privacy in Wireless and Mobile Networks – WISEC 2012*. ACM, 2012, pp. 113–124.

[120] L. Cai and H. Chen, "On the Practicality of Motion Based Keystroke Inference Attack," in *Trust and Trustworthy Computing – TRUST 2012*, ser. LNCS, vol. 7344. Springer, 2012, pp. 273–290.

[121] E. Miluzzo, A. Varshavsky, S. Balakrishnan, and R. R. Choudhury, "Tapprints: Your Finger Taps Have Fingerprints," in *Mobile Systems – MobiSys 2012*. ACM, 2012, pp. 323–336.

[122] D. Ping, X. Sun, and B. Mao, "TextLogger: Inferring Longer Inputs on Touch Screen Using Motion Sensors," in *Security and Privacy in Wireless and Mobile Networks – WISEC 2015*. ACM, 2015, pp. 24:1–24:12.

[123] M. Mehrnezhad, E. Toreini, S. F. Shahandashti, and F. Hao, "Stealing PINs via Mobile Sensors: Actual Risk versus User Perception," *arXiv ePrint Archive, Report 1605.05549*, 2016.

[124] R. Spreitzer, "PIN Skimming: Exploiting the Ambient-Light Sensor in Mobile Devices," in *Security and Privacy in Smartphones & Mobile Devices – SPSM@CCS*. ACM, 2014, pp. 51–62.

[125] L. Simon and R. Anderson, "PIN Skimmer: Inferring PINs Through the Camera and Microphone," in *Security and Privacy in Smartphones & Mobile Devices – SPSM@CCS*. ACM, 2013, pp. 67–78.

[126] T. Fiebig, J. Krissler, and R. Hänsch, "Security Impact of High Resolution Smartphone Cameras," in *Workshop on Offensive Technologies – WOOT 2014*. USENIX Association, 2014.

[127] S. Narain, A. Sanatinia, and G. Noubir, "Single-Stroke Language-Agnostic Keylogging Using Stereo-Microphones and Domain Specific Machine Learning," in *Security and Privacy in Wireless and Mobile Networks – WISEC 2014*. ACM, 2014, pp. 201–212.

[128] H. Gupta, S. Sural, V. Atluri, and J. Vaidya, "Deciphering Text from Touchscreen Key Taps," in *Data and Applications Security and Privacy – DBSec 2016*, ser. LNCS, vol. 9766. Springer, 2016, pp. 3–18.

[129] H. Bojinov, Y. Michalevsky, G. Nakibly, and D. Boneh, "Mobile Device Identification via Sensor Fingerprinting," *arXiv ePrint Archive, Report 1408.1416*, 2014.

[130] S. Dey, N. Roy, W. Xu, R. R. Choudhury, and S. Nelakuditi, "AccelPrint: Imperfections of Accelerometers Make Smartphones Trackable," in *Network and Distributed System Security Symposium – NDSS 2014*. The Internet Society, 2014.

[131] A. Das, N. Borisov, and M. Caesar, "Tracking Mobile Web Users Through Motion Sensors: Attacks and Defenses," in *Network and Distributed System Security Symposium – NDSS 2016*. The Internet Society, 2016.

[132] ——, "Do You Hear What I Hear?: Fingerprinting Smart Devices Through Embedded Acoustic Components," in *Conference on Computer and Communications Security – CCS 2014*. ACM, 2014, pp. 441–452.

[133] Z. Zhou, W. Diao, X. Liu, and K. Zhang, "Acoustic Fingerprinting Revisited: Generate Stable Device ID Stealthily with Inaudible Sound," in *Conference on Computer and Communications Security – CCS 2014*. ACM, 2014, pp. 429–440.

[134] T. Hupperich, H. Hosseini, and T. Holz, "Leveraging Sensor Fingerprinting for Mobile Device Authentication," in *Detection of Intrusions and Malware & Vulnerability Assessment – DIMVA 2016*, ser. LNCS, vol. 9721. Springer, 2016, pp. 377–396.

[135] A. Kurtz, H. Gascon, T. Becker, K. Rieck, and F. C. Freiling, "Fingerprinting Mobile Devices Using Personalized Configurations," *PoPETs*, vol. 2016, pp. 4–19, 2016.

[136] T. Hupperich, D. Maiorca, M. Kührer, T. Holz, and G. Giacinto, "On the Robustness of Mobile Device Fingerprinting: Can Mobile Users Escape Modern Web-Tracking Mechanisms?" in *Annual Computer Security Applications Conference – ACSAC 2015*. ACM, 2015, pp. 191–200.

[137] Y. Michalevsky, D. Boneh, and G. Nakibly, "Gyrophone: Recognizing Speech from Gyroscope Signals," in *USENIX Security Symposium 2014*. USENIX Association, 2014, pp. 1053–1067.

[138] B. Ho, P. D. Martin, P. Swaminathan, and M. B. Srivastava, "From Pressure to Path: Barometer-based Vehicle Tracking," in *Embedded Systems for Energy-Efficient Built Environments – BuildSys*. ACM, 2015, pp. 65–74.

[139] R. Schlegel, K. Zhang, X. Zhou, M. Intwala, A. Kapadia, and X. Wang, "Soundcomber: A Stealthy and Context-Aware Sound Trojan for Smartphones," in *Network and Distributed System Security Symposium – NDSS 2011*. The Internet Society, 2011.

[140] V. van der Veen, Y. Fratantonio, M. Lindorfer, D. Gruss, C. Maurice, G. Vigna, H. Bos, K. Razavi, and C. Giuffrida, "Drammer: Deterministic Rowhammer Attacks on Mobile Platforms," in *Conference on Computer and Communications Security – CCS 2016*. ACM, 2016, pp. 1675–1689.

[141] J. Han, E. Owusu, L. T. Nguyen, A. Perrig, and J. Zhang, "ACComplice: Location Inference Using Accelerometers on Smartphones," in *International Conference on Communication Systems and Networks – COMSNETS 2012*. IEEE, 2012, pp. 1–9.

[142] S. Nawaz and C. Mascolo, "Mining Users' Significant Driving Routes with Low-Power Sensors," in *Conference on Embedded Network Sensor Systems – SenSys 2014*. ACM, 2014, pp. 236–250.

[143] S. Narain, T. D. Vo-Huu, K. Block, and G. Noubir, "Inferring User Routes and Locations Using Zero-Permission Mobile Sensors," in *IEEE Symposium on Security and Privacy – S&P 2016*. IEEE, 2016, pp. 397–413.

[144] S. Hemminki, P. Nurmi, and S. Tarkoma, "Accelerometer-Based Transportation Mode Detection on Smartphones," in *Conference on Embedded Network Sensor Systems – SenSys 2013*. ACM, 2013, pp. 13:1–13:14.

[145] Y. Kim, R. Daly, J. Kim, C. Fallin, J. Lee, D. Lee, C. Wilkerson, K. Lai, and O. Mutlu, "Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors," in *International Symposium on Computer Architecture – ISCA 2014*. IEEE Computer Society, 2014, pp. 361–372.

[146] M. Seaborn and T. Dullien, "Exploiting the DRAM Rowhammer Bug to Gain Kernel Privileges," Blackhat 2015.

[147] D. Gruss, C. Maurice, and S. Mangard, "Rowhammer.js: A Remote Software-Induced Fault Attack in JavaScript," in *Detection of Intrusions and Malware & Vulnerability Assessment – DIMVA 2016*, ser. LNCS, vol. 9721. Springer, 2016, pp. 300–321.

[148] V. Lomné, T. Roche, and A. Thillard, "On the Need of Randomness in Fault Attack Countermeasures - Application to AES," in *Fault Diagnosis and Tolerance in Cryptography – FDTC 2012*. IEEE Computer Society, 2012, pp. 85–94.

[149] T. Kwon and S. Na, "TinyLock: Affordable Defense Against Smudge Attacks on Smartphone Pattern Lock Systems," *Computers & Security*, vol. 42, pp. 137–150, 2014.

[150] K. Krombholz, T. Hupperich, and T. Holz, "Use the Force: Evaluating Force-Sensitive Authentication for Mobile Devices," in *Symposium On*

Usable Privacy and Security – SOUPS 2016. USENIX Association, 2016, pp. 207–219.

[151] C. V. Wright, S. E. Coull, and F. Monrose, "Traffic Morphing: An Efficient Defense Against Statistical Traffic Analysis," in *Network and Distributed System Security Symposium – NDSS 2009*. The Internet Society, 2009.

[152] X. Luo, P. Zhou, E. W. W. Chan, W. Lee, R. K. C. Chang, and R. Perdisci, "HTTPOS: Sealing Information Leaks with Browser-side Obfuscation of Encrypted Flows," in *Network and Distributed System Security Symposium – NDSS 2011*. The Internet Society, 2011.

[153] K. P. Dyer, S. E. Coull, T. Ristenpart, and T. Shrimpton, "Peek-a-Boo, I Still See You: Why Efficient Traffic Analysis Countermeasures Fail," in *IEEE Symposium on Security and Privacy – S&P 2012*. IEEE Computer Society, 2012, pp. 332–346.

[154] X. Cai, R. Nithyanand, and R. Johnson, "CS-BuFLO: A Congestion Sensitive Website Fingerprinting Defense," in *Workshop on Privacy in the Electronic Society – WPES 2014*. ACM, 2014, pp. 121–130.

[155] R. Nithyanand, X. Cai, and R. Johnson, "Glove: A Bespoke Website Fingerprinting Defense," in *Workshop on Privacy in the Electronic Society – WPES 2014*. ACM, 2014, pp. 131–134.

[156] A. P. Felt, E. Ha, S. Egelman, A. Haney, E. Chin, and D. Wagner, "Android Permissions: User Attention, Comprehension, and Behavior," in *Symposium On Usable Privacy and Security – SOUPS 2012*. ACM, 2012, p. 3.

[157] G. Petracca, Y. Sun, T. Jaeger, and A. Atamli, "AuDroid: Preventing Attacks on Audio Channels in Mobile Devices," in *Annual Computer Security Applications Conference – ACSAC 2015*. ACM, 2015, pp. 181–190.

[158] P. Shrestha, M. Mohamed, and N. Saxena, "Slogger: Smashing Motion-based Touchstroke Logging with Transparent System Noise," in *Security and Privacy in Wireless and Mobile Networks – WISEC 2016*. ACM, 2016, pp. 67–77.

[159] R. Könighofer, "A Fast and Cache-Timing Resistant Implementation of the AES," in *Topics in Cryptology – CT-RSA 2008*, ser. LNCS, vol. 4964. Springer, 2008, pp. 187–202.

[160] C. Rebeiro, A. D. Selvakumar, and A. S. L. Devi, "Bitslice Implementation of AES," in *Cryptology and Network Security – CANS 2006*, ser. LNCS, vol. 4301. Springer, 2006, pp. 203–212.

[161] H. Wang, T. T. Lai, and R. R. Choudhury, "MoLe: Motion Leaks through Smartwatch Sensors," in *Mobile Computing and Networking – MOBICOM 2015*. ACM, 2015, pp. 155–166.

[162] X. Liu, Z. Zhou, W. Diao, Z. Li, and K. Zhang, "When Good Becomes Evil: Keystroke Inference with Smartwatch," in *Conference on Computer and Communications Security – CCS 2015*. ACM, 2015, pp. 1273–1285.

[163] A. Maiti, O. Armbruster, M. Jadliwala, and J. He, "Smartwatch-Based Keystroke Inference Attacks and Context-Aware Protection Mechanisms," in *Asia Conference on Computer and Communications Security – AsiaCCS*. ACM, 2016, pp. 795–806.

[164] C. Wang, X. Guo, Y. Wang, Y. Chen, and B. Liu, "Friend or Foe?: Your Wearable Devices Reveal Your Personal PIN," in *Asia Conference on Computer and Communications Security – AsiaCCS*. ACM, 2016, pp. 189–200.

[165] A. Sarkisyan, R. Debbiny, and A. Nahapetian, "WristSnoop: Smartphone PINs Prediction Using Smartwatch Motion Sensors," in *Workshop on Information Forensics and Security – WIFS 2015*. IEEE, 2015, pp. 1–6.

[166] A. Maiti, M. Jadliwala, J. He, and I. Bilogrevic, "(Smart)Watch Your Taps: Side-Channel Keystroke Inference Attacks using Smartwatches," in *International Symposium on Wearable Computers – ISWC 2015*. ACM, 2015, pp. 27–30.

[167] L. Arduser, P. Bissig, P. Brandes, and R. Wattenhofer, "Recognizing Text Using Motion Data From a Smartwatch," in *Pervasive Computing and Communication Workshops – PerCom 2016*. IEEE Computer Society, 2016, pp. 1–6.

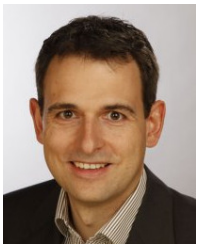
[168] B. Farshteindiker, N. Hasidim, A. Grosz, and Y. Oren, "How to Phone Home with Someone Else's Phone: Information Exfiltration Using Intentional Sound Noise on Gyroscopic Sensors," in *Workshop on Offensive Technologies – WOOT 2016*. USENIX Association, 2016.

[169] C. P. García, B. B. Brumley, and Y. Yarom, ""Make Sure DSA Signing Exponentiations Really are Constant-Time"," in *Conference on Computer and Communications Security – CCS 2016*. ACM, 2016, pp. 1639–1650.

Raphael Spreitzer is a researcher at Graz University of Technology. His main research interests are information security with a special focus on side-channel attacks on mobile devices, e.g., cache attacks and sensor-based attacks, and practical applications of privacy-enhancing technologies. He obtained a PhD degree in computer science with distinction from Graz University of Technology in 2017. Before, he finished the master's programme Software Engineering and Management with distinction at Graz University of Technology.

Veelasha Moonsamy is a postdoctoral researcher in the Digital Security group at Radboud University in The Netherlands. She obtained her PhD from Deakin University in Melbourne (Australia), under the supervision of Prof. Lynn Batten. Her research interests revolves around security and privacy on mobile devices, in particular side- and covert-channel attacks, malware detection and mitigation of information leaks at application and hardware level.

Thomas Korak received his MSc (Dipl.-Ing.) in Computer Engineering at Graz University of Technology in 2011 and his PhD in 2015. Until 2017 he was a postdoctoral researcher at the Institute for Applied Information Processing and Communications (IAIK), Graz University of Technology. His main research topics are side-channel attacks and fault attacks targeting embedded devices. Next to that he is also interested in countermeasures for hardening devices against this kind of attacks.



Stefan Mangard is full professor at Graz University of Technology since November 2013. Before moving to Graz, he was working as leading security architect at Infineon Technologies in Munich. In this role he was responsible for defining the security concepts for all the smart card platforms of Infineon, one of the largest manufacturers of smart card ICs worldwide. He is chair of the steering committee of CHES, which is the foremost conference on cryptographic hardware, and associate editor of the Journal on Cryptographic Engineering. He regularly serves on program committees of conferences in the field (CHES, CARDIS, COSADE...). In 2015, Stefan has received the highly prestigious ERC consolidator grant by the European Research Council for his research proposal "SOPHIA – Securing Software against Physical Attacks".