Hardware Platform Security for Mobile Devices

Other titles in Foundations and Trends® in Privacy and Security

Expressing Information Flow Properties
Elisavet Kozyri, Stephen Chong and Andrew C. Myers
ISBN: 978-1-68083-936-4

Accountability in Computing: Concepts and Mechanisms
Joan Feigenbaum, Aaron D. Jaggard and Rebecca N. Wright
ISBN: 978-1-68083-784-1

A Pragmatic Introduction to Secure Multi-Party Computation
David Evans, Vladimir Kolesnikov and Mike Rosulek
ISBN: 978-1-68083-508-3

Contextual Integrity through the Lens of Computer Science
Sebastian Benthall, Seda Gurses and Helen Nissenbaum
ISBN: 978-1-68083-384-3

Methods for Location Privacy: A comparative overview
Kostantinos Chatzikokolakis, Ehab ElSalamouny, Catuscia Palamidessi and Pazii Anna
ISBN: 978-1-68083-366-9

Hardware Platform Security for Mobile Devices

Lachlan J. Gunn
Aalto University
Finland
lachlan@gunn.ee

N. Asokan
University of Waterloo
Canada
asokan@acm.org

Jan-Erik Ekberg
Huawei
Finland

Hans Liljestrand
University of Waterloo
Canada

Vijayanand Nayani
Huawei
Finland

Thomas Nyman
Aalto University
Finland

now
the essence of knowledge
Boston — Delft

Foundations and Trends® in Privacy and Security

Published, sold and distributed by:
now Publishers Inc.
PO Box 1024
Hanover, MA 02339
United States
Tel. +1-781-985-4510
www.nowpublishers.com
sales@nowpublishers.com

Outside North America:
now Publishers Inc.
PO Box 179
2600 AD Delft
The Netherlands
Tel. +31-6-51115274

The preferred citation for this publication is

L. J. Gunn *et al.*. *Hardware Platform Security for Mobile Devices*. Foundations and Trends® in Privacy and Security, vol. 3, no. 3-4, pp. 214–394, 2022.

ISBN: 978-1-68083-977-7

Foundations and Trends® in Privacy and Security
Volume 3, Issue 3-4, 2022

Editorial Board

Editorial Scope

Topics

Foundations and Trends® in Privacy and Security publishes survey and tutorial articles in the following topics:

- Access control
- Accountability
- Anonymity
- Application security
- Artifical intelligence methods in security and privacy
- Authentication
- Big data analytics and privacy
- Cloud security
- Cyber-physical systems security and privacy
- Distributed systems security and privacy
- Embedded systems security and privacy
- Forensics
- Hardware security
- Human factors in security and privacy
- Information flow
- Intrusion detection
- Malware
- Metrics
- Mobile security and privacy
- Language-based security and privacy
- Network security
- Privacy-preserving systems
- Protocol security
- Security and privacy policies
- Security architectures
- System security
- Web security and privacy

Information for Librarians

Foundations and Trends® in Privacy and Security, 2022, Volume 3, 4 issues. ISSN paper version 2474-1558. ISSN online version 2474-1566. Also available as a combined paper and online subscription.

Contents

Hardware Platform Security for Mobile Devices

Lachlan J. Gunn[1], N. Asokan[2], Jan-Erik Ekberg[3], Hans Liljestrand[2], Vijayanand Nayani[3] and Thomas Nyman[1]

[1] *Aalto University, Finland; lachlan@gunn.ee*
[2] *University of Waterloo, Canada; asokan@acm.org*
[3] *Huawei, Finland*

ABSTRACT

Today, personal mobile devices like smartphones and tablets are ubiquitous. People use mobile devices for fun, for work, and for organizing and managing their lives, including their finances. This became possible because over the last two decades, mobile phones evolved from closed platforms intended for voice calls and messaging to open platforms whose functionality can be extended in myriad ways by third party developers. Such wide-ranging scope of use also means widely different security and privacy requirements for those uses. The mobile device ecosystem involved multiple different stakeholders such as mobile network operators, regulators, enterprise information technology administrators, and of course ordinary users. So, as mobile platforms became gradually open, *platform security* mechanisms were incorporated into their architectures so that the security and privacy requirements of all stakeholders could be met. Platform security mechanisms help to isolate applications from one another, protect persistent data and other on-device resources (like access to location or peripherals), and

Lachlan J. Gunn, N. Asokan, Jan-Erik Ekberg, Hans Liljestrand, Vijayanand Nayani and Thomas Nyman (2022), "Hardware Platform Security for Mobile Devices", Foundations and Trends® in Privacy and Security: Vol. 3, No. 3-4, pp 214–394. DOI: 10.1561/3300000024.

help strengthen software against a variety of attack vectors. All major mobile platforms incorporate comprehensive software and hardware platform security architectures, including mechanisms like *trusted execution environments (TEEs)*.

Over the past decade, mobile devices have been undergoing convergences in multiple dimensions. The distinction between "mobile" and "fixed" devices has blurred. Similar security mechanisms and concepts are being used across different platforms, leading to similar security architectures. Hardware enablers used to support platform security have gradually matured. At the same time, there have also been novel types of attacks, ranging from software attacks like return- and data-oriented programming to hardware attacks like side channels that exploit micro-architectural phenomena. It is no longer tenable to assume that the current hardware security mechanisms underpinning mobile platform security are inviolable.

The time is therefore right to take a new look at mobile platform security, which brings us to this monograph. We focus on *hardware platform security*. The monograph is divided into four parts: we begin by looking at the *why* and *how* of mobile platform security, followed by a discussion on *vulnerabilities and attacks*; we conclude by *looking forward* discussing emerging research that explores ways of dealing with hardware compromise, and building blocks for the next generation of hardware platform security.

Our intent is to provide a broad overview of the current state of practice and a glimpse of possible research directions that can be of use to practitioners, decision makers, and researchers.

Part I

Mobile Platform Security: Why?

1

Introduction

Today, mobile devices such as smartphones and tablets are very widely deployed. All modern mobile device platforms incorporate sophisticated software and hardware *platform security* mechanisms. To understand how this came to be, we need to start in the late 1990s.

1.1 What motivated mobile platform security?

The mobile phone revolution was well under way by the mid 1990s. Initially, mobile phones were simple embedded devices with fixed functionality: voice calls and text messages. Early on, the mobile phone industry recognized the power of billions of people having general-purpose computing devices in their hands. Personal digital assistants were already available and demonstrated the range of uses for portable, personal general-purpose computing devices. Therefore, already by the mid 1990s, the industry was working towards *opening up* mobile phone platforms so that users gain the ability to *extend* their functionality, by installing third-party software modules. This development directly led to today's smartphones and tablets with their "app" ecosystems.

The industry saw the potential for new types of applications like mobile payments, public transport ticketing, and digital media consump-

tion. But it also realized that for these applications to succeed, open mobile devices needed additional mechanisms to safeguard the security and privacy requirements of these novel, and potentially high-value, applications.

Furthermore, the mobile phone ecosystem already had well-established stakeholders. They were sensitive to the security and privacy concern that could arise in the transition from closed fixed-function devices to open platforms. Existing commodity general-purpose computing platforms at the time, like those for personal computers, did not incorporate the platform security mechanisms necessary to address these concerns. Consequently, they wanted to mediate this transition so that their own interests were safeguarded. This, too, drove the development of new platform security mechanisms. Mobile platform security architectures emerged because of the need to address these stakeholder concerns as mobile devices opened up (Matala *et al.*, 2019).

1.2 Stakeholders

An important class of stakeholders are mobile network operators (MNOs) (also known as "carriers") who are motivated by business interests. An example of a business interest of MNOs is the need to strongly authenticate their subscribers. This need led to the introduction of subscriber identity modules (SIMs) (discussed further in Section 2.2) right from the beginning. Another example of a business interest of MNOs is the need for robust technical mechanisms to support the *subsidy-lock business model* where a MNO gives a mobile phone to a subscriber for free or below cost, in return for a commitment to maintain the subscription for a specified period of time. The requirement to technically enforce subsidy locks translated into each device having an unforgeable unique identifier and the ability to run subsidy-lock enforcement software in a manner that cannot be bypassed.

Another, equally important, class of stakeholders are *regulators* who safeguard the public good. An example of a regulatory need is to ensure that radio-frequency transmission parameters, which are typically calibrated for each device at the time of manufacture, cannot be tampered with. This need can be met with secure (integrity-protected) storage for storing these parameters.

A third class of stakeholders are end-users. They were used to mobile phones that were reliable and trustworthy. They expected the same degree of reliability and trustworthiness to be maintained, even as mobile phone platforms were opening up.

There are other stakeholders in the ecosystem, like enterprise administrators, and of course the mobile phone manufacturers —also known as original equipment manufacturers (OEMs)—and operating system (OS) vendors themselves. To see what kinds of mechanisms are needed to protect the interest of different stakeholders, it is necessary to understand the threat models from the perspectives of these stakeholders.

1.3 Threat models

A threat model involves characterizing the adversary in terms of its capabilities, and the assets that need to be protected from these adversaries. For example a *software adversary* is assumed to be capable of influencing one or more software modules on the victim device. The adversary's control may be limited to a single application (software in user space) or can extend to privileged software like the OS itself. In contrast, a *hardware adversary* can directly interact with, and possibly manipulate, the hardware components on the victim device.

Rather than presenting an exhaustive treatment of all possible threat models, we will illustrate the concept with three informal examples.

First, consider the threat of a user's address book being exfiltrated from the device by a malicious third-party application that the user happened to install. We are concerned with a software adversary (the third-party developer) and the asset that needs protection is the address book. Standard hardware support (for memory management and process isolation) combined with a good OS security architecture (providing access-controlled persistent storage for each application) would be sufficient to provide the required protection. In Section 3 we will discuss OS security architectures.

Next, consider the same setting as above, but with a different asset: credentials for accessing financial transactions like online banking. While we are still concerned with a software adversary, the value of the asset is significantly higher, and its compromise can result in substantial

losses. Consequently, relying only on OS security is not reasonable because an OS is a complex software component with a large threat surface for the attacker to exploit. Additional hardware support for protecting high-value assets is justifiable. Hardware-assisted trusted execution environments (TEEs) allow small pieces of trusted software on a general-purpose computing device to be *isolated* from the rest of the software on the same device, including the OS and other applications. Today TEEs are ubiquitous. Nearly every smartphone or tablet is likely to have a processor with TEE capabilities. Many personal computers are also equipped with TEEs. The ubiquity of TEEs is not a recent phenomenon (Ekberg *et al.*, 2014): hardware-assisted TEEs started to appear in mobile phones from the early 2000s. For a technology that is so widely deployed, for so long, the origins and trajectory of TEE technologies are poorly understood. Our primary focus in this monograph is to explore hardware platform security for mobile devices, with a particular emphasis on TEEs.

Finally, consider the case of technical mechanisms for subsidy-lock enforcement. The adversary in this case is the user of the device who has physical access to the device. The asset that the adversary wants to compromise is the binding between the mobile device hardware and the MNO (so that a successful attack will result in breaking the binding, allowing the adversary to use the device with a different MNO subscription). OS security alone is not sufficient. Since we now deal with a potential hardware adversary, we must use hardware-security mechanisms that can withstand physical attack.

1.4 Chains of trust

In a given scenario, the party relying on the protection mechanism trusts the software and hardware components used to realize the mechanism. A *chain of trust* refers to the process of building up this trust, starting from one or more *roots of trust*. In the first example above, while OS security is sufficient, the relying party, the user, needs to trust that the correct OS is running on the device. *Platform integrity* (Section 4) makes it possible to build up this trust. Higher level platform security mechanisms like OS security rely on underlying building blocks like

platform integrity (Section 4), hardware-assisted isolation (Section 5), and cryptographic primitives realized in hardware (Section 6).

An important feature of hardware platform security mechanisms is allowing remote relying parties to build up trust in a device. In the second example above, a bank may need to convince itself that the user is accessing her bank account from a secure device before allowing access. This feature is called *remote attestation*, which is widely supported by modern TEEs. In Section 5 we will discuss the chains of trust involved in remote attestation in modern TEEs.

We begin with an overview of the history of mobile hardware platform security mechanisms (Section 2), and provide an overview of OS security (Section 3) to understand how an OS can make use of these mechanisms. We will explore the nuts and bolts of how platform security is implemented in today's devices, focusing on hardware platform security (Part II), and discuss attacks against hardware platform security mechanisms (Part III). We will conclude with a brief foray into a future outlook for hardware platform security (Part IV).

Notes on the scope of this monograph

The focus of this monograph is on hardware platform security in mobile devices. We do cover OS security in Section 3, but from the perspective of motivating hardware platform security. Mobile device platforms also incorporate sophisticated *software platform security* mechanisms. We refer readers interested in this topic to books dedicated to the topic such as Asokan *et al.* (2014). We also do not cover specific high-level attacks such as *jail-breaking* (removing manufacturer-imposed restrictions on what software can be installed on a mobile device) or *rooting* (obtaining the privileges of the maximally privileged "root" user on Unix-based mobile OSs). However, the basic attacks we describe in Part III can be, and often are, used as stepping stones for these high-level attacks.

2

Historical Overview

The requirements we saw in Section 1 led to mobile device and platform vendors developing and deploying software and hardware platform security architectures. Nokia Radio Application Processors are believed to be the first trusted execution environments (TEEs) deployed at a large scale (Matala *et al.*, 2019). These were followed shortly by Texas Instruments' M-Shield™ (Sundaresan, 2003) and subsequently by ARM's TrustZone™(Alves and Felton, 2004) which represents the overwhelming share of deployed mobile TEEs today.

In the non-mobile setting, hardware security modules (HSMs) used in the financial sector (starting with IBM's CryptoCard[1]) are an early example of a TEE. Trusted Computing Group's Trusted Platform Modules (TPMs) (Arthur and Challener, 2015) are widely deployed in personal computers, where they are used for boot integrity and disk encryption, but they have not found common use in the mobile space. Recently, Intel's Software Guard Extensions (SGX) (McKeen *et al.*, 2016) has become the most widely studied TEE architecture, thanks to the easy availability of both the software and hardware.[2] SGX is

[1] https://www.ibm.com/security/cryptocards/
[2] https://software.intel.com/en-us/sgx

primarily deployed in cloud settings to enable confidential computing use cases (Alibaba, 2020; McReynolds, 2021). Desktop use cases for SGX include Blu-ray digital rights management (DRM) (Toulas, 2021)

2.1 Hardware security modules

Early examples of the inclusion of a dedicated security co-processor were motivated by the need to perform sensitive cryptographic operations isolated from other computations in systems handling financial transactions. Transaction processing for Europay, Mastercard and Visa (EMV) payment cards use HSMs as the primary security device for key management (Cryptomathic, 2017). An HSM is a discrete computing device usually encapsulated in tamper-evident coating. HSMs in backend systems typically include specialized cryptographic hardware accelerators to enable high throughput because they need to process transactions in real-time. An HSM can be realized as either a standalone peripheral device or as an extension board connected directly to the internal bus of the host computer. The operational keys are generated in the cryptographic co-processor within the HSM and are then saved either in a keystore file or in application memory, encrypted under the master key of that co-processor. Any HSM with an identical master key can use those keys.

The first commercially available civilian HSMs were deployed already in the 1970s, originally for IBM mainframes. The IBM 3845 and 3846 data encryption devices (IBM, 1977) allowed exported encryption keys to be encrypted using the recently standardized DES algorithm. These early HSMs included secure key entry devices (cards and PIN pads) for master key loading, random number generation capabilities for seeding, and persistent storage for key materials. They were instrumental in securing early electronic banking, such as automatic teller machines (ATMs).

2.1.1 HSMs in radio communication

HSMs are also extensively deployed for modern military software-defined radio (SDR) communication. SDR refers to wireless communications

where the transmitter and receiver mixing, filtering, amplification, modulation/demodulation etc. occur in software instead of in conventional radio electronics. With SDR, software-based transmission algorithms can be downloaded and adapted over the lifecycle of the hardware. While analog military radio equipment include dedicated cryptographic chips for (proprietary) ciphers that are required for communication with compatible equipment, SDR equipment have to support a large number of cryptographic schemes, including legacy protocols and algorithms. Consequently military SDR equipment, such as the U.S. Joint Tactical Radio System (JTRS), employ embeddable HSMs specifically designed for communication security. The Advanced INFOSEC Machine (AIM) (General Dynamics Mission Systems, 2015a) is one such programmable, embeddable cryptographic unit developed by Motorola in the late 1990s. It consists of a hardware platform with three independent cryptographic processors, one for key management and two programmable processors for traffic encryption/decryption. The key management cryptographic engine (KMCE) is based on a 32-bit reduced instruction set computer (RISC) processor and includes a math co-processor designed for public key algorithm processing. The KMCE runs a read-only memory (ROM)-based Secure Operating System (SOS). The SOS provides a multi-security level, multi-tasking environment for the cryptographic applications which allowed the functionality of the AIM to be extended by software. The chip contains the necessary building blocks to implement encryption algorithms such as DES, and the classified SAVILLE and BATON cryptographic algorithms used by U.S. and NATO. Its successor, AIM II (General Dynamics Mission Systems, 2015b) is specifically designed for JTRS. Around the same time, a similar crypto-chip, called the General Crypto Device (GCD) (Lange, 1997), was developed in Europe by Dutch electronics giant Philips.

The use of HSMs such as AIM and GCD are early examples of the use of TEEs in telecommunications. The sensitivity of military communication justified the inclusion of dedicated components for security into end devices. However, for civilian telecommunication devices, the widespread use of TEE-technology only occurred when two conditions were met: 1) economic incentives emerged to justify requiring strong, hardware-based security, and 2) low-cost technological solutions that met those requirement were developed.

2.2 SIMs, mobile handsets, and smart cards

During the early 1990s, civilian wireless communication systems also began to employ hardware-assisted security. Mobile network operators (MNOs) required a reliable way of preventing illicit use of a subscriber identity for making phone calls from mobile phones. For this purpose, the subscriber identity module (SIM) card (GSMA, 2015) was developed by Munich smart-card maker Giesecke & Devrient, who sold the first 300 SIM cards to the Finnish MNO Radiolinja in 1991. The use of SIM cards became mandatory in the Global System for Mobile Communications (GSM) standard. Each SIM card contains an international mobile subscriber identity (IMSI) that uniquely identifies the user of the mobile network and a unique symmetric cryptographic key (K_i) assigned to it by the MNO during SIM card personalization. The SIM ensures the integrity of the IMSI and K_i, and the confidentiality of K_i. K_i allows the MNO to authenticate the SIM card when the mobile phone connects to the network. When the mobile phone connects, it obtains the IMSI from the SIM card, and requests network access by transmitting the IMSI to the MNO. The MNO looks up the corresponding K_i of the IMSI from its subscriber database, and generates a random nonce as a challenge which is transmitted to the mobile phone. The mobile phone passes the challenge to the SIM card, which signs it, and returns the signed response, which is transmitted back to the MNO by the mobile phone. The MNO compares the signed response to the response calculated using the MNO's copy of the K_i. If they match, the authentication is successful.

Modern SIM cards are based on tamper-resistant universal integrated circuit card (UICC) technology (SIMalliance Ltd., 2013) similar to smart cards. UICC cards can host multiple software applications, typically developed using Java Card software technology (ETSI, 2012). The applications include a SIM application for GSM, and universal subscriber identity module (USIM) for UMTS (3G), Long-Term Evolution (4G), and 5G network authentication. MNOs can also provision additional value-add applications to UICC cards that they issue, such as mobile banking and phone-based money transfer. UICC application can interface with mobile phone users or initiate actions via a card

application toolkit (CAT) part of the mobile phone operating system (OS): SIM Application Toolkit (STK) for GSM systems, and USIM Application Toolkit (USAT) for later generation networks. UICCs can support an optional bearer independent protocol (BIP), which allows MNOs to deliver over-the-air (OTA) updates to UICC applications either via cell broadcasts, or short message service packets.

All UICC applications are subject to authorization by the issuer security domain (ISD), namely the MNO who issued the UICC. Consequently UICCs are effectively closed application ecosystems; it is not possible for third-party developers to leverage UICC security without co-operating with MNOs in their region. This puts add-on services operated by large MNOs into an advantageous position compared to third-party alternatives, as is the case with M-Pesa (Mbiti and Weil, 2011), a money transfer application operated by Safaricom and Vodacom, the largest mobile MNO in Kenya and Tanzania. In developing countries, such as Kenya, low-cost feature phones are still prevalent, and UICC applications is the only ubiquitous application platform available to the majority of mobile phone users. Proprietary SIM overlay technology (a.k.a. "slim SIM" or "skin SIM") (Mondato, 2014) can enable third-party applications to operate independently of the underlying UICC.

The SIM overlay is a computer chip embedded into a thin plastic sheet that can be placed on top of a standard UICC card within a mobile phone. They were originally developed to support low-cost mobile roaming for Chinese customers traveling outside their home province. The overlay SIM acts as an independent security device, and allows additional functionality to be added to any mobile phone by attaching the overlay SIM to an MNO-issued UICC. However, an overlay SIM also has the potential to facilitate a man-in-the-middle attack by observing sensitive data such as personal identification numbers (PINs) being transmitted to the underlying UICC, or initiate, intercept and/or block mobile communications or CAT instructions (GSMA, 2014). By obtaining unauthorized access to the UICC SIM applications they could also change MNO configuration settings.

Embedded SIMs (eSIMs) (GSMA, 2015) are secure elements physically integrated into a mobile phone. eSIM chips can be directly soldered

onto the device or even embedded into the system on chip (SoC) itself. This physical integration necessitates MNO SIM or USIM profiles to be remotely provisioned. Additionally, unlike removable SIM cards, a single eSIMs may need to store multiple MNO profiles simultaneously.

2.3 Processor secure environments

Towards the late 1990s, mobile phones were transitioning from closed systems to open application platforms, for which third-party applications could be developed using the Java programming language. While not yet true smartphones, the feature phones of the time were gradually starting to resemble small, general-purpose computers. This brought with it new business opportunities, but also new challenges for device security; regulators and MNOs needed to ensure the protection of certain pieces of information after the mobile phone had left the manufacturing line. In particular, regulators required that the device identity, the international mobile equipment identifier (IMEI), remain unchanged in order to act as a theft deterrent. IMEIs of stolen mobile phones are blacklisted by network operators, thereby reducing the economic value of stolen mobile phones and deterring theft (GSMA, 2019).

Similarly, radio frequency parameters, which could affect the quality of service of other mobile phones in the area, or the safety of the user, should also remain unchanged. MNOs, who were the primary customers of large original equipment manufacturers (OEMs) such as Nokia, were concerned with ensuring that their subscribers receiving subsidized mobile devices do not break their contract terms. Consequently, they required a strong *subsidy lock* mechanism (colloquially known as SIM lock), which would tie the mobile phone to a particular MNO for the duration of the contract. Another emerging use case was DRM for digital content sold by the MNOs; initially ringtones, later games and music.

Nokia was the first to pursue a hardware-enforced processor secure environment. At the time, the security of Nokia's Digital Core Technology (DCT) generation phones was mainly based on obfuscated software solutions and protected by secrecy within the organization; even within the company, only few security professionals knew the exact design and

requirements of the DCT security architecture (Matala *et al.*, 2019). The leading market share of Nokia made it an attractive target for hackers who, (typically for a small fee) would "*unlock*" or "*unbrand*" subsidy-locked phones by either reverse engineering the valid unlock codes, or reflashing the phone with a different firmware version.

The fourth generation of DCT mobile phones included hardware components in the form of one-time-programmable memory to aid in the secure storage of sensitive device parameters. However, in the case of SIM locks, the economic motives to break device security were higher than the capabilities of the protection mechanism deployed at the time. Consequently, the revenue losses of important MNO customers resulting from SIM unlocking, increased the pressure to design a better security architecture for the upcoming generation of Nokia phones.

Within Nokia the idea of a coherent, hardware-enforced platform security originated within a team of engineers working with mobile payments and security (Matala *et al.*, 2019). Initial designs revolved around introducing a discrete security co-processor to ensure the physical isolation of the security-critical operations. However, the additional hardware chip in the bill of materials was deemed too expensive in the extremely cost-conscious organization, whose competitive advantage largely stemmed from its ability to keep manufacturing and components costs in control. Instead, Nokia engineers opted to implement a logically isolated secure processing mode within the main central processing unit (CPU). This solution was not only more cost effective in terms of component costs during manufacturing, but also functioned as common hardware platform for solutions to different use cases. This processor secure environment (Ekberg, 2013) would form the cornerstone of Nokia's Baseband 5 (BB5) generation mobile phone security architecture.

Initial hardware designs were based on Nokia's own radio application processors (RAPs), but from very early on Nokia collaborated with the U.S. semiconductor and Integrated Circuit (IC) manufacturer Texas Instruments (TI) with whom they had a close partnership at the time. The first BB5 mobile phone, the Nokia 6630 (codename "*Charlie*") was based on TI's Open Multimedia Applications Platform (OMAP) processors based on the ARM architecture. TI would brand the processor secure environment technology initially developed jointly with Nokia

as M-Shield (Sundaresan, 2003). It was however in Nokia's interest to ensure that it could invite bids from multiple hardware manufacturers for processors implementing a security architecture meeting Nokia's requirements. This became possible around 2003, when ARM proposed to develop system-wide hardware isolation architecture for secure execution for the ARMv6-A application processor architecture which included security extensions to the ARM SoC covering the processor, memory controllers and peripherals. ARM's design would become known as TrustZone (Alves and Felton, 2004). Integrating TrustZone in ARM processor architecture would ensure that any semiconductor manufacturer that implemented the TrustZone security extensions could supply Nokia with processor chips that met their requirements.

2.4 Trusted execution environments

In Section 1, we introduced the notion of TEEs – intuitively, a TEE is a computing environment on a device that a relying party trusts to a greater extent than the rest of the software running on the same device. Consider a device running a general-purpose operating system and applications, which, following standard practice, we will refer to as rich execution environment (REE) (GlobalPlatform, 2018c). For the purposes of this monograph we deem the device to have a TEE capable of running trusted code, if it has the following *capabilities*, possibly based on hardware support:

1. **Isolation**: The ability to run trusted code strongly isolated from the REE so that the REE cannot influence or learn the computations carried out by the trusted code,

2. **Secure Storage**: The ability for the trusted code to store persistent data guaranteeing its integrity and confidentiality with respect to an adversarial REE, even across reboots, and

3. **(Remote) Attestation**: The ability to convince a (possibly remote) party of the presence of the above attributes, and the characteristics of the trusted software protected by them.

This is an intentionally broad definition. It encompasses both physically distinct components—such as HSMs and TPMs—as well as processor secure environments where the isolation is logical and is enabled by extensions to the processor hardware.[3]

TEEs have largely evolved based on business needs, a number of commercial TEEs (Table 1) have emerged over the years. For mobile TEEs there is a framework of applicable standards, and a core set of these has reached critical mass in industry adoption. Standardization has followed in two contexts: 1) whenever and wherever common interfaces and application programming interfaces (APIs) are needed for interoperability, and 2) where common agreement for the formulation of the required security level for today's TEEs has been required.

The main standardization organization for mobile TEEs is the GlobalPlatform (GP) consortium.[4] GP provides a system architecture document (GlobalPlatform, 2018c) that describes the main components of the standards set related to TEEs, and how these individual standards contribute to the overall TEE system. Ostensibly the GP TEE architecture is not tied to any particular underlying hardware mechanism for ensuring isolation, but is, in practice, heavily influenced by the ARM TrustZone security architecture. Consequently GP standards are primarily adopted by TrustZone-based TEEs. Enclave architectures (Section 5.2), such as Intel SGX, do not yet have well-defined interoperability specification. But there are on-going efforts like the Linux Foundation's Confidential Computing Consortium which includes projects like the Open Enclave SDK[5] to provide a common development environment across different enclave architectures.

The GP TEE Client API (GlobalPlatform, 2010) is the common operating system interface (endpoint) to all TEE services. The specification primarily includes APIs for installing trusted applications (TAs) within the TEE, and for allowing REE applications – also known as

[3] Sometimes the term TEE is used as a synonym for the particular instance that we call "processor secure environments" in Section 5. The broad definition we adapt in this monograph is consistent with the terminology used by GlobalPlatform (GlobalPlatform, 2018c).

[4] https://globalplatform.org/

[5] https://openenclave.io/

client applications (CAs) – to communicate with their respective TAs, defining the data interaction model and the session management for this purpose. A separate Debug API, when available, enables a TA developer to receive logs from his TA, and also some post-mortem data in the case of critical crashes.

The GP TEE Internal API (GlobalPlatform, 2021a) is the specification against which TAs are written. For the time being, it provides C-language binding. The internal API defines the transactional model of TAs in the form of a set of standardized callback functions that are called when the TA is loaded, when it is connected to initially, and when it receives an incoming command. The data formats are TA-specific, but communication follows a paradigm of shared memory, allocated by the caller and accessible by the TA, when an incoming message is received. Another aspect of the internal API is the standardized programming framework, a "`libc`-like" interface that provides the TA developer with memory management, secure storage, time, peripheral access and cryptographic primitives. Due to the emphasis on security, the coverage of the cryptographic functionality in the internal API is extensive, and features most contemporary algorithms for public and private key cryptography, symmetric ciphers as well as digest and signature functions. Optional extensions (standards) to the GP internal API includes interfaces to smart cards and embedded secure elements (from within the TEE) (GlobalPlatform, 2021b), APIs by which trusted user interfaces can be setup and controlled (GlobalPlatform, 2013; GlobalPlatform, 2018d), and a socket API for network endpoints (GlobalPlatform, 2017).

For remote administration of TEEs, two separate specifications exist. Both are based on the notion that security domains are established on the device in a hierarchical fashion, after which the lifecycle of a security domain can be remotely managed, and secrets (data) and TA codes can be remotely provisioned to it. The two variants are the TEE Management Framework (TMF) (GlobalPlatform, 2016), and the Open Trust Protocol (OTrP) (Pei *et al.*, 2019; GlobalPlatform, 2019b). The latter is specified both in the context of the GP consortium (GlobalPlatform, 2019b) and in the context of IETF (Pei *et al.*, 2019). Even though both protocols accomplish the same thing, TMF is better suited to off-line (or store-and-forward) provisioning, whereas OTrP is explicitly an online protocol.

Another provisioning standard, used for virtually all smart cards with application update functionality (including UICC cards) is GP's *Card Specification* standards (GlobalPlatform, 2018a). These define the card commands by which software can be provisioned to the smart cards, and how security domains, i.e., keys identifying a certain card context, are managed. The secure communication between the provisioning entity and the card, as used by the Card Specification standard, is defined in the GP Secure Channel Protocols (GlobalPlatform, 2019a).

Appendix

Commercial TEE Deployments

Since TEE technology, and in particular TrustZone, has been deployed in large scale, a number of TEE vendors have emerged over the years. The majority of these are with proprietary implementations of the TEE software stack. Table 1 lists TEE vendors for TrustZone, TrustZone-M and the RISC-V architecture.

[1] https://globalplatform.org/wp-content/uploads/2019/07/01-CR-1.0_GP180004-Certificate-and-Certification-Report_20190712.pdf

[2] https://www.commoncriteriaportal.org/files/epfiles/CC%20Huawei%20iTrustee%20Software%20V2.0%20Security%20Target%202.1.pdf

[3] https://www.trustonic.com/solutions/iot-security/

[4] https://www.op-tee.org/

[5] https://www.provenrun.com/products/provencore/

[6] https://www.qualcomm.com/products/features/mobile-security-solutions

[7] https://www.rockycore.cn/index.html

[8] https://optimumdesk.com/it-solutions/data-loss-prevention-privacy

[9] https://www.sierraware.com/open-source-ARM-TrustZone.html

[10] https://www.trustkernel.com/en/products/tee/t6.html

[11] https://developer.samsung.com/teegris

[12] https://source.android.com/security/trusty

[13] https://www.taf.net.cn/Tee_detail.aspx?_ID=2349283b-6311-4617-862c-112234544354

[14] https://globalplatform.org/certified-products/watchtrust-2-1-1-on-sc9860-2/

[15] https://globalplatform.org/wp-content/uploads/2019/09/GP-SE-2019_02-CR-1.0_GP190006-Certificate-and-Certification-Report_20190909.pdf

[16] https://www.trustonic.com/technical-articles/kinibi-m/

[17] https://www.st.com/en/embedded-software/provencore-m.html

[18] https://hex-five.com/first-secure-iot-stack-riscv/

Table 1: Commercial TEE implementations. The GlobalPlatform (GP) Certification column indicates TEEs that have received TEE Initial Configuration v1.1 functional certification and/or TEE Security certification according to https://globalplatform.org/certified-products/.

TEE	Vendor	GP Certification Functional	GP Certification Security	License	Note
TrustZone					
Cloud Link TEE[1]	Pingtouge Semiconductor	✓	✓	proprietary	
iTrustee[2]	Huawei			proprietary	Formerly Secure Core
Kinibi[3]	Trustonic	✓		proprietary	Formerly Mobicore and <t-base
OP-TEE[4]	Linaro			BSD 2-Clause	
ProvenCore[5]	Prove & Run			proprietary	
Qualcomm TEE[6]	Qualcomm			proprietary	Formerly QSEE
Rocky Core[7]	Suzhou Rong Card Intelligent Technology	✓		proprietary	
SecuriTEE[8]	Solacia			proprietary	
SierraTEE[9]	SierraWare			proprietary	
T6[10]	TrustKernel			GNU GPL	
TEEGRIS[11]	Samsung		✓	proprietary	GP certification for Mediatek MT6737T
TLK NVIDIA Corporation, 2015	Nvidia			MIT	
Trusty[12]	Google			MIT	
TURBOTEE[13]	Eastcompeace Technologies	✓		proprietary	
Yunos TEE	Taobao Software	✓		proprietary	
WATCHTRUST[14]	Watchdata	✓	✓	proprietary	
Upteq NFC422 v1.0[15]	Gemalto (Thales Group)	✓		proprietary	
TrustZone-M					
Kinibi-M[16]	Trustonic	N/A		proprietary	
ProvenCore-M[17]	Prove & Run	N/A		proprietary	
RISC-V					
MultiZone[18]	Hex Five Security	N/A		MIT	

References

Abadi, M., M. Budiu, Ú. Erlingsson, and J. Ligatti. (2005). "Control-Flow Integrity". In: *Proceedings of the 2005 ACM Conference on Computer and Communications Security*. DOI: 10.1145/1102120.1102165.

Advanced Micro Devices, Inc. (2021). *AMD64 Architecture Programmer's Manual*. URL: https://www.amd.com/system/files/TechDocs/40332.pdf (accessed on 01/16/2022).

Akritidis, P., M. Costa, M. Castro, and S. Hand. (2009). "Baggy Bounds Checking: An Efficient and Backwards-Compatible Defense against Out-of-bounds Errors". In: *Proceedings of the 2009 USENIX Security Symposium*.

Alder, F. (2019). "TEE^2 – Combining Trusted Hardware to Enhance the Security of TEEs". Technical University of Darmstadt. URL: https://falder.org/tee2-thesis.pdf.

Alibaba. (2020). "Alibaba Cloud Released Industry's First Trusted and Virtualized Instance with Support for SGX 2.0 and TPM". URL: https://www.alibabacloud.com/blog/alibaba-cloud-released-industrys-first-trusted-and-virtualized-instance-with-support-for-sgx-2-0-and-tpm_596821 (accessed on 01/29/2021).

Alves, T. and D. Felton. (2004). "TrustZone: Integrated Hardware and Software Security". *Information Quarterly*. 3(4): 18–24.

Android Open Source Project. (2015). *Android 6.0 Compatibility Definition.*

Android Open Source Project. (2020a). *HWAddressSanitizer.* URL: https://source.android.com/devices/tech/debug/hwasan (accessed on 10/27/2020).

Android Open Source Project. (2020b). *Scudo.* URL: https://source.android.com/devices/tech/debug/scudo (accessed on 10/27/2020).

Apple. (2018). "Data protection". URL: https://support.apple.com/guide/security/data-protection-overview-secf6276da8a/web (accessed on 03/06/2022).

Apple Inc. (2018a). "Apple T2 Security Chip Security Overview". URL: https://www.apple.com/mac/docs/Apple_T2_Security_Chip_Overview.pdf.

Apple Inc. (2018b). "iOS Security — iOS 12". URL: https://www.apple.com/business/site/docs/iOS_Security_Guide.pdf.

Apple Inc. (2020a). *Preparing Your App to Work with Pointer Authentication.* URL: https://developer.apple.com/documentation/security/preparing_your_app_to_work_with_pointer_authentication (accessed on 10/27/2020).

Apple Inc. (2020b). *Swift.* URL: https://developer.apple.com/swift/ (accessed on 10/25/2020).

"Apple Platform Security". (2020). Apple.

Apple. Inc. (2020c). "About FaceID Advanced technology". URL: https://support.apple.com/en-gb/HT208108 (accessed on 03/06/2022).

Arbaugh, W. A., D. J. Farber, and J. M. Smith. (1997). "A secure and reliable bootstrap architecture". In: *Proceedings of the 1997 IEEE Symposium on Security and Privacy.* IEEE.

ARM Ltd. (2009). "ARM Security Technology - Building a Secure System using TrustZone Technology". URL: http://infocenter.arm.com/help/topic/com.arm.doc.prd29-genc-009492c.

ARM Ltd. (2016). "SMC Calling Convention". URL: http://infocenter.arm.com/help/topic/com.arm.doc.prd29-genc-009492c.

ARM Ltd. (2017a). "Power State Coordination Interface". URL: http://http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.den0022d.

ARM Ltd. (2017b). "Software Delegated Exception Interface". URL: http://http://infocenter.arm.com/help/topic/com.arm.doc.den0054a.

ARM Ltd. (2018a). "Arm TrustZone technology for ARMv8-M Architecture, Version 2.1". URL: https://static.docs.arm.com/100690/0201/armv8_m_architecture_trustzone_technology_100690_0201_01_en.pdf.

ARM Ltd. (2018b). "Arm® Platform Security Architecture Trusted Base System Architecture for Arm®v6-M, Arm®v7-M and Arm®v8-M 1.0". ARM. URL: https://armkeil.blob.core.windows.net/developer/Files/pdf/PlatformSecurityArchitecture/Architect/DEN0083-PSA_TBSA-M_1.0-bet1.pdf.

ARM Ltd. (2019a). "Armv8-M Architecture Reference Manual, Version A.k". URL: http://infocenter.arm.com/help/topic/com.arm.doc.ddi0553a.k/DDI0553A_k_armv8m_arm.pdf.

ARM Ltd. (2019b). "Armv8.5-A Memory Tagging Extension". *White Paper*.

ARM Ltd. (2019c). "Isolation using virtualization in the Secure world: Secure world software architecture on Armv8.4, Version 1.0". URL: https://developer.arm.com/-/media/Files/pdf/Isolation_using_virtualization_in_the_Secure_World_Whitepaper.pdf.

ARM Ltd. (2020a). "Armv8-A architecture reference manual, DDI 0487F.c". No. DDI 0487F.c.

ARM Ltd. (2021a). "Hardware Accelerated Crypto|Mbed OS 5 Documentation". URL: https://os.mbed.com/docs/mbed-os/v6.15/porting/hardware-accelerated-crypto.html.

ARM Ltd. (2021b). "Unlocking the power of data with ARM CCA". URL: https://community.arm.com/developer/ip-products/processors/b/processors-ip-blog/posts/unlocking-the-power-of-data-with-arm-cca (accessed on 03/06/2022).

Arm Ltd. (2020a). *Arm Morello Program*. Arm Developer. URL: https://developer.arm.com/architectures/cpu-architecture/a-profile/morello (accessed on 11/29/2020).

Arm Ltd. (2018). "FIPS 140-2 Non-Proprietary Security Policy". ARM. URL: https://csrc.nist.gov/CSRC/media/projects/cryptographic-module-validation-program/documents/security-policies/140sp3263.pdf.

"Arm® Platform Security Architecture Security Model 1.0". (2020). ARM Ltd.

"Arm® Platform Security Architecture Trusted Boot and Firmware Update 1.0". (2019). ARM Ltd.

Arthur, W. and D. Challener. (2015). *A Practical Guide to TPM 2.0: Using the Trusted Platform Module in the New Age of Security*. 1st. Berkely, CA, USA: Apress.

Asokan, N., K. Kostiainen, E. Reshetova, A.-R. Sadeghi, L. Davi, A. Dmitrienko, and S. Heuser. (2014). *Mobile Platform Security*. Vol. 9. *Synthesis Lectures on Information Security, Privacy, & Trust*. Morgan & Claypool Publishers. URL: http://search.ebscohost.com/login.aspx?direct=true&db=nlebk&AN=688023&site=ehost-live&authtype=sso&custid=ns192260.

Atsec information security corporation. (2016). "Cryptographic Module for Intel® vPro™ Platforms Security Engine Chipset". URL: https://csrc.nist.gov/CSRC/media/projects/cryptographic-module-validation-program/documents/security-policies/140sp2720.pdf (accessed on 07/18/2016).

Avanzi, R. (2017). "The QARMA Block Cipher Family. Almost MDS Matrices over Rings with Zero Divisors, Nearly Symmetric Even-Mansour Constructions with Non-Involutory Central Rounds, and Search Heuristics for Low-Latency S-Boxes". *IACR Transactions on Symmetric Cryptology*. 2017(1): 4–44. DOI: 10.13154/tosc.v2017.i1.4-44.

AWS. (2021). "FreeRTOS - Market leading RTOS (Real Time Operating System) for embedded systems with Internet of Things extensions". URL: https://www.freertos.org/ (accessed on 09/19/2021).

Azab, A. M., P. Ning, J. Shah, Q. Chen, R. Bhutkar, G. Ganesh, J. Ma, and W. Shen. (2014). "Hypervision Across Worlds: Real-time Kernel Protection from the ARM TrustZone Secure World". In: *Proceedings of the 2014 ACM Conference on Computer and Communications Security (CCS)*. 90–102. DOI: 10.1145/2660267.2660350.

Azad, B. (2019). *Project Zero: Examining Pointer Authentication on the iPhone XS*. URL: https://googleprojectzero.blogspot.com/2019/02/examining-pointer-authentication-on.html (accessed on 02/12/2020).

Bahmani, R., F. Brasser, G. Dessouky, P. Jauernig, M. Klimmek, A.-R. Sadeghi, and E. Stapf. (2021). "{CURE}: A Security Architecture with CUstomizable and Resilient Enclaves". In: *Proceedings of the 2021 USENIX Security Symposium*.

Bai, X. (2018). "The last line of defense: understanding and attacking Apple File System on iOS". URL: https://i.blackhat.com/eu-18/Thu-Dec-6/eu-18-Bai-The-Last-Line-Of-Defense-Understanding-And-Attacking-Apple-File-System-On-IOS.pdf (accessed on 03/06/2022).

Bar-El, H., H. Choukri, D. Naccache, M. Tunstall, and C. Whelan. (2006). "The Sorcerer's Apprentice Guide to Fault Attacks". *Proceedings of the IEEE*. 94(2): 370–382. DOI: 10.1109/JPROC.2005.862424.

Bar-El, H. (2002). "Security implications of hardware vs Software cryptographic modules". *Tech. rep.* Discretix Technologies.

Barker, E. and J. Kelsey. (2015). "Recommendation for Random Number Generation Using Deterministic Random Bit Generators". National Institute of Standards and Technology. DOI: 10.6028/NIST.SP.800-90Ar1.

Basse, F. (2016). "Amlogic S905 SoC: bypassing the (not so) Secure Boot to dump the BootROM". URL: https://fredericb.info/2016/10/amlogic-s905-soc-bypassing-not-so.html (accessed on 02/27/2022).

Baumann, A. (2017). "Hardware is the New Software". In: *Proceedings of the 2017 Workshop on Hot Topics in Operating Systems (HotOS)*. Whistler, BC, Canada: ACM. 132–137. DOI: 10.1145/3102980.3103002.

Beaupre, S. (2015). "TRUSTNONE". URL: http://theroot.ninja/disclosures/TRUSTNONE_1.0-11282015.pdf.

Beer, I. and S. Groß. (2021). *A deep dive into an NSO zero-click iMessage exploit: Remote Code Execution*. URL: https://googleprojectzero.blogspot.com/2021/12/a-deep-dive-into-nso-zero-click.html.

Bellom, Maxime Rossi and Melotti, Damiano and Teuwen, Philippe. (2021). "A Titan M Odyssey". URL: https://i.blackhat.com/EU-21/Wednesday/EU-21-Rossi-Bellom-2021_A_Titan_M_Odyssey-wp.pdf (accessed on 03/06/2022).

Beniamini, G. (2016a). *QSEE privilege escalation vulnerability and exploit (CVE-2015-6639)*. URL: https://bits-please.blogspot.com/2016/05/qsee-privilege-escalation-vulnerability.html (accessed on 03/01/2022).

Beniamini, G. (2016b). *TrustZone Kernel Privilege Escalation (CVE-2016-2431)*. URL: http://bits-please.blogspot.com/2016/06/trustzone-kernel-privilege-escalation.html (accessed on 03/01/2022).

Beniamini, G. (2017). *Trust Issues: Exploiting TrustZone TEEs*. URL: https://googleprojectzero.blogspot.com/2017/07/trust-issues-exploiting-trustzone-tees.html (accessed on 03/01/2022).

Berard, D. (2018). *Kinibi TEE: Trusted Application Exploitation*. URL: https://www.synacktiv.com/posts/exploit/kinibi-tee-trusted-application-exploitation.html (accessed on 03/01/2022).

Bernstein, D. J. (2005). "Cache-timing attacks on AES". URL: http://cr.yp.to/antiforgery/cachetiming-20050414.pdf.

Beyls, K. (2020). *[Llvm-Dev] Round Table on AArch64 Pauth ABI - Minutes. E-mail*. URL: http://lists.llvm.org/pipermail/llvm-dev/2020-October/145839.html (accessed on 10/27/2020).

Biondo, A., M. Conti, L. Davi, T. Frassetto, and A.-R. Sadeghi. (2018). "The Guard's Dilemma: Efficient Code-Reuse Attacks Against Intel SGX". In: *27th USENIX Security Symposium (USENIX Security 18)*. Baltimore, MD: USENIX Association. 1213–1227. URL: https://www.usenix.org/conference/usenixsecurity18/presentation/biondo.

Blazakis, D. (2011). *The Apple Sandbox*. URL: https://developer.android.com/guide/topics/permissions/overview (accessed on 02/01/2021).

Bletsch, T., X. Jiang, V. W. Freeh, and Z. Liang. (2011). "Jump-Oriented Programming: A New Class of Code-Reuse Attack". In: *Proceedings of the 2011 ACM Asia Conference on Information, Computer and Communications Security (ASIACCS)*. Hong Kong, China: ACM. 30–40. DOI: 10.1145/1966913.1966919.

Boak, D. G. (1973). *A History of U.S. Communications Security*. Vol. 1–2. National Security Agency.

Bock, C., F. Brasser, D. Gens, C. Liebchen, and A.-R. Sadeghi. (2019). "RIP-RH: Preventing Rowhammer-Based Inter-Process Attacks". In: *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security (ASIACCS). Asia CCS '19*. Auckland, New Zealand: Association for Computing Machinery. 561–572. DOI: 10.1145/3321705.3329827.

Borrello, P., D. C. D'Elia, L. Querzoni, and C. Giuffrida. (2021). "Constantine: Automatic Side-Channel Resistance Using Efficient Control and Data Flow Linearization". In: *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security (CCS)*. DOI: 10.1145/3460120.3484583.

Bourgeat, T., I. Lebedev, A. Wright, S. Zhang, Arvind, and S. Devadas. (2019). "MI6: Secure Enclaves in a Speculative Out-of-Order Processor". In: *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. Columbus, OH, USA. 42–56. DOI: 10.1145/3352460.3358310.

Bozzato, C., R. Focardi, and F. Palmarini. (2019). "Shaping the Glitch: Optimizing Voltage Fault Injection Attacks". *IACR Transactions on Cryptographic Hardware Embed. Syst.* 2019(2): 199–224. DOI: 10.13154/tches.v2019.i2.199-224.

Brasser, F., L. Davi, D. Gens, C. Liebchen, and A.-R. Sadeghi. (2017). "CAn't Touch This: Software-Only Mitigation against Rowhammer Attacks Targeting Kernel Memory". In: *Proceedings of the 2017 USENIX Conference on Security Symposium*. Vancouver, BC, Canada: USENIX Association. 117–130.

Brasser, F., B. El Mahjoub, A.-R. Sadeghi, C. Wachsmann, and P. Koeberl. (2015). "TyTAN: Tiny Trust Anchor for Tiny Devices". In: *Proceedings of the 2015 Annual Design Automation Conference (DAC)*. San Francisco, CA, USA. 34:1–34:6. DOI: 10.1145/2744769.2744922.

Brasser, F., D. Gens, P. Jauernig, A.-R. Sadeghia, and E. Stapf. (2019). "SANCTUARY: ARMing TrustZone with User-space Enclaves". In: *Proceedings of the 26th Annual Network and Distributed System Security Symposium (NDSS)*. S.

Brickell, E. and J. Li. (2009). "Enhanced Privacy ID from Bilinear Pairing". Cryptology ePrint Archive, Report 2009/095. URL: https://eprint.iacr.org/2009/095.

BSI. (2013). "Evaluation of random number generators". *Standard*. BSI.

Bulck, J. V., M. Minkin, O. Weisse, D. Genkin, B. Kasikci, F. Piessens, M. Silberstein, T. F. Wenisch, Y. Yarom, and R. Strackx. (2018a). "Foreshadow: Extracting the Keys to the Intel SGX Kingdom with Transient Out-of-Order Execution". In: *Proceedings of the 2018 USENIX Security Symposium*. Baltimore, MD: USENIX Association. 991–1008. URL: https://www.usenix.org/conference/usenixsecurity18/presentation/bulck.

Bulck, J. V., M. Minkin, O. Weisse, D. Genkin, B. Kasikci, F. Piessens, M. Silberstein, T. F. Wenisch, Y. Yarom, and R. Strackx. (2018b). "Foreshadow: Extracting the Keys to the Intel SGX Kingdom with Transient Out-of-Order Execution". In: *27th USENIX Security Symposium (USENIX Security 18)*. URL: https://www.usenix.org/conference/usenixsecurity18/presentation/bulck.

Carlini, N., A. Barresi, E. T. H. Zürich, M. Payer, D. Wagner, T. R. Gross, N. Carlini, A. Barresi, D. Wagner, and T. R. Gross. (2015). "Control-Flow Bending: On the Effectiveness of Control-Flow Integrity". In: *Proceedings of the 2015 USENIX Security Symposium*. Washington, DC, USA: USENIX Association. 161–176. URL: https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/carlini.

Carru, P. (2017). "Attack TrustZone with Rowhammer". URL: https://grehack.fr/data/2017/slides/GreHack17_Attack_TrustZone_with_Rowhammer.pdf.

Carter, N. P., S. W. Keckler, and W. J. Dally. (1994). "Hardware Support for Fast Capability-Based Addressing". *ACM SIGOPS Operating Systems Review*. 28(5): 319–327. DOI: 10.1145/381792.195579.

Cerdeira, D., N. Santos, P. Fonseca, and S. Pinto. (2020). "SoK: Understanding the Prevailing Security Vulnerabilities in TrustZone-assisted TEE Systems". In: *Proceedings of the 2020 IEEE Symposium on Security and Privacy (S&P)*. San Francisco, CA, USA. DOI: 10.1109/SP40000.2020.00061.

Charles Garcia-Tobin, ARM Ltd. (2021). "ARM CCA Hardware Architecture". URL: https://static.linaro.org/connect/armcca/presentations/CCATechEvent-210623-CGT-2.pdf (accessed on 03/06/2022).

Cheeseman, L. (2019). *D51429 [AArch64] Return Address Signing B Key Support*. URL: https://reviews.llvm.org/D51429 (accessed on 10/26/2020).

Chen, L., J. Franklin, and A. Regenscheid. (2012). "Guidelines on HardwareRooted Security in Mobile Devices". No. SP 800-16. Gaithersburg, MD, United States.

Chen, N. (2016). "The Benefits Of Antifuse OTP". *Semiconductor Engineering*. Dec. URL: https://semiengineering.com/the-benefits-of-antifuse-otp/ (accessed on 03/07/2022).

Cheng, L., H. Liljestrand, M. S. Ahmed, T. Nyman, T. Jaeger, N. Asokan, and D. Yao. (2019). "Exploitation Techniques and Defenses for Data-Oriented Attacks". In: *Proceedings of the 2019 IEEE Cybersecurity Development (SecDev)*. Tysons Corner, VA, USA. 114–128. DOI: 10.1109/SecDev.2019.00022.

Clang team. (2020). *Hardware-Assisted AddressSanitizer Design Documentation*. URL: https://clang.llvm.org/docs/HardwareAssistedAddressSanitizerDesign.html (accessed on 09/06/2020).

Cormen, T. H., C. E. Leiserson, R. L. Rivest, and C. Stein. (2009). *Introduction to Algorithms*. 3rd. MIT Press.

Costan, V., I. Lebedev, and S. Devadas. (2016). "Sanctum: Minimal Hardware Extensions for Strong Software Isolation". In: *Proceedings of the 2016 USENIX Security Symposium*. Austin, TX: USENIX Association. 857–874. URL: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/costan.

Cowan, C., S. Beattie, R. F. Day, C. Pu, P. Wagle, and E. Walthinsen. (1999). "Protecting Systems from Stack Smashing Attacks with StackGuard". In: *Linux Expo*.

Cryptomathic. (2017). "EMV Key Management – Explained". URL: https://www.cryptomathic.com/hubfs/docs/cryptomathic_white_paper-emv_key_management.pdf.

Davis, D., R. Ihaka, and P. Fenstermacher. (1994). "Cryptographic Randomness from Air Turbulence in Disk Drives". In: *Proceedings of Advances in Cryptology — CRYPTO '94*. Ed. by Y. G. Desmedt. Berlin, Heidelberg: Springer Berlin Heidelberg. 114–120.

Denis-Courmont, R., H. Liljestrand, C. Chinea, and J.-E. Ekberg. (2020). "Camouflage: Hardware-Assisted CFI for the ARM Linux Kernel". en. *Proceedings of the 2020 ACM/IEEE Annual Design Automation Conference (DAC)*.

Dennis, J. B. (1965). "Segmentation and the design of multiprogrammed computer systems". *Journal of the ACM (JACM)*. 12(4): 589–602.

Dent, A. W. (2012). "Secure Boot and Image Authentication". No. SP 800-16.

Dmitrienko, A., S. Heuser, T. D. Nguyen, M. d. Silva Ramos, A. Rein, and A.-R. Sadeghi. (2015). "Market-driven code provisioning to mobile secure hardware". In: *Proceedings of the 2015 International Conference on Financial Cryptography and Data Security (FC)*. Springer. 387–404.

Douceur, J. R. (2002). "The Sybil Attack". In: *Proceedings the 2002 International Workshop on Peer-to-Peer Systems (IPTPS)*. Ed. by P. Druschel, M. F. Kaashoek, and A. I. T. Rowstron. Vol. 2429. *Lecture Notes in Computer Science*. Cambridge, MA, USA: Springer. 251–260. DOI: 10.1007/3-540-45748-8_24.

Eck, W. van. (1985). "Electromagnetic radiation from video display units: An eavesdropping risk?" *Computers & Security*. 4(4): 269–286. DOI: 10.1016/0167-4048(85)90046-x.

Ekberg, J.-E., K. Kostiainen, and N. Asokan. (2014). "The Untapped Potential of Trusted Execution Environments on Mobile Devices". *IEEE Security & Privacy Magazine*. 12(4): 29–37. DOI: 10.1109/MSP.2014.38.

Ekberg, J.-E. (2013). "Securing Software Architectures for Trusted Processor Environments; Programvarusystem för säkra processorarkitekturer". en. *Ph.D Thesis*. 91 + app. 139. URL: http://urn.fi/URN:ISBN:978-952-60-3632-8.

Ekberg, J.-E. and N. Asokan. (2009). "External authenticated non-volatile memory with lifecycle management for state protection in trusted computing". In: *Proceedings of the 2009 International Conference on Trusted Systems (INTRUST)*. Springer. 16–38.

Ekberg, J.-E. and N. Asokan. (2010). "External Authenticated Non-volatile Memory with Lifecycle Management for State Protection in Trusted Computing". In: *Proceedings of the 2010 International Conference on Trusted Systems (INTRUST)*. Berlin, Heidelberg: Springer Berlin Heidelberg. 16–38. DOI: 10.1007/978-3-642-14597-1_2.

Eldefrawy, K., A. Francillon, D. Perito, and G. Tsudik. (2012). "SMART: Secure and Minimal Architecture for (Establishing a Dynamic) Root of Trust". In: *Proceedings of the 2012 Annual Network and Distributed System Security Symposium (NDSS)*. URL: http://www.eurecom.fr/publication/3536.

ETSI. (2012). "UICC Application Programming Interface for Java Card, Release 11". URL: https://www.etsi.org/deliver/etsi_ts/102200_102299/102241/11.00.00_60/ts_102241v110000p.pdf.

Fabry, R. S. (1973). "Dynamic verification of operating system decisions". *Communications of the ACM*. 16(11): 659–668.

Filardo, N. W., B. F. Gutstein, J. Woodruff, S. Ainsworth, L. Paul-Trifu, B. Davis, H. Xia, E. T. Napierala, A. Richardson, J. Baldwin, D. Chisnall, J. Clarke, K. Gudka, A. Joannou, A. T. Marekttos, A. Mazzinghi, R. M. Norton, M. Roe, P. Sewell, S. Son, T. M. Jones, S. W. Moore, P. G. Neumann, and R. N. M. Watson. (2020). "Cornucopia: Temporal Safety for CHERI Heaps". In: *Proceedings of the 2020 IEEE Symposium on Security and Privacy (SP)*. San Francisco, CA, USA.

"Security Requirements for Cryptographic Modules". (2019). NIST. DOI: 10.6028/NIST.FIPS.140-3.

Freivalds, R. (1977). "Probabilistic Machines Can Use Less Running Time". In: *Proceedings of the 1977 IFIP Congress*. Toronto, Canada.

Gallagher, M., L. Biernacki, S. Chen, Z. B. Aweke, S. F. Yitbarek, M. T. Aga, A. Harris, Z. Xu, B. Kasikci, V. Bertacco, S. Malik, M. Tiwari, and T. Austin. (2019). "Morpheus: A Vulnerability-Tolerant Secure Architecture Based on Ensembles of Moving Target Defenses with Churn". In: *Proceedings of the 2019 International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*. New York, NY, USA: Association for Computing Machinery. 469–484. DOI: 10.1145/3297858.3304037. (Accessed on 02/24/2021).

GCC Team. (2018). *GCC 9 Release Series — Changes, New Features, and Fixes*. URL: https://gcc.gnu.org/gcc-9/changes.html (accessed on 10/26/2020).

GCC Wiki. (2018). *Intel® Memory Protection Extensions (Intel® MPX) Support in the GCC Compiler*. URL: https://gcc.gnu.org/wiki/Intel%20MPX%20support%20in%20the%20GCC%20compiler (accessed on 10/28/2020).

General Dynamics Mission Systems. (2015a). "Advanced INFOSEC Machine (AIM)". URL: https://gdmissionsystems.com/-/media/General-Dynamics/Cyber-and-Electronic-Warfare-Systems/PDF/Brochures/cyber-advanced-infosec-machine-aim-datasheet.ashx (accessed on 03/06/2022).

General Dynamics Mission Systems. (2015b). "AIM II — Embeddable Programmable Security)". URL: https://gdmissionsystems.com/-/media/General-Dynamics/Cyber-and-Electronic-Warfare-Systems/PDF/Brochures/cyber-aim2-embeddable-programmable-security-datasheet.ashx.

Gil, R., H. Okhravi, and H. Shrobe. (2018a). "There's a Hole in the Bottom of the C: On the Effectiveness of Allocation Protection". In: *Proceedings of the 2018 IEEE Cybersecurity Development. SecDev '18*. Cambridge, MA, USA. 102–109. DOI: 10.1109/SecDev.2018.00021.

Gil, R., H. Okhravi, and H. Shrobe. (2018b). "There's a Hole in the Bottom of the C: On the Effectiveness of Allocation Protection". In: *Proceedings of the 2018 IEEE Cybersecurity Development*. 2018 IEEE Cybersecurity Development. *SecDev '18*. Cambridge, MA, USA: IEEE. 102–109. DOI: 10.1109/SecDev.2018.00021. (Accessed on 03/25/2019).

GlobalPlatform. (2010). "TEE Client API Specification, Version 1.0". URL: https://globalplatform.org/specs-library/tee-client-api-specification/.

GlobalPlatform. (2013). "TEE Trusted User Interface API v1.0". URL: https://globalplatform.org/specs-library/trusted-user-interface-api-v1/.

GlobalPlatform. (2016). "TEE Management Framework, Version 1.0". URL: https://globalplatform.org/specs-library/tee-management-framework-including-asn1-profile/.

GlobalPlatform. (2017). "TEE Sockets API Specification v1.0.1, 1.0.2 & 1.0.3". URL: https://globalplatform.org/specs-library/tee-sockets-api-specification/.

GlobalPlatform. (2018a). "Card Specification v2.3.1". URL: https://globalplatform.org/specs-library/card-specification-v2-3-1/.

GlobalPlatform. (2018b). "Root of Trust Definitions and Requirements v1.1". URL: https://globalplatform.org/specs-library/globalplatform-root-of-trust-definitions-and-requirements/.

GlobalPlatform. (2018c). "TEE System Architecture, Version 1.2". URL: https://globalplatform.org/specs-library/tee-system-architecture-v1-2/.

GlobalPlatform. (2018d). "TEE Trusted User Interface Lowe-level API v1.0.1". URL: https://globalplatform.org/specs-library/globalplatform-technology-tee-trusted-user-interface-low-level-api-v1-0-1/.

GlobalPlatform. (2019a). "Secure Channel Protocol '03' - Amendment D v1.1.2". URL: https://globalplatform.org/specs-library/secure-channel-protocol-03-amendment-d-v1-1-2/.

GlobalPlatform. (2019b). "TEE Management Framework: Open Trust Protocol (OTrP) Profile v1.0". URL: https://globalplatform.org/specs-library/tee-management-framework-open-trust-protocol.

GlobalPlatform. (2021a). "TEE Internal Core API Specification, Version 1.3.1". URL: https://globalplatform.org/specs-library/tee-internal-core-api-specification/.

GlobalPlatform. (2021b). "TEE Secure ElementAPI v1.1.2". URL: https://globalplatform.org/specs-library/tee-secure-element-api/.

GNU. (2018). *GCC 8.4 Manual*. URL: https://gcc.gnu.org/onlinedocs/gcc-8.4.0/gcc (accessed on 11/10/2020).

Google. (2016). "Pixel Security: Better, Faster, Stronger". URL: https://blog.google/products/android-enterprise/pixel-security-better-faster-stronger/ (accessed on 03/06/2022).

Google. (2020a). *Android permissions*. URL: https://developer.android.com/guide/topics/permissions/overview (accessed on 02/01/2021).

Google. (2020b). "DM-verity". URL: https://source.android.com/security/verifiedboot/dm-verity (accessed on 03/06/2022).

Google. (2020c). "FS-verity". URL: https://www.kernel.org/doc/html/latest/filesystems/fsverity.html (accessed on 03/06/2022).

Google. (2020d). *Kotlin and Android*. URL: https://developer.android.com/kotlin (accessed on 10/25/2020).

Google. (2021). *Security-Enhanced Linux in Android*. URL: https://source.android.com/security/selinux (accessed on 02/01/2021).

Graham, R. M. (1968). "Protection in an information processing utility". *Communications of the ACM*. 11(5): 365–369.

GSMA. (2014). "Generic Overlay SIM Security Assessment?" URL: https://www.gsma.com/publicpolicy/wp-content/uploads/2014/08/GSMA-Security-Group-Overlay_SIM_Security_Assessment_August_18_2014.pdf.

GSMA. (2015). "Understanding SIM Evolution". URL: https://www.gsmaintelligence.com/research/?file=81d866ecda8b80aa4642e06b877ec265 (accessed on 03/06/2022).

GSMA. (2019). "IMEI Blacklisting". URL: https://www.gsma.com/security/resources/imei-blacklisting/.

Gueron, S. (2016). "A Memory Encryption Engine Suitable for General Purpose Processors". Cryptology ePrint Archive, Report 2016/204.

Halderman, J. A., S. D. Schoen, N. Heninger, W. Clarkson, W. Paul, J. A. Calandrino, A. J. Feldman, J. Appelbaum, and E. W. Felten. (2009). "Lest We Remember: Cold-boot Attacks on Encryption Keys". *Communications of the ACM*. 52(5): 91–98. DOI: 10.1145/1506409.1506429.

Hasarfaty, S. and Y. Moyal. (2019). "Behind the Scenes of Intel Security and Manageability Engine". URL: https://i.blackhat.com/USA-19/Wednesday/us-19-Hasarfaty-Behind-The-Scenes-Of-Intel-Security-And-Manageability-Engine.pdf (accessed on 03/06/2022).

Holman, W., J. Connelly, and A. Dowlatabadi. (1997). "An integrated analog/digital random noise source". *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*. 44(6): 521–528. DOI: 10.1109/81.586025.

Houdek, M. E., F. G. Soltis, and R. L. Hoffman. (1981). "IBM System/38 Support for Capability-Based Addressing". In: *Proceedings of the 1981 Annual Symposium on Computer Architecture (ISCA)*. Washington DC, USA: IEEE Computer Society Press. 341–348. DOI: 10.5555/800052.801885.

Hu, H., S. Shinde, S. Adrian, Z. L. Chua, P. Saxena, and Z. Liang. (2016). "Data-Oriented Programming: On the Expressiveness of Non-Control Data Attacks". In: *Proceedings of the 2016 IEEE Symposium on Security and Privacy (SP)*. San Jose, CA, USA. 969–986. DOI: 10.1109/SP.2016.62.

Huawei. (2020). "Huawei EMUI Security Whitepaper". URL: https://consumer-img.huawei.com/content/dam/huawei-cbg-site/common/campaign/privacy/whitepaper/emui-10-security-technical-white-paper-v1.pdf (accessed on 03/06/2022).

IBM. (1977). "3845/3846 Data Encryption Devices". URL: http://ed-thelen.org/comp-hist/IBM-ProdAnn/3845.pdf (accessed on 03/06/2022).

Intel. (2019). "Control-Flow Enforcement Technology Specification (Revision 3.0)". 358. URL: https://software.intel.com/sites/default/files/managed/4d/2a/control-flow-enforcement-technology-preview.pdf (accessed on 11/09/2020).

Intel Corporation. (2012). "Intel Atom Processor Z2760 Datasheet". URL: https://www.intel.com/content/dam/www/public/us/en/documents/product-briefs/atom-z2760-datasheet.pdf (accessed on 03/06/2022).

Intel Corporation. (2019). "Proof of Elapsed Time". URL: https://github.com/hyperledger/sawtooth-poet (accessed on 03/06/2022).

ISO. (2011). "Information technology — Security techniques — Random bit generation". *Standard*. ISO.

Jakobsson, M. and A. Juels. (1999). "Proofs of Work and Bread Pudding Protocols". In: *Proceedings of the 2019 IFIP TC6/TC11 Joint Working Conference on Communications and Multimedia Security (CMS)*. Ed. by B. Preneel. Vol. 152. Leuven, Belgium: Kluwer. 258–272.

Jakobsson, M., E. Shi, P. Golle, and R. Chow. (2009). "Implicit authentication for mobile devices". In: *Proceedings of the 2009 USENIX Conference on Hot topics in Security*. Vol. 1. USENIX Association. 25–27.

Jang, I., A. Tang, T. Kim, S. Sethumadhavan, and J. Huh. (2019). "Heterogeneous Isolated Execution for Commodity GPUs". In: *Proceedings of the 2019 International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*. Providence, RI, USA: ACM. 455–468. DOI: 10.1145/3297858.3304021.

Kim, Y., R. Daly, J. Kim, C. Fallin, J. H. Lee, D. Lee, C. Wilkerson, K. Lai, and O. Mutlu. (2014). "Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors". In: *Proceeding of the 2014 Annual International Symposium on Computer Architecuture (ISCA)*.

Kocher, P., J. Horn, A. Fogh, D. Genkin, D. Gruss, W. Haas, M. Hamburg, M. Lipp, S. Mangard, T. Prescher, M. Schwarz, and Y. Yarom. (2019). "Spectre Attacks: Exploiting Speculative Execution". In: *Proceedings of the 2019 IEEE Symposium on Security and Privacy (SP)*. 1–19. DOI: 10.1109/SP.2019.00002.

Kocher, P., J. Jaffe, and B. Jun. (1999). "Differential Power Analysis". In: *Proceedings of 1999 Advances in Cryptology (CRYPTO)*. Ed. by M. Wiener. Berlin, Heidelberg: Springer Berlin Heidelberg. 388–397. DOI: 10.1007/3-540-48405-1_25.

Kocher, P., J. Jaffe, B. Jun, and P. Rohatgi. (2011). "Introduction to differential power analysis". *Journal of Cryptographic Engineering*. 1(Apr.): 5–27. DOI: 10.1007/s13389-011-0006-y.

Koeberl, P., S. Schulz, A.-R. Sadeghi, and V. Varadharajan. (2014). "TrustLite: A Security Architecture for Tiny Embedded Devices". In: *Proceedings of the 2014 European Conference on Computer Systems (EuroSys)*. Amsterdam, The Netherlands: ACM. 10:1–10:14. DOI: 10.1145/2592798.2592824.

Koeune, F. and F.-X. Standaert. (2005). "A Tutorial on Physical Security and Side-Channel Attacks". In: *Foundations of Security Analysis and Design III: FOSAD 2004/2005 Tutorial Lectures*. Ed. by A. Aldini, R. Gorrieri, and F. Martinelli. Berlin, Heidelberg: Springer Berlin Heidelberg. 78–108. DOI: 10.1007/11554578_3.

Kömmerling, O. and M. G. Kuhn. (1999). "Design Principles for Tamper-Resistant Smartcard Processors". In: *Proceedings of the 1999 USENIX Workshop on Smartcard Technology*. URL: https://www.usenix.org/conference/usenix-workshop-smartcard-technology/design-principles-tamper-resistant-smartcard.

Koning, K., X. Chen, H. Bos, C. Giuffrida, and E. Athanasopoulos. (2017). "No Need to Hide: Protecting Safe Regions on Commodity Hardware". In: *Proceedings of the 2017 European Conference on Computer Systems*. Belgrade, Serbia: ACM. 437–452. DOI: 10.1145/3064176.3064217.

Krajci, I. and D. Cummings. (2013). "The Intel Mobile Processor". In: *Android on x86: An Introduction to Optimizing for Intel® Architecture*. Berkeley, CA: Apress. 33–46. DOI: 10.1007/978-1-4302-6131-5_5.

Kuhn, M. G. and R. J. Anderson. (1998). "Soft tempest: Hidden data transmission using electromagnetic emanations". In: *Proceedings of the 1998 International Workshop on Information Hiding*. Springer. 124–142.

Kuvaiskii, D., O. Oleksenko, S. Arnautov, B. Trach, P. Bhatotia, P. Felber, and C. Fetzer. (2017). "SGXBOUNDS: Memory Safety for Shielded Execution". In: *Proceedings of the 2017 European Conference on Computer Systems (EuroSys)*. Belgrade, Serbia: ACM. 205–221. DOI: 10.1145/3064176.3064192.

Lakos, C. A. (1980). "Implementing BCPL on the Burroughs B6700". *Software: Practice and Experience*. 10(8): 673–683. DOI: 10.1002/spe.4380100806.

Lampson, B. W. (1974). "Protection". *ACM SIGOPS Operating Systems Review*. 8(1): 18–24.

Lange, N. (1997). "Single-chip implementation of a cryptosystem for financial applications". In: *Financial Cryptography*. Ed. by R. Hirschfeld. Berlin, Heidelberg: Springer Berlin Heidelberg. 135–144.

Larabel, M. (2020a). *Intel Confirms CET Security Support For Tiger Lake*. URL: https://www.phoronix.com/scan.php?page=news_item&px=Intel-CET-Tiger-Lake (accessed on 11/15/2020).

Larabel, M. (2020b). *Intel MPX Support Is Dead With Linux 5.6*. URL: https://www.phoronix.com/scan.php?page=news_item&px=Intel-MPX-Is-Dead (accessed on 11/15/2020).

Laurie, A., J. McMaster, C. Morgan, and J. Exum. (2013). "rompar: Masked ROM optical data extraction tool". URL: https://github.com/AdamLaurie/rompar (accessed on 08/12/2020).

Lee, D., D. Kohlbrenner, S. Shinde, K. Asanović, and D. Song. (2020). "Keystone: An Open Framework for Architecting Trusted Execution Environments". In: *Proceedings of the 2020 European Conference on Computer Systems (EuroSys)*. Heraklion, Greece: Association for Computing Machinery. DOI: 10.1145/3342195.3387532.

Liljestrand, H., T. Nyman, K. Wang, C. C. Perez, J.-E. Ekberg, and N. Asokan. (2019). "PAC It up: Towards Pointer Integrity Using ARM Pointer Authentication". en. In: *Proceedings of the 2019 USENIX Security Symposium*. Santa Clara, CA, USA: USENIX Association. 177–194.

Lin, J. (2021). *Developoer Guidance for Hardware-enforced Stack Protection — Microsoft Tech Community*. URL: https://techcommunity.microsoft.com/t5/windows-kernel-internals-blog/developer-guidance-for-hardware-enforced-stack-protection/ba-p/2163340 (accessed on 01/16/2022).

Lipp, M., D. Gruss, R. Spreitzer, C. Maurice, and S. Mangard. (2016). "ARMageddon: Cache Attacks on Mobile Devices". In: *Proceedings of the 2016 USENIX Security Symposium*. 25th USENIX Security Symposium. Austin, TX, USA. 549–564. URL: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/lipp (accessed on 03/06/2022).

Lipp, M., M. Schwarz, D. Gruss, T. Prescher, W. Haas, A. Fogh, J. Horn, S. Mangard, P. Kocher, D. Genkin, Y. Yarom, and M. Hamburg. (2018). "Meltdown: Reading Kernel Memory from User Space". In: *Proceedings of the 2018 USENIX Conference on Security Symposium*. Baltimore, MD, USA: USENIX Association. 973–990.

LLVM. (2020). *Scudo Hardened Allocator*. URL: https://llvm.org/docs/ScudoHardenedAllocator.html (accessed on 10/27/2020).

Loscocco, P. and S. Smalley. (2001). "Integrating Flexible Support for Security Policies into the Linux Operating System." In: *Proceedings of the 2001 USENIX Annual Technical Conference*. 29–42.

Machiry, A., E. Gustafson, C. Spensky, C. Salls, N. Stephens, R. Wang, A. Bianchi, Y. R. Choe, C. Kruegel, and V. Giovanni. (2017). "BOOMERANG: Exploiting the Semantic Gap in Trusted Execution Environments". In: *Proceedings of the 2017 Annual Network and Distributed System Security Symposium (NDSS)*. S. DOI: 10.14722/ndss.2017.23227.

Mandt, T., M. Solni, and D. Wang. (2016a). "Demystifying the Secure Enclave Processor". URL: http://mista.nu/research/sep-paper.pdf (accessed on 03/06/2022).

Mandt, T., M. Solni, and D. Wang. (2016b). "Demystifying the Secure Enclave Processor". URL: https://www.blackhat.com/docs/us-16/materials/us-16-Mandt-Demystifying-The-Secure-Enclave-Processor.pdf (accessed on 03/06/2022).

Martsenko, K. (2020). *Arm64: Compile the Kernel with Ptrauth Return Address Signing*. URL: https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=74afda4016a7437e6e425c3370e4b93b47be8ddf (accessed on 10/26/2020).

Mashtizadeh, A. J., A. Bittau, D. Boneh, and D. Mazières. (2015). "CCFI: Cryptographically Enforced Control Flow Integrity". en. In: *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. Denver, CO, USA: ACM Press. 941–951. DOI: 10.1145/2810103.2813676.

Maslowski, D. (2020). "Look at ME! Intel ME Firmware Investigation". FOSDEM 2020. URL: https://archive.fosdem.org/2020/schedule/event/firmware_lam/attachments/slides/3872/export/events/attachments/firmware_lam/slides/3872/look_at_me_fosdem20.pdf (accessed on 03/07/2022).

Matala, S., T. Nyman, and N. Asokan. (2019). "Historical insight into the development of Mobile TEEs". URL: https://blog.ssg.aalto.fi/2019/06/historical-insight-into-development-of.html (accessed on 03/06/2022).

Mbiti, I. and D. N. Weil. (2011). "Mobile banking: The impact of MPesa in Kenya". *Working Paper* No. 011-13. Brown University, Department of Economics. URL: http://hdl.handle.net/10419/62662.

McCune, J. M., B. Parno, A. Perrig, M. K. Reiter, and H. Isozaki. (2008). "Flicker: An Execution Infrastructure for TCB Minimization". In: *Proceedings the 2008 ACM SIGOPS/EyroSys European Conference on Computer Systems*. Glasgow, UK. 315–328. DOI: 10.1145/3246965.

McKeen, F., I. Alexandrovich, I. Anati, D. Caspi, S. Johnson, R. Leslie-Hurd, and C. Rozas. (2016). "Intel Software Guard Extensions (Intel SGX) Support for Dynamic Memory Management Inside an Enclave". In: *Proceedings of the 2016 International Workshop on Hardware and Architectural Support for Security and Privacy (HASP). HASP 2016*. Seoul, Republic of Korea: ACM. 10:1–10:9. DOI: 10.1145/2948618.2954331.

McKeen, F., I. Alexandrovich, A. Berenzon, C. V. Rozas, H. Shafi, V. Shanbhogue, and U. R. Savagaonkar. (2013). "Innovative Instructions and Software Model for Isolated Execution". In: *Proceedings of the 2013 International Workshop on Hardware and Architectural Support for Security and Privacy (HASP)*. Tel-Aviv, Israel: Association for Computing Machinery. DOI: 10.1145/2487726.2488368.

McReynolds, M. (2021). "Azure announces next generation Intel SGX confidential computing VMs". URL: https://techcommunity.microsoft.com/t5/azure-confidential-computing/azure-announces-next-generation-intel-sgx-confidential-computing/ba-p/2839934 (accessed on 01/29/2021).

Mehrabi Koushki, M., B. Obada-Obieh, J. H. Huh, and K. Beznosov. (2020). "Is Implicit Authentication on Smartphones Really Popular? On Android Users' Perception of "Smart Lock for Android"". In: *Proceedings of the 2020 International Conference on Human-Computer Interaction with Mobile Devices and Services*. 1–17.

Microsoft. (2006). *A Detailed Description of the Data Execution Prevention (DEP) Feature in Windows XP Service Pack 2, Windows XP Tablet PC Edition 2005, and Windows Server 2003*. URL: https://support.microsoft.com/en-us/help/875352/a-detailed-description-of-the-data-execution-prevention-dep-feature-in (accessed on 09/05/2019).

Miettinen, M., S. Heuser, W. Kronz, A.-R. Sadeghi, and N. Asokan. (2014). "Conxsense: automated context classification for context-aware access control". In: *Proceedings of the 9th ACM Asia Symposium on Information, Computer and Communications security (ASIACCS)*. 293–304. DOI: 10.1145/2590296.2590337.

Minsky, M. (1988). "Memoir on Inventing the Confocal Scanning Microscope". *Scanning*. 10: 128–138.

Modadugu, N. and B. Richardson. (2018). "Building a Titan: Better security through a tiny chip". URL: https://security.googleblog.com/2018/10/building-titan-better-security-through.html (accessed on 03/06/2022).

Mohan, V., P. Larsen, S. Brunthaler, K. W. Hamlen, and M. Franz. (2015). "Opaque Control-Flow Integrity". In: *Proceedings of the 2015 Network and Distributed System Security Symposium*. San Diego, CA, USA: Internet Society. DOI: 10.14722/ndss.2015.23271. (Accessed on 10/07/2019).

Mondato. (2014). "Skin SIM Technology: A Serious Challenge for Safaricom?" URL: https://blog.mondato.com/skin-sim-safari/ (accessed on 03/06/2022).

Morris, J., S. Smalley, and G. Kroah-Hartman. (2002). "Linux Security Modules: General security support for the Linux kernel". In: *Proceedings of the 2002 USENIX Security Symposium*. ACM Berkeley, CA. 17–31.

Mutlu, O., S. Ghose, J. Gómez-Luna, and R. Ausavarungnirun. (2020). "A Modern Primer on Processing in Memory". *arXiv preprint arXiv:2012.03112*.

Nagarakatte, S., J. Zhao, M. M. Martin, and S. Zdancewic. (2009). "SoftBound: Highly Compatible and Complete Spatial Memory Safety for C". In: *Proceedings of the 2009 ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*. Dublin, Ireland: ACM. 245–258. DOI: 10.1145/1542476.1542504.

National Geographic. (1947). "First computer bug". URL: https://www.nationalgeographic.org/thisday/sep9/worlds-first-computer-bug/.

Neustadter, D. (2020). "True Random Number Generators for Heightened Security in Any SoC". URL: https://www.synopsys.com/designware-ip/technical-bulletin/true-random-number-generator-security-2019q3.html.

Nguyen, P. Q. and I. E. Shparlinski. (2003). "The Insecurity of the Elliptic Curve Digital Signature Algorithm with Partially Known Nonces". *Designs, Codes and Cryptography*. 30(2): 201–217. DOI: 10.1023/A:1025436905711.

Nguyen and Shparlinski. (2002). "The Insecurity of the Digital Signature Algorithm with Partially Known Nonces". *Journal of Cryptology*. 15(3): 151–176. DOI: 10.1007/s00145-002-0021-3.

NVIDIA Corporation. (2015). "Trusted Little Kernel (TLK) for Tegra: FOSS Edition". URL: http://nv-tegra.nvidia.com/gitweb/?p=3rdparty/ote_partner/tlk.git;a=blob_plain;f=documentation/Tegra_BSP_for_Android_TLK_FOSS_Reference.pdf;hb=HEAD.

Nyman, T., G. Dessouky, S. Zeitouni, A. Lehikoinen, A. Paverd, N. Asokan, and A.-R. Sadeghi. (2019). "HardScope: Hardening Embedded Systems Against Data-Oriented Attacks". In: *Proceedings of the 2019 Annual Design Automation Conference (DAC)*. Las Vegas, NV, USA: ACM. 63. DOI: 10.1145/3316781.3317836.

Nyman, T., J.-E. Ekberg, L. Davi, and N. Asokan. (2017). "CFI CaRE: Hardware-Supported Call and Return Enforcement for Commercial Microcontrollers". In: *Proceedings of the 2017 Symposium on Research in Attacks, Intrusions, and Defenses (RAID)*. Ed. by M. Dacier, M. Bailey, M. Polychronakis, and M. Antonakakis. Cham: Springer International Publishing. 259–284. DOI: 10.1007/978-3-319-66332-6_12.

Oleksenko, O., D. Kuvaiskii, P. Bhatotia, P. Felber, and C. Fetzer. (2019). "Intel MPX Explained: A Cross-Layer Analysis of the Intel MPX System Stack". *Proceedings of the ACM on Measurement and Analysis of Computing Systems*. 2(2): 28:1–28:30. DOI: 10.1145/3224423.

Osvik, D. A., A. Shamir, and E. Tromer. (2006). "Cache Attacks and Countermeasures: The Case of AES". In: *Proceedings of the 2006 CT-RSA*. Red. by D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, and G. Weikum. Vol. 3860. Berlin, Heidelberg: Springer Berlin Heidelberg. 1–20. DOI: 10.1007/11605805_1. (Accessed on 02/27/2022).

Page, B. (1988). "A Report on the Internet Worm". URL: https://www.ee.ryerson.ca/~elf/hack/iworm.html (accessed on 03/06/2022).

Patterson, D. A. and J. L. Hennessy. (2005). *Computer Organization and Design*. 3rd ed. Morgan Kaufmann.

Paul, G. and J. Irvine. (2015). "Take Control of Your PC with UEFI Secure Boot". *Linux Journal*. 2015(257).

PaX Team. (2006). *PaX PAGEEXEC Documentation*. URL: https://pax.grsecurity.net/docs/pageexec.txt (accessed on 11/15/2020).

Pearson, S. (2002). *Trusted Computing Platforms: TCPA Technology in Context*. Upper Saddle River, NJ, USA: Prentice Hall PTR.

Pei, M., A. Atyeo, N. Cook, M. Yoo, and H. Tschofenig. (2019). "The Open Trust Protocol (OTrP)". *Internet-Draft* No. draft-ietf-teep-opentrustprotocol-03. IETF Secretariat. URL: http://www.ietf.org/internet-drafts/draft-ietf-teep-opentrustprotocol-03.txt.

Peslyak, A. (D. (1997). *Getting around Non-Executable Stack (and Fix)*. URL: https://seclists.org/bugtraq/1997/Aug/63 (accessed on 09/05/2019).

Piessens, F. (2020). "Security across abstraction layers: old and new examples". In: *Proceedings of the 2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. Genoa, Italy. DOI: 10.1109/EuroSPW51379.2020.00043.

Pinto, S. and N. Santos. (2019). "Demystifying Arm TrustZone: A Comprehensive Survey". *ACM Computing Surveys*. 51(6): 130:1–130:36. DOI: 10.1145/3291047.

Pomonis, M., T. Petsios, A. D. Keromytis, M. Polychronakis, and V. P. Kemerlis. (2017). "kRˆX: Comprehensive Kernel Protection against Just-in-Time Code Reuse". In: *Proceedings of the 2017 European Conference on Computer Systems (EuroSys)*. Belgrade, Serbia: ACM. 420–436. DOI: 10.1145/3064176.3064216.

Pyo, C. and G. Lee. (2002). "Encoding Function Pointers and Memory Arrangement Checking against Buffer Overflow Attack". en. In: *Information and Communications Security*. Vol. 2513. Berlin, Heidelberg: Springer Berlin Heidelberg. 25–36. DOI: 10.1007/3-540-36159-6_3.

Qualcomm. (2017). "Pointer Authentication on ARMv8.3: Design and Analysis of the New Software Security Instructions".

Qualcomm Technologies, I. (2019a). "Qualcomm Secure Processing Unit SPU230 Core Security Target Lite". URL: https://www.commoncriteriaportal.org/files/epfiles/1045b_pdf.pdf.

Qualcomm Technologies, I. (2019b). "Qualcomm SPU FIPS 140-2 Non-Proprietary Security Policy V1.3". URL: https://csrc.nist.gov/CSRC/media/projects/cryptographic-module-validation-program/documents/security-policies/140sp3549.pdf.

Quisquater, J. and D. Samyde. (2001). "ElectroMagnetic Analysis (EMA): Measures and Counter-Measures for Smart Cards". In: *Proceedings of the 2001 International Conference on Research in Smart Cards*.

Ravi, S., A. Raghunathan, and S. Chakradhar. (2004). "Tamper resistance mechanisms for secure embedded systems". In: *Proceedings of the 2004 International Conference on VLSI Design*. 605–611. DOI: 10.1109/ICVD.2004.1260985.

Reshetova, E., F. Bonazzi, and N. Asokan. (2017). "Randomization Can't Stop BPF JIT Spray". In: *Proceedings of the 2017 International Conference on Network and System Security. Lecture Notes in Computer Science*. Cham: Springer International Publishing. 233–247. DOI: 10.1007/978-3-319-64701-2_17.

Reshetova, E., H. Liljestrand, A. Paverd, and N. Asokan. (2018). "Toward Linux Kernel Memory Safety". *Software: Practice and Experience*. 48(12): 2237–2256. DOI: 10.1002/spe.2638.

Rosenberg, D. (2014). "QSEE Trustzone Kernel Integer Overflow Vulnerability". URL: https://www.blackhat.com/docs/us-14/materials/us-14-Rosenberg-Reflections-On-Trusting-TrustZone-WP.pdf.

Ruan, X. (2014). *Platform Embedded Security Technology Revealed: Safeguarding the Future of Computing with Intel Embedded Security and Management Engine*. 1st. Berkely, CA, USA: Apress.

Rutland, M. (2018). *Arm64: Enable Pointer Authentication*. URL: https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=04ca3204fa09f5f55c8f113b0072004a7b364ff4 (accessed on 10/26/2020).

Samsung. (2020a). "Samsung Knox". URL: https://docs.samsungknox.com/admin/whitepaper.

Samsung. (2020b). "Samsung Knox file encryption". URL: https://www.samsungknox.com/en/blog/samsung-knox-file-encryption-1-0-the-first-certified-integrated-dual-data-at-rest-solution-for-mobile-devices (accessed on 03/06/2022).

Sandhu, R. S. and P. Samarati. (1994). "Access control: principle and practice". *IEEE Communications Magazine*. 32(9): 40–48.

Santesson, S., M. Myers, R. Ankney, A. Malpani, S. Galperin, and D. C. Adams. (2013). "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP". RFC 6960. DOI: 10.17487/RFC6960.

Savagaonkar, U., N. Porter, N. Taha, B. Serebrin, and N. Mueller. (2017). "Titan in depth: Security in plaintext". URL: https://cloud.google.com/blog/products/gcp/titan-in-depth-security-in-plaintext (accessed on 03/06/2022).

Schroeder, M. D. and J. H. Saltzer. (1971). "A Hardware Architecture for Implementing Protection Rings". In: *Proceedings of the 1971 ACM Symposium on Operating Systems Principles (SOSP)*. Palo Alto, CA, USA: ACM. 42–54. DOI: 10.1145/800212.806498.

Sebastian, A., M. Le Gallo, R. Khaddam-Aljameh, and E. Eleftheriou. (2020). "Memory devices and applications for in-memory computing". *Nature Nanotechnology*. 15(7): 529–544. DOI: 10.1038/s41565-020-0655-z.

ARM Ltd. (2020b). *Security IP*. URL: https://developer.arm.com/ip-products/security-ip (accessed on 10/22/2020).

Arm Ltd. (2020b). *Security IP | CryptoCell-300 family – Arm Developer*. URL: https://developer.arm.com/ip-products/security-ip/cryptocell-300-family (accessed on 10/22/2020).

Arm Ltd. (2019). *Security IP | CryptoCell-700 Family – Arm Developer*. URL: https://developer.arm.com/ip-products/security-ip/cryptocell-700-family (accessed on 12/01/2019).

Serebryany, K., D. Bruening, A. Potapenko, and D. Vyukov. (2012). "AddressSanitizer: A Fast Address Sanity Checker". In: *Proceedings of the 2012 USENIX Annual Technical Conference (ATC)*. Boston, MA, USA: USENIX. 309–318. URL: https://www.usenix.org/conference/atc12/technical-sessions/presentation/serebryany.

Serebryany, K. (2019). "ARM Memory Tagging Extension and How It Improves C/C++ Memory Safety". *USENIX ;login:* 44(2): 12–16.

Shacham, H. (2007). "The Geometry of Innocent Flesh on the Bone: Return-into-libc Without Function Calls (on the x86)". In: *Proceedings of the 2007 ACM Conference on Computer and Communications Security (CCS)*. Alexandria, Virginia, USA: ACM. 552–561. DOI: 10.1145/1315245.1315313.

Shen, D. (2015). "Attacking your "Trusted Core" Exploiting TrustZone on Android". URL: https://www.blackhat.com/docs/us-15/materials/us-15-Shen-Attacking-Your-Trusted-Core-Exploiting-Trustzone-On-Android.pdf.

SIMalliance Ltd. (2013). "Device Implementation Guidelines version 1.1". URL: https://simalliance.org/wp-content/uploads/2015/03/SIMalliance_UICC_Device_Implementation_Guidelines-1.1.pdf .

Skorobogatov, S. P. and R. J. Anderson. (2003). "Optical Fault Induction Attacks". In: *Proceedings of the Conference on Cryptographic Hardware and Embedded Systems (CHES)*. Ed. by B. S. Kaliski, ç. K. Koç, and C. Paar. Berlin, Heidelberg: Springer Berlin Heidelberg. 2–12. DOI: 10.1007/3-540-36400-5_2.

Slowinska, A. and H. Bos. (2009). "Pointless Tainting? Evaluating the Practicality of Pointer Tainting". In: *Proceedings of the 2009 ACM European Conference on Computer Systems (EuroSys)*. New York, NY, USA: Association for Computing Machinery. 61–74. DOI: 10.1145/1519065.1519073.

Smalley, S. and R. Craig. (2013). "Security Enhanced (SE) Android: Bringing Flexible MAC to Android." In: *Proceedings of the 2013 Network and Distributed Systems Symposium (NDSS)*. Vol. 310. 20–38.

Snow, K. Z., F. Monrose, L. Davi, A. Dmitrienko, C. Liebchen, and A. Sadeghi. (2013). "Just-in-Time Code Reuse: On the Effectiveness of Fine-Grained Address Space Layout Randomization". In: *Proceedings of the 2013 IEEE Symposium on Security and Privacy (SP). SP '13*. San Francisco, CA, USA. 574–588. DOI: 10.1109/SP.2013.45.

Spafford, E. H. (1989). "The Internet Worm Program: An Analysis". *ACM SIGCOMM Computer Communication Review*. SIGCOMM 19(1): 17–57. DOI: 10.1145/66093.66095.

Stepanov, E., K. Serebryany, M. Phillips, and V. Buka. (2020). "Memory Tagging in LLVM and Android". 2020 Virtual LLVM Developers' Meeting.

Stephens, N. (2019). *Developments in the Arm A-Profile Architecture: Armv8.6-A*. URL: https://community.arm.com/developer/ip-products/processors/b/processors-ip-blog/posts/arm-architecture-developments-armv8-6-a (accessed on 04/20/2020).

Suh, G. E., D. Clarke, B. Gassend, M. van Dijk, and S. Devadas. (2003). "AEGIS: Architecture for Tamper-evident and Tamper-resistant Processing". In: *Proceedings of the 2003 Annual International Conference on Supercomputing (ICS)*. San Francisco, CA, USA: ACM. 160–171. DOI: 10.1145/782814.782838.

Suh, G. E., J. W. Lee, D. Zhang, and S. Devadas. (2004). "Secure Program Execution via Dynamic Information Flow Tracking". *ACM SIGPLAN Notices*. 39(11): 85–96. DOI: 10.1145/1037187.1024404. (Accessed on 10/25/2021).

Sundaresan, H. (2003). "OMAP platform security features". URL: https://www.ti.com/pdfs/wtbu/omapplatformsecuritywp.pdf (accessed on 03/07/2022).

Synopsys Inc. (2014). *Heartbleed Bug*. URL: https://heartbleed.com/ (accessed on 06/21/2021).

Szekeres, L., M. Payer, T. Wei, and D. Song. (2013). "SoK: Eternal War in Memory". In: *Proceedings of the 2013 IEEE Symposium on Security and Privacy (SP)*. Washington, DC, USA: IEEE Computer Society. 48–62. DOI: 10.1109/SP.2013.13.

Tal, A. (2020). *Using Intel® MPX with the Intel® Software Development Emulator*. URL: https://www.intel.com/content/www/us/en/develop/articles/using-intel-mpx-with-the-intel-software-development-emulator.html (accessed on 11/15/2020).

Tang, A., S. Sethumadhavan, and S. Stolfo. (2017). "CLKSCREW: Exposing the Perils of Security-Oblivious Energy Management". In: *Proceedings of the 2017 USENIX Security Symposium*. Vancouver, BC: USENIX Association. 1057–1074. URL: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/tang.

Toulas, B. (2021). "New Intel chips won't play Blu-ray disks due to SGX deprecation". URL: https://www.bleepingcomputer.com/news/security/new-intel-chips-wont-play-blu-ray-disks-due-to-sgx-deprecation/ (accessed on 01/29/2021).

Tramer, F. and D. Boneh. (2019). "Slalom: Fast, Verifiable and Private Execution of Neural Networks in Trusted Hardware". In: *Proceedings of the 2019 International Conference on Learning Representations*. URL: https://openreview.net/forum?id=rJVorjCcKQ.

Turan, M. S., E. Barker, J. Kelsey, K. A. McKay, M. L. Baish, and M. Boyle. (2018). "Recommendation for the Entropy Sources Used for Random Bit Generation". National Institute of Standards and Technology. DOI: 10.6028/NIST.SP.800-90B.

Van Bulck, J., D. Moghimi, M. Schwarz, M. Lipp, M. Minkin, D. Genkin, Y. Yuval, B. Sunar, D. Gruss, and F. Piessens. (2020). "LVI: Hijacking Transient Execution through Microarchitectural Load Value Injection". In: *Proceedings of the 2020 IEEE Symposium on Security and Privacy (SP)*.

Viand, A., P. Jattke, and A. Hithnawi. (2021). "SoK: Fully Homomorphic Encryption Compilers". In: *Proceedings of the 2021 IEEE Symposium on Security and Privacy (SP)*. 1092–1108. DOI: 10.1109/SP40001.2021.00068.

Volos, S., K. Vaswani, and R. Bruno. (2018). "Graviton: Trusted Execution Environments on GPUs". In: *Proceedings of the 2018 USENIX Symposium on Operating Systems Design and Implementation (OSDI)*. Carlsbad, CA: USENIX Association. 681–696. URL: https://www.usenix.org/conference/osdi18/presentation/volos.

Von Neumann, J. (1993). "First Draft of a Report on the EDVAC". *IEEE Annals of the History of Computing*. 15(4): 27–75.

Watson, R., B. Feldman, A. Migus, and C. Vance. (2003). "Design and implementation of the Trusted BSD MAC framework". In: *Proceedings of the 2003 DARPA Information Survivability Conference and Exposition*. Vol. 1. IEEE. 38–49.

Watson, R. N. M., P. G. Neumann, J. Woodruff, M. Roe, H. Almatary, J. Anderson, J. Baldwin, D. Chisnall, B. Davis, N. W. Filardo, A. Joannou, B. Laurie, A. T. Markettos, S. W. Moore, S. J. Murdoch, K. Nienhuis, R. Norton, A. Richardson, P. Rugg, P. Sewell, S. Son, and H. Xia. (2019). "Capability Hardware Enhanced RISC Instructions: CHERI Instruction-Set Architecture (Version 7)". No. UCAM-CL-TR-927. University of Cambridge, Computer Laboratory. URL: https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-927.html (accessed on 11/30/2020).

Weiser, S., M. Werner, F. Brasser, M. Malenko, S. Mangard, and A.-R. Sadeghi. (2019). "TIMBER-V: Tag-Isolated Memory Bringing Fine-grained Enclaves to RISC-V". In: *Proceedings of the 2019 Network and Distributed System Security Symposium (NDSS)*. DOI: 10.14722/ndss.2019.23068.

Woodruff, J., R. N. M. Watson, D. Chisnall, S. W. Moore, J. Anderson, B. Davis, B. Laurie, P. G. Neumann, R. Norton, and M. Roe. (2014). "The CHERI Capability Model: Revisiting RISC in an Age of Risk". In: *Proceedings of the ACM/IEEE 2014 International Symposium on Computer Architecture (ISCA)*. ACM/IEEE ISCA '14. 457–468. DOI: 10.1109/ISCA.2014.6853201. (Accessed on 12/03/2020).

Wright, P. (1987). *Spycatcher*. Heinemann Publishers Australia.

Xu, L. (2010). "Secure the Enterprise with Intel® AES-NI". *White Paper*. Intel. URL: https://www.intel.com/content/www/us/en/enterprise-security/enterprise-security-aes-ni-white-paper.html.

Yarom, Y. and K. Falkner. (2014). "FLUSH+RELOAD: A High Resolution, Low Noise, L3 Cache Side-Channel Attack". In: *Proceedings of the 2014 USENIX Security Symposium*.

Yoshizawa, Y., H. Kimura, H. Inoue, K. Fujita, M. Toyama, and O. Miyatake. (1999). "Physical random numbers generated by radioactivity". *Journal of the Japanese Society of Computational Statistics*. 12(1): 67–81. DOI: 10.5183/jjscs1988.12.67.

Yu, Y.-c. (2021). *[PATCH v30 00/32] Control-flow Enforcement: Shadow Stack*. URL: https://lore.kernel.org/linux-mm/20210830181528.1569-3-yu-cheng.yu@intel.com/T/ (accessed on 01/16/2022).

Yu, J., L. Hsiung, M. El'Hajj, and C. W. Fletcher. (2019a). "Data Oblivious ISA Extensions for Side Channel-Resistant and High Performance Computing". In: *Proceedings of the 2019 Network and Distributed System Security Symposium (NDSS)*. Network and Distributed System Security Symposium. San Diego, CA: Internet Society. DOI: 10.14722/ndss.2019.23061. (Accessed on 09/23/2021).

Yu, J., M. Yan, A. Khyzha, A. Morrison, J. Torrellas, and C. W. Fletcher. (2019b). "Speculative Taint Tracking (STT): A Comprehensive Protection for Speculatively Accessed Data". In: *Proceedings of the 2019 Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. New York, NY, USA: Association for Computing Machinery. 954–968. DOI: 10.1145/3352460.3358274.

Zhang, N., K. Sun, D. Shands, W. Lou, and Y. T. Hou. (2016a). "TruSpy: Cache Side-Channel Information Leakage from the Secure World on ARM Devices". *Cryptology ePrint Archive* No. 2016/980. URL: https://eprint.iacr.org/2016/980 (accessed on 03/06/2022).

Zhang, T., Y. Zhang, and R. B. Lee. (2016b). "CloudRadar: A Real-Time Side-Channel Attack Detection System in Clouds". In: *Proceedings of the 2016 International Symposium on Research in Attacks, Intrusions, and Defenses (RAID)*. Paris, France. DOI: 10.1007/978-3-319-45719-2_6.

Zhang, X., Y. Xiao, and Y. Zhang. (2016c). "Return-Oriented Flush-Reload Side Channels on ARM and Their Implications for Android Devices". In: *Proceedings of the 2016 ACM Conference on Computer and Communications Security (ACM)*. New York, NY, USA: Association for Computing Machinery. 858–870. DOI: 10.1145/2976749.2978360.

Zhou, Q., X. Liao, K.-w. Wong, Y. Hu, and D. Xiao. (2009). "True Random Number Generator Based on Mouse Movement and Chaotic Hash Function". *Information Sciences*. 179(19): 3442–3450. DOI: 10.1016/j.ins.2009.06.005.

Zhu, J., R. Hou, X. Wang, W. Wang, J. Cao, B. Zhao, Z. Wang, Y. Zhang, J. Ying, L. Zhang, and D. Meng. (2020). "Enabling Rack-Scale Confidential Computing Using Heterogeneous Trusted Execution Environment". In: *Proceedings of the 2020 IEEE Symposium on Security and Privacy*. 2020 IEEE Symposium on Security and Privacy (SP). San Francisco, California, USA. 16. DOI: 10.1109/SP40000.2020.00054.

Zinzindohoué, J.-K., K. Bhargavan, J. Protzenko, and B. Beurdouche. (2017). "HACL*: A Verified Modern Cryptographic Library". In: *Proceedings of the 2017 ACM Conference on Computer and Communications Security (CCS)*. DOI: 10.1145/3133956.3134043.