ROYAL HOLLOWAY, UNIVERSITY OF LONDON

DOCTORAL THESIS

Techniques for Establishing Trust in Modern Constrained Sensing Platforms with Trusted Execution Environments

---

Author:
Carlton SHEPHERD

Supervisor:
Prof. Konstantinos MARKANTONAKIS

A thesis submitted in fulfillment of the requirements for the degree of Doctor of Philosophy

in the

Information Security Group
Department of Mathematics

January 17, 2019

Declaration of Authorship

I, Carlton SHEPHERD, declare that this thesis titled, "Techniques for Establishing Trust in Modern Constrained Sensing Platforms with Trusted Execution Environments" and the work presented in it is my own. I confirm that:

This doctoral thesis and the work therein was conducted under the supervision of Prof. Konstantinos Markantonakis. The work presented in this thesis is the result of original research carried out by myself, or in collaboration with others, while enrolled in the Department of Mathematics as a candidate for the degree of Doctor of Philosophy. This work has not been submitted for any other degree or award at other university or educational establishment.

Signed:

Date:

Abstract

The Internet of Things (IoT) – the notion that interconnected everyday objects will acquire the ability to monitor and act upon their environment – is anticipated to benefit multiple domains, including manufacturing, health and social care, finance, and within the home. However, a plethora of security and trust concerns surround the deployment of millions of devices that transmit sensing data to inform critical decision-making, with potentially serious consequences for end-users. Trusted Execution Environments (TEEs) are emerging as a robust and widely-available solution for protecting the confidentiality and integrity of sensitive applications on IoT devices. TEEs continue a succession of secure execution technologies, including smart cards and embedded Secure Elements, by employing hardware-assistance for protecting run-time accesses to sensitive memory locations, input/output (I/O) devices, and persistent data. TEEs can also provide many of the mechanisms provided by other trusted computing primitives, namely the Trusted Platform Module (TPM), like remote attestation.

Given their recent inception, however, TEEs lack the maturity and the ecosystem of long-standing solutions such as TPMs, particularly for constrained devices. This thesis identifies and analyses a multitude of such challenges, resulting in the proposal and evaluation of contributions in five areas of concern. This includes applying TEEs to sensor-driven continuous authentication schemes, an emerging paradigm for addressing the shortfalls of conventional biometrics; secure and mutually trusted communication between two TEEs on remotely located devices; tamper-resistant system logging for constrained platforms with TEEs; remote TEE credential management with respect to centralised IoT deployments, e.g. smart cities and industrial IoT; and a critical evaluation of proposed solutions to relay attacks in contactless transactions, to which existing TEEs are vulnerable. This thesis concludes by identifying open research challenges surrounding the deployment and management of constrained device TEEs in IoT applications.

Acknowledgements

First and foremost, I would like to thank my supervisor Prof. Konstantinos Markantonakis, who was always understanding, accommodating, supportive, and provided helpful advice at every turn. My appreciation also goes to the current and previous members of the Smart Card Centre (SCC) at Royal Holloway, who provided many hours of insightful and interesting discussions on a myriad of topics, ranging from Android to FPGAs, NFC to automotive security, and many, many more. Special thanks go to Raja N. Akram and Iakovos Gurulian from the SCC for many interesting and rewarding collaborations. Raja was also indispensable in identifying any flaws and weaknesses in my ideas, and providing many helpful suggestions throughout my studies. Iakovos, meanwhile, was there for any much-needed pints, suggestions, and discussions into the early hours of the morning. I would also like to thank Fabien Petitcolas at OneSpan (formerly VASCO Data Security), who hosted me for my internship in Wemmel, Belgium. Although I didn't realise at the time, the experience helped to shape my Ph.D. direction towards more interesting and fruitful grounds. Furthermore, I greatly appreciate the support from the EPSRC, who funded my studies, and the staff and students involved with the Centre for Doctoral Training (CDT) at Royal Holloway; pursuing this Ph.D. would have been impossible otherwise. Special thanks go to Dusan Repel and Blake Loring, in particular, for many interesting discussions. Last, but certainly not least, my thanks go to my family and Ellen for the support they provided over the past four years.

Publications

[1] C. Shepherd, G. Arfaoui, I. Gurulian, R. P. Lee, K. Markantonakis, R. N. Akram, D. Sauveron, and E. Conchon, "Secure and Trusted Execution: Past, Present, and Future - A Critical Review in the Context of the Internet of Things and Cyber-Physical Systems", in *Proceedings of the 15th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, ser. TrustCom '16, IEEE, 2016, pp. 168–177.

[2] C. Shepherd, R. N. Akram, and K. Markantonakis, "Towards Trusted Execution of Multi-modal Continuous Authentication Schemes", in *Proceedings of the 32nd Symposium on Applied Computing*, ser. SAC '17, ACM, 2017, pp. 1444–1451.

[3] C. Shepherd, I. Gurulian, E. Frank, K. Markantonakis, R. N. Akram, K. Mayes, and E. Panaousis, "The Applicability of Ambient Sensors as Proximity Evidence for NFC Transactions", in *Mobile Security Technologies, IEEE Security and Privacy Workshops*, ser. SPW '17, IEEE, 2017.

[4] C. Shepherd, R. N. Akram, and K. Markantonakis, "Establishing Mutually Trusted Channels for Remote Sensing Devices with Trusted Execution Environments", in *Proceedings of the 12th International Conference on Availability, Reliability and Security*, ser. ARES '17, ACM, 2017, 7:1–7:10.

[5] C. Shepherd, F. A. P. Petitcolas, R. N. Akram, and K. Markantonakis, "An Exploratory Analysis of the Security Risks of the Internet of Things in Finance", in *Proceedings of the 14th International Conference on Trust and Privacy in Digital Business*, ser. TrustBus '17, Springer, 2017, pp. 164–179.

[6] G. Haken, K. Markantonakis, I. Gurulian, C. Shepherd, and R. N. Akram, "Evaluation of Apple iDevice Sensors As a Potential Relay Attack Countermeasure for Apple Pay", in *Proceedings of the 3rd ACM Workshop on Cyber-Physical System Security*, ser. CPSS '17, ACM, 2017, pp. 21–32.

[7] I. Gurulian, C. Shepherd, E. Frank, K. Markantonakis, R. N. Akram, and K. Mayes, "On the Effectiveness of Ambient Sensing for NFC-based Proximity Detection by Applying Relay Attack Data", in *Proceedings of the 16th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, ser. TrustCom '17, IEEE, 2017.

[8] I. Gurulian, K. Markantonakis, C. Shepherd, E. Frank, and R. N. Akram, "Proximity Assurances Based on Natural and Artificial Ambient Environments **(Invited Paper)**", in *Proceedings of the 10th International Conference on Innovative Security Solutions for Information Technology and Communications*, ser. SECITC '17, Springer, 2017.

[9] C. Shepherd, R. N. Akram, and K. Markantonakis, "EmLog: Tamper-Resistant System Logging for Constrained Devices with TEEs", in *Proceedings of the 11th IFIP International Conference on Information Security Theory and Practice*, ser. WISTP '17, Springer, 2018.

[10] ——, "Remote Credential Management with Mutual Attestation for Trusted Execution Environments **(In Press)**", in *Proceedings of the 12th IFIP International Conference on Information Security Theory and Practice*, ser. WISTP '18, Springer, 2019.

Contents

List of Figures

List of Tables

List of Abbreviations

AES	**A**dvanced **E**ncryption **S**tandard
API	**A**pplication **P**rogramming **I**nterface
CA	**C**ontinuous **A**uthentication
CC	**C**ommon **C**riteria
DAA	**D**irect **A**nonymous **A**ttestation
DH	**D**iffie-**H**ellmann
EAL	**E**valuation **A**ssurance **L**evel
ECC	**E**lliptic **C**urve **C**ryptography
ECDSA	**E**lliptic **C**urve **D**igital **S**ignature **A**lgorithm
EER	**E**qual **E**rror **R**ate
EMV	**E**uropay **M**astercard **V**isa
FAR	**F**alse **A**cceptance **R**ate
FRR	**F**alse **R**ejection **R**ate
GP	**G**lobal**P**latform
GPS	**G**lobal **P**ositioning **S**ystem
HAL	**H**ardware **A**bstraction **L**ayer
HCE	**H**ost-based **C**ard **E**mulation
IoT	**I**nternet **o**f **T**hings
IIoT	**I**ndustrial **I**nternet **o**f **T**hings
MAE	**M**ean **A**bsolute **E**rror
NFC	**N**ear **F**ield **C**ommunication
OS	**O**perating **S**ystem
PIN	**P**ersonal **I**dentification **N**umber
PCR	**P**latform **C**onfiguration **R**egister
PRAD	**P**roximity and **R**elay **A**ttack **D**etection
RA	**R**emote **A**ttestation
REE	**R**ich **E**xecution **E**nvironment
RoT	**R**oot **o**f **T**rust
SE	**S**ecure **E**lement
SGX	**S**oftware **G**uard E**X**tensions
SIM	**S**ubscriber **I**dentity **M**odule
STET	**S**ecure and **T**rusted **E**xecution **T**echnology
TCG	**T**rusted **C**omputing **G**roup
TEE	**T**rusted **E**xecution **E**nvironment
TPM	**T**rusted **P**latform **M**odule
TXT	**T**rusted E**X**ecution **T**echnology
UICC	**U**niversal **I**ntegrated **C**ircuit **C**ard
VM	**V**irtual **M**achine
VMM	**V**irtual **M**achine **M**anager

Chapter 1

Introduction

The Internet of Things (IoT) – the notion that everyday objects will monitor and potentially actuate upon the environment – is anticipated to significantly benefit a number of domains, such as manufacturing, logistics, health and social care, financial services, personal fitness, and home efficiency. Many potential applications within these areas are long-standing, including early detection of illness using wearable physiological monitors [1]; improving energy conservation using room occupancy data [2]; floor sensors for fall/injury detection in the homes of the elderly and disabled [3]; and the monitoring of ambient conditions to protect perishable foods in transit [4] and livestock at rest [5].

However, only in recent years have technological and economic factors converged so relatively small, powerful and sensor-rich devices are available at unprecedented cost. This has been driven by advances in high-density IC[1] fabrication techniques, e.g. Ball Gate Array (BGA) – described in Chapter 2 – System-on-Chip (SoC) architectures, and the mass-production of passive and active sensors, like GPS[2] and tri-axial inertial measurement units (IMUs). Now, modern single-board computers (SBCs), most famously the credit-card sized Raspberry Pi 3[3] available at $35 (USD), often contain quad-core CPUs, multi-gigabyte RAM modules, embedded GPUs, and can host a fully-fledged OS. Such devices are capable of performing high-definition video playback; selected computer vision tasks, e.g. motion and object detection; the collection of various environmental measurements from I/O devices; and utilise a range of wireless mediums, like Wi-Fi (IEEE 802.11n) and Bluetooth.

Small, powerful devices are useful as standalone devices, say, as compact home media centres; however, it is the synergy of multiple data-rich devices, alongside the application of data analytics and machine learning techniques, that fuels the high expectations surrounding IoT. In fitness monitoring, systems such as Fitbit[4] and the Nike FuelBand[5] allow users to visualise their progress

[1] IC: Integrated Circuit.
[2] GPS: Global Positioning System.
[3] Raspberry Pi 3: https://www.raspberrypi.org/
[4] Fitbit: https://www.fitbit.com/
[5] Nike FuelBand: https://secure-nikeplus.nike.com/plus/what_is_fuel/

from aggregated physiological and contextual measurements, such as location, step-count and heart rate. This allows users to reflect on their progress and, if necessary, reconfigure their fitness regimes to attain their desired goals. In freight transportation, it is proposed to deploy sensing platforms in long-haul vehicles that to report environmental variables, e.g. humidity and temperature, in order to raise alarms and reactively trigger environmental control systems if conditions deviate from safe parameters [4], [5]. In smart homes, commercial systems already exist that intelligently control the brightness and colour of ambient lighting, e.g. Phillips Hue[6]; control the temperature of central heating; and report household energy usage to a smartphone or smartwatch (see Nest[7]).

The forecasted number of deployed IoT devices differs widely according to device classification. Gartner [6] estimates the number of Internet of Things (IoT) devices to reach 20.8 billion by 2020, excluding smartphones and tablets. Meanwhile, Statista [7] forecast the number of installed IoT devices to reach 30.73 billion in 2020 before reaching 74.55 billion by 2025. In financial terms, a 2017 report by IDC [8] projected that global IoT spending is, on average, due to grow annually at 15.6% over the 2015-2020 forecast period – reaching $1.29 trillion (USD) by 2020. According to the same report, the largest investments in 2016 originated from manufacturing ($178bn), focussing on field servicing and production asset management; transportation ($78bn), primarily for freight condition tracking; and utilities ($69bn), for electricity and gas infrastructure monitoring. IDC state that insurance telematics is the leading use case in financial services, while *"remote health monitoring will see the greatest investment in the healthcare industry"* [8].

Business and industrial IoT purchases are generally projected to exceed consumer sales. A 2017 report by PwC [9] estimated that business IoT spending will reach $832bn in 2020 (from $215bn in 2015), with only $236bn in 2020 (from $72bn in 2015) coming from consumer spending. This sentiment is shared by Bain & Company [10], who estimate business-to-business (B2B) IoT sales to reach $300bn annually by 2020, while consumer applications will, by the same year, generate approximately $150bn. While scepticism has been directed at some projected figures – notably IBM's 2012 forecast of one-trillion connected devices by 2015 – the trend of billions of connected devices is shown to generally hold [11].

1.1 Motivation

In the face of high expectations, the prospect of billions of connected devices raises a plethora of severe security, trust and privacy challenges. An obvious

[6]Phillips Hue: https://www.philips.co.uk/c-m-li/hue
[7]Nest: https://nest.com/uk/

security concerns is the participation of millions of small, powerful and Internet-connected devices launching Distributed Denial of Service (DDoS) attacks. In 2014, Symantec discovered the `Linux.Darlloz` worm that exploited a PHP vulnerability to infect home routers, printers and industrial control monitors in order to mine cryptocurrencies [12]. The Mirai botnet, discovered in August 2016, infected up to 2.5 million devices according to McAfee [13], including IP web cameras, digital cameras, set-top boxes, home routers and printers. The botnet was responsible for a 620GB/s attack targeting the security blog Krebs on Krebs[8] and a 1TB/s DDoS attack, believed to be the largest ever recorded [12], on French web host OVH and the DNS provider Dyn. Mirai exploited insecurely configured default usernames and passwords for administrative accounts, and caused outages to Github, Twitter, Reddit, Netflix and Airbnb [14].

While DDoS attacks are undoubtedly costly, it is the effect of malware on devices responsible for the monitoring of manufacturing equipment, critical infrastructure, sensitive substances in logistics, or assistive health and social care technologies that raise the gravest concerns [15], [16]. A compromised sensing device that surreptitiously feeds falsified measurements, e.g. temperature and battery consumption, to a reactive control system could disrupt its operation or falsely suppress alarms. At worst, devices may endanger personnel and industrial equipment if it influences climate controls or heavy machinery. In assistive technologies, compromised fall or object detection systems could endanger vulnerable users [17]. Falsified sensor measurements within transport monitoring systems, such as temperature monitors in rail-lines and train brakes [18], could conceal the presence of dangerous environmental conditions and operating parameters. In finance, malicious telematics data may allow users to construe an intentionally misleading model of behaviour to attain inaccurate insurance premiums. While not presented in this thesis, our previous work [19] showed that untrustworthy sensor data is one of the greatest risks facing the deployment of IoT in financial applications.

Economic factors, moreover, are of little help: IoT devices are often manufactured to a minimal price-per-unit, and competitive pressures from being the 'first to market' can lead to security and privacy concerns playing a secondary role [15], [20], [21]. Timely patch development, and the adoption thereof, has already been identified as a major IoT security issue, particularly for smaller vendors who lack the capability to quickly remedy potentially millions of devices [20].

While various software security countermeasures exist – application sandboxing, shadow stacks, and user-input sanitisation – these are not used solely for protecting the most sensitive assets. This is principally the concern for key material, biometrics and other credentials used to authenticate devices (and their

[8] Krebs on Krebs: `https://krebsonsecurity.com/`

users) in operations with potentially damaging consequences to end-users. A severe enough vulnerability – an unsecured network port, unsatisfactory cryptographic parameters, or even a social engineering attack – in the 'wrong' IoT device and environment could enable an attacker to potentially endanger users, whether physically, financially or otherwise. Indeed, the perceived trustworthiness of devices may be the differentiating factor in realising the projected potential of IoT. A 2014 report into the Internet of Things by the UK Government's Chief Scientific Advisor stated that, while *"real value is created for businesses, consumers and governments [by IoT]...a major data breach or cyber-attack is likely to have extremely damaging consequences on public attitudes"* [22]. Secure and trustworthy computing is vital in deployments that require increased levels of assurances that IoT devices are performing to expectations in order to mitigate the rise of harmful behaviour. For further reading, the reader is directed to surveys on security and trust in IoT security by Scelari et al. [23] (general introduction and security challenges), Yen et al. [24] (trust management), Islam et al. [25] (IoT and healthcare), and Nguyen et al. [26] (secure communication protocols).

One major security countermeasure in recent years has been Security-Enhanced Linux (SELinux) [27], initially developed by the US National Security Agency (NSA), which provides mandatory access control to the Linux kernel. SELinux utilises policies that define the operations that each system object, say, a process, may perform on its subjects, such as files and I/O devices. Through appropriate policy definition, which is itself a difficult task in continually evolving systems, unsafe processes can be executed without compromising safe processes. While SELinux is an important construct for protecting server and Android devices [28], it is still wholly software-based. Following the principle of security in layers, *hardware-assisted* trusted computing has been employed to protect the assets of the most sensitive application domains, particularly in banking and biometrics. Here, inherently trusted hardware components – also known as a hardware Root of Trust (RoT) – are used to enforce access control; perform cryptographic operations, such as tamper-resistant key generation and storage; and for interacting with sensitive Input/Output (I/O) peripherals, such as fingerprint readers. Many of these technologies also allow remote parties to gain assurances that the target platform is executing to expectations using *remote attestation*. The most prominent examples of these technologies include the Trusted Platform Module (TPM) and Secure Element (SE), described in greater detail in Chapter 2.

One of the major drawbacks of for many hardware-assisted mechanisms is the requirement for additional hardware, which increases costs and occupies limited PCB space on constrained devices. Moreover, these mechanisms are often relatively limited computationally, and offer a reduced feature set compared to the main execution environment, also known as the Rich Execution Environment

(REE). TPMs, for example, do not natively provide isolated application execution, while SEs lack the ability to gather evidence of boot components (measured boot) and securely providing such evidence to remote verifiers (remote attestation). An increasingly popular solution is the Trusted Execution Environment (TEE), which provides SE-like secure and isolated application execution, as well as TPM-like trust mechanisms like remote attestation. Notable TEEs include Intel SGX and the GlobalPlatform TEE specifications, which are also explored further in Chapter 2.

However, TEEs are relatively new and lack the maturity of previous solutions, such as the TPM, especially in their use in protecting embedded sensing devices. This thesis identifies a number of challenges surrounding the use of TEEs in this domain, such as their application in protecting emerging sensor-based authentication technologies (continuous authentication), mutual TEE remote attestation, tamper-resistant system logging and remote credential management in centralised IoT deployments. Throughout this thesis, implementations on genuine TEEs are used frequently in order to evaluate the real-world practicality of the proposed schemes. Formal symbolic verification is also used heavily in the proposal of any protocols and procedures for assuring correctness against the stated adversarial models. Lastly, this thesis goes one step further to examine state-of-the-art attacks that are not protected by current TEEs, namely *relay attacks*, on NFC-based contactless transactions used in many payment, transportation and physical access control systems.

1.2 Thesis Outline and Contributions

Chapter 2 serves as a primer for the remainder of this thesis by introducing and describing the architecture and relative computational capability of modern mobile and embedded systems. This is followed by a detailed review of hardware-assisted secure and trusted execution technologies for protecting such systems, ranging from traditional solutions, such as smart cards and TPMs, to the most recent advancements like Intel SGX and the GlobalPlatform TEE specifications. This chapter also includes a security and feature comparison of each technology and, where applicable, a discussion of relevant industry standards, including the Common Criteria framework for platform security evaluation. This chapter is a substantially expanded version of our following published work:

- C. Shepherd, G. Arfaoui, I. Gurulian, R. P. Lee, K. Markantonakis, R. N. Akram, D. Sauveron, and E. Conchon. "Secure and Trusted Execution: Past, Present and Future – A Critical Review in the Context of the Internet of Things and Cyber-Physical Systems," in *Proceedings of the 15th IEEE*

International Conference on Trust, Security and Privacy in Computing and Communications, ser. TrustCom '16, IEEE, 2016, pp. 168-177.

The first core contribution of this thesis is presented in Chapter 3, which investigates the application of secure and trusted execution technologies to provide additional security assurances for Continuous Authentication (CA). CA is an emerging user-authentication paradigm that employs machine learning to model user behaviour in order to infer the device's authentication state from sensing data. This chapter is an extended version of our following publication:

- C. Shepherd, R. N. Akram, and K. Markantonakis. "Towards Trusted Execution of Multi-Modal Continuous Authentication Schemes," in *Proceedings of the 32nd ACM Symposium on Applied Computing*, ser. SAC '17, ACM, 2017, pp. 1444-1451.

Chapter 4 develops the concept of remote attestation in greater detail, and raises the challenge of *mutual attestation* and TEE-to-TEE intercommunication in which TEEs aboard remotely located devices wish to communicate in a secure and trusted fashion with mutual trust assurances. To this end, we present the first investigation into achieving this in a single protocol run for TEEs, which is implemented and evaluated on two devices hosting GlobalPlatform-compliant TEEs. The proposed protocols are also subjected to symbolic formal verification using the Scyther protocol analysis tool [29]. This chapter is based on the following work:

- C. Shepherd, R. N. Akram, and K. Markantonakis. "Establishing Mutually Trusted Channels for Remote Sensing Devices with Trusted Execution Environments," in *Proceedings of the 12th International Conference on Availability, Reliability and Security*, ser. ARES '17, ACM, 2017, 7:1-7:10.

Chapter 5 presents this thesis's third core contribution: EmLog – a novel tamper-resistant logging scheme for preserving system logs on constrained devices with TEEs, with potential applications in IoT device forensics and auditing. This chapter is based on our work in:

- C. Shepherd, R. N. Akram, and K. Markantonakis. "EmLog: Tamper-Resistant System Logging for Constrained devices with TEEs," in *Proceedings of the 11th IFIP International Conference on Information Security Theory and Practice*, ser. WISTP '17, Springer, 2018.

Following this, Chapter 6 addresses the open challenge of secure and trusted remote credential management for constrained sensing devices in centralised IoT environments. A suite of protocols is introduced, motivated, proposed and analysed for the revocation, migration, backup and update of TEE-based

device credentials, as well as the retrieval of audit system logs as an additional development of our work from Chapter 5. This chapter is based on the following work:

- **(In Press)** C. Shepherd, R. N. Akram, and K. Markantonakis. "Remote Credential Management with Mutual Attestation for Trusted Execution Environments," in *Proceedings of the 12th IFIP International Conference on Information Security Theory and Practice*, ser. WISTP '18, Springer, 2019.

The penultimate chapter of this thesis, Chapter 7, segues into a long-standing challenge in NFC-based contactless transactions – *relay attacks* – that circumvent even the protection offered by TEEs. In this chapter, a series of past proposals based on ambient sensing for thwarting relay attacks are evaluated for their effectiveness. This work assesses 15 sensors available through the Android platform and, using data collected from an emulated relay attack setup, uses an array of similarity- and supervised machine learning-based methods proposed in existing literature. The results indicate that, under industry-stipulated time constraints, no single evaluated sensor is appropriate for use in high-value transactions without significantly detracting from security and usability. This chapter is an aggregation of our work presented in the following papers:

- C. Shepherd, I. Gurulian, E. Frank, K. Markantonakis, R. N. Akram, K. Mayes, and E. Panaousis. "The Applicability of Ambient Sensors as Proximity Evidence for NFC Transactions," in *Mobile Security Technologies*, IEEE Security and Privacy Workshops, ser. SPW '17, IEEE, 2017.

- I. Gurulian, C. Shepherd, E. Frank, K. Markantonakis, R. N. Akram, and K. Mayes. "On the Effectiveness of Ambient Sensing for NFC-based Proximity Detection by Applying Relay Attack Data," in *Proceedings of the 16th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, ser. TrustCom '17, IEEE, 2017.

- **(Invited Paper)** I. Gurulian, K. Markantonakis, C. Shepherd, E. Frank, and R. N. Akram. "Proximity Assurances Based on Natural and Artificial Ambient Environments," in *Proceedings of the 10th International Conference on Innovative Security Solutions for Information Technology and Communications*, ser. SECITC '17, Springer, 2017.

The final chapter, Chapter 8, concludes this thesis and presents several open challenges in the realm of constrained device TEEs and secure and trusted sensing.

Chapter 2

Background

This chapter describes the motivation, capability, architectures and evolution of computing platforms designed to realise secure and trusted execution of security-critical applications and data.

2.1 Mobile and Embedded System Architectures

A *mobile* system, as contained within a smartphone or tablet, is a portable computing system that can be used for general-purpose computing, such as video/audio file playback, sending/receiving SMSs and electronic mail, and accessing video games. An *embedded* system, meanwhile, is a computing system with a dedicated function, potentially with real-time computing constraints, and exists within a larger mechanical or electrical system. It may or may not be portable.

The precise computational capability of a mobile or embedded system is difficult to define given the continuing technological advancements in processor design, semiconductor fabrication techniques, and deflationary hardware prices. However, both mobile and embedded devices are associated with low power consumption, smaller physical size, lower cost per unit, and limited processing capacity relative to a workstation or supercomputer.

Modern personal devices, such as digital TVs, fitness monitors, baby monitors and smartwatches, rely heavily upon embedded systems. As do more complex units like vehicle subsystems, e.g. collecting and transmitting telematics data, such as location and speed; digital kitchen appliances, e.g. washing machines and microwaves; and for controlling heating, ventilation and air conditioning (HVAC) units. Due to their computational limitations, embedded systems are deployed typically with fewer software applications and an operating system (OS) with reduced functionality. For the smallest microcontroller-based devices, only a single dedicated application may be available. For applications requiring greater flexibility and computational power, such as arcade systems and home media centres, single-board computers (SBCs) can be used that host a general purpose OS, like Linux, and multiple applications. The differences between

microcontrollers, SBCs and, indeed, System-on-Chips (SoCs) are discussed in greater detail in the following sections.

2.1.1 System-on-Chip (SoC)

A System-on-Chip (SoC) incorporates the components of a computing system – the CPU, memory units (RAM[1], ROM[2], EEPROM[3] and flash memory), system buses, real-time clocks (RTCs), analog-to-digital converters and vice-versa (ADCs and DACs), timers etc. – onto a single integrated circuit (IC). SoCs were developed from the need for greater power efficiency, smaller product PCBs (printed circuit boards), and lower assembly costs [30]. This is achieved through the removal of redundant components and the reduction of wiring distances and material, thus improving power and cost efficiency. SoCs allow the installation of only a single chip during assembly as opposed to its individual sub-components.

A SoC's exact components and capabilities varies between vendors and the intended application. The NVIDIA Tegra X1[4], for example, comprises a 256-core GPU for applications requiring high-definition graphics rendering, such as gaming and video playback. The Texas Instruments TDA2x[5] contains interfaces for six camera inputs, CAN bus connectivity, digital video output and Audio Video Bridging (AVB) support for low-latency video streaming intended for automobile park assist systems. The hardware design of SoC components typically begins with pre-built initial IP blocks (cores) that are selected and integrated to the vendor's specifications; any proprietary, application-specific components can be integrated thereafter. The designed SoC is usually tested and validated for correctness using software-based emulation and FPGA prototypes (functional verification) before being fabricated physically on substrate.

Some of the most common SoC components are described below, while an example ARM-based SoC is depicted in Figure 2.1.

- **CPU**. The ARM Cortex-A/-M/-R CPUs are virtually ubiquitous in modern SoCs. The Cortex-A line is designed for high performance applications, supports 32- or 64-bit execution and virtual memory via a Memory Management Unit (MMU), and is capable of running a fully-featured OS. The Cortex-M family, aimed at low-cost systems such as microcontrollers (Section 2.1.3), do not feature full MMUs and contain a smaller instruction set. Indeed, some models, e.g. Cortex-M0 and -M3, do not even support division or floating point operations. Cortex-R processors cater for real-time

[1] RAM: Random Access Memory.
[2] ROM: Read-Only Memory.
[3] EEPROM: Electrically Erasable Programmable Read-Only Memory.
[4] NVIDIA Tegra: http://www.nvidia.com/object/tegra.html
[5] Texas Instruments TDA2x: http://www.ti.com/lit/ml/sprt681/sprt681.pdf

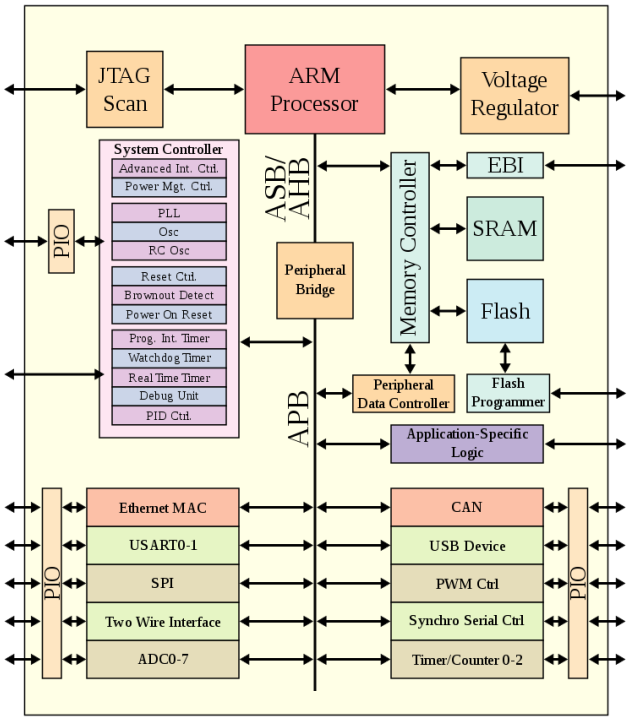


FIGURE 2.1: High-level ARM-based SoC architecture [31].

systems requiring deterministic timing and low interrupt latency; however, like the Cortex-M, there is no full MMU support. SoCs may contain multiple processors – Multiple Processor SoCs, or MPSoCs – in which subcomponents, such as sensor hubs, contain their own CPU. These CPUs control further sub-units, like individual sensors, to reduce the number of interrupts to the primary SoC for greater power conservation.

- **Interrupt controllers**, e.g. the ARM Generic Interrupt Controller (GIC), are IP cores that monitor and collate interrupt requests (IRQs) and urgent, low-latency fast interrupt requests (FIQs) from other SoC modules. Such controllers are used to combine interrupts onto one or more CPU lines, to control and issue CPU interrupt requests only when necessary, and in managing request prioritisation.

- **Memory**. Any number of ROM, RAM and flash memory blocks may be incorporated into an SoC design. The ROM is a non-writable medium typically used for holding power-on self-test (POST) routines, passing the boot control flow to the main bootloader – often held in flash memory – and acting as a root of trust (RoT) for implementing secure boot (discussed in Sections 2.3 and 2.4.4). Faster but more expensive Static Random Access Memory (SRAM) is used for processor caches, while slower but lower-cost Dynamic RAM (DRAM) is usually used for general-purpose storage of

application data under execution. Flash memory is re-writeable[6], non-volatile memory used for long-term persistent data storage, including hosting rewritable firmware. (The rewriting of flash-resident firmware is also known as 'flashing').

- **Input/Output (I/O)**. SoCs receive and transmit information with the outside world – collecting sensor data, receiving commands over a keypad, outputting data to an LCD display, and so on – over a multitude of I/O interfaces. Most SoCs support several interfaces, like the Serial Peripheral Interface (SPI), Universal Asynchronous Receiver-Transmitter (UART), Universal Serial Bus (USB), General Purpose I/O (GPIO), and Ethernet. Some specialist SoCs provide domain-specific interfaces, such as a Controller Area Network (CAN) bus interface for vehicular systems.

- **Memory controllers** contain the logic for reading/writing to RAM, like implementing double-data rate (DDR) transfers[7], and wear-levelling for flash memory controllers in which read/writes are distributed over blocks to prevent uneven degradation. Other controllers include Direct Memory Access (DMA) controllers for allowing I/O devices to directly read/write data to RAM independently of the CPU. This improves energy efficiency by eliminating interruptions that occupy the CPU during the acquisition and transfer of I/O data to RAM. I/O Memory Management Units (IOMMUs) are also used to prevent DMA-enabled peripherals from manipulating unauthorised RAM (DMA attacks). This is achieved by isolating memory space through the translation of virtual addresses visible to the peripherals to the host's physical addresses.

- **Testing and debugging** manufactured PCBs was standardised by the Joint Test Action Group (JTAG) in the IEEE 1149 series [32]. JTAG access ports are ubiquitous on modern SoCs, FPGAs (field-programmable gate arrays), and microcontrollers. JTAG is used for verifying potentially thousands of IC connections without testing each pin manually – a practically infeasible task for modern high-density circuits. In addition to testing PCB interconnects (boundary scan), JTAG is used for inspecting registers and memory, initialising flash memory, and re-flashing corrupt firmware.

- **Buses and bridges** are the mediums through which the SoC's subcomponents are connected and data is transferred. The ARM Advanced Microcontroller Bus Architecture (AMBA) is a common standard for defining SoC

[6]Note that flash memory blocks degrade with use. The electrical energy dispersed during erasure procedures causes the silicon substrate to degrade a small amount; over time, even error-correcting codes fail to compensate and the medium loses reliability before, ultimately, failing. Manufacturers guarantee flash memory within a certain number of Program/Erase (P/E) cycles.

[7]DDR data transfers on both the rising and falling edges of the clock, or two per clock cycle.

buses and interconnects. This includes the Advanced High-performance Bus (AHB) and Advanced eXtensible Interface (AXI) specifications for high bandwidth, high clock frequency blocks like the CPU and RAM, and the Advanced Peripheral Bus (APB) for low bandwidth peripherals, e.g. via UART and USB. Bridges are used to translate transactions between differing bus architectures, such as AXI to APB and vice-versa.

- **Security**. SoCs may contain security-centric blocks that provide hardware implementations of widely-used cryptographic algorithms, such as AES, (3)DES, SHA-256 and RSA, and hardware-based pseudo-random number generators (PRNGs). All of these may be integrated into a security co-processor block or provided separately. One of the most popular security extensions for ARM-based SoCs is TrustZone for establishing a secure world of execution, which is described in greater detail in Section 2.4.4.

- **Wireless communication**. Silicon vendors can integrate wireless mediums directly onto the SoC (see the Nordic Semiconductor nRF51822[8]). This may include transceivers for Near-Field Communication (NFC), Bluetooth LE (Low Energy), WiFi (IEEE 802.11), and ZigBee (IEEE 802.15.4).

- **Video and audio**. Specialist SoCs may contain GPU blocks for delivering high quality video at low frame rates. One such SoC, the NVIDIA Tegra, contains a 256-core NVIDIA Maxwell GPU supporting DirectX 12, OpenGL 4.5, NVIDIA CUDA, OpenGL ES 3.1 and Vulkan. Hardware-based video codec processors are also provided, e.g. for H.265 and MPEG4, and support for HDMI output. Home media centre SoCs, e.g. Freescale STMP3700[9], may also feature headphone and speaker amplifiers; high quality audio ADCs and DACs; and FM radio, microphone and line input support.

2.1.2 System-in-Package (SiP) and Package-on-Package (PoP)

A System-in-Package (SiP) refers to the stacking of *several* dies into a single physical package. This is opposed to SoCs, which fabricate a system onto a *single* die. SiPs enable greater flexibility by allowing the co-existence of SoC and IC dies from different silicon vendors, but adds additional complexity relating to die placement, package design, and thermal and (electromagnetic) noise management [33]. An example SiP is the Apple S2[10] used in the Apple Watch.

[8]Nordic Semiconductor nRF51822: https://www.nordicsemi.com/eng/Products/Bluetooth-low-energy/nRF51822

[9]Freescale STMP3700: https://www.nxp.com/docs/en/fact-sheet/STMP3700FS.pdf

[10]Apple S2: https://www.apple.com/newsroom/2016/09/apple-introduces-apple-watch-series-2/

Original Equipment Manufacturers (OEMs) may also incorporate Package-on-Package (PoP) designs in which several IC packages are stacked and soldered vertically using BGA onto a mounting PCB (see Figure 2.2). This naturally utilises less PCB area by packaging compatible components – a common configuration being a large dedicated memory package atop a SoC package – that would have otherwise been located separately. This allows device manufacturers to maximise PCB density while shifting the responsibility of package development to IC vendors. This provides greater flexibility by allowing the customisation of packages without handling the acquisition, integration, testing and packaging of dice; however, it relinquishes control over the package's development.

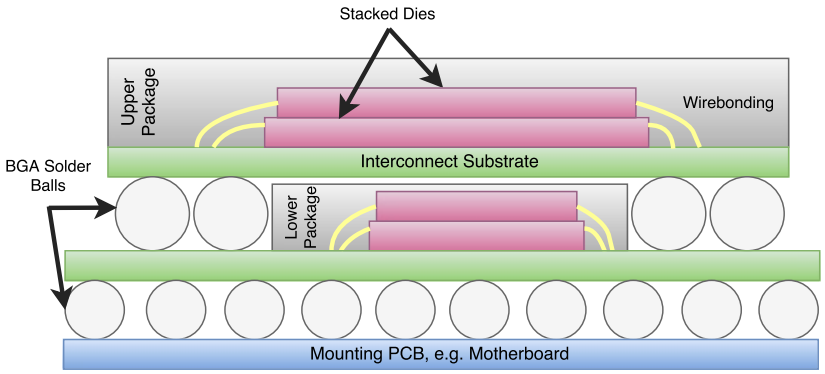


FIGURE 2.2: Overview of Package-on-Package (PoP) assembly.

2.1.3 Microcontroller (MCU)

A microcontroller unit (MCU), or simply 'microcontroller', is a computing system fabricated on a single IC. They are designed for applications with very limited power and computational requirements, such as toys, implantable medical devices and simple remote environmental monitors. MCUs typically host a single-purpose application and do not host a general-purpose OS. Its CPU may be a simple 8-bit CPU, e.g. Microchip ATtiny102[11], or a more complex 32-bit CPU, like the ARM Cortex-M33. MCU CPUs usually feature lower clock frequencies (usually under 100MHz), reduced instruction pipelines (2-3 stages), and smaller instruction sets. MCU RAM and persistent flash memory is also significantly restricted (typically less than 1MB). A limited number of I/O interfaces may also be provided, e.g. UART and SPI, and a JTAG access port for debugging. During active operation, some microcontrollers may draw as little as 1mA, or even <1μA in a low-power state [34].

[11] Atmel ATtiny102: https://www.mouser.co.uk/new/microchip/atmel-attiny-102-104-mcus/

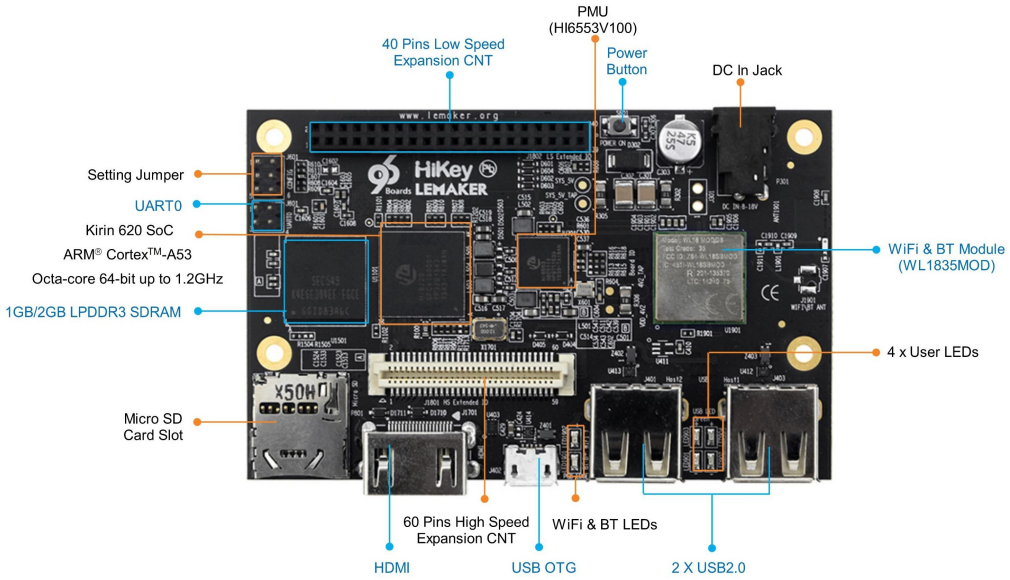


FIGURE 2.3: HiKey LeMaker single-board computer [37].

2.1.4 Single-Board Computer (SBC)

A single-board computer (SBC) is a complete computing system fabricated upon a single PCB. SBCs, unlike MCUs and SoCs, typically include full physical connectors – rather than simply interfaces – for video and audio outputs, e.g. HDMI, VGA and 1/8-inch stereo line out; network connectivity, such as Ethernet ports; storage inputs, e.g. SD card slots; and generic I/O connectors, e.g. General Purpose I/O (GPIO), USB and SPI headers. SBCs typically comprise one or more SoCs or even MCUs (sometimes referred to as single-board microcontrollers, like the Arduino unit[12]). The Hikey LeMaker SBC, pictured in Figure 2.3, is driven by a HiKey Kirin 620 SoC comprising an eight-core ARM Cortex-A53, 2GB (off-SoC) DDR3 RAM, a Mali-450 MP4 GPU, and support for WiFi (802.11b/g/n) and Bluetooth LE. SBCs can be used to host general purpose OSs, such as Android, and are used regularly for computationally intensive but space- and power-constrained applications. Example applications include robotics, home entertainment, and commercial unmanned aerial vehicle (UAV) platforms [35], [36].

2.1.5 Discussion

SiPs, SoCs and PoPs are primarily design and fabrication differences, each with its own advantages and drawbacks with respect to acquisition, design and assembly complexity. However, whether an MCU, SoC-based SBC, or traditional computing architecture[13] is chosen depends mostly on the intended device's

[12] Arduino: https://www.arduino.cc/

[13] A traditional architecture is one whose components reside on separate PCBs that communicate via expansion ports, e.g. SATA and PCI, over a primary PCB or motherboard.

requirements. Ultimately, the wide variety of SoCs, MCUs, SBCs and SiPs means that OEMs can tailor a system to a configuration that best suits its energy, computational, I/O and physical space requirements.

Mobile devices, such as tablets, smartphones and certain laptops, are usually architected using high-end, microprocessor-based SoCs, like the aforementioned HiSilicon Kirin 620 SoC, as used in the HiKey LeMaker SBC (Figure 2.3), Huawei P8 Lite and Honor 4X and 4C smartphones. Modern smartphones often contain multiple SoCs other than the primary application SoC – particularly sensor hub and wireless communication SoCs – to which low-intensity background tasks can be delegated without waking the main application SoC. The application SoC can then remain in a low-power state in order to minimise power consumption and prolong battery life. With respect to larger mobile devices, the HP Envy X2[14] laptop is powered by a Qualcomm Snapdragon 835 SoC. Notably, SoC-based architectures tend to transition to traditional architectures at this point, i.e. lower-to-mid range laptops. Devices intended for high-performance gaming and video editing, for example, typically use discrete PCBs for GPUs, memory, audio, and so on, in order to maximise computational performance.

2.2 Secure Execution Platforms

In this thesis, a secure execution platform is designed to offer strong assurances regarding the preservation of the integrity, authenticity, availability and confidentiality of its host applications. This includes any associated data, both persistently and at run-time. Examples of such platforms include traditional smart cards, Java Card and Secure Elements (SEs). The platform is built to resist significantly greater security threats, including hardware- and software-based attacks.

The next section (Section 2.2.1) describes the Common Criteria for Information Technology Security Evaluation (Common Criteria or CC, for short), which is a widely-used computer security framework for evaluating and certifying the security of secure and trusted execution platforms. This is introduced to discuss and briefly compare the assurance levels to which platform is evaluated. The subsequent sections present a review of widely-deployed, hardware-assisted secure execution platforms in mobile and embedded systems. This includes a brief examination of smart cards (Section 2.2.2); Secure Elements (SEs), e.g. UICC and SIM Cards (Section 2.2.3); and Java Card (Section 2.2.4). Additionally, Host-based Card Emulation is discussed in Section 2.2.5 for emulating smart card-based services on mobile platforms.

[14]HP Envy X2: http://www8.hp.com/us/en/campaigns/envy-x2/overview.html

2.2.1 Common Criteria

The Common Criteria for Information Technology Security Evaluation – commonly referred to as Common Criteria (CC) – is an international standard for evaluating product security, which is formalised in ISO/IEC 15408 [38]. Anderson [39] defines security evaluation as *"the process of assembling evidence that a system meets, or fails to meet, a prescribed assurance target"*. CC replaced individual national schemes, such as the US Department of Defense's Orange Book and European Information Technology Security Evaluation Criteria (ITSEC), which previously required vendors to ascertain multiple evaluation certificates based on the intended deployment region. CC is used to convince procurers and other stakeholders, such as government agencies, that a security product robustly conforms to a recognised specification.

In CC, the product presented for evaluation by its vendor is known as the Target of Evaluation (TOE). A Protection Profile (PP) is then chosen that stipulates the security requirements of a *class* of systems; for example, CC PPs have been developed by for TEEs (GlobalPlatform TEE PP [40]) and SEs (GlobalPlatform SE PP [41]). CC PPs explicitly include list of threats, assumptions, organisational policies, security objectives, rationales, security functional (SFR) and assurance (SAR) requirements. A separate but closely-related Security Target (ST) is also often defined that contains implementation-specific details of a TOE, which claims conformance to one or more CC PPs. The vendor makes claims regarding the ST's attributes to those in the PP, and an independent testing laboratory evaluates the TOE to assess the rigour of those claims.

The degree to which the TOE satisfies the assurances is graded using Evaluation Assurance Levels (EALs), ranging from EAL1 for which basic functional testing is required, up to EAL7 requiring a formally verified design. The ST specifies the Evaluation Assurance Level (EAL) that the TOE ought to fulfil. The EALs provide confidence that the TOE's supposed security features are implemented reliably. The CC EALs are described in greater detail in Table 2.1.

2.2.2 Smart Cards

A smart card is a lightweight, pocket-sized device containing an embedded integrated circuit(s). They are securely packaged using a polymer, such as polyvinyl chloride (PVC), and used to store sensitive personal credentials. Notable application domains include physical access control systems, like security gates and doors; financial transactions, e.g. credit and debit cards; public transportation tickets and passes; and national identity card schemes, like those used in Belgium, Estonia and Finland [43]. Contact-based smart cards use connective pads for

TABLE 2.1: Common Criteria Evaluation Assurance Levels (EALs) based on US-CERT definitions [42].

EAL	Description
1	Functionally Tested: *Confidence is required in the TOE's correct operation against non-serious threats; provides evidence that the TOE functions consistently with its documentation and provides useful protection against identified threats.*
2	Structurally Tested: *Low-to-moderate security assurances, but the complete development record is not readily available. This situation may arise when there is limited developer access or when securing legacy systems.*
3	Methodically Tested and Checked: *Moderate assurances and a thorough investigation of the TOE and its development without substantial re-engineering.*
4	Methodically Designed, Tested and Reviewed: *Moderate-to-high assurances in conventional commodity products; prepared to incur additional security-specific engineering costs.*
5	Semiformally Designed and Tested: *High assurances in a planned development environment; requires rigorous development approach that does not incur unreasonable costs from specialist security engineering techniques.*
6	Semiformally Verified Design and Tested: *Applies to TOEs in high-risk situations where the protected asset value(s) justifies the additional costs.*
7	Formally Verified Design and Tested: *Applies to TOEs in extremely high-risk situations and where the high value of the assets justifies the higher costs.*

electrical activation and data transmission. Contactless smart cards communicate with a terminal occurs over a radio frequency interface, usually at a <10cm distance. Contactless and contact-based smart cards are standardised in ISO/IEC 14443 [44] and 7816 [45] respectively. These standards define, among others, the physical characteristics of cards, including dimensions and location of electrical contacts; operational frequency (13.56 MHz); radio power; and initialisation, transmission, anti-collision, application and data management protocols. Standards also exist specifying the operational requirements of smart cards within particular domains, such as EMV[15] for the technical details of payment cards and terminals.

A typical smart card architecture comprises a 8-/16-/32-bit CPU (usually with <35MHz clock); ROM for hosting a slimline OS and its applications (<32kB); EEPROM for data storage (<24kB); RAM (<8kB) for temporary in-execution storage; and a security co-processor for performing cryptographic operations, e.g. 3DES, AES and RSA [46]. High-security smart cards, such as credit and national ID cards, are engineered to be tamper-resistant. This includes a co-processor built to resist a range of hardware/physical security threats, such as simple and differential power analysis (SPA/DPA), layered packaging that deters invasive physical attacks without damaging the card, and a rigorously-tested OS. Smart

[15] EMV: Europay, Mastercard, Visa.

cards are also manufactured and provisioned in a secure production environment. Cards for high-security commercial applications, e.g. healthcare access control cards, are typically evaluated to CC EAL4 [47].

2.2.3 Secure Elements (SEs)

A Secure Element (SE) is a tamper-resistant platform capable of securely hosting code and confidential data, such as cryptographic keys, according to the processes established by its owner. SEs are currently available in three form factors: the Universal Integrated Circuit Card (UICC), commonly found in the SIM card format; the embedded SE (eSE), which is integrated into the NFC chip or directly onto the device PCB; and an enhanced secure microSD card. All SE forms accommodate a smart card microcontroller for storing sensitive credentials and hosting a limited set of applications. The use of a microcontroller means SEs typically have restricted memory and processing capacity relative to an SBC or traditional computing system, as discussed in Section 2.1.3. SEs are often used in conjunction with NFC for performing payment tokenisation for contactless payments using multi-use tokens provisioned into the SE before deployment [48]. Another common application is the execution of fingerprint matching algorithms and the storage of users' biometric templates [49]. In general, SEs aim to defend against threats similar to that of a smart card (Section 2.2.2), including advanced side-channel analyses, e.g. power analysis, and fault injection [48]. GlobalPlatform has standardised SEs in the Financial [50], UICC [51], Card Common [52], and Card Contactless [53] configuration specifications.

2.2.4 Java Card

Java Card [54] provides an interoperable secure platform for SEs, which allows them to run multiple applications (applets) written in a subset of the Java language. The fundamental components of a Java Card platform are: 1), a Virtual Machine (VM) composed of a bytecode interpreter providing hardware-independence and, from version 3.x, a mandatory embedded bytecode verifier. 2), a set of APIs that abstract the complexity of the underlying smart card communications protocols, offer access to cryptographic functionalities, and secure inter-applet data sharing. 3), a Java Card Runtime Environment (JCRE) with additional security mechanisms, such as atomically updating persistent data. And, lastly, 4), a firewall that maintains strong isolation between applets and the system, and between applets from differing packages (see Figure 2.4).

The firewall enforces the security rules by restricting the rights of an applet to applets from the same provider but in different packages, or in the same package from different providers. Applets from the same package are in the same *context*.

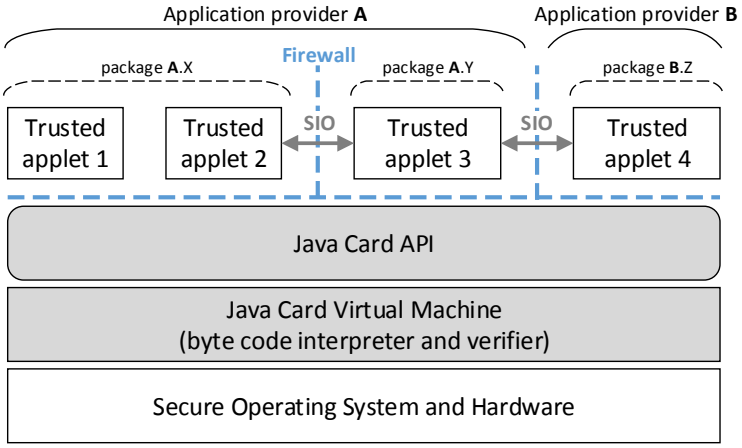
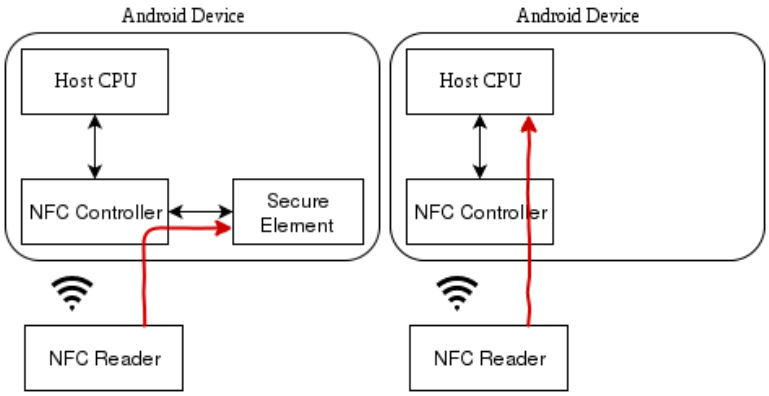


FIGURE 2.4: Java Card architecture.

An applet may only access objects owned by another applet in a different context if they are explicitly tagged as 'shared' items. Notably, while Java Card may host multiple applets, it is a single-threaded environment. The traditional ownership model for Java Card is issuer-centric (ICOM), where only the Java Card issuer can install applications. GlobalPlatform has standardised mechanisms through which authorised third-parties can install applications via the use of Security Domains (SDs), which are described later in Section 2.4.7.

2.2.5 Host-based Card Emulation (HCE)

Host-based Card Emulation (HCE), introduced in Android 4.4 (API 19), enables NFC-enabled mobile devices to emulate contactless smart cards [55]. HCE's aim is to allow mobile devices to fulfil services with existing NFC-based card-reader infrastructures, which would otherwise have been deployed using smart cards, without using discrete SEs [48]. Prior to HCE, data sent and received to a terminal was routed through an NFC controller connected directly to a SE aboard the mobile device (Figure 2.5a). With HCE, the host OS handles the routing of the messages, which *may* use a hardware-based SE or an application executing on the host CPU. In Android, this is implemented as an application service that executes as a long-running background task. This allows the user to place the device against an NFC reader to execute transactions with the correct service in the background, without manually launching a dedicated mobile application. In both SE- and HCE-based deployments, data is routed to a particular application based on its 16-byte Application ID (AID), as stipulated in ISO/IEC 7816-4 [45]. The AIDs are registered with the mobile's NFC controller, which maintains a routing table of the AIDs belonging to which applications [55]. In NFC transactions, the first Application Protocol Data Unit (APDU) is the `SELECT` command, also



(A) SE-based (B) HCE-based

FIGURE 2.5: Card emulation using a SE (a) and HCE (b).

defined in ISO/IEC 7816-4 [45], containing the AID of the application with which to process further NFC transaction messages.

The confusion lies in that, while HCE aims to supplant the role played by hardware SEs and contactless cards, it does *not* provide the same degree of tamper-resistance. In itself, HCE is a routing mechanism, while credential *storage* and application *execution* may, at a minimum, be protected by the security mechanisms afforded by the OS itself, such as application sandboxing, i.e. software-only. This is opposed to SEs and contactless cards where application execution and credential storage is conducted within a tamper-resistant platform. As such, it is not recommended that critical transactions, like mobile payments and high-security access control, are based solely on HCE [55]. A cloud-based SE is seen as one solution whereby credentials are managed off the device and where the HCE-enabled application acts a medium for acquiring dynamically-generated, limited-use credentials from a remote server [56]. However, this has the drawback of requiring network connectivity. Another solution is the Trusted Execution Environment (TEE), described in Section 2.4, to host these credentials; we return to mobile-based NFC transactions in conjunction with TEEs in greater detail in Chapter 7.

2.3 Trusted Platform Module (TPM)

The Trusted Computing Group (TCG) defines trust as the *"expectation that a device will behave in a particular manner for a specific purpose"* [57]. In highly constrained environments, establishing trust may be achievable if devices are vetted off-line beforehand; isolated from the outside world, e.g. air-gapped; and frequently inspected thereafter. However, this is not typically the case in reality, and a range of trusted computing technologies have been developed to provide remote assurances that individual platforms are in a trusted state.

The Trusted Platform Module (TPM) is one such technology – defined in a suite of specifications developed by the TCG and standardised in ISO/IEC 11889 [58]. The TPM is a tamper-resistant processor intended to securely perform cryptographic operations – key generation, a PRNG and various algorithms, e.g. AES, SHA-2 and RSA – and providing evidence of the platform's operating state. Each TPM is provisioned with module-specific keys during manufacturing, which are used in *remote attestation* and data *sealing* and *binding*. Remote attestation (Section 2.3.2) is the mechanism through which the platform's operating state is measured, i.e. in terms of firmware and software components collected during measured boot (Section 2.3.1), which are securely transmitted to a remote verifier upon request. Notions of *binding* and *sealing* for secure storage are discussed further in Section 2.3.3. The TPM is typically used as a Root of Trust (RoT): a component that is inherently trusted and used for providing assurances that the platform is behaving in the expected manner.

This section briefly describes these core abstractions provided by the TPM. The reader is referred to ISO/IEC 11889 [58], Balfe et al. [59] and Sadegi [60] for detailed technical descriptions of the TPM and its potential applications.

2.3.1 Measured Boot

Measured boot is used for measuring and storing the integrity state of critical system components at boot-time. This is performed using a set dedicated 160-bit Platform Configuration Registers (PCRs) contained within the TPM for storing cryptographic hash measurements of these boot components. The TCG TPM Profile Specification stipulates a minimum of one PCR bank containing 24 registers [61].

After a system reboot, executable code known as the Core Root of Trust for Measurement (CRTM), which is stored in ROM, is used to initiate the component measurement process that computes and extends the PCR values (usually performed by the BIOS's boot block). Each component is measured into a PCR by *extending* its initial value using the `TPM_Extend` command that computes $PCR_i = h(PCR'_i||X)$. Here, i is the PCR index, PCR'_i represents the old PCR value at i; X is the data argument appended to the old PCR value; h is a cryptographic one-way hash function, e.g. SHA-256; and $||$ is the concatenation operation. Note that the PCRs are zeroed initially upon boot.

A typical configuration is to use PCRs 0-1 for storing BIOS measurements including the CRTM, 2-3 for any optional ROMs, 4-5 the master boot record (MBR) and MBR configuration, 6-7 for state transitions and platform-specific functionality, 8-15 the OS, and 16 onwards for any additional specific application measurements; some PCRs may also remain unused [61]. The information and execution flow for measured boot from an initial RoT is illustrated in Figure 2.6.

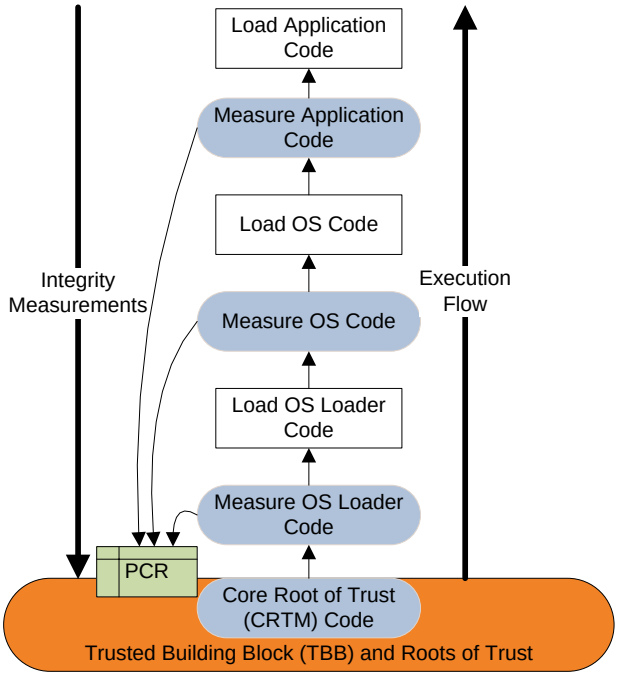


FIGURE 2.6: Information and execution flow for measured boot.

The system's integrity state is thus represented in the values extended through the PCRs; if any component is compromised and modified or replaced maliciously, then the measured hash at that PCR index will deviate from its expected value. After boot, the Trusted Building Block (TBB) is established and the CRTM proceeds with loading the primary OS. Importantly, measured boot does not capture the state of *every* possible application present on the platform; if a particular application is not measured and extended into a PCR, then it may still be compromised, even if the remaining PCR values are as expected.

2.3.2 Remote Attestation

Remote attestation (RA) is an interaction protocol between a prover, P, and verifier, V, in which V ascertains the current state of a remote platform, i.e. P. The goal of RA is assure V that P is operating with an expected platform configuration, which is ascertained remotely using a secure channel over a network. The TPM PCR values comprise the platform integrity measurements sent to a remote verifier in order to evidence the system components loaded at boot-time. In traditional attestation, the PCR values are packaged into a data structure known as a *quote* report and signed by the TPM's attestation identity key (AIK), which is certified by the TPM vendor to assure its authenticity and integrity. The most recent TCG TPM specification (version 2.0) stipulates the use of Direct Anonymous Attestation (DAA). DAA tackles the privacy challenge that signing quotes with a AIK unique to each TPM allows an adversary to monitor the activity of particular

TPM. DAA solves this using group signature schemes, where one public-key may be mapped to multiple private AIKs, which prevents an adversary from learning the activity of a particular TPM from signed TPM quotes.

Numerous RA protocols have been proposed in the literature for use with differing trusted technologies, e.g. TPMs or TEEs; adversarial models; privacy requirements; and its integration with existing protocols, e.g. TLS and IPsec [62]–[72]. The reader is referred to Abera et al. [62] for an IoT-centric review of remote attestation mechanisms. Recent literature has tackled the challenge of using a single remote attestation request to evaluate the state of multiple systems, such as drone fleets [73]. The technicalities of RA are described later in greater detail in Chapter 4.

2.3.3 Binding and Sealing for Secure Storage

Two other functionalities provided by the TPM are data *binding* and *sealing*, which are used to realise secure storage from a master Storage Root Key (SRK) that is generated when a user assumes ownership of the TPM. The SRK is a uniquely generated key that never leaves the confines of the TPM.

The first abstraction, *binding*, is the process by which the TPM generates a cryptographic binding key-pair, k, derived from its SRK, whose public component is returned to the calling program. k may then be used to encrypt arbitrary binary large objects (blobs) outside the TPM using the `TSS_Bind` command. (Auxiliary functions used to support operations outside the TPM are defined in the TCG Software Stack (TSS) specifications [74]). An associated authentication token, also known as authdata – akin to a password [75] – can be specified in order to restrict 'unbinding' operations to applications with that token. Encrypted blobs may be decrypted using the `TPM_Unbind` command, which, given a binding key, k, and an encrypted object, decrypts it within the TPM using the private component of k, which never leaves the TPM.

The second abstraction, *sealing*, is performed similarly, but with the crucial distinction that the TPM encrypts data blobs such that they can only be decrypted under the same PCR configuration. This prevents data being exposed to a compromised system or if the encrypted blobs and keys are migrated to another platform with different PCR values. Sealing and unsealing is performed using the TPM commands `TPM_Seal` and `TPM_Unseal` respectively.

2.3.4 TCG TPM 1.2 and 2.0

The features and capabilities of the TCG TPM have been developed and revised, resulting principally in the TPM 1.2 and 2.0 specifications. The main differences centre around the flexibility in the available cryptographic algorithms and the

initial key hierarchies. TPM 2.0 enables so-called 'algorithm agility' that allows a greater variety of cryptographic algorithms to be used, rather than a single asymmetric (RSA) and hashing algorithm (SHA-1) as per the TPM 1.2 specification. TPM 2.0 modules support elliptic curve-based operations, e.g. ECDSA and ECDH, including ECC-based DAA by Chen et al. [63] for remote attestation, as well as SHA-256 for HMACs and general-purpose hashing, rather than SHA-1 alone.

Another difference is that TPM 1.2 contains a single key hierarchy rooted in the storage root key (SRK) – a 2048-bit RSA key – while TPM 2.0 comprises *storage*, *platform* and *endorsement* hierarchies. The TPM 1.2 SRK is generated randomly and cannot leave the TPM. Child keys are encrypted (wrapped) by the SRK, which may subsequently be used to wrap their own child keys. This hierarchy is controlled by a single owner, or administrator. The TPM 2.0 hierarchies intend to separate the use of keys from platform firmware that cannot distinguish whether the TPM is enabled and initialised. It also aims to separate privacy-sensitive and non-sensitive application key uses. The TPM 2.0's *storage* hierarchy is analogous to the TPM 1.2's hierarchy, which is intended for use by the user. The *platform* hierarchy allows the OEM to provision a list of trusted public keys in the TPM's NVRAM, which are used to verify system component signatures. Lastly, the *endorsement* hierarchy is usually for wrapping keys used for signing remote attestation quotes; each endorsement key (EK) is unique to the TPM, and the TPM vendor provisions an accompanying certificate for authentication.

Other differences include the use of multiple algorithms per key hierarchy in TPM 2.0 and the inclusion of a timer, clock and counters in NVRAM. The reader is referred to [61] in which the revisions between TPM 1.2 and 2.0 are described in greater detail.

2.4 Trusted Execution Environments (TEEs)

Trusted Execution Environments (TEEs) are platforms that operate alongside standard mobile operating systems. TEEs provide two primary features: 1), strong isolated execution of trusted applications – usually hardware-enforced – both at run-time and at rest, and 2), secure storage of persistent data, such as cryptographic keys. TEEs may also provide remote attestation for allowing remote entities to verify its operating state, and a trusted path with I/O peripherals. Broadly, TEEs address the shortcomings of previous technologies with greater performance and integration with the underlying hardware. Standards-wise, GlobalPlatform specifies a TEE system architecture [76]; a range of TEE APIs, particularly the GP Internal and Client APIs [77], [78]; and a TEE protection profile for Common Criteria compliance [40]. The GlobalPlatform TEE and its

specification suite is discussed further in Section 2.4.7. A range of non-compliant TEEs have been developed, however, particularly using the TPM as a root of trust; a broad range of technologies are described in this section.

2.4.1 TPM-backed Virtualisation

Virtualisation is the process of creating virtual computing systems or resources, such as a storage device or network. Virtual Machines (VMs) can imitate independent computing platforms (guest machines) with potentially different OSs that run upon the hardware of the host platform (host machine). VMs are instantiated and managed by a hypervisor or Virtual Machine Manager (VMM). Hypervisors are designed to provide strong isolation between guest machines to prevent a compromised guest machine from easily influencing other guest machines. Software-only virtualisation, such as Android application sandboxing, is used in preventing inter-application compromises, as well as in commercial hypervisors, such as Xen, for guest-to-guest and guest-to-host security in cloud computing.

In this section, several key technologies are explored that instantiate TEEs using a TPM to launch a trusted hypervisor layer. These technologies make use of a Dynamic Root of Trust for Measurement (DRTM) and the concept of 'late launch'. Here, secure boot is performed as usual in which hashes of critical boot components, e.g. BIOS and MBR, are loaded into PCRs 1–16 (see Section 2.3.1). Next, without requiring a reboot, the remaining PCRs (17–23) are reset to zero, and then the code necessary to load the TEE is measured and stored into these PCRs before it is executed. This assists in launching a trusted state without rebooting the entire machine in order to remeasure the relevant components. In this section, the major commercial and highly-cited academic technologies are examined.

Intel Trusted eXecution Technology (TXT)

Intel Trusted eXecution Technology (TXT) [80], formerly known as LaGrande technology, aims to defend computing platforms from generalised software attacks, including firmware, BIOS and rootkit attack vectors. TXT constructs a chain of trust using secure boot, which utilises the TPM as a root of trust. Late launch, performed using the `SKENTER` Intel CPU, is subsequently used to measure the state of the Intel TXT VMM launcher code. Any underlying system modifications to the BIOS, MBR or VMM, such as by rootkits and other surreptitious software, can be detected from deviations in the launch configuration from the TPM's PCR values. TXT interoperates with Intel's Virtualisation Technology (Intel VT) [81], which provides native hardware virtualisation extensions for providing high degree of isolation while retaining performance. Facilities are also offered for

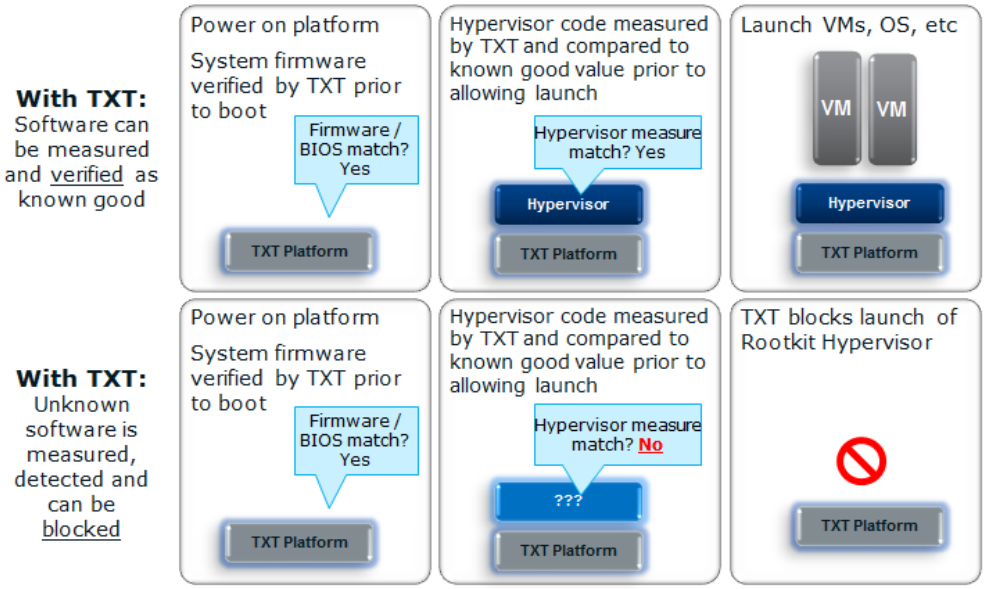


FIGURE 2.7: Using Intel TXT for trusted hypervisor launch and VM instantiation [79].

platform attestation in which a local or remote party is able to determine the trust status of the launch configuration of the platform or VM, i.e. whether or not it was instantiated correctly. A disadvantage of TXT is its significant hardware Trusted Computing Based (TCB), comprising the TPM, CPU chipset, motherboard and system buses. Figure 2.7 illustrates a high-level architecture for instantiating trusted launch of VMs using TXT.

AMD Secure Virtual Machine (SVM)

AMD Secure Virtual Machine (SVM) is a TXT-like, DRTM-based environment for AMD chipsets that launches a trusted hypervisor using a TPM 1.2 on the computer's motherboard. SVM provides a set of hardware extensions to maximise the efficiency of world switches between the hypervisor and guest domains, and allows the allocation of I/O device that can only be accessed by a particular guest machine (but not others) [82]. Similar to Intel TXT, SVM provides support for attestation of the boot values measured with late-launch, which is performed using the `SKINT` CPU instruction. SVM provides an additional security mechanism, 'automatic memory clear', which erases the system's memory contents after a reset to prevent an adversary from accessing secrets by simply reading memory contents.

Flicker & TrustVisor

McCune et al. [83] propose Flicker – a system architecture for protecting sensitive code execution from an adversary that controls the BIOS, OS and DMA-enabled devices. The aim of Flicker is to allow trusted execution of small arbitrary applications. Like Intel TXT and SVM, Flicker uses late-launch but, rather than launching a hypervisor, it pauses execution in order to run a small piece of application logic known as a Flicker module containing security-sensitive code. Flicker executes the remainder of the paused execution flow, i.e. in the untrusted OS, after completing the execution of the Flicker module. Flicker modules are linked with and managed by another entity, the Secure Loader Block (SLB), for erasing secrets kept in memory, returning values through a bespoke Linux kernel module, and triggering the resumption of the untrusted OS's execution. Flicker application code is extended into PCR 17 by the SLB Core, which has an intentionally small TCB (0.312kB or 94 lines of code). This PCR value is included in remote attestation quotes such that remote entities can be convinced of the Flicker application's operating state. Flicker is implemented and evaluated across a range of applications, including rootkit detection (up to 1022ms overhead), SSH password authentication (up to 937ms), and certificate signing (906ms).

TrustVisor is a second proposal by McCune et al. [84] that, like Flicker [83], allows the execution of sensitive applications while retaining a small TCB. Its primary difference is the use of an attestable, slimline *hypervisor* upon which sensitive applications are executed, rather than attesting an individual application. TrustVisor is, hence, similar to Intel TXT and AMD SVM, but purposely omits scheduling and inter-process communication in order to minimise its software TCB. TrustVisor is larger than Flicker in terms of its software TCB – <10k lines of C and X86 Assembly, compared to Flicker's <1k LOC – but yields greater performance, with only 7% overhead versus unprotected code. TrustVisor, like Flicker, is launched using a TPM using late-launch and attested similarly by extending a PCR responsible for the hypervisor code. TrustVisor also exposes individually virtualised 'micro TPMs' to each descendent application, in which the keys of each virtualised TPM is derived from the host (hardware) TPM, thus preventing applications from unsealing other applications' sealed secrets. The state of each application is measured upon launch into a single PCR held in each micro TPM launched by TrustVisor. During remote attestation, these micro TPM PCRs are aggregated into a single value alongside the TPM's PCR values collected at boot-time, rather than extending a virtually arbitrary number of application states into the TPM's PCRs.

2.4.2 Microsoft Palladium

The Next-Generation Secure Computing Base (NGSCB), otherwise known as Palladium or Trusted Windows, was an architectural proposal by Microsoft that promised *"users greater data security, personal privacy, and system integrity...[and] enterprise customers significant new benefits for network security and content protection"* [85]. Palladium, shown in Figure 2.8, separates execution into two abstract worlds: 'standard' and 'trusted' worlds, in which the trusted world hosts a bespoke security kernel, known as Nexus, containing a set of critical security applications known as 'agents'. The standard OS communicated with Nexus and its agents using a Windows kernel module known as `NexusManager.sys`. A proprietary TPM, referred to as the Security System Component (SSC), was employed for attestation, Nexus' measured boot, and secure storage of application data from the standard and trusted worlds.

Additionally, Palladium allowed Nexus to access main memory in order to move data written from DMA-enabled peripherals into Nexus space for secure I/O. Palladium also provided means to counter keyloggers, even with administrative privileges, by encrypting keystrokes at its USB controller under a key shared with a Nexus agent for further processing, e.g. email. Nexus also had read/write access to the video memory buffer of on-board graphics adapters in order to neutralise screen-scrapers executing in the untrusted world [86].

A shortcoming of Palladium was its requirement for extensive modifications to the CPU chipset and motherboard, particularly in light of competing proposals at the time, namely Intel TXT [86]. Moreover, Palladium faced strong criticism from privacy and free software activists, who expressed concerns over its potential role in enforcing Digital Rights Management (DRM), vendor lock-in, and commercial and even political censorship [87]. The Palladium project was cancelled in 2005.

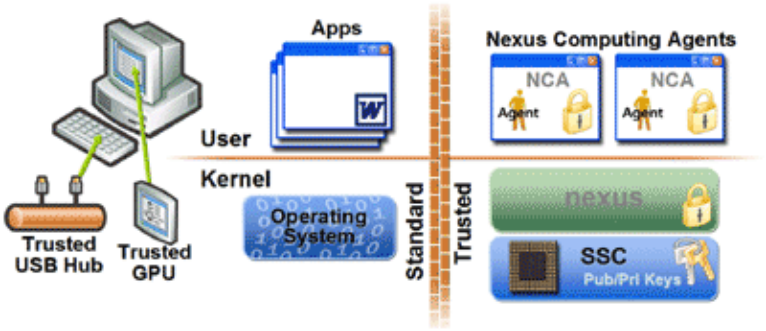


FIGURE 2.8: Overview of Microsoft Palladium in 2003 [88].

2.4.3 Texas Instruments M-Shield

Texas Instruments developed an early inception of the TEE known as M-Shield. M-Shield is a platform that resembles a SE with several additional hardware modules for realising a trusted path, including over DMA, without any intermediary untrusted components, such as kernel drivers residing in an untrusted OS. It features a hardware cryptographic processor and implements a secure state machine (SSM) that monitors and enforces system security policies for preventing unauthorised accesses to sensitive material. The SSM enforces peripheral and memory accesses from the secure environment, including the management of DMA channels that guarantee data confidentiality from peripherals to secure memory regions [89]. M-Shield also provides one-time programmable ROM modules for storing long-term key information, e.g. device-specific keys, accessible only to the secure environment. Comprehensive details, however, remain sparse in publicly-available documentation. Later iterations of M-Shield from 2006 integrated ARM TrustZone software and APIs. M-Shield underpinned the security of the TI OMAP SoCs deployed on late-2000s/early-2010s smartphones and tablets; however, production was ceased in 2013.

2.4.4 ARM TrustZone

ARM TrustZone is a collection of hardware IP blocks that enable the instantiation of a TEE on ARM-based SoCs. Fundamentally, TrustZone establishes 'secure' and 'non-secure' worlds[16] intended to host security-sensitive and non-sensitive applications respectively. This is maintained through a separate NS register bit that denotes the world in which execution is currently occurring. The NS bit is propagated through the SoC's bus transactions to memory, peripheral and debug controllers for providing hardware-enforced access control to sensitive memory regions, debug interfaces and I/O devices. As a result, each world may only access the resources, namely memory resources and peripherals, associated with it; non-secure applications may not arbitrarily access secure world RAM and sensitive I/O peripherals unless configured to do so. This is illustrated at a high-level in Figure 2.9.

The creation and maintenance of the worlds is handled primarily by the processor and SoC bus components. On the processor, each physical core is associated with two virtual cores for secure and non-secure execution that operate in a time-sliced fashion. For Cortex-A CPUs, world context switches are mediated by a secure monitor that enforces that the departing and arrival worlds are being correctly saved and restored. Monitor mode is entered, for software-based switching, using Secure Monitor Call (`smc`) instructions, or by using a small set

[16] Also referred to as 'trusted' and 'untrusted' worlds.

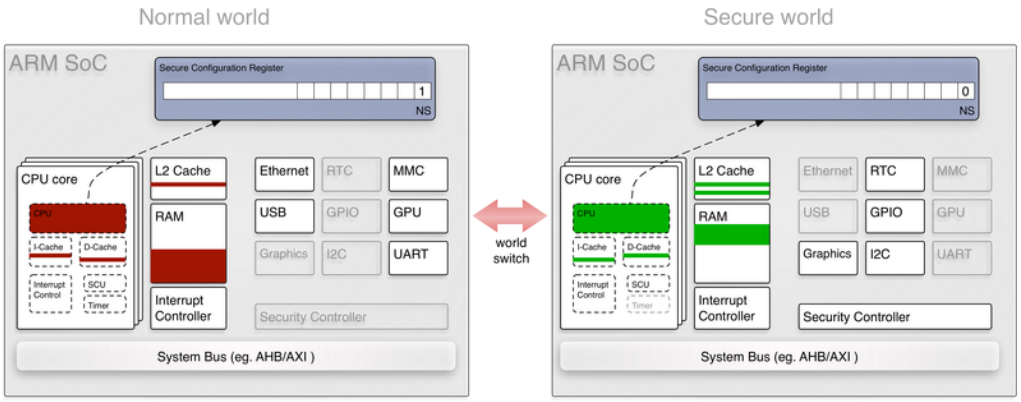


FIGURE 2.9: TrustZone separates trusted and untrusted system components using the NS-bit [90].

of pre-configured hardware exceptions (see [91]). SMC instructions generate an exception that takes the processor's execution mode into Secure Monitor mode, which is code that is part of ARM's Trusted Firmware framework[17] running at a higher privilege level (EL3 in ARM nomenclature) and outside the untrusted world's context (operating at EL1). The monitor saves the current world's processor context, such as register banks, switches to the other world, and sets the NS-bit accordingly if successful.

Further IP cores, such as the TrustZone Address Space (TZASC) and TrustZone Protection Controllers (TZPC) are used in enforcing access control to secure memory regions and I/O peripherals based on SoC bus transactions. TZASC is used to program the partitioning of DRAM into secure and non-secure memory address regions. Meanwhile, TZPC is used to assign protection bits to peripherals to mark them as available to either the only the secure or both worlds, which is used to prevent unauthorised accesses to protected peripherals from the untrusted world by the AXI-to-APB SoC bus bridge. Lastly, the TrustZone Memory Adapter (TZMA) is used to protect sensitive static memory modules, like secure SRAM and ROM, contained within the SoC.

To fully implement the TEE, additional software is necessary for securely booting the secure world, configuring the protection controllers, implementing the secure monitor, a trusted OS, and trusted applications. TrustZone's authenticated boot sequence operates similarly to the TPM's by measuring a chain of trusted components from secure ROM. A TrustZone OS is also necessary to enable the hosting of multiple applications, as well as associated APIs with the untrusted world, which are standardised in the GlobalPlatform TEE specifications (Section 2.4.7). Derivative technologies have been developed that use TrustZone as a

[17] ARM Trusted Firmware: https://github.com/ARM-software/arm-trusted-firmware

root of trust (RoT) on other processor architectures, as well as providing additional features, such as remote attestation, which are not provided natively by TrustZone. An example TrustZone-based SoC is illustrated in Figure 2.10.

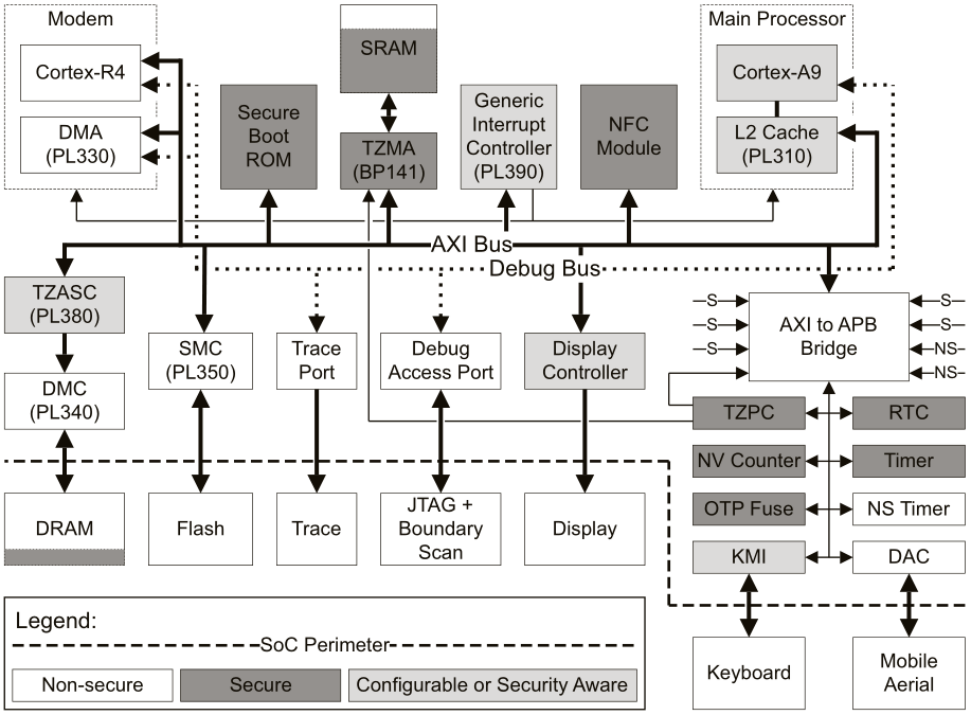


FIGURE 2.10: Example ARM-based SoC with TrustZone, showing non-secure and security-aware IP blocks [92].

TrustZone for ARMv8-M (TrustZone-M)

The previous discussion pertains to ARM TrustZone as originally realised for Cortex-A application processors using the ARMv8-A architecture (TZ-A). TrustZone technology has also been extended to the ARM Cortex-M processor family, marketed for use in embedded microcontroller units. Here, TrustZone for ARMv8-M [93], or TrustZone-M (TZ-M), shares the same high-level security features as TZ-A in dividing execution and physical memory into secure and non-secure worlds; however, they feature significant differences in how it is realised.

For world switching, as stated previously, TZ-A uses enters secure world functions via secure monitor code in EL3 that mediates the final world switch; that is, the secure monitor is the sole access point for performing world transitions. In comparison, TZ-M uses a memory map-based approach, where secure and non-secure regions are defined in a programmable or fixed fashion using the Secure Attribution Unit (SAU) and Implementation Defined Attribution Unit (IDAU) aboard the Cortex-M [93]. World context switches occur automatically without the need for an secure monitor exception handler as program execution

flows from non-secure to secure regions and vice-versa. To prevent unauthorised secure world accesses, transitions from non-secure memory must first follow a Secure Gateway (`sg`) instruction held in Non-Secure Callable (NSC) memory – a separate region initialised by the SAU or IDAU containing 'secure gate veneers' that point to valid entry points that secure world code can be branched into [94]. Attempting to access secure memory without a preceding `sg` instruction triggers an exception handled in the secure world. Unlike TZ-A, TZ-M shares all general purpose registers except the stack pointer (`sp`) registers during world switches to further minimise switching latency and energy consumption [93]. These differences accumulate to a reduction from *"thousands"* to a *"few"* processor cycles when comparing TZ-A to TZ-M for performing world switches [94].

AMD Secure Processor

AMD Secure Processor, formerly AMD PSP, is a technology for AMD chipsets for executing sensitive software in a TEE. AMD Secure Processor uses an ARM-based SoC packaged into the AMD chipset that uses TrustZone for establishing secure and untrusted worlds for realising secure application execution [95]. Besides marketing material [95] and technical presentation slides [96], few published details exist regarding the actual functionality and capability of AMD Secure Processor. However, it is known to contain a Trustonic TEE kernel, and is used to implement a TEE-based TPM and host DRM, payment and identity applications from trusted third-party providers on commercially-available AMD chipsets [96].

Samsung KNOX

Samsung KNOX is a mobile-based security platform built upon a TrustZone TEE. The secure boot process of the untrusted and trusted worlds is enhanced to set a KNOX Warranty Bit in one-type programmable ROM if boot component measurements deviate from their expected values. This terminates the KNOX platform indefinitely, which is persists across device reboots and resets [97], [98]. KNOX also offers remote attestation using a device-specific root key for signing attestation quotes [97]. KNOX is underpinned by a Trustonic TEE, which uses ARM TrustZone, on all flagship Samsung handsets, e.g. Galaxy S6–S9 smartphones [99]. However, few publicly-available technical details exist, particularly regarding the secure boot process and attestation protocol.

2.4.5 Nokia On-board Credentials (ObC)

On-board Credentials (ObC) was a project developed at Nokia Research Center between 2005–2013 for securely storing and using credentials intended for mobile-centric services, e.g. payments and transport. It was intended to address the

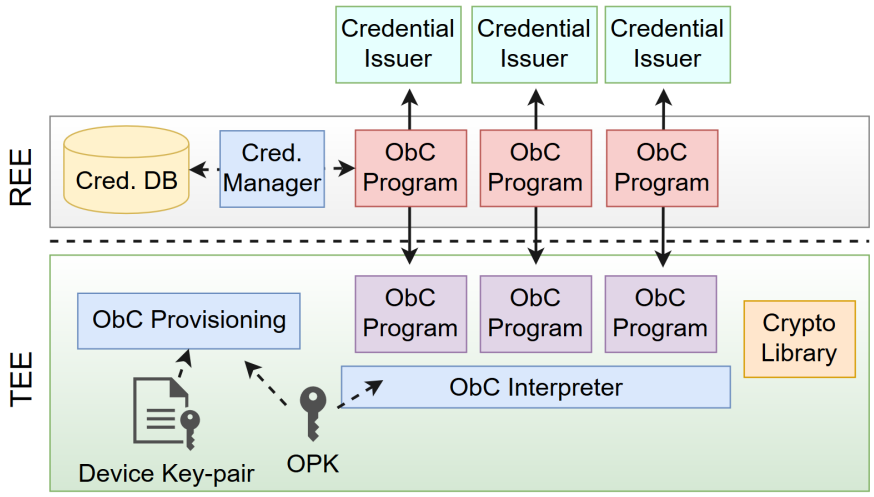


FIGURE 2.11: ObC system architecture [101]).

challenge of protecting wholly software-based credentials stored in a standard OS environment without incurring the expense and inflexibility of hardware-bound credentials [100]. Fundamentally, in its most recent form, ObC used ARM TrustZone, which hosted a VM, the ObC interpreter, for providing isolated execution of ObC programs from untrusted developers within the TEE.

Each ObC device contains a unique root seal key, the ObC Platform Key (OPK), accessible only to the TEE. This is used to derive subsequent, per-ObC program keys for sealing data to the device filesystem.Each ObC-enabled device is deployed with a certified key-pair by the manufacturer, accessible only to the TEE, which is used for device authentication in the provisioning of credentials from their issuers. This key-pair is also used to sign remote attestation quote responses. Each ObC program has access to a TEE-based cryptographic library and a secure source of randomness provided by the underlying hardware implementation. ObC also contains a corresponding set of REE applications for networking with external entities, namely credential issuers. ObC was deployed on all Nokia Symbian Belle and Windows Phone 8 devices [100]. ObC's system architecture is illustrated in Figure 2.11; the reader is referred to work by Kostiainen [101] (esp. Chapter 4 of [101]) and Ekberg [102] (esp. Section 9.1 of [102]) for detailed analyses regarding ObC.

2.4.6 Intel Software Guard eXtensions (SGX)

Intel Software Guard eXtensions (SGX) is an extension to the X86-64 instruction set architecture (ISA) that enables the creation and management of 'enclaves' in which sensitive code and data is hosted and executed. In general, Intel SGX aims to protect primarily against software-based adversaries originating from any protection mode, e.g. untrusted user- (ring 3) and kernel-mode (ring 0)

applications. At run-time, enclave code and data is protect by a hardware TCB including only the Intel CPU. This is achieved predominantly by allocating a DRAM region known as Processor Reserved Memory (PRM) containing the Enclave Page Cache (EPC) that holds four-kilobyte pages for enclave code and data. The CPU protects enclave pages in PRM from non-enclave and DMA memory accesses from external devices. An additional region, the Enclave Page Cache Metadata (EPCM), is used to store meta-data regarding the mapping of EPC pages to enclave identities. Enclave pages are encrypted outside the CPU using a proprietary Intel Memory Encryption Engine (MEE) to prevent divulging secrets stored in DRAM using certain hardware-based attacks, namely bus probing and cold-boot attacks. During execution, enclaves are still subject to conventional OS techniques like context switching and exceptions, e.g. interrupts.

When launching an SGX-enabled application, the Intel SGX run-time module requests the CPU to load SGX data from untrusted memory into EPC pages. This is performed using dedicated CPU instructions: `ECREATE` for creating a new Secure Enclave Control Structure (SECS) that stores the enclave meta-data; `EADD` to load code and data to new enclave pages; and `EEXTEND` for incrementally constructing the enclave measurement hash in 256-byte chunks. The initial application state is considered untrustworthy; it is urged that no secrets are statically stored (hard-coded) within the enclave code [103]. After the enclave code and data is loaded, the `EINIT` instruction is used to finalise the enclave measurement SHA-256 hash constructed through the `EEXTEND` instructions (also known as the `MRENCLAVE` value).

Enclave developers, or Independent Software Vendors (ISVs), must provide a certificate that includes the enclave's identity represented by a SHA-256 of its code, data and meta-data, such as author name and version number, which is produced during the build procedure (known as the `MRENCLAVE` value). This value is used when sealing data – akin to TPM sealing – such that only an enclave with the same `MRENCLAVE` value can unseal that data when the enclave is relaunched. Another identity, the signing identity or `MRSIGNER`, is the authority who signs the enclave prior to distribution. Either `MRSIGNER` or `MRENCLAVE` may be used when sealing data persistently, either to allow data to be (un-)sealed between enclaves with the same developer identity, or restricted only to a single enclave respectively.

Each SGX-enabled CPU supports two 128-bit keys stored in ROM: the root provisioning (RPK) and sealing key (RSK). Each RPK is created off-site at a Intel Key Generation Facility and retained for assuring that the chipset is a genuine Intel SGX CPU. The RSK is created randomly at manufacture-time and used for secure storage using the sealing abstraction. Intel SGX uses a set of specialist enclaves for accessing ROM-resident keys: the provisioning enclave (PE) and

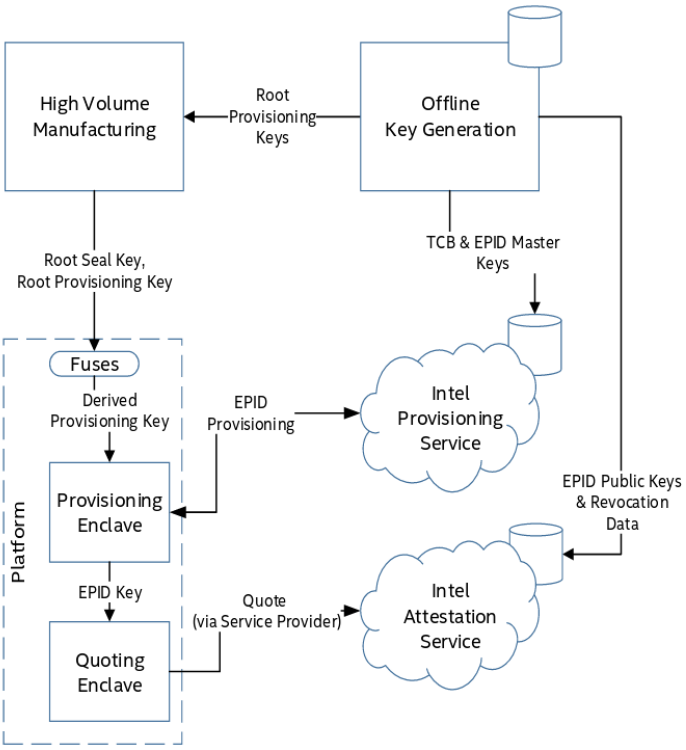


FIGURE 2.12: Intel SGX key infrastructure [64].

a quoting enclave (QE) used for remote attestation. All keys, except the RPK, require the RSK for derivation, which renders them unknown to Intel. The Intel SGX key infrastructure is shown in Figure 2.12.

SGX supports remote attestation based on the Enhanced Privacy ID (EPID) protocol – a DAA-based group signature scheme with revocation support. It allows the SGX enclave to demonstrate to a remote authority that it is utilising an authentic Intel SGX CPU. When SGX software is deployed, the PE is used to contact the Intel EPID Provisioning Server that 'joins' the CPU to the EPID group from its RPK. A separate EPID attestation key is then transmitted to the PE, which is used by the QE to sign quotes of the measured value of the target enclave during the remote attestation process. The EPID protocol allows a secure channel to be bootstrapped after the target enclave has been attested successfully. This is the intended method to provision secrets – keys, passwords and other credentials – into enclaves, rather than hard-coded secrets. The high-level attestation process is shown in Figure 2.13 and described message-by-message below.

In (1), the host application receives an attestation request from the remote service that its enclave must produce an attestation quote (2). Next, a procedure known as *local attestation* occurs between the quoting enclave (QE) – a special enclave that is launched specifically for with access to the EPID attestation key – and the enclave to be attested. Here, the target enclave requests a local attestation report from the QE in which the CPU computes an AES128-CMAC tag on the

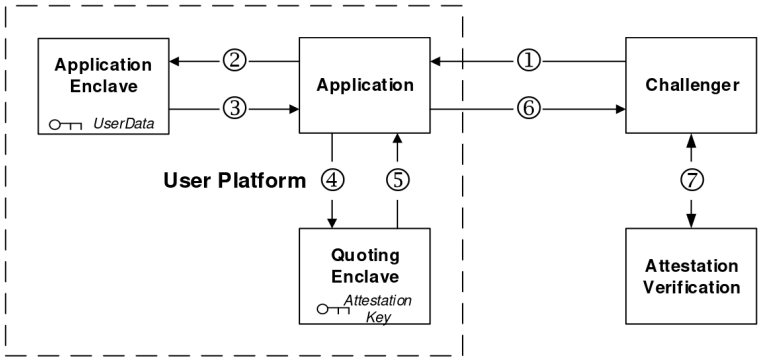


FIGURE 2.13: Intel SGX remote attestation process [104].

MRENCLAVE value using the enclave's report key (using EREPORT), which is verified and then replaced by a signature by the QE using the EPID attestation key. This produces the final *quote* that is returned to the host application (5) which, in turn, returns it to the challenger in (6). Lastly, the remote entity verifies the quote using the Intel Attestation Authority (see Figure 2.12), which possesses public portions of the provisioned EPID keys.

2.4.7 GlobalPlatform TEE

The GlobalPlatform (GP) TEE is a series of specifications for trusted execution environments, including TEE networking (GP TEE Sockets API); system architecture (GP System Architecture API); the Internal API for TEE applications, e.g. for cryptographic operations; debugging APIs; the Client API, for communication with applications in the untrusted world; and interoperability with secure elements [40], [41], [76]–[78], [105], [106]. The GP TEE divides execution into two distinct 'worlds': the untrusted world, comprising a standard OS, such as Windows or Linux, and the trusted world containing a TEE OS and one or more Trusted Applications (TAs).

TAs interact with the TEE OS using the GP Internal API, while untrusted client applications in the REE communicate with the TEE using the GP Client API, which transmits messages to the TEE kernel over a hardware-assisted secure monitor. The kernel, in turn, routes the messages to the target TA. The TEE kernel and TAs are considered trusted; it follows that any errors in the TA code, e.g. the API function definitions it exposes to client applications, or the TEE kernel could potentially compromise the TEE. The high-level architecture of the GP TEE is shown in Figure 2.14.

Hardware-wise, a GP TEE should have access to a secure clock, volatile and non-volatile memory, cryptographic accelerators and peripherals if applicable; the GP TEE should be able to utilise access resources without trusting the REE. The GlobalPlatform Internal API contains many of the primary abstractions

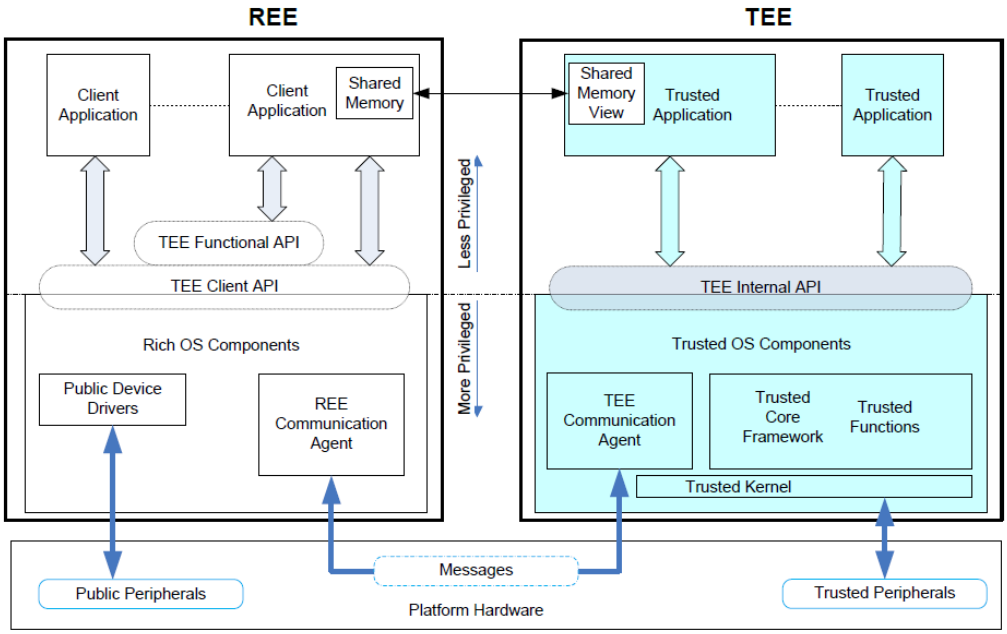


FIGURE 2.14: GlobalPlatform TEE system architecture overview.

described in previous sections, including *sealing* – similar to TPMs and Intel SGX – in which TAs can seal secrets to a TEE-specific key, and a host of general-purpose cryptographic primitives, such as AES, RSA, and elliptic curve cryptography. Notably, the GP TEE does not specify protocols for performing remote attestation.

It is important to note that the GP TEE is only a series of interface specifications; it does not contain details, for example, regarding the *implementation* of the REE and TEE partitioning. Generally speaking, ARM TrustZone (Section 2.4.4) is the predominant technology for instantiating a GP TEE on mobile and embedded systems. GlobalPlatform-compliant TEEs include OP-TEE[18], an open-source TEE developed by Linaro, and commercial closed-source solutions like Qualcomm's QSEE[19] and Trustonic's Kinibi[20] – all of which utilise TrustZone in practice.

The management of the GlobalPlatform TEE is defined in the TEE Management Framework [107] (TMF) specifications. The GP TEE utilises Security Domains (SDs) as the abstraction with which to manage TAs between multiple provisioning authorities. SDs are abstract isolated regions that manage a set of descendent TAs, which have no administrative capability themselves, using the GP Client API. Initially, a root SD (rSD) is provisioned by a trusted master authority, e.g. OEM, which is used to install and manage descendent SDs in a tree-based structure; these SDs may then install subsequent TAs themselves. It is possible for a TEE to contain multiple rSDs – belonging to the OEM, mobile operator and

[18]OP-TEE: https://www.op-tee.org/
[19]Qualcomm QSEE: https://www.qualcomm.com/snapdragon/security
[20]Trustonic Kinibi: https://www.trustonic.com/solutions/

government authority, for example – each comprising independent SD trees. A comprehensive set of management functions, including the installation, deletion and locking of TAs in order to prevent further updates and the configuration and personalisation of SDs, are found in the GP TMF specification [107].

The following section gives special attention is given to the GP TEE's protection profile and deployment model (as assumed in the CC PP), as it forms the security basis of the systems, protocols and architectures proposed in the forthcoming chapters of this thesis.

Protection Profile

The GlobalPlatform Protection Profile [40] is designed for a target of evaluation (TOE) based on the GP TEE. The TOE must implement the GP System Architecture, Internal and Client APIs; it comprises any hardware, firmware and software used to provide the TEE security functionality and secure usage of the TEE after delivery. The GP TEE PP does not consider the REE or any individual client or trusted applications. At present, the minimum assurance level for compliance is CC EAL2.

The profile is split into TEE Base, which covers *any* GP TEE, and the Time and Rollback modules that are implemented separately. The TEE Time and Rollback modules address the requirement for monotonic TA persistent time, integrity verification of TA code, configuration and trusted persistent storage. The TOE may require additional, non-TOE hardware to operate, such as other flash memory storage units, but the TOE must not rely on the correct behaviour of such units. The profile lists a series of assets aboard the TEE and the security properties that should be upheld; Table 2.2 lists these assets and their security requirements, aggregated from the TEE Base, Time, Rollback and Debug modules. The current version *"targets threats to the TEE assets that arise during the end-usage phase and can be achieved by software means...focuses on non destructive software attacks that can be easily widespread...and constitute a privileged vector for getting undue access to TEE assets without damaging the device itself"* [40]. This may, for example, include threats that attempt to recover keys for accessing corporate data or enforcing DRM video playback, which is implemented by a TA in the TEE. It does *not* protect against poorly-designed TAs containing improper use of cryptographic

algorithms and TA programming errors in general.

TABLE 2.2: Security requirements of GP TEE assets.

Asset	Property									
	C	**I**	**AU**	**U**	**UP**	**AT**	**DB**	**M**	**CO**	**IM**
TEE Identifier				✓						✓
RNG					✓					
TA Code			✓						✓	
TA Data and Keys	✓	✓	✓			✓	✓		✓	
TA Instance Time								✓		
TA Run-time Data	✓	✓							✓	
TA Persistent Data	✓	✓	✓				✓		✓	
TEE Firmware		✓	✓							
TEE Init. Code and Data		✓								
TEE Storage RoT	✓	✓								
TA Persistent Time								✓		
Rollback Detection Data		✓								
TEE Debug Auth. Key	✓	✓								

C: Confidentiality, **I**: Integrity, **AU**: Authenticity, **U**: Uniqueness, **UP**: Unpredictability, **AT**: Atomicity, **DB**: Device Binding, **M**: Monotonicity, **CO**: Consistency, **IM**: Immutability.

The reader is referred to [40] for a full list of all the security and organisational controls, delivery and deployment assumptions and requirements. Importantly, the PP defines two untrusted world adversaries that *any* GP TEE must protect against:

- **Basic remote attacker**: *"Performs the attack on a remotely-controlled device or alternatively makes a downloadable tool that is very convenient to end-users. The attacker retrieves details of the vulnerability identified in the identification phase and [...] makes a remote tool or malware and uses techniques such as phishing to have it downloaded and executed by a victim"* [40].

- **Basic (on-)device attacker**: *"Has physical access to the target device; it is the end-user or someone on his behalf. The attacker is able to retrieve exploit code, guidelines written on the internet on how to perform the attack, and downloads and uses tools to jailbreak/root/reflash the device in order to get privileged access to the REE allowing the execution of the exploit. The attacker may be a layman or have some level of expertise but the attacks do not require any specific equipment"* [40].

Additionally, the the GP TEE Exploitation Profile (Appendix A.2 of [40]) covers four capability levels of these adversaries (in increasing order):

1. **Remote** – an adversary without physical access who uses malware or phishing through which to conduct an attack on the TEE.

2. **Local layman** – a low capability adversary with physical access, who may jailbreak/root/reflash the device, and use standard equipment available through the Internet with information from an Internet resource, e.g. tutorial.

3. **Local proficient attacker** – same as (2) but with greater expertise and more time; assumes the use of standard equipment, such as oscilloscope, voltage supply and low-end visible light microscope.

4. **Local proficient attacker with equipment** – same as (3), but possesses specialist equipment, including chemical etching tools, UV-light microscopes, laser apparatus and micro-probe workstations.

Specifically, the GP TEE addresses 14 threats, which are described in Table 2.3. The current GP TEE CC requires resistance to attackers with `TEE-Low` attack potential. This is deduced from the Attack Quotation Grid using the sum of the adversary capability, elapsed time, and any organisational controls, including those that inhibit access to the TOE TEE's designs (VHDL/Verilog SoC designs) and source code, like requiring non-disclosure agreements (detailed further in Annex A.1 in [40]). While the security threats in Table 2.3 imply protection against hardware/physical attack vector, the TOE must only satisfy the `TEE-Low` criteria, which, in practice, does not address threat actors with significant expertise and bespoke equipment. Comparatively, the Eurosmart Security IC PP stipulates conformance to EAL4+ with protections against expert adversaries possessing specialist tools found in *'solid -state physics research and IC failure analysis'*. As such, while a TEE *may* protect against expert adversaries with bespoke equipment (reaching the security rating of `TEE-High` or beyond), the PP states that this *"is often beyond reach"* of most TEEs, nor is it it necessary for CC certification.

Deployment Model

GlobalPlatform provides a reference deployment model comprising six stages, which are listed as follows:

1. The *TEE hardware designer* is responsible for designing the IP cores for processors and other components used for hosting TEE software, maintaining inter-world hardware-enforced separation, and other hardware security resources used by the TEE, e.g. secure clocks and cryptographic co-processors.

TABLE 2.3: GlobalPlatform TEE CC PP threats.

CC Threat Code	Description
T.ABUSE_DEBUG	*An attacker is granted access to TEE Debug features, allowing them to obtain sensitive data or bypass, deactivate or change security services.*
T.TA_PERSISTENT_TIME_ROLLBACK	An attacker modifies TA persistent time; for instance, to extend expired rights.
T.ROLLBACK	*An attacker backs up part or all storage spaces and restores them later in order to use obsolete TA services or have the TA use obsolete data.*
T.RNG	Attacker exploits an RNG with insufficient entropy, thus comprising TEE-generated secrets derived from random numbers.
T.SPY	*Attacker who accesses TEE storage by means of run-time attacks, thus violating data confidentiality (this may also include bus probing, timing and power consumption side-channels).*
T.RAM	Partial or total recovery of confidential RAM content, potentially allowing the attacker to interfere with TEE code and data.
T.IMPERSONATION	*Impersonating another TA to cause another TA to reveal confidential data.*
T.STORAGE_CORRUPTION	An attacker corrupts the TEE's non-volatile storage in order to trigger unexpected behaviour from secure storage mechanisms.
T.TEE_FIRMWARE_DOWNGRADE	*An attacker backs up the TEE firmware and restores it later in order to use obsolete TEE functionalities.*
T.PERTURBATION	An attacker modifies the behaviour of the TEE or a TA in order to disclose or modify sensitive data and services.
T.ROGUE_CODE_EXECUTION	*An attacker imports malicious code into the TEE to disclose or modify sensitive data.*
T.CLONE	An attacker copies TEE-related data of a first device on a second device in order to make it accept the data as genuine.
T.ABUSE_FUNCT	*An attacker accesses TEE functionalities outside of their expected availability range, thus violating irreversible phases of the TEE life-cycle/state machine.*
T.FLASH_DUMP	The dumping of non-volatile flash memory.

2. The *TEE software developer* develops the TEE's software resources, including the TEE kernel and procurement of TAs. It also develops code used in initialising and instantiating the TEE, such as secure boot code, and any requisite drivers for interfacing with I/O peripherals in the trusted world. The TEE software developer is responsible for testing TEE software correctness and its compliance with the GlobalPlatform specifications.

3. The *silicon vendor* is responsible for producing the TEE chipset that integrates the hardware and software components from the previous steps. The silicon vendor may assume the role of hardware and software designer, or procure them externally. It produces the secure ROM code used as the root of trust, and incorporates any additional hardware, such as wireless (e.g. Ethernet and 802.11 WiFi), I/O (e.g. GPIO and SPI) and GPU support, into the final chipset.

4. *Device manufacturing* is the production of the final product into its housing, including installing the previously developed chipset from the silicon vendor. Here, the REE software, root security domain(s), and any additional TAs are provisioned.

5. *End-usage* involves any final quality assurance steps before delivering the product to the client, and the process of aftercare is prepared. This includes the use of a Trusted Service Manager (TSM), which may be incorporated into the device manufacturer or outsourced, for managing the TEE throughout its lifecycle. This includes remotely performing updates, factory resets, and uninstalling and installing TAs.

2.5 Feature Comparison

This section provides a comparison of all the above technologies using two generic types of adversary based on the level of access to a particular device. These are drawn from the aforementioned GlobalPlatform adversaries.

2.5.1 On-Device Adversary

On-device adversaries are malicious users that can potentially compromise the device. By doing so, depending upon the depth of compromise, they can control the execution of any sensitive applications and kernel space services. We divide the capabilities of this adversary further into three general categories:

1. *User-mode Adversary:* An adversary that develops a malicious application and has it installed on the target device or compromises an application currently executing in user-mode (Ring 3).

2. *Kernel-mode Adversary:* An adversary that comprises the kernel space (Ring 0) of a device through a vulnerability discovered in the OS and/or kernel, or another application currently executing with root-level privileges. The adversary is assumed to have itself gained root-level access over the system.

3. *Hardware Adversary:* An informed adversary with the ability to perform hardware-based shack attacks, such as bus probing, cold-boot and power analysis attacks using commercially-available equipment.

2.5.2 Off-Device Adversary

Off-device adversaries are malicious users that are located remotely and aim to manipulate the target's communication interfaces. The objective is to exfiltrate stored data or introduce malicious code to provide control the device to the remote adversary. This includes a standard Dolev-Yao adversary with the ability to intercept, read, modify and replay protocol messages across a network (e.g. during remote attestation), and with access to API specifications used for remotely managing the technology.

2.5.3 Criteria

In this section, the evaluation comparison criteria is introduced regarding the aforementioned secure and trusted execution technologies (STETs). These are described as follows:

1. *User Control:* The user may freely (un-)install arbitrary applications that use the chosen STET.

2. *Centralised Control:* The STET requires total or some co-operation with its issuer or maintainer, i.e. a centralised authority other than the end-user, in order to fully utilise it securely.

3. *Open Access:* Developers are freely able to access and utilise the STET's features in their own applications.

4. *Static Verification:* The STET has the ability to provide static integrity verification of the device platform and applications running on it.

5. *Continuous Verification:* The STET has the ability to monitor run-time states to provide continuous integrity verification and/or access control between applications.

6. *Tamper-Resistant Hardware:* The STET is packaged in tamper-resistant hardware.

7. *Secure Storage (Internal):* The STET contains internal secure storage for protecting data/applications, which never leaves the STET.

8. *Secure Storage (External):* The STET has access to external secure storage for protecting data/applications. This also includes encrypted data/applications stored on external storage, where the storage medium itself may be insecure.

9. *Isolated Execution:* The STET allows applications to be hosted and execution in complete isolation, at run-time and persistently, without interference from the untrusted device OS(s) and its applications.

10. *Software/Hardware Binding:* Individual applications and their data aboard the device are securely bound to the underlying hardware; applications and associated data cannot be easily transferred between platforms.

11. *Remote Attestation:* The trust technology can provide evidence to assure remote verifiers that it is functioning to expectations, including its platform and applications.

12. *Trusted Path:* The technology provides a trusted path such that I/O peripherals can communicate data to that technology without relying on the co-operation of untrusted entities, e.g. untrusted OS drivers.

13. *Protection Against On-Device Adversary 1:* The STET can protect against user-mode adversaries from an untrusted element, as described in the previous section.

14. *On-Device Adversary 2:* The STET defends against a kernel-mode adversary from an untrusted element.

15. *On-Device Adversary 3:* The STET defends against an informed adversary with the capability of performing hardware-based 'shack attacks' with commercially-available tools.

16. *Off-Device Adversary:* The STET defends against the off-device adversary described in the previous section.

Table 2.4 illustrates the degree to which each of the previous secure and trusted execution technologies compares against the above criteria. AMD Secure Processor and Samsung KNOX are ommitted from the comparison due to a lack of publicly-available information for a reliable analysis. The comparison also aggregates TPM-backed hypervisors – Intel TXT, AMD SVM, Flicker and TrustVisor – because of their shared security features, namely remote attestation, TPM-assisted secure storage and similar adversary protection. The differences in

TABLE 2.4: Feature comparison of Secure and Trusted Execution Technologies (STETs).

STET	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15	S16
Smart Card [45]	✗	✓	✗	✗	✗	✓	✓	✗	✓	✓	✗	✗	✓	✓	✓	✓
Secure Element [50]	✗	✓	▼	✓	✗	✓	▼	❋	✓	❋	✗	▼	✓	✓	✓	▼
Java Card [54]	✗	✓	▼	✓	✓	✓	✓	✗	✓	❋	✗	✗	✓	✓	✓	❋
TPM [108]	✓	✓	▼	✓	✗	✓	❋	✓	✗	✓	✓	✗	✓	✓	✓	❋
Intel TXT [80]	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✗	✓	✓	✓	❋
AMD SVM [82]	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✗	✓	✓	✓	❋
Flicker [83]	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✗	✓	✓	✓	❋
Trustvisor [84]	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✗	✓	✓	✓	❋
Microsoft Palladium [85]	✓	✓	–	✓	✓	–	✓	✓	✓	–	✓	✓	✓	✓	–	–
TI M-Shield [109]	❋	✓	✗	✓	✓	❋	❋	–	✓	–	✗	✓	✓	✓	–	–
Nokia ObC [101]	✓	✓	✓	✓	✓	❋	❋	✓	✓	✓	✓	▼	✓	✓	❋	✓
Intel SGX [110]	✓	✓	✓	✓	✓	❋	✗	✓	✓	✓	✓	✗	✓	✓	❋	✓
GP TEE with ARM TZ [76]	❋	✓	▼	▼	✓	❋	▼	✓	✓	✓	✗	✓	✓	✓	❋	▼
HCE [111]	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	❋	✗	✗

✓ : Supports. ✗ : No support. ❋ : Limited support or with significant caveats. ▼ : Implementation dependent.
– : Insufficient information available.

these technologies are largely functional in terms of performance and the target architecture.

2.6 Discussion

In this chapter, a range of hardware-assisted security technologies were introduced for establishing platform trust, focussing principally on embedded and mobile platforms. One immediate observation is that no single technology satisfies all security criteria. Some technologies, such as Intel SGX and the GP TEE, provide strong security guarantees of isolated application execution against kernel-level adversaries. However, these forgo robust resistance against hardware-based attacks, particularly in the case of Intel SGX, while the GP TEE is significantly implementation-dependent. Comparatively, technologies built to resist strong hardware-based adversaries – Secure Elements and Java Card, as used for payments, access control for health care, and national ID schemes – forgo the flexibility of other technologies. This includes limited internal storage (and, indeed, general computing processing performance) and requiring stringent co-operation with the technology's vendors for acquiring access.

A further observation is the fragmentation of the TEE landscape. SEs, Java Card and TPMs have converged towards a set of widely understood, implemented and maintained specifications, developed by GlobalPlatform, Oracle and TCG respectively. In comparison, TEEs primarily remain a disparate set of technologies and, while most share a set of similar features, such as secure storage and isolated execution, they are predominantly incompatible across platform architectures. Despite standardisation efforts by GlobalPlatform (Section 2.4.7), some widely-deployed technologies, such as Intel SGX, do not conform.

Beyond security, the requirement for additional hardware and auxiliary software, e.g. for drivers and communication protocols, is a disincentive to a manufacturer looking to create secure platforms without significantly increasing the size of the device's physical PCB and hardware and software TCBs. STETs integrated into the platform's SoC or chipset are desirable from an architectural perspective for embedded systems. At present, the GP TEE upon ARM TrustZone is a highly attractive solution with respect to deployability, availability, performance and security features offered. This notably includes trusted I/O and protection against the strongest of software-based adversaries. However, the absence of a standardised remote attestation mechanism is still a major shortcoming, as is the lack of current open-access implementations at the time of writing.

Lastly, attention is drawn to the shortcomings of the adversaries listed in Sections 2.5.1 and 2.5.2. In particular, we do not consider attacks requiring significant access to expertise, specialist non-commercial equipment, time and

capital, confined typically to state-level and similar actors. Related attacks also include the influencing of the supply chain and the ability to perform insider attackers upon the technology itself.

Furthermore, we are also aware of the emerging class of threats involving *microarchitectural attacks*, such as Spectre [112], Meltdown [113] and Foreshadow [114] that exploit the side-effects of optimisation techniques used by modern CPUs, namely speculative and out-of-order (OoO) execution. Briefly, Spectre [112] exploits a side-effect of speculative execution, where the CPU performs certain tasks in advance, e.g. branch prediction, in parallel without knowing for certain whether the results will be used by the program under execution. The authors demonstrate how a victim program's execution can be induced to speculatively execute sequences that should not have occurred under correct program execution in order to leak unauthorised information from the victim program's memory address space using a cache covert channel. Meltdown [113], meanwhile, exploits OoO on certain processor models to leak kernel memory from user space; OoO is an execution technique in which subsequent program instructions are executed in parallel with the preceding instructions in order to maximise CPU utilisation. At a high level, Meltdown exploits a race condition where unauthorised memory accesses are available to OoO before the processor issues a fault and reverts speculatively executed operations, which can be leaked via a cache covert channel. Foreshadow, presented by Van Bulck et al. [114], exploits a speculative execution bug that enables the execution of an extended Meltdown attack to recover Intel SGX enclave secrets, including the keys of SGX's quoting enclave (see Section 2.4.6). After extraction, the keys can then be used to forge local and remote attestation responses. While Foreshadow was remedied by Intel in August 2018 [115], we note that, due to their recent emergence, the reader is urged to be mindful of the continually evolving landscape of microarchitectural attacks on STETs.

The coming chapters will introduce, propose and evaluate solutions for a number of existing domain challenges where TEEs could benefit constrained sensing platforms. It will be examined how the application of TEEs can enhance the security and trust assurances of such platforms while retaining performance and deployability.

Chapter 3

Towards Trusted Execution of Multi-Modal Continuous Authentication Schemes

3.1 Introduction

The emergence of powerful, sensor-rich devices has prompted the development of *continuous authentication* (CA) schemes on commodity hardware, where user behaviour is compared to past experience to produce an authentication decision. CA schemes aim to address many of the existing challenges with traditional authentication schemes, such as tokens and conventional biometrics (elaborated upon in Section 3.2.1). Current proposals, however, have largely neglected system-level adversaries and offer little in the way of assurances to a remote authority that CA decisions can be trusted. This has particular importance if a third-party controls access to assets based on CA decisions from a potentially compromised device. This could be, for example, an employee's phone whose decisions are used to control access to assets belonging to their employer based on a CA-based authentication decision. A software compromise, either on the platform or scheme implementation, may enable an adversary to modify authentication scores to gain unauthorised access to restricted assets, conduct a Denial of Service (DoS) by denying legitimate accesses, or potentially gain insights into user behavioural patterns. This could be, for instance, if the host application contains a vulnerability in which the CA scheme resides. Worse, this could be by exploiting an OS-level vulnerability through malicious software (malware) delivered, say, in a phishing email.

In this chapter, we investigate the use of secure and trusted execution technologies to preserve the security and trust of CA schemes, even when an adversary has root-level privileges, while retaining deployability. After an analysis of these technologies, we present the first forays into evaluating TEE-based CA, which is implemented on a Intel-based laptop and ARM-based development board using Intel SGX and the GlobalPlatform TEE respectively. Our proposal provides

additional confidentiality, integrity and trust assurances over existing untrusted world implementations. A comparative performance evaluation is presented of Intel SGX and GlobalPlatform TEE versus non-TEE performance using a test-bed of algorithms proposed in related work. The results indicate that trusted CA can be performed in an efficient fashion while reducing the TCB relative to an untrusted world implementation.

3.1.1 Motivation

Modern mobile devices are often used to store sensitive user data, such as contact, financial, corporate email and social media information. The physical nature of mobile devices, however, introduces a range of attack vectors typically avoided with desktop workstations. Small form-factor devices can be misplaced, dropped or stolen, such as in restaurants and on public transport. Despite this, previous studies by Harbach et al. [116] and Micallef et al. [117] indicate that in the region of 25% of smartphone users do not use a secret-based locking mechanism, such as a PIN or fingerprint, depending on demographic. Moreover, Hayashi et al. [118] show that the all-or-nothing nature of mobile authentication, where access to *all* device functionality is granted/denied upon the completion/failure of a secret, is a *"remarkably poor fit"* with users' preferences. Instead, users prefer explicitly unlocking a device for only certain applications – those holding sensitive data, such as banking – while avoiding explicit authentication for benign ones, like navigation. Similarly, in separate work, Hayashi et al. [119] showed that users favour setting different authentication challenges in locations of varying perceived safety; that is, reducing the strength or frequency of authentication challenges in perceived safe locations, like the home, while maximising strength in previously unvisited areas.

These factors have inspired the exploration of *continuous authentication*[1], where the device's authentication state is influenced by environmental and contextual data. This includes measurements from mobile sensors, such as ambient light, temperature, humidity and GPS location, as well as application usage, nearby Bluetooth MAC addresses and WiFi access points (APs). However, while numerous schemes have been proposed in the literature, little attention has been paid towards secure and trustworthy on-device execution of CA schemes in practice. If any element is modified – the sensor data, underlying user model or the decision itself – access could be granted to sensitive assets and services maliciously.

[1] Also known as 'implicit', 'on-going' and 'transparent' authentication in related literature.

3.1.2 Contributions and Chapter Structure

For the first time, we explore Secure Elements (SEs) and Trusted Execution Environments (TEEs) in order to provide stronger confidentiality, integrity and trust assurances regarding the execution of CA schemes. After introducing CA and examining recently proposed schemes (Section 3.2), the potential applicability of SEs and TEEs in addressing our extended threat model is discussed in Section 3.3. Next, the implementation of test-beds for evaluating TEE-based CA using Intel SGX and the GlobalPlatform TEE with ARM TrustZone is described in Section 3.4. The test-beds are evaluated with a selection of machine learning algorithms proposed in past literature using a publicly-available dataset from real-world users (Section 3.5). Finally, the conclusions of our results are discussed, as well as identifying potential avenues for future research (Sections 3.6). This chapter provides the following contributions:

- The design and implementation of a test-bed for performing trusted execution of multi-modal CA using Intel SGX and the GlobalPlatform TEE with ARM TrustZone. To our knowledge, this is the first investigation into assessing TEE-based CA.
- An analysis of TEEs and SEs for executing continuous authentication schemes, along with their associated benefits and drawbacks relating to security, performance and deployability. This also includes a high-level threat analysis relevant to CA schemes generally.
- Performance results regarding two implementations of TEE-based CA. The results show that TEE-based CA can be conducted with a projected maximum overhead of 404ms (training) and 2.0ms (testing), versus an implementation executing in the untrusted world.

The work presented in this chapter is an extended version of work from the following publication:

- C. Shepherd, R. N. Akram, and K. Markantonakis. "Towards Trusted Execution of Multi-Modal Continuous Authentication Schemes," in *Proceedings of the 32nd ACM Symposium on Applied Computing*, ser. ACM SAC '17, ACM, 2017, pp. 1444-1451.

3.2 Related Work

This section presents a more detailed examination of CA and its motivations (Section 3.2.1), and trusted biometric schemes in recent literature (Section 3.2.2).

3.2.1 Continuous Authentication (CA)

User authentication relies upon what one *knows*, e.g. passcodes and PINs; *has*, e.g. tokens and smart cards; or *is*, e.g. behavioural and physical biometrics. Password and pattern schemes, however, suffer from well-studied issues relating to memorability and shoulder-surfing attacks [120]. Token-based methods are generally seen to be more secure, but are liable to theft and considered burdensome when managing multiple credentials across numerous providers [121]. Biometrics comprise physiological and behavioural measures, and recent work [118] shows that users are receptive to these due to their convenience. However, physiological systems such as fingerprint recognition, suffer from inherent reliability issues from ageing, injury and the environment [121]. Physiological biometrics also have inherent difficulties with 'revoking' or 'replacing' physical characteristics once divulged. With behavioural biometrics, like gait authentication, achieving acceptable error rates remains an open problem [121]. CA aims to address the shortcomings of traditional schemes by transparently monitoring the user's environment and context via device sensors. Various schemes have been proposed that rely upon keystroke- [122], motion- [123], environmental- [124] and touch-based [125] features. Recent research – discussed in the following subsection – has tended to focus upon a *multi-modal* approach, where multiple data sources are used to authenticate users in order to reduce error rates. Generally, CA aims to address one or more of the following:

- Reducing the number of unnecessary login attempts through modulating the lock-screen strength [119], [126].
- A primary or second line of defence for detecting unauthorised users who have bypassed the initial lock-screen, e.g. via shoulder-surfing [127]–[129].
- Adjusting access control policies based on context [130].

A typical approach involves training a supervised learning classifier using labelled sensor data, and classifying subsequent data according to these labels. Classes may correspond to locations, such as 'safe' and 'unsafe' locations as per [130], or the accept/reject status directly [126], [128], [129], [131]–[134]. Other work does not focus on classifying data, but produces a probability value that is subsequently thresholded at rates decided by the operator [127], [135].

CA schemes usually follow two phases (Figure 3.1):

1. **Enrollment/training**: sensor data is collected over a period of N days, often with user prompts to ascertain labelled data, and training a classifier to model user behaviour with respect to those labels.

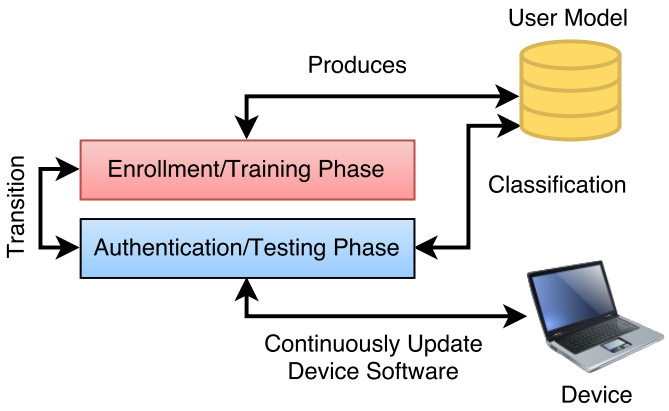


FIGURE 3.1: High-level state flow of CA Schemes.

2. **Authentication/testing**: occurs after the model is generated. Sensor data is collected over a time window – hours, minutes, or per sample – that is inputted to the classifier to produce the corresponding class indicating the authentication state.

In both stages, data is collected at predetermined intervals of M samples at a rate of T milliseconds. Data is classified directly or after a feature extraction stage, as described in Section 3.3.1. A classification model may be retrained after certain intervals; that is, transitioning periodically between the authentication and enrollment stages in order to address gradual natural shifts in user behaviour [119], [128], [135]. We summarise the key attributes of related schemes developed for mobile platforms in Table 3.1. Many existing proposals use modalities beyond the capabilities of modern mobile platforms, such as ECG, body temperature and perspiration measurements from wearable monitors [138]–[140]. Rather, this work concentrates on modalities collected from a single conventional mobile device.

3.2.2 Secure and Trusted Execution of Biometrics

Trusted execution of biometrics is a well-understood domain. This section briefly examines measures on consumer devices and those proposed in academic literature for securing the confidentiality, integrity and availability of biometric measurements and templates.

Consumer Devices

At present, fingerprint authentication is the most widely-deployed biometric authentication mechanism on consumer mobile devices [121]. Critically, fingerprint matching algorithms must be executed securely such that the user's fingerprint images enrolled onto the device – also known as *templates* – cannot easily be read

TABLE 3.1: Multi-modal CA schemes in related work.

Proposal	Method(s)	Modalities
Hayashi et al. [119]	NB	GPS†
Micallef et al. [117]	DT	Acc., WiFi, Light, Sound, Mag.
Shi et al. [129]	NB	GPS, Cell ID, Acc., Sound, Touch
Miettenen et al. [130]	RF, kNN, NB	GPS, WiFi
Riva et al. [126]	SVM	Light, Temp., Hum., BT, Touch, Logins
Li et al. [128]	SVM	Touch, Acc., Ori., Mag.
Verlinde & Chollet [136]	kNN, LR, NN, DT	Sound, Face + Profile Images
Bo et al. [137]	SVM	Touch, Acc., Ori, Gyro.
Saevanee et al. [131]	NN	GPS, Text Messages, WS, Keystrokes
Ketabdar et al. [132]	NN	Acc., Sound
Wang et al. [134]	SVM, RF	Touch Size, Pressure, Swipe Dynamics
Fridman et al. [133]	NB, SVM	Keystrokes, CC, WS

† Only GPS evaluated, but the scheme is generalisable to N inputs.
* Acc.: Accelerometer; BT: Bluetooth; DT: Decision Tree; Hum.: Humidity; Gyro.: Gyroscope; Mag.: Magnetometer; NB: Naive Bayes; LR: Logistic Regression; NN: Neural Network; Ori.: Orientation; RF: Random Forest; SVM: Support Vector Machine; Temp.: Temperature; CC.: Cursor Coordinates; WS.: Writing Style (Stylometry).

by untrusted system elements, including the host OS. The lack of a revocation and replacement mechanism for physical biometrics, as alluded to in Section 3.2.1, means the disclosure of a fingerprint could affect a multitude of other services used by the same user.

Towards addressing this on consumer devices, the Android architecture is designed to allow only the vendor's TEE OS to access fingerprint hardware directly. Its architecture, shown in Figure 3.2, is intended to prevent arbitrary accesses to raw fingerprint images from any part of the untrusted world, including user-installed applications and kernel-mode services. The Android SDK [141] describes the following components:

- The `FingerprintManager` API is exposed to Android application developers for conducting fingerprint authentication. The API (version 23) exposes only three public functions: 1), `authenticate()` initiates the fingerprint reader, displays a system dialog box indicating the reader's activation, and triggers a callback if the authentication attempt fails or succeeds; 2), `hasEnrolledFingerprints()` returns a boolean value for whether

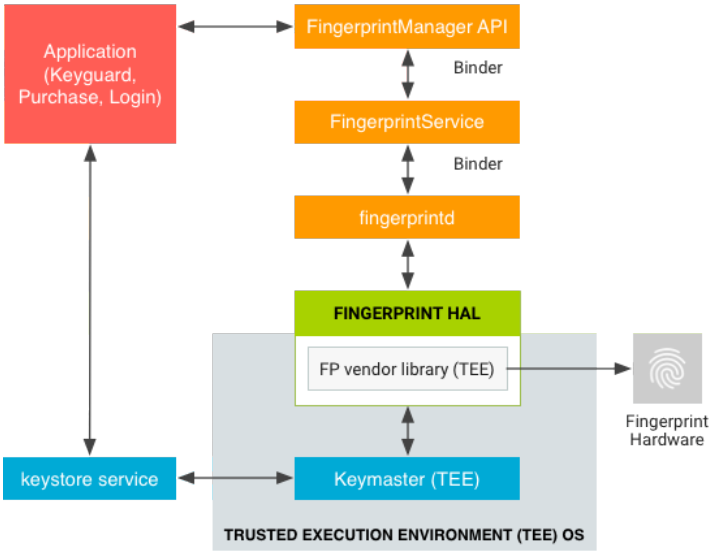


FIGURE 3.2: Fingerprint authentication system architecture for Android platforms [141].

the user has any enrolled fingerprints; and 3), `isHardwareDetected()` returns whether fingerprint hardware is present and functional. Each application contains an instance of `FingerprintManager`, which communicates with `FingerprintService`.

- `FingerprintService` is a singleton service that operates in a system process and handles communication with `fingerprintd`.
- `fingerprintd` is a C/C++ implementation of the binder interface from `FingerprintService`. The `fingerprintd` daemon executes in its own process for communicating with the Fingerprint Hardware Abstraction Layer (HAL) that abstracts vendor-specific code.
- The `Keystore` API and `Keymaster` are key management services for storing, managing and protecting keys in the device's TEE.

Android stipulates that *"raw images and processed fingerprint features must not be passed in untrusted memory. All such biometric data needs to be secured within sensor hardware or trusted memory"* and that, notably, *"Rooting*[2] *must not compromise*

[2]'Rooting' is the process by which a user gains administrative (superuser) access permissions in a UNIX-type environment. For Android, this allows users to modify existing system applications and preferences, and execute third-party applications with root-level privileges whose functionality would otherwise be sandboxed by its `uid` and `gid`. On Apple iOS platforms, 'jailbreaking' is the process of exploiting a known privilege escalation vulnerability to break out of chroot 'jail' that restricts application access to parts of the filesystem, namely reads/writes to the device's OS partition and execution of the data partition. This is used to acquire root access, which is further utilised to disable signature verification checks in iOS that enforce the installation and launching of only Apple-signed system components and applications. This subsequently allows the user to install unauthorised, unsigned system components and applications on the device.

biometric data" [141]. The aforementioned `FingerprintManager` API exposes a binary authentication result to application developers, i.e. success or failure. Raw fingerprint data is not accessible from outside the TEE; indeed, the fingerprint's SPI interface *"must be accessible only to the TEE"*, and that fingerprint image acquisition, enrollment and matching must also occur within the TEE [141]. Additionally, only encrypted forms of the fingerprint data can be stored on the (untrusted) file system, and that *"templates must be signed with a private, device-specific key"* using AES such that encrypted templates cannot be transferred to another device and reused [141].

On Apple platforms, fingerprint authentication is performed using a proprietary security processor – isolated from the primary application processor – known as a 'Secure Enclave' from the Apple A7 SoC used in Apple iPhone models from Q3 2013 [142]. Few publicly-available details exist regarding the Secure Enclave[3], but the Apple iOS Security documentation [142] states that it hosts a microkernel capable of performing fingerprint and facial authentication, payment tokenisation, and managing a keystore of application-generated keys. It also states that *"data saved to the file system by the Secure Enclave is encrypted with a key entangled with the UID[4]."* More generally, NXP, STMicroelectronics and Infineon produce embedded Secure Elements (eSEs) marketed towards fingerprint authentication; the state-of-the-art Infineon SLE97 eSE[5] offers a 32-bit CPU at 300MHz with 32 kB RAM and 1MB flash storage; hardware-accelerated AES, 3DES, RSA and ECC; and is certified to CC EAL5+.

Academic Literature

While the trustworthy execution of CA has not been addressed in the literature to the best of our knowledge, work *has* been conducted in protecting sensor values and securely and privately computing authentication decisions remotely. The latter is based principally on the application of garbled circuits and homomorphic encryption. A review of related work and their respective contributions is provided below.

Liu et al. [143] propose an architecture offering confidential, integral and trusted sensor values using Credo – a TPM-backed hypervisor for Intel TXT – for x86 platforms, and ARM TrustZone. In this work, a trusted GPS sensor is implemented that signs resulting measurements within the TEE over a trusted

[3]The absence of publicly-available documentation has precluded a detailed security discussion regarding the capabilities of Apple's Secure Enclave in this thesis.

[4]The Unique ID (UID) is a device-specific value that is generated by the Secure Enclave on first launch and unknown to Apple Inc.

[5]Infineon SLE97 eSE: https://www.infineon.com/cms/en/product/security-smart-card-solutions/security-controllers/sle-97/

path, and provides sealing of values – protecting a given secret by binding it to a policy based on sensor readings – and remote attestation.

Safa et al. [144] propose a scheme for computing authentication decisions remotely from encrypted feature vectors using homomorphic encryption. Here, an encrypted behavioural model is stored on a remote server and a decision is computed from incoming feature vectors (also encrypted) without revealing either to a potentially intrusive server. Authentication is performed by computing the dissimilarity between incoming features and the stored user profile; the user is authenticated if the dissimilarity is within an acceptable threshold set by the remote server. A security proof is provided; however, the scheme is neither implemented nor evaluated. This method of cloud-based authentication also requires constant online connectivity between the device and remote server; the authors also note that the scheme does not cover scenarios in which the user's device is compromised, which we aim to tackle in this work.

Domingo-Ferrer et al. [145] develop the work by Safa et al. [144] and present an approach based on the set intersection of homomorphically encrypted feature vectors to authenticate users. The work operates similar to [144], but adds set intersection to compute the dissimilarity function between the encrypted user profile and incoming feature vectors to support categorical as well as numerical features ([144], meanwhile, supports only numerical features). The authors provide a performance evaluation of the proposal in which authentication takes 0.08–31.2 seconds depending on the number of input features (1–50 input features respectively). Like [144], however, constant internet connectivity is required and on-device adversaries are also not considered.

Sedenka et al. [146] propose protocols for secure outsourcing of biometrics using garbled circuits and homomorphic encryption, which enables remote computation of decisions under the honest-but-curious server model. A security analysis is provided and the schemes are evaluated using a consumer laptop and smartphone. The results suggest that outsourcing is possible, but not without a communication and time penalty, ranging from 4–174MB and 0.85s–45.9s based on the submitted data size. As with [144] and [145], our primary concern is security issues relating to *on-device execution*, rather than outsourcing computation to a remote party.

3.2.3 Discussion

The question of executing a CA scheme in a *secure* and *trustworthy* fashion, on the device itself, has not been addressed directly in related work. While attention has duly been given to scheme accuracy and outsourcing decisions in a secure and privacy-preserving way, it has yet to be directed to the extended threats faced 'in the wild' from on-device system adversaries. This may be, for example,

through the exploitation of a vulnerable mobile application in which the CA scheme resides, and also OS-level adversaries, whether as a result of a malicious application with root access (e.g. on an already-rooted device), or the exploitation of an OS-level vulnerability. The threat model is described further in Section 3.3.2.

Because of such threats, we note that a CA scheme operator, such as a corporation altering access to remote resources based on context, is unlikely to trust authentication decisions produced from users' devices for high-value assets. The information stored and used by such a scheme during execution may provide privacy-intrusive insights into users' behavioural patterns that are represented by a history of GPS coordinates, ambient sound, current application usage, and other measurements used by a particular CA scheme. Secure and trusted CA is, hence, necessary to protect against such adversaries, while remaining mindful of deployment costs. Additionally, the high volume of sensor data and the maintenance of a behavioural model provide unique challenges over traditional biometric schemes like fingerprints (discussed further in Section 3.3.3).

3.3 CA Security Analysis

Next, we introduce the high-level components of a generic CA scheme in order to ground the subsequent discussion on the threat model and attack surfaces that we consider. After this, TEEs and SEs are explored as potential candidates towards addressing these threats.

3.3.1 High-level CA Components

The high-level procedure for acquiring raw measurements collected from sensing hardware, constructing and managing a CA behavioural model, and generating the authentication decision can be modelled in six stages:

1. **Sensor data acquisition**: Raw measurements are acquired from the relevant sensor I/O devices. This may include a vector of real values from a tri-axial accelerometer (representing x-, y- and z-axis values in three-dimensional space), latitude and longitude GPS co-ordinates, key presses, two-dimensional cursor co-ordinates, or even a waveform of desired length recorded from a microphone. The sampling rate at which data is polled from sensing hardware is scheme-dependent.

2. **Feature extraction**: Raw measurements are transformed into more meaningful or efficient representations for more effective classification. This may comprise standard dimensionality reduction algorithms, such as principal component analysis (PCA); noise reduction; normalisation; or computing

simple statistical metrics, such as the arithmetic mean or standard deviation of sensor measurements.

3. **Classification**: In past work, authentication is often modelled as a binary classification problem, i.e. 'authenticated' and 'not authenticated' [128], [129], [131]–[133], [135]–[137]. During the enrollment phase (see Figure 3.1), the classifier is trained from features collected after a period of monitoring the user's regular behaviour, which serve as the authenticated examples, while unauthenticated examples are sourced from artificially perturbed values or data samples sourced from other users.

4. **User Model**: During enrollment, the saved classifier parameters act as the behavioural model. This may include, for example, learned network weights and node activation values for a neural network, or feature weights for logistic regression. In the authentication phase, the parameters are restored from file and used to classify future segments of sensor data.

5. **Decision/post-processing**: During the authentication stage, a segment of sensor data, e.g. using a sliding window of a given time-frame, is used as input to the classifier that outputs the corresponding authentication state.

6. **Update device status**: The classifier label may be transmitted to a remote authority over a secure channel or returned to a security monitor that modifies the system state to reflect the authentication state.

3.3.2 Threat Model

In general, current CA schemes aim generally to protect against the following: **a)** a primary or second-line of defence against a device thief who bypasses the initial lock-screen through shoulder-surfing, successful guessing, or where no mechanism is present, with the aim of accessing sensitive data; and **b)** a curious adversary, e.g. a co-worker or friend, who accesses the device to browse private data. In our work, the threat model is expanded to incorporate adversaries present in a practical deployment, and we specifically consider software threats that aim to influence or subvert the operation of the scheme on the device.

An on-device CA scheme and its constituent components – described in Section 3.3.1 – would typically execute as a part of a user-mode application (ring 3) or the host operating system (ring 0). We aim to address a strong adversary operating at either of these protection levels; that is, at worst, a root-level adversary with extensive control over the OS, including the CA code itself and run-time and persistent states accessible to the OS. The threats under consideration are now summarised:

- Observing run-time states, such as memory contents, including raw sensor data polled from hardware, the extracted features, classification decisions, and the user model under execution.
- Observing data structures contained within the device's persistent storage, such as the trained user model stored at rest on the device's filesystem.
- Modifying sensor measurements after they are polled from hardware, the extracted features, classification decisions, and the user model both at rest and under execution.
- Disrupting or modifiying the intended CA scheme execution flow; for example, bypassing feature extraction in order to give rise to denial of service conditions.

In the above cases, the intentions of the attacker include learning the behaviour of the device user, thus violating user privacy, and generating inaccurate authentication states in order to access restricted device assets or deny legitimate accesses to the device. To be clear, we do not consider physical attacks on the sensing hardware itself to be in scope; for example, replacing sensors with ones that output adversely inaccurate measurements, or removing them completely to give rise to denial of service conditions. Moreover, defending against contextual adversaries that engineer the surrounding environmental conditions to influence classification decisions (see Shretha et al. [147]) is also beyond the scope of this work. However, we *do* place trust in the security assurances provided by a TEE or SE, which were discussed in greater detail in Chapter 2. Related to this, we consider threats outside the remit of TEEs and SEs to be out-of-scope in this work; for example, developer-induced TEE and SE programming errors, supply chain attacks and, for TEEs, the use of fault injections and other related attacks requiring significant expertise, as discussed in Sections 2.5.1 and 2.6 of Chapter 2. The reader is also referred back to Section 2.4.7 and [40] for a description of protection scope of the GlobalPlatform TEE specifically; Section 2.4.6 and, in particular, Costan and Devadas [110] for Intel SGX; and Section 2.2.3 for SEs.

3.3.3 TEEs and SEs as Secure CA Candidates

As per Chapter 2, secure execution is the ability to maintain confidentiality and integrity of run-time states during code execution, while trusted execution typically includes extensions for attesting to the state of executed code. SEs and TEEs are two high-level constructs for preserving the security of CA schemes at run-time, which would be implemented as a trusted application (TA) and provisioned into the TEE or SE before deployment. This section contrasts these

two candidates in order to protect the sensitive assets of a CA scheme – authentication decisions, feature extraction, user model and sensor data – and the code under execution. The focus on these standardised constructs minimises potential deployment overhead commercially.

The principle behind separating a CA scheme from the Rich OS is to minimise the Trusted Computing Base (TCB) – the set of hardware, software and firmware components critical to maintaining the security of a system. Conventional OSs, such as Windows and Android, are generally too large in size to formally verify their security properties; TEEs and SEs, meanwhile, are significantly smaller and more suitable for such analysis [148]. By relocating CA schemes to a TEE or SE, the TCB is reduced largely to that of the TEE/SE and the CA scheme's application logic, thus reducing the attack surface significantly. A TEE or SE could be used to address the threat model by protecting the code under execution, including run-time states, and protecting the internal behavioural model from untrusted world adversaries. Using a TEE, the behaviour model can be protected by using its secure storage after use, i.e. at power-down, to encrypt it on the untrusted world filesystem using a TEE-resident storage key, and subsequently restoring it on relaunch, e.g. upon booting the device. A TEE could also be used to host key material for signing authentication decisions, using ECDSA or RSA, before transmission to a remote authority.

The potential performance benefits of TEEs, i.e. its hardware sharing with a REE, make it attractive in pursuit of our design goals, namely for computationally-intensive applications like those requiring machine learning. Our initial investigations using the GCU dataset [135] found that, on average, sensor data surpassed 0.5MB per day per user – far exceeding the memory capacity of many commercial embedded SEs at the time of writing when over 3 days of data is used, as suggested in related work [119], [127], [128], [149], [150]. This could be a deciding factor when considering that other data, such as payment and transportation credentials and applications, must reside simultaneously on one SE.

TEEs are also supported directly by underlying device CPU/SoC, RAM and storage, thus offering greater speed and capacity over SEs. Moreover, many TEEs, such as Intel SGX, also offer native remote attestation, where third-parties may attest to the integrity of the enclave application. This property allows remote services reliant on authentication decisions to verify the state of the platform (post-boot) or applications themselves. As stated in Chapter 2, a TEE's precise TCB depends primarily on the chosen platform, but for widespread TEEs, such as TrustZone and SGX, the hardware TCB generally comprises a trusted CPU chipset or SoC. The software TCB of a TEE comprises the trusted world containing a trusted OS, firmware and applications (for GlobalPlatform and TrustZone), or enclave logic for SGX [110].

TABLE 3.2: Summary of CA-relevant features.

Feature	REE	SE	TEE
Isolated Execution	✗	✓	✓
Processing Performance	Best	Poor	Good^{+}
Memory Capacity	Great	Limited	Great^{+}
HW Tamper-Resistance	✗	✓	†
Remote Attestation	✗	✗	◊
Secure I/O	✗	✓	✓^
Hardware TCB	All	SE MCU	CPU/SoC*
Software TCB	Large	Smallest	Small

◊ TEE-dependent. † Limited protection.
* Applicable for TrustZone and SGX. ^ Applicable for TrustZone.
$^{+}$ Applicable for SGX and TrustZone-A. We note that TrustZone-M, marketed for microcontrollers, is significantly more limited.

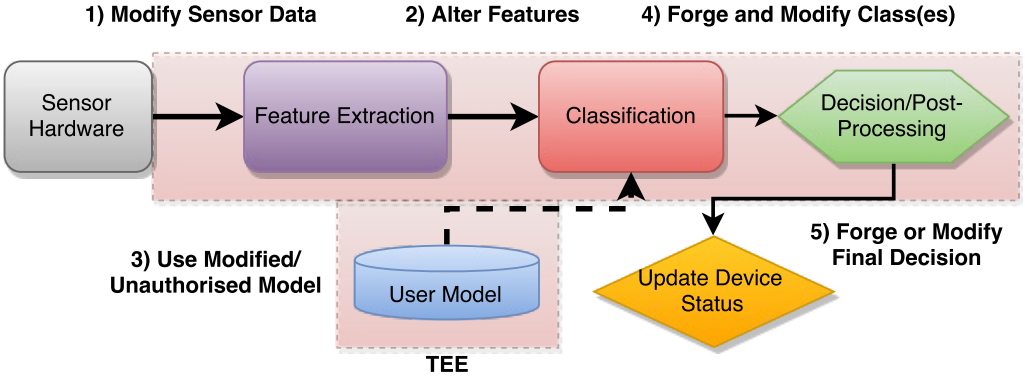


FIGURE 3.3: Example CA attack vectors; this chapter proposes using a TEE to protect those areas in red.

While SEs are significantly less powerful than TEEs, their primary benefit is strong hardware tamper-resistance, thus making it an attractive solution for key and other small-sized auxiliary data storage. Note that TEEs and SEs are not mutually exclusive; it is possible for a TEE to rely upon an SE, or indeed a TPM, for certain operations such as cryptographic operations and key storage, as noted in GlobalPlatform guidelines [40]. We summarise the key properties of each architecture in Table 3.2, and compare this to a regular application running in user-mode in the Rich OS (implicit in existing proposals).

3.4 Test-bed Implementations

In light of the above discussion, we describe our test-bed implementations for evaluating TEE-based CA. Our system uses the Intel SGX and OP-TEE along with implementations of three learning algorithms for performing trusted CA on

ARM and Intel chipsets. We present the details and challenges of implementing this on two commercially-available devices.

3.4.1 Intel SGX

Intel SGX was described previously in greater detail in Section 2.4.6 of Chapter 2. The first thing we note is that Intel SGX supports only a subset of C/C++ libraries. We discovered, for example, that libraries found ubiquitously elsewhere, e.g. `<locale>`, `<sstream>` and `<iostream>`, were either partially or wholly unavailable. This is because, in present versions, SGX enclaves cannot use methods reliant upon OS systems calls. Consequently, basic functions reliant on I/O, such as `printf`, had to be redefined – printing from enclaves requires a context switch to the Rich OS through an `OCALL`, where a system call is used to redirect characters to standard output. Intra-enclave thread creation is also unsupported, along with the `rand` and `srand` PRNGs, thus forcing calls to the CPU's hardware RNG via the SGX-only function `sgx_read_rand`. Originally, we planned to use a popular C++ machine learning library, such as Shark[6], that offered implementations of a variety of learning algorithms. Such libraries, however, rely heavily upon system calls and I/O functions, meaning bespoke, SGX-compatible algorithms had to be re-developed for use within enclaves. For complex learning algorithms, like SVMs and Random Forests, redesigning and reimplementing libraries with large code bases is difficult while retaining correctness without expert oversight. Indeed, popular SVM libraries have non-trivial code bases (>5,000 lines of code for LibSVM[7]). This issue was less problematic for kNN, LR and NB – all relatively simpler in nature – which were favoured in the test-bed implementation. These algorithms are described briefly in the following section.

3.4.2 Machine Learning Algorithms

In machine learning, classification is a supervised learning task in which a function, f, which maps input features to output labels, is inferred/'trained' using a set of example input-output (feature-label) pairs. This example set is also known as the training set: $\mathbf{T} = \{(F^{(1)}, C^{(1)}), (F^{(2)}, C^{(2)}), \ldots, (F^{(n)}, C^{(n)})\}$. A feature vector, $F = (x_1, x_2, \ldots, x_m)$, comprises a vector of real-values, where m is the number of observed features, while $C \in \{1, \ldots, c\}$ represents the corresponding label or class ($c = 2$ for binary classification, e.g. authentication). Numerous classification algorithms exist in order to model f using various techniques, such as Bayesian probability, finding the closest members in feature space, or minimising a cost function of a linear function that maps feature vectors to their labels (linear

[6]Shark Machine Learning: http//image.diku.dk/shark/
[7]LibSVM: https://www.csie.ntu.edu.tw/~cjlin/libsvm/

optimisation). The process of systematically classifying new feature vectors after training is known as 'testing'. We describe the algorithms explored in this work below.

Naïve Bayes (NB) is a widely-used probabilistic classifier that uses Bayes' Theorem with a strong independence assumption between features. Given a labelled training set, **T**, the training phase involves computing the label priors (the probabilities of each label occurring in the set, i.e. $p(C_k)$), and feature conditional probabilities (the probabilities of features occurring given the label, i.e. each $p(x_i|C_k)$). A feature vector is subsequently classified by maximising the posterior probability for each class label C_k, given in Equation 3.1. While strong independence between features may seem intuitively unwise, Naïve Bayes has been applied successfully in wider literature and, indeed, in CA successfully [119], [129], [130], [133].

$$C_{new} = \underset{k \in \{1,\dots,c\}}{\arg\max}\, p(C_k) \prod_{i=1}^{m} p(x_i|C_k) \tag{3.1}$$

k-Nearest Neighbour (kNN) classifies a feature vector, F, using the k closest neighbours to F in feature space ($F \in \mathbb{R}^m$). That is, given a pre-existing training set, **T**, kNN measures the distance (typically Euclidean distance) between F_{new} and each $F^{(i)} \in \mathbf{T}$. F_{new} assumes the label of the nearest k neighbours in **T** using majority vote. kNN has no dedicated training period, other than storing feature vectors with which to reference future examples.

Logistic Regression (LR) is a binary classification method in which the probability of the occurrence of a positive example is found by computing the logit function (Equation 3.2) over a weighted sum of the feature vector. That is, the probability of the label, $c = 1$, given an example feature vector F is found by computing the logit function, g, on the sum of weights multiplied by each feature $x_i \in F$.This is denoted in Equation 3.3, where w_i is the weight of the i^{th} feature. The set of weights is found by minimising the ordinary least-squares cost function of the training examples and labels using gradient descent [151]. An open-source C++ implementation[8] of logistic regression was adapted for the test-bed implementations.

$$g(z) = \frac{1}{1 + e^{-z}} \tag{3.2}$$

$$p(c = 1|F) = g\Big(\sum_{i=0}^{m}(w_i x_i)\Big) \tag{3.3}$$

[8]'A Simple LR Implementation in C++': https://github.com/liyanghua/logistic-regression-in-c--

The test-beds implement TEE-compatible versions of the above algorithms and a simple API is exposed through which application developers can train and test data in the TEE from the Rich OS. All functions use purely `ECALL`s for SGX; no CA-specific data is divulged to the Rich OS other than: 1), a value indicating successful training, and 2), the authentication decision. The test-bed spawns a enclave that hosts each algorithm, which was implemented in C++ using Microsoft Visual Studio and the Intel SGX SDK, and compiled using the Intel compiler provided. The Intel SGX SDK allows the development of applications that instantiate enclaves natively. We implement the training and testing phases inside the enclave environment, which are managed through a restricted API from the host application. Only two context switches occur between SGX and the Rich OS: **1)**, to transfer raw sensor data from the Rich OS to the TEE; and **2)**, transferring an acknowledgement message (after training) or a boolean authentication value (testing) from the TEE to the Rich OS.

3.4.3 OP-TEE

OP-TEE [152] is an open-source, GlobalPlatform-compliant TEE framework developed by Linaro, which currently supports executing alongside a custom minimal Linux-based kernel; Debian (also Linux-based); or, for certain development boards, Android OS. OP-TEE implements the GlobalPlatform system architecture (see Section 2.4.7 of Chapter 2) using ARM TrustZone, and allows host applications executing in the untrusted REE to communicate over the GlobalPlatform Client API to access Trusted Applications (TAs) executing in OP-TEE. A Linux driver is provided that uses ARM-based Secure Monitor Calls (SMC) to implement world transitions between client applications in the REE and TEE TAs (see Section 2.4.4). The OP-TEE OS provides interfaces for TAs to use, amongst others, software- and hardware-based cryptographic operations (depending on the development board), external SEs, clocks and timers, and secure storage using the sealing abstraction in the GlobalPlatform Internal API. Internally, OP-TEE currently uses cryptographic implementations from LibTomCrypt, such as for AES, SHA-2, RSA, Diffie-Hellman and Elliptic Curve Cryptography (ECC[9]). The reader is referred to the OP-TEE repository[10] for a list of currently supported development boards, current issues, and existing and planned feature support.

Unsurprisingly, memory consumption quickly became problematic when working with large datasets within OP-TEE. For the current OP-TEE release (July 2018), 32MB RAM is allocated for the TEE kernel and all resident TAs, with the rest allocated to the untrusted world OS. For a standard TA, the Linaro Working

[9] At present, GlobalPlatform ECC support is limited to NIST-approved curves up to P521.
[10] OP-TEE: `https://github.com/OP-TEE/optee_os`

Group[11] recommends a default stack and heap size at 1kB (stack) and 32kB (data) respectively. While this can adjusted to a virtually arbitrary limit based on the RAM capabilities of the chosen platform – 2GB of RAM on our development board (HiKey LeMaker), which must also accommodate the REE – the Linaro Working Group recommends a maximum of 1MB for the stack and heap. These recommendations are advised to maximise the RAM available to the REE. For the purposes of this test-bed, this was exceeded in order to accommodate data above 1-day's worth: to 1MB and 40MB for the stack and heap respectively in order to accommodate 21-days worth of training data for each user. (In comparison, Intel SGX supports up to 128MB PRM in RAM by default). As such, a potential OEM wishing to deploy TEE-based CA must be aware of CA's larger run-time memory requirements.

The design of the OP-TEE test-bed comprises a REE-side application that communicates through the GlobalPlatform Client API, over the SMC secure monitor, and to the corresponding TA hosted by the TEE. Currently, OP-TEE does not support C++ TA development – requiring TAs and accompanying REE-side client applications to be written in C (C99, precisely) and compiled with GCC[12]. The OP-TEE CA TA exposed simple training and testing API functions for each algorithm, which were available to the untrusted REE client application. The algorithm code-base was identical to the SGX implementation in nature, but rather than communicating over the Intel SGX enclave boundary with its host application, the OP-TEE client application communicated over the GP Client API with its respective TA. Both test-beds exposed the same training and testing function definitions.

3.5 Evaluation

The immediate concern was the overhead incurred with the maintenance of a TEE application, such as world context switches (between enclave and non-enclave environments for SGX, and SMC monitor calls using OP-TEE and TrustZone between the client and trusted applications). For SGX, this also includes the real-time enclave page encryption performed for preserving the integrity and confidentiality of CPU-DRAM traffic. In this section, the evaluation of TEE versus non-TEE CA is described using a series of benchmarks from the test-beds in Section 3.4.

We employ the GCU dataset provided by Kayacik et al. [135], which provides mobile sensor data collected from 7 users (staff and students at a UK university) over a period of 2–14 weeks. The dataset is sampled at a rate of 5 minutes,

[11] Linaro Working Group TEE recommendations: https://wiki.linaro.org/WorkingGroups/Security/OP-TEE

[12] C/C++ support in OP-TEE: https://github.com/OP-TEE/optee_os/issues/1708

consisting of nearby WiFi access points, cell tower IDs and current applications running on the device. Optimal sampling rates and window sizes depend entirely upon scheme and the performance/security requirements of the scheme operator (this trade-off is analysed in [149]). To avoid ambiguity, we use the GCU dataset without re-sampling. Training set size also depends on the requirements of the operator: more data/large training sets will invariably imposes greater computational demands, but larger training sets have been shown to improve scheme accuracy [128]. Because of this, we evaluate a set of training periods discussed in related work, ranging from 1-day to 3-weeks [119], [149]. Concretely, the following components are evaluated:

- **Training**: the performance of training each algorithm using sets lasting 1-, 5-, 14- and 21-days, reflecting related work and beyond. kNN is unique in having no dedicated training phase, so we measure the time to load necessary sensor data into internal data structures in preparation for testing. This is implicit in the timings of NB and LR.

- **Testing**: the overhead of producing the authentication decision from a vector of sensor data. We measure the average time taken to classify one hour of sensor data using all three of our implemented algorithms after training each classifier for each of the aforementioned training times.

The round-trip time was measured between issuing the request from the untrusted world and receiving the corresponding result, i.e. an acknowledgement and authentication decision for training and testing respectively. For Intel SGX, this was measured using the `<chrono>` library from C++11, which provides a high-resolution clock with nanosecond precision. For the strictly C99 TA implementation aboard OP-TEE, the time taken was implemented in the client application using the C-compatible `<time.h>`, with microsecond precision. Aside from this discrepancy, the algorithm implementations were identical. Compiler-level optimisations were disabled for both the SGX and OP-TEE implementations.

The GCU data was separated into the training and test sets for each user using a Python script. The training set represents the first 1, 5, 14 and 21 days of sensor data from each user, thus simulating an enrollment period after purchasing the device. We performed this for each user where possible; in total, all 7 users had training data available for up to 14 days, while only 3 users had over 21 days of available data. For training, we follow Li et al. [128] and Fridman et al. [133], and randomly pair each user with another to construct negative/positive valued training sets, which was performed pre-training within the aforementioned Python script. Here, the test user's readings are considered as legitimate (positive) values, while the paired user's values are considered illegitimate (negative). For the authentication phase, we measure the average time taken to classify a single

TABLE 3.3: Mean F1-scores for each algorithm across all GCU users using X-days of enrollment data (S.D. in brackets).

Method	1-day	5-days	14-days	21-days
NB	0.51 (0.02)	0.59 (0.11)	0.71 (0.16)	0.70 (0.18)
kNN, $k = 3$	0.55 (0.08)	0.65 (0.14)	0.80 (0.20)	0.80 (0.20)
kNN, $k = 5$	0.52 (0.06)	0.67 (0.19)	0.81 (0.22)	0.79 (0.19)
kNN, Avg.	0.54 (0.07)	0.66 (0.17)	0.81 (0.21)	0.80 (0.20)
LR	0.52 (0.05)	0.61 (0.12)	0.73 (0.21)	0.72 (0.20)

TABLE 3.4: Mean classification accuracy for each algorithm across all GCU users using X-days of enrollment data.

Method	1-day	5-days	14-days	21-days
NB	0.52 (0.03)	0.58 (0.13)	0.71 (0.23)	0.68 (0.16)
kNN, $k = 3$	0.56 (0.06)	0.67 (0.15)	0.82 (0.20)	0.80 (0.20)
kNN, $k = 5$	0.54 (0.06)	0.66 (0.18)	0.82 (0.19)	0.81 (0.20)
kNN, Avg.	0.55 (0.06)	0.67 (0.17)	0.82 (0.20)	0.81 (0.20)
LR	0.54 (0.05)	0.58 (0.13)	0.74 (0.21)	0.69 (0.17)

feature vector. These feature vectors comprise hourly segments of all the readings after the enrollment date until the end of the file for each user. The average time was computed to classify each individual feature vector across users.

For SGX, the evaluation was performed using a consumer laptop – a Lenovo Thinkpad T460s with an SGX-enabled Intel i5-6200U CPU (2.8GHz clock) and 8GB DDR4 RAM (2133MHz) – using Microsoft Windows 10 as the deployment platform. For the OP-TEE implementation, a HiKey LeMaker development board was chosen featuring a HiSilicon Kirin 620 SoC with 2GB DDR3 RAM (800MHz) and an ARM Cortex A53 CPU (eight-cores at 1.2 GHz with TrustZone extensions).

While we use methods and modalities from related work, it is not an exact replication of an existing multi-modal CA scheme; the reasons for this, along with other limitations, are discussed later in Section 3.5.2. As such, we indicate some error rates for our baseline scheme using the GCU dataset in Tables 3.3 and 3.4 to contextualise the performance results presented in Section 3.5.1. Table 3.3 indicates the F1-scores[13], as used in [129], [132], [134], defined as a function of each classifier's *precision*, the number of positive identifications that were actually correct, i.e. true positives (TP) divided by TP and false positives (FP); and *recall*, the number of actual positives that were correctly identified, i.e. TP divided by TP and false negatives (FN). The equations for precision and recall, and the F1-score are given in Equations 3.4 and 3.5 respectively. We also provide the classification accuracy in Table 3.4, as used in [137], defined as the number of correctly identified true positive (TP) and negative (TN) results as a proportion of all outcomes (Equation 3.6). To clarify, a 'positive' result is where the user

[13] Also known as the F-score or F-measure.

TABLE 3.5: Training phase. Mean wall-clock training times across all users using X-days of enrollment data conducted ten times (in milliseconds).

Method	1-Day	5-Days	14-Days	21-Days
Lenovo T460s				
NB	80.45 (16.07)	1040 (278.0)	3650 (606.6)	3534 (636.2)
kNN, $k = 3$	64.50 (16.50)	655.2 (95.84)	2218 (227.2)	1978 (429.3)
kNN, $k = 5$	66.57 (15.09)	660.4 (76.80)	2278 (304.9)	1868 (315.9)
kNN, Avg.	65.54 (15.80)	657.8 (86.32)	2248 (266.1)	1923 (372.7)
LR	2027 (26.11)	8843 (101.3)	12114 (421.8)	13018 (843.5)
SGX NB	3.652 (0.557)	20.29 (2.479)	55.49 (7.027)	57.19 (13.95)
SGX kNN, $k = 3$	2.005 (0.189)	11.46 (1.619)	30.74 (3.129)	25.97 (9.170)
SGX kNN, $k = 5$	1.938 (0.347)	11.74 (1.339)	31.13 (4.053)	30.21 (1.828)
SGX kNN, Avg.	1.971 (0.268)	11.60 (1.479)	30.93 (3.591)	28.09 (5.499)
SGX LR	15.68 (3.73)	90.30 (5.77)	139.65 (15.90)	120.12 (14.11)
HiKey LeMaker				
NB	133.7 (20.39)	1952 (261.6)	4322 (557.2)	4334 (497.0)
kNN, $k = 3$	116.4 (19.74)	1359 (98.08)	3989 (336.4)	3897 (399.2)
kNN, $k = 5$	122.3 (18.22)	1328 (153.3)	4060 (349.5)	3904 (273.7)
kNN, Avg.	119.3 (18.98)	1344 (125.7)	4025 (342.9)	3901 (336.5)
LR	3840 (261.4)	14611 (571.3)	20901 (1018)	20472 (1277)
OP-TEE NB	145.2 (17.10)	1997 (147.4)	4401 (327.3)	4476 (433.4)
OP-TEE kNN, $k = 3$	130.3 (24.05)	1398 (150.0)	4165 (343.2)	4089 (353.8)
OP-TEE kNN, $k = 5$	136.8 (27.82)	1424 (102.6)	4173 (275.4)	4062 (385.0)
OP-TEE kNN, Avg.	133.6 (25.94)	1411 (126.3)	4169 (299.3)	4076 (369.4)
OP-TEE LR	4186 (110.1)	14758 (488.5)	21305 (982.5)	20736 (1146)

is authenticated, while a 'negative' is where the user is not authenticated. We compute the metrics across all users for the enrollment periods and algorithms considered.

Note that, while our results are below the state of the art – 0.81 (F1) and 0.82 (Acc.) in the best case using kNN with 14-days of enrollment data, versus 0.97 (F1) in [129], 0.98 (Acc.) in [137], 0.95 (F1) in [134], and 0.90 (F1) in [132] – the aim of this work is to provide a first step in investigating the feasibility of TEE-based CA, rather than proposing a novel authentication scheme.

$$Precision = \frac{TP}{TP + FP} \qquad Recall = \frac{TP}{TP + FN} \tag{3.4}$$

$$F1_{score} = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{3.5}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.6}$$

TABLE 3.6: Testing phase. Mean wall-clock time to classify one-hour of sensor data from X-days of training data across all users (in microseconds).

Method	1-Day	5-Days	14-Days	21-Days
Lenovo T460s				
NB	125.7 (5.936)	142.7 (8.712)	158.3 (40.33)	153.3 (35.64)
kNN, $k = 3$	458.4 (64.23)	2539 (319.0)	6554 (972.0)	5976 (1623)
kNN, $k = 5$	464.1 (55.69)	2481 (258.5)	6634 (998.9)	5949 (1803)
kNN, Avg.	461.3 (59.96)	2510 (288.8)	6594 (985.5)	5963 (1713)
LR	95.2 (8.838)	102.8 (30.39)	112.9 (36.90)	113.5 (40.01)
SGX NB	14.28 (2.498)	14.17 (0.951)	12.86 (1.864)	14.67 (9.333)
SGX kNN, $k = 3$	100.4 (20.02)	584.7 (63.30)	1598 (231.6)	1481 (468.5)
SGX kNN, $k = 5$	100.8 (16.79)	590.6 (63.90)	1593 (197.8)	1452 (431.0)
SGX kNN, Avg.	100.6 (11.26)	587.7 (63.60)	1596 (214.7)	1467 (449.8)
SGX LR	8.876 (1.002)	9.405 (2.803)	9.384 (0.776)	9.523 (1.249)
HiKey LeMaker				
NB	241.0 (14.22)	250.2 (45.41)	267.5 (30.18)	280.4 (37.60)
kNN, $k = 3$	878.2 (70.68)	3801 (291.9)	11560 (1745)	10007 (1419)
kNN, $k = 5$	873.0 (66.35)	3988 (282.7)	12010 (2327)	10554 (2136)
kNN, Avg.	875.6 (68.51)	3895 (287.3)	11785 (2036)	10281 (1778)
LR	203.3 (9.334)	250.8 (9.750)	257.4 (12.81)	252.3 (10.19)
OP-TEE NB	535.5 (17.65)	560.2 (25.05)	552.1 (30.18)	531.9 (49.56)
OP-TEE kNN, $k = 3$	1139 (93.81)	4212 (302.5)	12388 (883.0)	12046 (1071)
OP-TEE kNN, $k = 5$	1180 (88.02)	4376 (355.8)	12505 (1420)	12154 (1367)
OP-TEE kNN, Avg.	1160 (90.92)	4294 (328.2)	12447 (1152)	12100 (1219)
OP-TEE LR	484.7 (12.57)	495.6 (11.03)	521.0 (20.65)	528.7 (16.21)

3.5.1 Results Discussion

We present our findings in Tables 3.5 and 3.6. As expected, training times for Logistic Regression and Naïve Bayes far exceed kNN, which requires no dedicated training procedure. Instead, during testing, kNN uses an exhaustive search through every member of the training set to find its k closest neighbours. As such, we expected kNN to yield the reverse effect, i.e. becoming dramatically slower in testing. This is compared to NB where, once priors and conditionals are computed, computing the class is a series of joint probability multiplications, which performs in near-constant time. LR simply computes the logit function for a given feature vector and the parameters acquired by minimising the least-squares cost function (see Section 3.4.2). In general, LR was the costliest to train, while kNN was the costliest to test. These trends were consistent across all days worth of data.

The results of training and testing drop slightly between 14 and 21 days, which we found was due to the variance in device usage between users. While the dataset is sampled at 5-minute intervals, this does not account for periods when the device is disabled. Thusly, Figure 3.4 depicts the mean number of samples captured per day for each participant. Only three users had data lasting 21-days – users 1, 2 and 5 – constituting 198 samples per day per user on average,

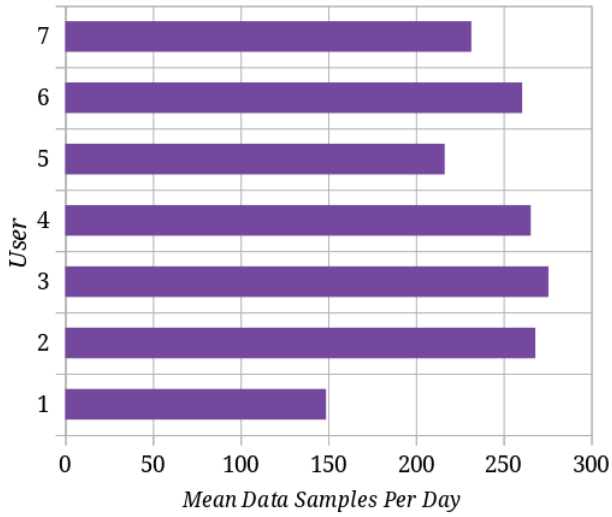


FIGURE 3.4: Mean samples per day per participant in the GCU dataset [135].

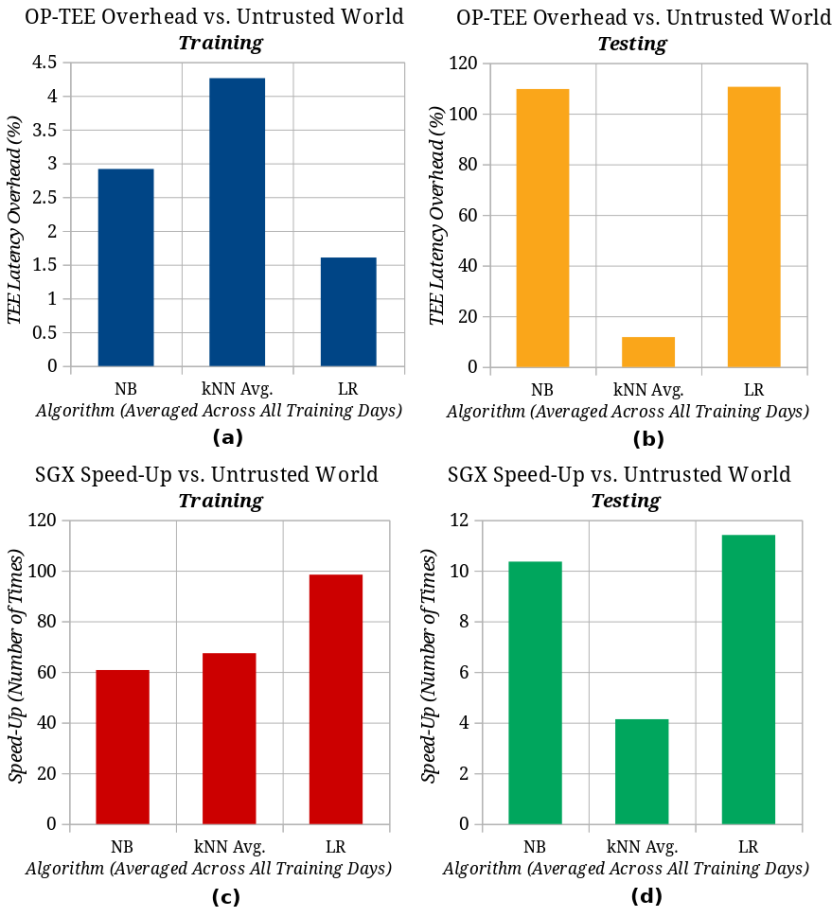


FIGURE 3.5: TEE versus non-TEE overhead averaged across training and testing. **(a)** training and **(b)** testing overhead of OP-TEE versus untrusted world; and relative SGX speed-up of **(c)** training and **(d)** testing versus non-SGX implementation.

while, for 14 days, this rises to 245 samples per day. The differences in sample 'density', due to the usage variance between users, is the primary driver behind the cause for similar timings between 14-day and 21-day training and testing. This lower training times of 21-days worth of data is hence biased heavily by the unusual sparsity of User 1. Evidently, device usage is a significant influence in determining CA training and testing times.

The comparative performance of each platform is illustrated in Figure 3.5. As expected, OP-TEE yields a small overhead during testing and training. In training, this yields a small overhead of approximately 1.5–4.25% depending on method. In absolute terms (Table 3.5), this ranges between a 12ms overhead (one-day training average with NB) to 404ms (14-days training average with LR). While certainly noticeable in isolation, a maximum overhead of 404ms is unlikely to be burdensome, as training/enrollment is either a one-off or occasional task. Occasional re-training may occur, for example, once every week or month to refresh a model that no longer reflects the user's current behaviour, say, due to natural changes in the user's (permanent) location, e.g. residency, or application usage (after installing new applications or deleting unused ones). During testing, it seems that OP-TEE yields a high relative overhead for NB and LR of approximately 110% versus 10% for kNN. However, this relative overhead is somewhat misleading as the absolute kNN testing time is the highest of all (at most 12,154μs for kNN versus 528.7μs for LR with 21-days of training data). While kNN has a substantially lower training time, its exhaustive search during testing yields a 20-24 times overhead relative to the absolute testing times NB and LR. One reason for the high relative TEE overhead of NB and LR is the greater impact that context switching plays relative to the actual testing time.

To our knowledge, no literature exists regarding the precise performance overhead incurred by OP-TEE and ARM TrustZone versus a suite of untrusted applications. Benchmarking the latency incurred by SMC calls and OP-TEE OS internals in isolation is out-of-scope in this paper. However, we believe that such an investigation would be worthwhile for the benefit of TEE applications in general, e.g. DRM and payment tokenisation, as a separate research contribution.

Most surprising is the substantial difference between SGX and non-SGX performance. At first, we believed SGX would incur some overhead, with the additional memory enforcement, e.g. encryption of enclave pages between the CPU and DRAM, and context switching. Yet, it exceeds the REE by approximately 60-100 times for training and 4–11 times for testing, depending on the algorithm. Upon investigation, while SGX and non-SGX implementations were identical, enclave applications are linked with Intel's own C (`sgx_tstdc.lib`) and C++ standard libraries (`sgx_tstdcxx.lib`), which must be compulsorily linked to any SGX enclave application. These 'trusted' libraries are used to provide

functions that can only be used within enclaves. At present, enclaves do not support I/O operations or functions that rely upon OS system calls. Indeed, attempting to link non-Intel C/C++ standard libraries will *"either fail the enclave signing process or cause a runtime failure due to the use of restricted instructions"* [153]. Intel's trusted libraries provide an optimised subset of compatible functions based on Intel's Integrated Performance Primitives (IPP) [154]. This suite offers parallelised, multi-threaded optimisations of standard routines based on the most recent AVX and SSE instruction sets, which *"significantly increases performance"* over unoptimised implementations [154]. Comparatively, the non-SGX portion uses the Microsoft Visual C++ standard library with no such enhancements.

Interestingly, training times benefit most, due likely to the computationally intensive methods involved, namely computing prior and conditional probabilities for Naïve Bayes, and linear optimisation for LR. Further investigations revealed that IPP *does* favour such arithmetical[14] and string-based[15] operations strongly. Unfortunately, the fact that SGX's C/C++ standard libraries must be compulsorily linked is an unwelcome distortion in gauging the overhead of SGX versus non-TEE performance directly.

What we can conclude, however, is that the observed performance of SGX raises security concerns in itself: there is a clear incentive for developers to move non-critical components to enclaves if some perceived security benefits can be gained with little performance overhead. This also applies to OP-TEE as the timing overhead, in real terms, is not dramatically greater than the untrusted world. Yet, recklessly shifting code to a TEE can dangerously increase the TCB of the trusted application, which may expose security-critical components to logical flaws or programming bugs from non-security related code sections. The essence of a secure application in any TEE is to minimise its TCB to that which is functionally necessary, and any incentive to increase the TCB for performance gains may undermine the application's overall security in reality. A second observation is that, for both TEEs, algorithm choice has a far more significant effect on training and testing times than whether to use a TEE.

3.5.2 Limitations Discussion

Currently, our test-bed does not evaluate the small number of proposed CA approaches using hand-crafted probability models or unsupervised learning, as in e.g.[135] and [127]. Rather, we favour the majority of work by using binary supervised learning with dedicated enrollment and testing phases, as per [119], [126], [128], [130], [133]. At first, it would seem obvious to re-implement the schemes from related work on commodity handsets. However, this task is

[14] https://software.intel.com/en-us/node/502094
[15] https://software.intel.com/en-us/node/501987

critically impeded by the closed nature of commercial smartphone TEEs, such as Trustonic Kinibi, as used on the Samsung Galaxy S6/7/8 [99]. GlobalPlatform-complaint TEEs on mobile devices are generally locked before deployment (see Section 2.4.7 of Chapter 2), meaning developing and installing TAs is not possible without OEM cooperation. Another note is that TEEs on consumer handsets host and execute a number of other TEE applications simultaneously, such as for DRM (e.g. WideVine[16] for video copyright protection) and payment security, which could potentially affect the latency of a CA scheme in practice. We attempted to mitigate this limitation by using commercially-available hardware in both test-beds, with similar specifications to a commodity device and using open-source tooling (OP-TEE). The SGX implementation used the Intel SGX SDK on an off-the-shelf laptop (Lenovo T460s) to reflect multi-modal CA on a more powerful mobile device.

Furthermore, only three algorithms were implemented as part of the test-bed – Naïve Bayes, Logistic Regression and k-Nearest Neighbour – due primarily to the difficulty of developing significantly more intricate learning algorithms correctly for each TEE without the availability of compatible, peer-reviewed libraries. As mentioned previously, SGX does not support fundamental C++ libraries, such as `<sstream>`, which severely impeded the adoption of open-source machine learning implementations. The development of bespoke, SGX-compatible algorithms was hence necessary, and classification algorithms with smaller code bases were favoured to assure correctness. Given the widespread deployment of SGX, it is hoped that correct SGX implementations will emerge over time in order to evaluate a greater set of learning algorithms correctly.

A major but practically insurmountable limitation is that, in a real-life deployment, one would expect sensor data to be collected in real-time over a trusted path directly between the TEE and the relevant sensing hardware, without exposure to the untrusted world. Ideally, a system-on-chip should protect the relevant sensors from untrusted world accesses using the TrustZone Protection Controller (TZPC), described previously in Section 2.4.4 of Chapter 2. Unfortunately, OP-TEE and our associated development board did not support this. An appropriate SoC would have to be designed and manufactured, alongside the development of kernel-level OP-TEE extensions to support the configuration of protection controllers with the appropriate sensing hardware. This problem is inherent with Intel SGX, which does not currently support a secure I/O path between peripherals and enclaves.

A related limitation was the inability to conduct an on-line analysis using sensors connected directly to the TEE. As a result, we chose to utilise the modalities

[16] WideVine: https://storage.googleapis.com/wvdocs/Widevine_DRM_Architecture_Overview.pdf

available from the GCU dataset provided by Kayacik et al. [135]. Acquiring data for CA is a costly and non-trivial task, often requiring users to carry devices for many weeks, or potentially months, for a realistic evaluation. In future work, extra modalities would allow a comparison for how trusted CA scales with input sources, particularly with denser and higher-dimensional data sources like touch-points and keystrokes. Because of these limitations – primarily the lack of access to TEEs aboard commodity mobile handsets – we stress that our experiments should be considered as *indicative* results. However, we show that the application of TEEs to CA is a promising solution in protecting its assets from root-level on-device adversaries.

3.6 Conclusion

In this work, we introduced the need for trusted execution CA and broadened the existing threat model to incorporate system-level adversaries encountered in a practical deployment. To this end, we examined two standardised constructs for offering greater trust and security provisions without compromising deployability: Secure Elements and Trusted Execution Environments. After analysing the potential security threats to CA schemes in practice, the features and suitability of SEs and TEEs were analysed as potential solutions. Following this, we proposed using a TEE to encapsulate sensitive aspects of a CA scheme in order to address the extended threat model. We implemented and evaluated this using two test-bed environments using Intel SGX and OP-TEE – a GlobalPlatform TEE using ARM TrustZone – with commercially-available hardware. This was followed by a suite of indicative performance measurements from a variety classifiers and sensor modalities used in related work using a public dataset from real-world users. Our results indicate that CA *can* be performed on Intel- and ARM-based TEEs with only a relatively modest performance overhead (up to a maximum of 404ms in absolute terms). Our work indicates that applying TEEs to CA is practical towards protecting the run-time execution state of CA schemes from a powerful adversarial model.

3.6.1 Future Work

As part of our ongoing research, we aim to investigate the following:

- Evaluating the performance overhead of context switching, SMC calls and OP-TEE TEE internals. Currently, there is little literature on the precise overheads imposed by individual TEE components.

- A real-time evaluation of trusted CA on various device types, such as wearables, and the associated performance penalty on differing computational platforms.
- Collecting a new dataset comprising touch-points, keystrokes and additional environmental sensors, and analysing how trusted CA scales with input sources from additional modalities.

Chapter 4

On Mutually Trusted Channels for Remote Sensing Devices with TEEs

4.1 Introduction

The past two chapters explored the range of secure and trusted execution technologies available to modern embedded and mobile systems in Chapter 2, before analysing and evaluating their potential application to continuous authentication in Chapter 3. This chapter continues this theme of trusting data from remote sensing devices in sensitive IoT deployments by examining the challenges surrounding the transmission of data *between* TEEs hosted on remotely located devices.

4.1.1 Motivation

The introductory chapter of thesis (Chapter 1) introduced and motivated the challenges of trusting data transmitted from remote sensing devices in order to, ultimately, inform critical-decision making. This may be conducted by a largely automated system based on artificial intelligence (AI), or the data could be aggregated and visualised on a human-machine interface (HMI) to inform user input. These devices, which are more than likely to be largely unattended – that is, constant or frequent expert inspection is likely to be prohibitively expensive – are being, or have been, proposed to be deployed in sensitive environments.

Previous literature in trusted sensing has focused heavily on the TPM as a root of trust, but these forgo desirable features of recent developments in TEEs, such as secure I/O. In TPM-based literature, remote attestation has underpinned previous solutions by enabling a remote verifier to securely verify the operating state of a target platform at a distance based upon its measurements collected at boot-time (or shortly thereafter, also known as DRTMs, which were seen in Chapter 2). In this chapter, we examine the state-of-the-art of remote attestation, including the benefits and challenges of static attestation, dynamic attestation, attesting groups of devices (group attestation), and providing trust assurances of both communicating devices in a single protocol run (mutual attestation). From

this examination, we raise the challenge of secure TEE-to-TEE communication between remote devices with mutual trust assurances, which has not hitherto been addressed in existing literature.

To this end, we develop two novel secure and trusted channel protocols that perform mutual or one-way remote attestation in a single run with TEEs. We specify the security and functional goals relevant to TEE mutual attestation aimed at in this work, including the challenge of remaining agnostic to particular TEE architectures. This includes any differences in deployment assumptions, and the crucial challenge of avoiding the disclosure of security-sensitive data outside the TEEs on either device, i.e. their untrusted worlds.

4.1.2 Chapter Structure

In this paper, we first review related work in trusted sensing platforms generally (Section 4.2.1), before proceeding to covering the state-of-the-art in remote attestation (Section 4.2.2). In particular, the issue of mutual trusted channels for TEEs – the notion of secure TEE-to-TEE communication with trust assurances – has received little attention in past literature. To address this, we present a novel protocol that provides secure and trusted channels between two remote TEEs (Sections 4.4 and 4.5), where trust can be verified *uni-directionally* or *bi-directionally* through one- and two-way remote attestation respectively. This provides resilience to sophisticated integrity and confidentiality software attacks conducted in the intermediate components from REE- and network-level adversaries. Our work is applicable to sensing applications requiring strong mutual trust assurances. We also discuss measures for facilitating interoperability between disparate TEEs, such as Intel SGX and the GlobalPlatform TEE. The protocols are subjected to formal symbolic analysis using Scyther, before presenting a performance evaluation using GlobalPlatform-compliant TEEs on two networked ARM development boards (Sections 4.6 and 4.7). Lastly, we conclude our findings and identify future research directions in Section 4.8.

4.1.3 Contributions

The work presented in this chapter provides the following contributions:

- The first investigation into the challenge of TEE intercommunication between remotely-located TEEs on distinct constrained devices.
- The design and proposal of two secure and trusted protocols for facilitating TEE intercommunication based upon the relevant security and functional goals. The protocols provide either uni- or bi-directional trust assurances for addressing cases where one or both devices contain TEEs respectively.

Both proposed protocols are subjected to formal symbolic verification using the Scyther analysis tool by Cremers [29], which found no attacks under a Dolev-Yao adversarial model. The Scyther protocol scripts are provided in Appendix A.

- A performance evaluation in a test-bed implementation containing two ARM development boards hosting GlobalPlatform TEEs. The protocols execute in reasonable time (under 1.7 seconds round-trip on average), with approximately four-times overhead versus TLS and SSH without optimisation.

This chapter is based on work from our following publication:

- C. Shepherd, R. N. Akram, and K. Markantonakis. "Establishing Mutually Trusted Channels for Remote Sensing Devices with Trusted Execution Environments", in *Proceedings of the 12th International Conference on Availability, Reliability and Security*, ser. ARES '17, ACM, 2017, 7:1-7:10.

4.2 Related Work

This section reviews related work concerning trusted sensing generally, before moving to an examination of remote attestation mechanisms.

4.2.1 Trusted Sensing Platforms

Sensing with standardised trust technologies has attracted notable attention in past literature; we discuss relevant work and highlight their contributions. Liu et al. [155] propose two abstractions for providing trust in sensing platforms and their data. This comprises **(1)**, *sensor attestation*, which signs each reading with a TPM's Attestation Identity Key (AIK) and verified with its public key; platform integrity is preserved using static remote attestation, defined in Section 4.2.2; and **(2)**, *sensor seal*, in which encrypted secrets are bound to the TPM and released once sensor readings satisfy a given policy. These two concepts are evaluated using ARM TrustZone with a TPM emulated in software, and Credo, a TPM-backed hypervisor, for X86 systems. Both are evaluated yielding moderate overhead.

Gilbert et al. [156] examine TPMs to provide data assurance in sensitive sensing applications. Similarly, the authors propose a TPM-backed hypervisor to protect device drivers and sensing processing applications from software integrity attacks. Untrusted user applications are executed in a separate guest domain to facilitate isolated execution, with the hypervisor acting as a secure monitor. It is proposed to sign sensing measurements with a signed TPM quote,

which includes the TPM Platform Configuration Register (PCR) responsible for the sensing application. Remote attestation can then be used by the verifier to ascertain assurances regarding the expected operating state of the device.

Saroiu et al. [157] propose two sensing architectures using TPMs: **(1)**, a TPM-backed hypervisor that sandboxes users' software in a guest Virtual Machine (VM), and a root VM with privileged access to sensor drivers and the TPM. The root VM signs sensor readings with the TPM to prevent modification once passed to the user VM. Design **(2)** proposes attaching a TPM to each sensing microcontroller to protect against a malicious kernel. Here, the microcontroller signs readings using the TPM independently of the OS. Remote attestation can then be used to either attest the state of a single TPM, or to attest all TPMs collectively that are attached to each sensing microcontroller. The authors note that **(1)** is less costly – both performance-wise and economically – but provides fewer security guarantees.

This thesis continues its focus on providing greater trust assurances in modern constrained sensing platforms – a known barrier to deploying beneficial services to end-users [157]. Providing evidence of platform integrity is a core component to assuring remote providers that may rely on sensor data sourced from the device. Next, we examine the process of securely reporting platform integrity measurements – remote attestation – in greater detail below.

4.2.2 Remote Attestation

Remote Attestation (RA) was first introduced in Section 2.3.2 of Chapter 2. RA is an interaction protocol between a prover, P, and verifier, V, in which V securely ascertains the current state of the remote of P. The goal of RA is to assure V that P is operating with an expected platform configuration, which is conducted remotely, such as over the Internet or Local Area Network (LAN). In this section, we examine a range of RA techniques based on *static*, *hybrid*, *dynamic*, *property-based*, *group* and *mutual* attestation. We note that attestation is a continually evolving domain; the reader is referred to work by Steiner and Lupu [158] and Abera et al. [159] for a study of existing attestation techniques in greater detail.

Hardware-based Static Attestation

This section briefly describes hardware-based static attestation as used by the TPM and Intel SGX. TPMs collect a set of platform configuration measurements upon boot, call this set $M = \{PCR_1, PCR_2, \ldots, PCR_n\}$, which is used in conjunction with some secure channel protocol to transmit M to V. M is a chain of integrity measurements, comprising critical software components, that serves as the evidence with which V is reasonably convinced that the system is running

to expectations. Any deviation in these measurements, e.g. bootloader or BIOS, which are stored in the PCRs is an indication that it has been modified. (Note that the TPM does not take decisions itself from these measurements: it accepts and stores them in a secure and reliable manner. It is the responsibility of V to decide and enact upon the values in M). For reasons of authenticity and integrity, the measurements are transmitted as part of a signed data structure known as a *quote* report, Q, using the TPM's AIK. TPM 2.0 stipulates DAA to solve the issue of AIKs being used to link particular TPMs. DAA uses a group signature scheme in which one public-key is responsible for verifying multiple private AIKs, thus preventing an adversary from linking signed quotes by unique, TPM-specific keys to an individual platform [63].

Intel SGX's remote attestation mechanism, which was described in detail in Section 2.4.6 of Chapter 2, operates using a similar abstraction. In short, SGX uses the EPID protocol by Brickell et al. [160] – a DAA-based RA scheme with revocation support – in order to attest a target enclave. After receiving an attestation challenge, the untrusted host application requests its enclave to produce an attestation. Next, SGX uses a quoting enclave, with access to a hardware-bound EPID attestation key, to measure and sign a report comprising properties of target enclave, including its code, version and ID. The final quote is transmitted to the challenger over a secure session based on the Sigma protocol [161], as used in Internet Key Exchange (IKE)v1 and v2. Here, ECDH is used for the actual key exchange, while ECDSA is used for authentication. The verifier can inspect the contents of the quote for assessing the enclave's operating state, i.e. that the correct version enclave is executing, and may use the Intel Attestation Authority to verify the quote's authenticity, which possesses public portions of the hardware-bound EPID keys. Evidently, this process is controlled stringently by Intel, which functions as a trusted third-party for CPU key provisioning. Comparatively, the GlobalPlatform TEE specifications are yet to standardise a remote attestation mechanism.

Software-based Static Attestation

Both Intel SGX and TPM-based attestation operate from a hardware-based root-of-trust. To eliminate the need for additional hardware, attempts been made to provide software-based static attestation; we now briefly review some notable schemes to this end. Shaneck et al. [162] propose a mechanism for detecting the unauthorised tampering of device memory contents on embedded systems. Here, the verifier constructs an attestation routine that randomly measures the target device's static memory, which is sent to and executed by the device. A checksum of the measurements is then returned to the verifier, who compares it to a known value computed beforehand. To increase the difficulty of attacks that

dynamically return tampered memory addresses to their expected (untampered) values, the attestation routine is obfuscated and a timeout is used by the verifier, after which any responses are rejected. Seshadri et al. [163] propose SWATT for software-based static attestation using timings, which employs a similar approach in measuring program memory contents against a value known to the verifier. In this work, the verifier sends a randomly-generated challenge to the device, which uses *pseudo-random memory traversal* to measure its memory contents. Here, the verifier's challenge seeds a pseudo-random number generator (PRNG) to generate the target memory addresses. A checksum is then iteratively constructed from the memory contents of these addresses on the device, which is subsequently returned to the verifier who possesses a checksum of the expected value measured *a priori*. Like [162], the time taken to measure and return the checksum to the verifier is used as a countermeasure against attempts to alter tampered memory contents back to their expected state. Using time as a side-channel to perform attestation is utilised similarly by the Pioneer system by Seshadri et al. [164], which studies its feasiblity on complex CPUs, and presents an implementation and evaluation on an Intel Pentium IV Xeon CPU.

The above approaches, however, present challenges with respect to security and reliability [158], [159], [165]. Castelluccia et al. [165] demonstrate the susceptibility of memory-measuring schemes to return-oriented programming (ROP) and compression attacks. Furthermore, the authors conclude that *"time-based attestation schemes are very difficult, if not impossible, to design correctly"* due to the challenge of selecting secure and reliable timing thresholds across differing platform configurations [165]. Additionally, they are generally limited to 'one-hop' networks, and are not resilient to impersonation attacks from adversaries who control two identical devices; in this case, an authenticated channel is necessary, such as a secure physical connection [158], [159].

Hybrid Attestation

Hybrid attestation schemes have also been explored in the literature that use an amalgam of the aforementioned hardware- and software-based attestation mechanisms. In this vein, El Defrawy et al. [166] propose the SMART architecture that introduces small hardware changes to a micro-controller unit (MCU) acting as the prover. These changes provide read-only and guaranteed execution of attestation code in ROM, and secure key storage. At the beginning of SMART, the verifier transmits parameters to the proving device, including the memory boundary to attest (between a and b) and a random nonce to address replay attacks. Next, the ROM-resident code measures an HMAC of the contents of the memory region $[a, b]$ under a key in the MCU's secure storage, which is available only to the ROM attestation code. The resulting HMAC measurement is returned

to the verifier, who verifies it by recomputing the measurement under the same key and parameters, i.e. memory boundaries.

Koeberl et al. [167] propose TrustLite – a generic hardware architecture for software isolation on embedded devices. The proposal introduces a new hardware unit aboard the SoC, known as the Execution-Aware Memory Protection Unit (EA-MPU), for protecting security software modules known as 'trustlets'. Upon device initialisation, a software-based Secure Loader is used to load the desired trustlets and their data into on-chip memory; it also programs a Memory Protection Unit (MPU) to protect trustlet memory regions and its own code and data regions from unauthorised accesses. The MPU is akin to a standard MMU, but without support for virtual memory and paging; the MPU is augmented to provide 'execution awareness' (hence EA-MPU), which also accounts for the target and origin of memory access requests. Ultimately, the EA-MPU aims to ensure that a particular trustlet's data can only be accessed by the code of that trustlet, which can be used to implement a SMART-like attestation mechanism in [166]. Notably, unlike SMART, TrustLite can execute multiple trustlets concurrently and provide updates. Related to this, we note that the TyTAN proposal by Brasser et al. [168] utilises an EA-MPU for similar ends and provides additional features over TrustLite, such as secure interprocess communication, dynamic loading and real-time scheduling; the proposal is implemented and evaluated on an Intel Siskiyou Peak embedded platform.

Dynamic Attestation

Dynamic attestation techniques focus on the shortcomings of static attestation schemes in the detection of run-time attacks that hijack application data and control flow, e.g. return-oriented programming (ROP) attacks. As such, the dynamic attestation paradigm concentrates on producing attestation values at run-time, rather than being measured statically. An example scheme, C-FLAT, is proposed by Abera et al. [67] for remotely attesting the target device based on application control flow. Firstly, the vendor creates an application control flow graph (CFG) that is distributed to the verifier and proving device(s). During attestation, the verifier generates a challenge, i, and transmits it to the prover; the prover executes the application with input i, and creates an incremental hash, h, of each traversed CFG node. This is performed within a TEE or other trusted entity that signs h using a device-specific key, akin to a quote, and transmits this signed value to the verifier over a secure channel. C-FLAT is implemented on a Raspberry Pi 3 with ARM TrustZone. The authors note, however, that C-FLAT is limited to simple applications. Applications with large and looping CFGs, or require explicit I/O, are unsuitable candidates for timely attestation responses.

Kil et al. [169] and Davi et al. [170] propose an alternative approach to dynamic attestation based on the use of a trusted monitor to measure and assess application states aboard the device, akin to an intrusion detection system (IDS). This includes approaches based on monitoring system calls and marking and tracking specific data structures loaded into memory using dynamic taint analysis. Upon receiving an attestation challenge, the trusted monitor co-signs attestation response quotes containing the TPM's PCR values if and only if the measured data structures and other measured attributes do not deviate from their expected parameters.

In general, dynamic attestation is affected by several issues: 1), it is non-trivial to identify 'good' states of run-time application objects [169]; 2), the system must access a potentially large number of dynamic objects to assess their state, which significantly increases overhead [170]; 3), the presence of false positives and negatives can undermine trust in the attestation values themselves [169]; and, 4), a CFG-based approach, as per [67], implies storage of expected CFGs on all possible communicating devices. The latter three points are particularly major drawbacks in relation to the constrained embedded devices targeted in this work. Such devices are likely to have limited persistent storage capacities to store CFG graphs, and limited computational and run-time memory for performing dynamic state analysis.

Property-based Attestation

Property-Based Attestation (PBA) was first proposed by Sadeghi and Stüble [171] to address the shortfalls of TPM-like static attestation. The first shortfall occurs when a platform is significantly updated and the TPM PCR values change from its previous configuration. If data is sealed under this previous platform configuration, it becomes inaccessible under the new configuration. Secondly, during attestation, the challenger must theoretically account for a large number of differing PCR configurations of the 'same' software, due to system-level variations in patch penetration and target OS. Thirdly, a forceful content provider could unreasonably demand the presence of certain software (represented in the PCRs), such as a particular word processing package, which has led to concerns over vendor lock-in and infringing users' freedom [172]. To address this, [171] proposes attestation based on 'properties' that conform to the attesting party's security requirements. This way, platform configurations may have slight deviations, but still fulfil that particularly property. A shortfall of PBA is identifying, defining and scoping properties, but it may, for example, include a platform's ability to collect measurements over a trusted I/O path. It may also include its ability to defend against a predetermined set of vulnerabilities, as suggested by Poritz et al. [173].

Sadeghi and Stüble [171] introduce a trusted third party (TTP), the certificate issuer, which issues and maintains credentials that endorse a mapping of platform configurations to known security properties. That is, $S_i \rightarrow P = \{p_0, p_1, \ldots, p_n\}$, where S_i is a configuration mapping to a set of known properties, P. A Trusted Attestation Service (TAS) aboard the proving platform is used to map the binary PCR values to properties, before signing and transmitting the response to the verifier. This abstraction allows a platform to prove its conformance to a trusted platform configuration without revealing the precise software configuration as raw PCR values. PBA has has spawned a range of variations covering revocation, virtualisation, and without TASs using zero-knowledge proofs [174]–[177]. However, its general criticisms include poor scalability from the need to manage potentially an arbitrary number of configuration-property mappings. The reader is referred to [178] for a detailed analysis of PBA and its challenges.

Group and Mutual Attestation

Another paradigm involves attesting groups or a pair of remotely communicating devices in a single protocol run.

Asokan et al. [73] tackle attesting groups of devices organised in a swarm, where a verifier, V, is assured of the integrity of swarm $S = \{s_1, s_2, \ldots, s_n\}$ managed by a swarm operator, OP. S is considered trusted if all of its members are running the expected software configuration. Attestation is conducted using static attestation from the swarm members' software configuration; each swarm member is provisioned with public-private key pairs certified by OP for signing and authenticating response quotes. It is assumed that each swarm member can communicate only with its direct neighbours; S may be dynamic in terms of its membership. The topology of S is not known to OP and V. First, V contacts an arbitrary member $s_i \in S$, which recursively attests its neighbours by way of constructing a minimal spanning tree. The attestation responses of each member are aggregated back to s_i, which appends its attestation response, before replying to V who verifies the responses and the reported configurations.

Gasmi et al. [71] first presented the need for secure and trusted channel protocols (STCPs) where during a protocol's key exchange, e.g. Diffie-Hellman (DH) or Elliptic Curve Diffie Hellman (ECDH), each party transmits its signed platform configuration measurements before establishing the shared secret. The authors discuss modifying the TLS 1.1 (client-server) handshake protocol to provide authentication of each TPM, along with the transfer of PCRs representing the devices' platform configurations. This is performed primarily by replacing the transfer of TLS certificates (for entity authentication) with another certificate containing a long-lived key. This separate certificate is enhanced with Subject Key Attestation Evidence (SKAE) – a TCG proposal that extends X.509 certificates

to indicate that the key was created by a TPM. This certificate contains signed PCR values from the TPM, and stipulates that a certain TCB configuration, i.e. PCR values, must be present before it is released. The key is signed by the TPM's AIK using the `TPM_CertifyKey` operation. The proposal, however, is neither implemented nor evaluated for potential overhead.

Greveler et al. [66] propose a mutual attestation protocol for two TPM-equipped devices. The protocol establishes a shared session key using a 2048-bit Diffie-Hellman (DH) key exchange, along with trust assurances based on the transmission of a signed set of PCR values between each party. These PCR values are signed by each TPM's AIK as quotes; each device is transmitted the other's certified public AIK portion. The DH key exchange and the subsequent verification of signed PCR values – verifying both the signatures and quote contents – establishes a shared session key with the added benefit that both devices are operating to expectations. Like the previous proposal, however, the protocol is not empirically evaluated.

Similarly, Akram et al. [70] propose a mutual attestation protocol for establishing trust in two TPM-enabled nodes in avionics wireless networks. This protocol is also based on a 2048-bit Diffie-Hellman key exchange and the exchange of signed PCR values from each device. The protocol, which is intended to run pre-flight, supports forward secrecy as well as session resumption to compensate for in-flight device power losses. It is implemented on two Raspberry Pi B models over WiFi in *ad hoc* mode; the evaluation uses emulated PCR values to compensate for the absence of actual TPMs in its implementation. The full protocol executes in 4.582 seconds, with approximately three- to four-times overhead versus TLS and SSH respectively.

Akram et al. [179] present a similar STCP using the transmission of signed device-specific authentication values with a DH-based key exchange. This is intended for the secure and trusted application management of smart cards operating under the User-Centric Ownership (UCOM) model. The STCP comprises a service provider (SP) that manages applications, e.g. installation and deletion, upon request from users possessing an authorised smart card. The protocol, known as P-STCP, is used between the SP and the smart card's Trusted Execution Manager (TEM), which performs mutual device authentication using values from Physically Unclonable Functions (PUFs) before authorising the application management operation. The proposal is implemented on a Java Card and laptop acting as the SP with a 1.83GHz CPU and 2GB RAM, and exhibits a round-trip wall-clock time of approximately 3.0 seconds, versus 4.0s and 4.2s for TLS and SSH. Both protocols in [70] and [179] are subjected to formal symbolic verification using CasperFDR [180] to assure correctness under a Dolev-Yao adversarial model. (As a side note, PUFs and even seeding PRNGs [69], [181]

have been suggested as a primitive with which to fingerprint and authenticate particular devices. However, concerns still exist surrounding low PUF entropy and high error rates [182]–[184], and protecting against statistical and machine learning attacks [185]–[187]. These are out-of-scope in this thesis, which focuses upon hardware-assisted *execution* technologies, rather than device fingerprinting measures).

4.3 The Challenge of TEE Intercommunication

TEEs offer potential in sensing-based applications, but the absence of a generic remote attestation solution impedes interoperability. This also leads to the challenge of secure and trusted TEE-to-TEE communication on distinct remotely-located devices with mutual trust assurances. This is particularly pertinent for sensing applications: it is conceivable that differing TEE architectures may wish to communicate with each other directly, e.g. an Intel-based server-side analytics service receiving sensitive health measurements from an ARM-based wearable, without revealing anything to the untrusted worlds on either end-point.

As we seen, a multitude of remote attestation mechanisms exist that allows the bootstrapping of a secure channel through which security-sensitive data can be provisioned to the target environment. The most notable example of this is EPID in Intel SGX [64]; however, it is only a one-way mechanism that allows the attestation of a target *Intel-based* platform given the manufacturing and key infrastructure assumptions in place, which were discussed heavily in Section 2.4.6 of Chapter 2. In other words, EPID does not generalise to chipsets outside the auspices of Intel and, while this may be acceptable for server or more powerful mobile device, it does not reconcile attesting ARM-based SoCs, which are predominant among constrained devices. Ideally, both end-points should undergo trust verification to provide this assurance, i.e. each node remotely attesting the other prior to further communications involving sensitive data, which ought to be performed without trusting intermediate untrusted components, such as either REE.

To our knowledge, this remains unaddressed with remote TEEs. An immediate solution is to perform remote attestation twice: one per party, which may include EPID for Intel SGX, or a TPM-based protocol for past TEEs with TPMs, e.g. Chen et al. [63]. However, the complexity overhead of two protocol executions is excessive in terms of potential network and computational overhead. This is supported by previously discussed existing literature [66], [70], [71] that propose secure and mutually trusted channels in a *single* run. Another drawback of executing multiple attestation protocols is that it implies the maintenance of a suite of protocols for providing the necessary attestation mechanism used by all

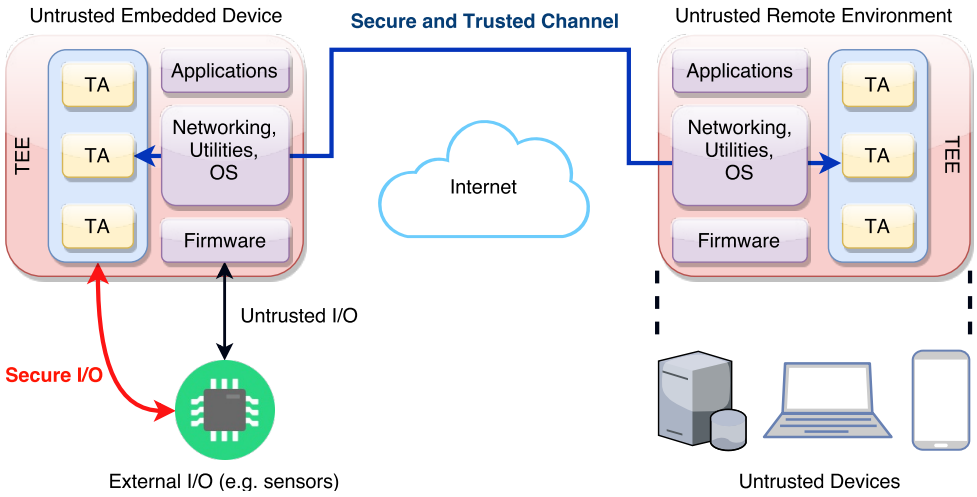


FIGURE 4.1: High-level mutual attestation of two TEE-enabled devices.

possible communicating devices. To this end, we develop a generic and flexible TEE-based mutual attestation mechanism focussing on benefitting embedded sensing devices. In the coming sections, we begin formalising a single protocol that provides intercommunication with bi-directional trust assurances, while accounting for potential interoperability issues.

4.4 Protocol Design

This section outlines the threat model, security goals and assumptions we operate from for designing a protocol for providing secure TEE-to-TEE intercommunication using mutual attestation.

4.4.1 Threat Model

We refer to the high-level architecture in Figure 4.1 in our discussion. At a high-level, our threat model assumes any component between the TEEs to be untrusted, including the medium across which data is transmitted, such as the Internet, a short-range (e.g. Bluetooth), local area, or peer-to-peer network, and the untrusted worlds on both device. More specifically, on the end-devices, we consider user- and kernel-level adversaries aiming to observe or modify protocol message exchanges within the untrusted worlds before they are received by either TEE. This includes, for example, protocol session keys; attestation values; and application-specific data, e.g. credentials and sensor measurements, held by either TEE.

From a network perspective, we consider a Dolev-Yao type adversary operating over any network medium over which the protocol messages are exchanged,

such as Bluetooth, Wi-Fi (802.11), a low-power wide-area network (LPWAN), or otherwise. Specifically, we aim to defend against an adversary that attempts to:

- Observe and modify arbitrary protocol messages transmitted between each end-device.
- Masquerade as either end-point, including the use of man-in-the-middle attacks, the transmission of previously observed messages (replay attacks), and the attempted generation of fraudulent protocol messages.
- Use previously compromised protocol session keys in order to compromise subsequent sessions.
- Deny or repudiate that a protocol instance had occurred.

In this work, we do not consider on-device denial-of-service (DoS) that drop messages within the untrusted world before they reach either TEE. As discussed by Sabt et al. [148], TEEs rely implicitly upon co-operation between the untrusted and trusted world, and adversaries that prevent this co-operation, say, by impeding world switches, would subsequently impede the protocol generally. This on-going, TEE-wide challenge is considered out-of-scope in this work. Additionally, the TEE and its applications are considered to be trusted, and should satisfy the GlobalPlatform TEE Protection Profile [40] discussed previously in Chapter 2. As such, we do not consider threats generally considered outside the protection scope of TEEs, such as advanced physical attacks requiring substantial time and expertise, e.g. laser fault injections; supply chain attacks; and pre-existing, unintentional programming and logic errors within the TEE or TA code that give rise to software vulnerabilities [40], [148]. A description of the GlobalPlatform TEE Protection Profile was given in Section 2.4.7, while the protection scope of TEEs generally was also analysed in Sections 2.5.1, 2.5.2 and 2.6 of Chapter 2. We revisit the limitations of this work later in Section 4.7.3.

4.4.2 Security Goals

To scope this work and ground future discussion, we identify a series of goals for establishing trusted channels between remote TAs, which serve as a minimum, but not exhaustive, baseline that we aim to support:

1. **Mutual Key Establishment**: A shared secret key is established for communication between the two entities.
2. **Mutual Entity Authentication**: Each entity shall authenticate the other's identity in order to counter masquerading.

3. **Mutual Non-Repudiation**: Neither entity may deny that a protocol instance had occurred.

4. **Trust Assurance**: Entities shall be able to remotely attest the application and platform integrity of the other.

5. **Mutual Trust Verification**: Both entities shall successfully attest the state of the other within the protocol and before creation of the secure channel.

6. **Freshness**: Session keys and their derivatives must be fresh in order to counter replay attacks.

7. **Forward Secrecy**: The compromise of a particular session key should not affect past or subsequent protocol runs.

8. **Denial of Service (DoS) Prevention**: Resource allocation shall be minimised at both end-points to prevent DoS conditions from arising.

In this work, the following goals are also established from a functional perspective:

1. **Avoid additional trust hardware**: To maximise compatibility with constrained embedded devices, the protocol shall avoid mandating additional security hardware, namely SEs or discrete TPMs.

2. **TEE agnosticism**: The protocol shall be generic and avoid manufacturing and verification processes specific to any particular TEE.

3. **Session resumption**: The ability to resume a session securely without reconducting the protocol.

4.4.3 Trust Measurement and Attestation – Operational Perspective

We now describe a basic approach for attesting TAs and platform configurations using a TEE. As discussed previously, TEEs and TPMs typically follow the quoting abstraction in which platform integrity information is measured and signed by a trusted entity for assuring a verifier of the platform's trustworthiness. This takes the form of PCR values with TPMs, but may take the form of an application binary hash (as used in Intel SGX), or both. In SGX, a separate trusted application – the quoting enclave – functions as a verifier and signatory of the requested enclave's state. This comprises a signed hash of the enclave's code and related data, e.g. version number. The quote, which is signed under a CPU-accessible key provisioned in single-write ROM, is transmitted back to the challenger for verification. The challenger then uses the Intel Attestation Service (IAS) for verifying the authenticity of the signed quote, i.e. that it was signed

using a genuine Intel SGX chipset. The IAS maintains a repository of public keys to achieve this, and functions as a trusted third party.

This abstraction is not specified in the GlobalPlatform specifications, but such an architecture can be adopted generically for the purposes of this work. This does not necessitate hardware-bound key-pairs in each device: GP TEEs are capable of hosting data persistently in secure storage, whether as part of the TEE OS itself, as a separate 'keymaster' TA, simliar to the Android's TEE-based Keystore[1]. It could also be a Secure Element (SE) for hardware tamper-resistance using the GlobalPlatform TEE Secure Element API [41]. The exact implementation ultimately depends on the OEM and its security, cost and physical space requirements. Nevertheless, the primary requirement is that, akin to a TPM's Attestation Identity Key (AIK), it is certified by a trusted third party to the verification authority, which may even be the OEM itself, and that its private components are never revealed outside the TEEs.

Maintaining the security and trust of a TEE and its TAs differs according to implementation. OP-TEE verifies a TA's binary signature when it is loaded into memory such that only verified TAs are launched; TAs are signed by the developer and provisioned into the TEE before deployment. The fact that the device was booted and the TA was launched implies some notion of trust, but this does not extend to convincing a verifier, V, that the target TA, or even TEE OS, is running the expected version. This could be a consequence of patch delivery failure, which is seen as a major issue affecting IoT security generally, as discussed in Chapter 1. For stronger assurances, proprietary TEE remote attestation mechanisms have emerged, but the specifics remain largely undocumented publicly. Samsung KNOX is known to transmit boot measurements as part of its attestation response, which is signed by a device-unique attestation key certified by Samsung [188]. Attestations that generate both TA and platform measurements, i.e. a static binary hash *and* boot measurements in PCR-like values, could be used to provide V with heightened evidence of platform integrity.

In this vein, we suggest the use of a separate TEE OS component – a Trusted Measurer (TM) verified in the TEE's secure boot sequence – to measure and produce quotes based on the known state information of the platform and target TA. This includes the TA's binary hash, any known personalisation data, UUID (Universally Unique IDentifier), and version number. TM subsequently signs the resulting quote using the private portion of a certified attestation key-pair provisioned by the operator before deployment. This could be in software using native TEE secure storage or, at the cost of additional hardware and Functional

[1] Android Open Source Project – Hardware-backed Keystore: https://source.android.com/security/keystore/

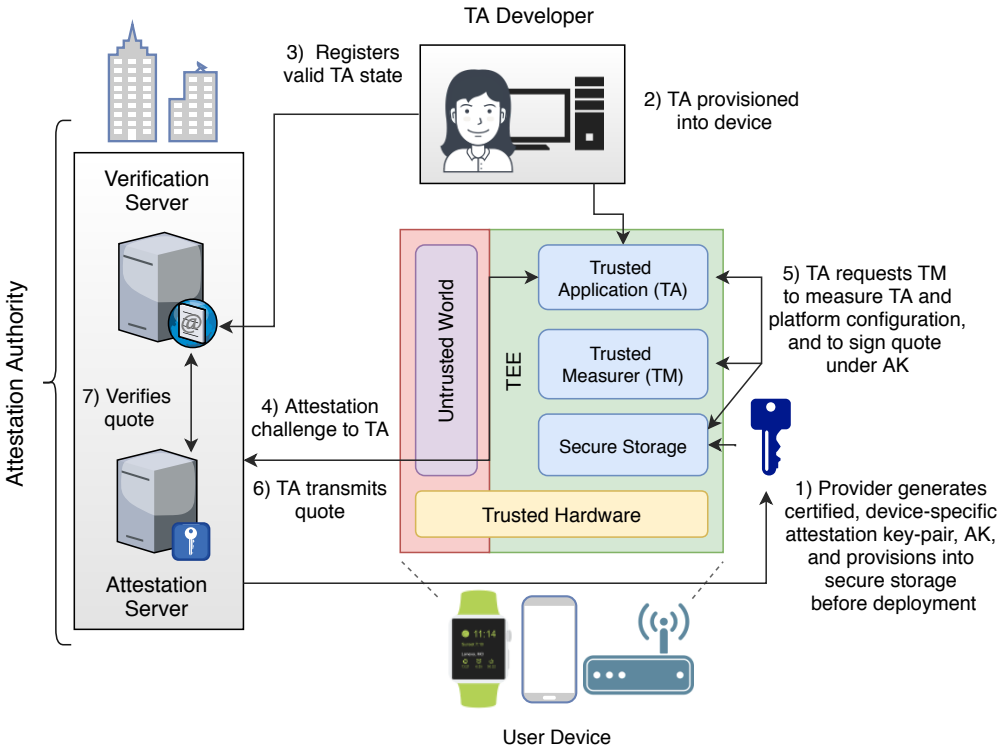


FIGURE 4.2: Generic TEE remote attestation architecture.

Requirement 1, within an SE accessible only to the TEE to provide greater hardware tamper resistance, as suggested by the GP specifications [41]. (We note that TEE-based TPM implementations – also known as firmware-based TPMs, or fTPMs, as described by Raj et al. [189] – could serve as a helpful reference to an OEM in implementing a potential TM candidate. Intel SGX already contains a TM in the form of the quoting enclave, or QE). Lastly, V verifies the received quote: its signature and the contained trust measurements. The OEM must be aware that, if one of the communicating TEEs uses Intel SGX, it is difficult to avoid using its remote attestation infrastructure for the quote verification portion [64]. Figure 4.2 depicts a generic attestation process reflected in this discussion and described as follows:

1. An Attestation Authority, AA generates and certifies a device-specific attestation key-pair, AK, using a desired public-key algorithm, e.g. 3072-bit RSA or 512-bit ECDSA, which is securely stored in the TEE and accessible to TM.

2. The OEM procures and provisions TAs into the TEE from a trusted developer.

3. The initial expected TA state and platform configuration is registered with AA.

4. AA issues a remote attestation request to the target TA to produce a quote representing its operating state. This may, for example, be prior to accessing any restricted assets, or before the transfer of sensitive data that informs some other activity, e.g. sensing data.

5. TM creates the attestation quote, Q, comprising boot measurements and the hashed TA binary currently in use, and signs it with the private portion of AK provisioned into TEE secure storage.

6. TM returns the signed Q to the TA, which transmits it to V over a secure channel.

7. AA verifies the signature of Q using the public component of AK. This step also includes a look-up with a trusted verification service that inspects that the contents of Q, i.e. TA in use and boot measurements, conform to expected parameters registered *a priori*.

4.5 Proposed Protocols

This section now presents the proposed protocols for creating a secure and trusted channel between remotely located TAs. Based on the goals established in Section 4.4, we present two protocols under two differing deployment scenarios: *bi-directional* trust, in which both communicating devices contain TEEs, and *uni-directional* trust, in which only one device contains a TEE. The latter scenario is likely to arise for legacy devices that existed before the recent developments in TEEs, but still wish to communicate in a secure and trusted with a remote entity that *does* host a TEE. In this case, the trust guarantees in Section 4.4 apply only to the TEE-enabled device. These two protocols, which are described and analysed in detail throughout this section, are presented and listed as the Bi-directional Trust Protocol (BTP) in Protocol 1 and the Uni-directional Trust Protocol (UTP) in Protocol 2. In design, BTP is influenced by Greveler et al. [66] and Akram et al. [70] protocols that establish mutually trusted channels using TPMs over a Diffie-Hellman based key exchange. Next, this section discusses the pre- and post-protocol procedures, provides a high-level descriptive analysis in Section 4.5.3, and discusses formal symbolic verification in Section 4.5.4.

4.5.1 Setup Assumptions

For BTP, it is assumed that each TA on the remotely located devices possess means for verifying attestation quotes received from the other, as stated in Section 4.4.3. Moreover, each TEE also contains TM for generating and signing quotes from application and platform state data upon request. For UTP, only the TEE-enabled

Protocol 1 Proposed Bi-directional Trust Protocol (BTP)

1: $TA_{SD} \rightarrow TA_{RE} : ID_{SD} \,||\, ID_{RE} \,||\, n_{SD} \,||\, g^{SD} \,||\, AR_{RE} \,||\, S_{cookie}$
$S_{cookie} = H(g^{SD} \,||\, n_{SD} \,||\, ID_{SD} \,||\, ID_{RE})$
2: $TA_{RE} \rightarrow TA_{SD} : ID_{RE} \,||\, ID_{SD} \,||\, n_{RE} \,||\, g^{RE} \,||\, [\sigma_{TA_{RE}}(X_{TA_{RE}}) \,||\, \sigma_{TA_{RE}}(V_{TA_{RE}})]^{K_E}_{K_{MAC}} \,||\, AR_{SD}$
$X_{TA_{RE}} = H(ID_{SD} \,||\, ID_{RE} \,||\, g^{RE} \,||\, g^{SD} \,||\, n_{RE} \,||\, n_{SD})$
$V_{TA_{RE}} = Q_{TA_{RE}} \,||\, n_{RE} \,||\, n_{SD}$
3: $TA_{SD} \rightarrow TA_{RE} : [\sigma_{TA_{SD}}(X_{TA_{SD}}) \,||\, \sigma_{TA_{SD}}(V_{TA_{SD}})]^{K_E}_{K_{MAC}} \,||\, S_{cookie}$
$X_{TA_{SD}} = H(ID_{SD} \,||\, ID_{RE} \,||\, g^{RE} \,||\, g^{SD} \,||\, n_{RE} \,||\, n_{SD})$
$V_{TA_{SD}} = Q_{TA_{SD}} \,||\, n_{RE} \,||\, n_{SD}$

Protocol 2 Proposed Uni-directional Trust Protocol (UTP)

1: $UA_{SD} \rightarrow TA_{RE} : ID_{SD} \,||\, ID_{RE} \,||\, n_{SD} \,||\, g^{SD} \,||\, AR_{RE} \,||\, S_{cookie}$
$S_{cookie} = H(g^{SD} \,||\, n_{SD} \,||\, ID_{SD} \,||\, ID_{RE})$
2: $TA_{RE} \rightarrow UA_{SD} : ID_{RE} \,||\, ID_{SD} \,||\, n_{RE} \,||\, g^{RE} \,||\, [\sigma_{TA_{RE}}(X_{TA_{RE}}) \,||\, \sigma_{TA_{RE}}(V_{TA_{RE}})]^{K_E}_{K_{MAC}}$
$X_{TA_{RE}} = H(ID_{SD} \,||\, ID_{RE} \,||\, g^{RE} \,||\, g^{SD} \,||\, n_{RE} \,||\, n_{SD})$
$V_{TA_{RE}} = Q_{TA_{RE}} \,||\, n_{RE} \,||\, n_{SD}$
3: $UA_{SD} \rightarrow TA_{RE} : [\sigma_{UA_{SD}}(X_{UA_{SD}})]^{K_E}_{K_{MAC}} \,||\, S_{cookie}$
$X_{UA_{SD}} = H(ID_{SD} \,||\, ID_{RE} \,||\, g^{RE} \,||\, g^{SD} \,||\, n_{RE} \,||\, n_{SD})$

end-point must respond to attestation requests and provide quotes. It is also assumed that the TEE has a secure, preferably hardware-based, means of key generation and derivation, and random number generation.

4.5.2 Post-protocol

Trusted sensing necessitates secure session resumption to alleviate the overhead of re-executing the protocol. This is useful for constrained devices that transmit sensitive data frequently but not necessarily constantly, such as a home monitoring system reporting occupancy data during the day. Resumption is not cost-free, however; the longer a session exists, the greater the impact of one particular compromised session. As such, sessions should be re-established at regular intervals to minimise this risk, which ought to constitute part of the service provider's risk model.

4.5.3 Protocol Analysis

Analyses of the protocols are now provided, comprising a message-by-message descriptive analysis and the use of formal symbolic verification with Scyther [29].

TABLE 4.1: Protocol notation.

Notation	Description
SD	Sensing device.
RE	Remote entity.
TA_X	TEE trusted application on device X.
UA_X	Untrusted application on device X.
n_X	Random number (nonce) generated by X.
$H(D)$	A secure one-way hash function, H, applied to D.
$X \rightarrow Y$	Message transmission from X to Y.
ID_X	Identity of X.
$A \mid\mid B$	Concatenation of A and B.
g^X	Diffie-Hellman exponentiation of X.
AR_X	Attestation request on target entity X.
Q_X	Quote from TEE X.
$\sigma(A)^P_K$	Signature of A with private-public key-pair (K, P).
$[D]^{K_e}_{K_{MAC}}$	Message D is encrypted using session encryption and MAC keys K_e and K_{MAC} respectively, both generated during the protocol.

Message Description

1. The sensing device, which is trusted in BTP (TA_{SD}) or untrusted in UTP (UA_{SD}), computes and transmits its Diffie-Hellman (DH) exponential g^{SD}, along with an attestation request AR_{RE} to instruct the remote entity to provide its quote. S_{cookie}, comprising a hash of the nonce, IDs and DH exponentiation may be used as a session resumption cookie, *a la* Akram et al. [70]. This satisfies the session resumption functional requirement (F3) in Section 4.4.

2. Next, TA_{RE} transmits its DH exponentiation, g^{RE}; nonce, n_{RE}; and signatures of both DH exponentiations and its quote, $Q_{TA_{RE}}$. The quote and both DH exponentiations (now known by TA_{RE} at this point), IDs and nonces are signed using its (certified) device-specific attestation key-pair, and then encrypted under an encrypt-then-MAC construction under keys derived from the shared session key. The transmission of this message satisfies requirement S1 (mutual key establishment). For BTP, TA_{RE} now requests an attestation from the sensing device's TA, TA_{SD}. By design, mutual transmission of attestation requests is not used in UTP.

3. Finally, the sensing device computes its shared DH session key. It then acknowledges with the nonces, DH exponentiations and IDs, which it signs using its certified device-specific attestation key-pair and subsequently encrypted under encrypt-then-MAC. SD also transmits the session cookie for resumption. This satisfies S2 (mutual entity authentication), S3 (mutual non-repudiation), S6 (freshness) and S7 (forward secrecy, by generation of

ephemeral DH key). Lastly, the sensing device responds with its respective quote, thus satisfying S5 (bi-directional trust). For UTP, no responding quote is generated and $\sigma_{TA_{SD}}(V_{TA_{SD}})$ is not transmitted. S8 (DoS prevention) is largely ensured by avoiding either party to perform overly-demanding operations: two signatures, a single attestation, DH exponentiation, nonce generation and shared-key generation are required by each party. Evidently, this is biased in favour of SD during UTP, which does not have to participate in producing remote attestation responses; the service provider should be aware of this when using UTP. A performance evaluation of the protocols is presented later for an indication regarding this in Section 4.7.

4.5.4 Formal Verification

We make use of automated protocol verification to evaluate the correctness of the proposed protocols using the Scyther tool by Cremers [29]. Verification tools are motivated by the difficulty of verifying protocol correctness manually, particularly those involving a many messages and communicating parties [190]. In the following sub-sections, symbolic protocol analysis is briefly introduced before discussing the analysis of the proposed protocols using Scyther. Following this, we stress the limitations of symbolic analysis tools that arise predominantly when implementing the protocols in practice. We do not intended to provide a comprehensive review of symbolic analysis – for this, the reader is referred to work by Cremers [29], Lafourcade and Puys [190], Scott [191], and Blanchet [192], [193] – but, rather, we aim to identify its salient features and limitations.

Symbolic Analysis

Tools such as Scyther [29], Tamarin [194], AVISPA [195], CasperFDR [180] and ProVerif [196] operate in the *symbolic* model in which protocols comprise messages containing symbols that represent black-box cryptographic primitives [190]. These primitives – hash functions, asymmetric and symmetric encryption, digital signature schemes, and so on – are treated as abstract entities with certain properties without being concerned about their implementation. For example, one may model symmetric encryption in Tamarin by defining its functionality with decryption (`sdec`) and encryption (`senc`) functions as such: `sdec(senc(m, k), k) = m`, where `m` is message and `k` is a shared secret key, without defining how it is actually realised, say, using AES [191]. Scyther provides a set of built-in definitions of common cryptographic primitives, such as hash functions, digital signatures, and symmetric encryption; an extensive list is found in [29].

This leads to the important assumption that these primitives are assumed to exhibit *perfect cryptography*; that is, cryptographic operations can only be performed with knowledge of the required key [191]. This assumption is revisited later when discussing the limitations of verification tools. This assumption is used in conjunction with a Dolev-Yao [197] adversary where, in short, the attacker is assumed to control the network with the ability to eavesdrop, forge, replay, modify and drop transmitted protocol messages. Symbolic verification tools operate by exploring whether the desired security properties hold against all possible behaviours of this adversary [192], with the tools providing feature differences relating to execution time, memory consumption, automation of desired security properties, expressiveness, and termination guarantees. A detailed comparison of symbolic verification tools is provided by Lafourcade and Puys [190] and Cremers et al. [198].

Automated Symbolic Analysis of the Proposed Protocols

The Scyther verification tool by Cremers [29] is employed to verify the correctness of both protocols. A protocol is first specified using Scyther's high-level Security Protocol Description Language (SPDL) that defines the communicating parties *(roles)*; the messages represented in terms of built-in primitive types, such as nonces, hash functions, keys and signatures, and user-defined types; and the desired security properties to test *(claims)*. After the definition of the roles, protocol messages and claims, Scyther analyses whether the specification holds against all possible behaviours of a Dolev-Yao adversary, known as *traces* [199], under the perfect cryptography assumption. Rather than formally specifying security properties as lemmata, as required in some tools like Tamarin [194] and ProVerif [192], Scyther supports a number of built-in security properties with which to form claims, namely: *aliveness*, *secrecy*, *non-injective agreement*, *non-injective synchronisation*, and *reachability*. These properties are formalised by Lowe [200], Cremers et al. [201], Abadi and Blanchet [202], and Cremers [29], and are described below.

Lowe [200] defines *aliveness* as the weakest form of authentication notion that guarantees that the communicating party has previously participated in the protocol: *"We say that a protocol guarantees to an initiator A aliveness of another agent B if, whenever A (acting as initiator) completes a run of the protocol, apparently with responder B, then B has previously been running the protocol"* [200]. A protocol can fail the aliveness property in a *mirror attack* whereby an adversary simply reflects messages back to the initiator without actually participating in the protocol.

Non-injective agreement is a stronger form of authentication provided by Lowe [200], where the communicating entities agree on a set of data items, such as keys and other variables: *"We say that a protocol guarantees to an initiator A*

non-injective agreement with a responder B on a set of data items ds (where ds is a set of free variables appearing in the protocol description) iff, whenever A (acting as initiator) completes a run of the protocol, apparently with responder B, then B has previously been running the protocol, apparently with A, and B was acting as responder in his run, and the two agents agreed on the data values corresponding to all the variables in ds" [200].

Non-injective synchronisation is introduced by Cremers et al. [201] to address the issue of replay attacks. Informally, it states that a message received by the communicating parties is one unique to that protocol; the use of *nonces* is typically employed by security protocols to realise this property. The definition states: *"Initiator I considers a protocol injectively synchronising if the protocol synchronises and each run of I corresponds to a unique run with [responder] R"* [201]. This relies on the following definition of *synchronisation*: *"Initiator I considers a protocol synchronising whenever I as initiator completes a run of the protocol with responder R, then R as responder has been running the protocol with I. Moreover, all messages are received exactly as they were sent, in the order described by protocol"* [201].

Secrecy is the property of preserving the confidentiality of secret information, e.g. session keys, during message exchanges. It is defined as: *"Protocol P preserves the secrecy of data M if P never publishes M, or anything that would permit the computation of M, even in interaction with an adversary Q. Equivalently, a protocol P preserves the secrecy of data M if P in parallel with an adversary Q will never output M on a public channel"* [202].

Reachability is the simplest property tested by Scyther that determines whether there exists some trace involving the tested entity (*"It is true iff there exists a trace in which this claim occurs"* [29]). This is useful in verifying whether at least one message is transmitted to a given party in order to detect specification errors.

In Scyther, a claim holds if no traces are found that violates its stated security property. We note that Scyther lacks some features provided by other tools, such as its inability to model more advanced cryptographic primitives, like blind signatures, and in defining custom security properties and adversaries. However, it is shown to outperform most other tools with respect to run-time performance [190], [198], and has been used in verifying IKEv1 and IKEv2 [203], and the ISO/IEC 9798 [204] and ISO/IEC 11770 protocol families [205]. We analysed all protocols testing for the secrecy of quotes from both parties, e.g. (`Secret, qre`); aliveness (`Alive`); entity authentication, i.e. non-injective agreement (`Niagree`) and non-injective synchronisation (`Nisynch`); session key secrecy (`SKR, K`), which is a synonym of the `Secrecy` claim for keys; and the reachability of all communicating entities (`TARE/TASD/UASD, Reachable`). The full Scyther specification scripts for the bi-directional (BTP) and uni-directional (UTP) protocols can be found in Appendices A.1 and A.2 respectively. Scyther found no attacks on either protocol; however, this must be considered alongside its limitations, which are

discussed below.

Limitations of Protocol Verification Tools

At this point, the limitations of symbolic verification tools must be stressed. Notably, the 'perfect cryptography' assumption does not readily translate into reality: cryptographic primitives are regularly subverted by weaknesses in their implementation, whether due to programming errors, side-channels and other vulnerabilities outside the scope of such tools [191], [196]. Moreover, secure key management, secure sources of randomness, and adequate parameter selection for such primitives are not covered. We discuss some of these limitations below:

- *Cryptographic primitive and parameter selection*. The use of 'perfect' primitives abstracts from the importance of using *secure* implementations and associated parameters. For example, the use of insufficiently large key sizes is not covered, nor is the improper use of cryptographic primitives, such as initialisation vector (IV) reuse in AES; insecure modes of operation, e.g. AES in ECB mode; and the selection of weaker primitives, e.g. DES for symmetric encryption.

- *Protocol implementation errors*. Even assuming the accurate specification of a particular protocol, verification tools do not cover its actual implementation. For example, programming errors may improperly verify the validity of nonces, hence giving rise to replay attacks, or incorrectly verifying digital signatures and message authentication codes, thus enabling integrity attacks.

- *Side-channel and physical attacks*. Verification tools are also unable to model side-channels, such as power analysis and timing attacks against cryptographic primitives [196]. This is also applicable to physical attacks, e.g. bus probing and fault analysis, conducted by an attacker on the device.

- *Secure randomness*. Protocol verification tools do not consider the implementation of secure sources of randomness with which to generate unpredictable numbers used in security protocols, such as nonces and IVs; this is simply assumed under the perfect cryptography assumption.

- *Software security*. Verification tools do not model system-level security controls, such as protection rings, sandboxing measures, or, indeed, the use of TEEs or SEs to protect protocol material against on-device adversaries.

- *Key management challenges*. Verification tools assume the existence of secure key management practices; for example, the secure generation, distribution and provisioning of keys, and their revocation. Consequently, the tools

do not cover the attempted use of revoked keys, insider or supply chain attacks that lead to the use of stolen keys, and the storage of keys at rest on insecure media.

For these reasons, caution is urged in the interpretation when a verification tool states that no attacks were discovered on a given protocol. However, such tools are useful in detecting and remedying well-known flaws in security protocols, such as replay attacks, and gaining some assurances regarding the secrecy of variables as they are exchanged in the protocol at a high level [191].

4.6 Implementation

Both BTP and UTP were implemented using OP-TEE [152] – an open-source, GP-compliant TEE based on ARM TrustZone, which was utilised and described in Chapter 3. OP-TEE executes two execution environments simultaneously: the untrusted world OP-TEE Client, containing Debian (Linux) that implements the GP TEE Client API [77]; and OP-TEE OS, running the TEE kernel that implements the GP TEE Internal API [78]. Two applications were developed that executed in the trusted and untrusted worlds independently, with communications mediated by the OP-TEE secure monitor. The reader is referred back to Section 2.4.7 and Figure 2.14 in Chapter 2 for a description of the GP TEE.

Both protocols were implemented using the cryptographic operations defined in the GP Internal API, all of which occurred in the trusted world application. The untrusted world application was used for handling TCP/IP sockets and transferring protocol messages into the corresponding TA. In OP-TEE, cryptographic operations are implemented by the LibTomCrypt[2] library and, where available, ARM Cryptography Extensions for providing instruction-level AES and SHA [206]. Based on the NIST key size recommendations [207], we used 2048-bit Diffie-Hellman, 128-bit AES in CBC mode, and SHA-256 (including for HMACs). Also based on those recommendations, we used the Elliptic Curve Digital Signature Algorithm (ECDSA) for quote signing and verification using the NIST `secp256r1` curve. At present, only the NIST curves are specified in the GlobalPlatform TEE Internal API specifications [78], and ECC support in general is not mandated for GlobalPlatform compliance. Each TA was preloaded with the known signature of the remote TA and 256-bit ECDSA public key (also `secp256r1`). We emulated a software TM as another TA that, upon request, returned an ECDSA-signed hash of the binary and mock platform data, which constituted the quote. In practice, as per Section 4.4.3, this key would be certified and provisioned by the developer, and the TM would, ideally, be verified as part of the device's authenticated boot process.

[2] LibTomCrypt: `https://github.com/libtom/libtomcrypt`

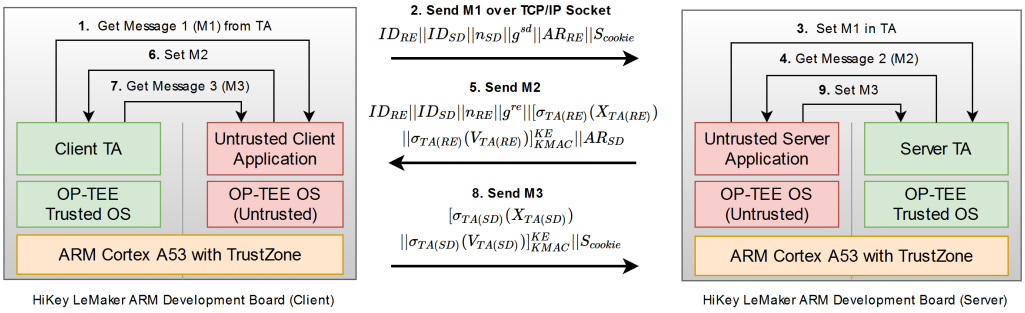


FIGURE 4.3: High-level implementation information flow for BTP.

We used two HiKey LeMaker ARM development boards, hosting an HiSilicon Kirin 620 SoC with an eight-core ARM Cortex-A53 processor (1.2GHz), 2GB DDR3 RAM and TrustZone extensions. Such specifications are typical of modern SBCs; the Raspberry Pi 3, for example, also uses an ARM Cortex-A53 with 1GB RAM. Notably, our board is supported and reasonably documented by the OP-TEE project; however, it has not been CC certified under the GlobalPlatform TEE PP. We are currently unaware of any publicly-available, CC-certified TEE platforms with open-access for the installation and testing of TAs. The protocol was implemented in C – the only development language supported fully by OP-TEE at the time of writing (April 2018) – and cross-compiled for ARM 64-bit platforms using the GNU C Compiler (GCC) with the default optimisation flags (`-O0`). World context switches were kept to two per message for setting and retrieving messages into and from the TA respectively. This is illustrated in Figure 4.3.

4.7 Evaluation

We measured 1,000 runs of the BTP and UTP protocols, which were benchmarked against the OpenSSL 1.0.1 implementation of TLS v1.2 and SSH (OpenSSH 6.7) executing in the untrusted world. For TLS, the modes `DHE-RSA-AES128-SHA256`, `DHE-DSS-AES128-SHA256` and `ECDHE-ECDSA-AES128-SHA256` were used. This selection stems from the similarities with our proposal: the use of ephemeral Diffie-Hellman (DHE), 256-bit ECDSA and 128-bit AES, in addition to RSA and DSS signatures for comparison. These were measured using `openssl s_client`. For SSH, 128-bit AES in CBC mode was used with SHA-256 HMACs, as per our implementation. This was tested with RSA-, ECDSA- and DSA-based key pairs; the `ssh` utility was used with default settings except for specifying AES and HMAC modes, and the desired key. The Diffie-Hellman group sizes were set at 2048 bits (with pre-generated moduli) in accordance with our implementation, and 256-bit ECDSA for the relevant SSH and TLS modes. 3072-bit RSA was also used, in line with the NIST recommendations [207]. 3072-bit DSS was used for

TABLE 4.2: Mean client and server round-trip wall-clock times (in milliseconds; S.D. in brackets).

Message	BTP			UTP		
	Client (TA_{SD})	Server (TA_{RE})	Round-trip	Client (UA_{SD})	Server (TA_{RE})	Round-trip
M1: $TA/UA_{SD} \rightarrow TA_{RE}$	**302.7** (4.1)	**21.5** (2.9)	**347.1** (3.5)	**305.9** (2.5)	**20.8** (3.0)	**350.2** (2.6)
M2: $TA_{RE} \rightarrow TA/UA_{SD}$	**346.9** (6.6)	**682.6** (6.9)	**1192.0** (7.2)	**350.3** (6.1)	**689.4** (7.3)	**1189.2** (10.3)
M3: $TA/UA_{SD} \rightarrow TA_{RE}$	**61.6** (4.8)	**39.7** (4.1)	**117.2** (4.6)	**39.2** (3.9)	**26.7** (3.1)	**73.8** (3.4)

TABLE 4.3: Full protocol mean wall-clock times in milliseconds.

Proposal		TLS			SSH		
BTP	UTP	DHE+RSA	DHE+DSS	ECDHE+ECDSA	RSA	DSA	ECDSA
1692.3 (11.0)	**1618.2** (9.6)	**410.5** (3.9)	**374.3** (4.1)	**102.5** (2.2)	**535.1** (13.6)	**446.1** (11.8)	**453.8** (15.0)

TLS, but only 1024-bit DSA was used with OpenSSH due to its deprecation; however, we still include it for comparison purposes. The protocols were measured using wall-clock time via the `time` UNIX utility with microsecond precision. This was assisted by a shell script in the untrusted world for initialising the target IP address and port number, and for result logging.

Message-specific times were measured for BTP and UTP using the `<time.h>` C library provided by OP-TEE. For these, the mean wall-clock time was calculated from 1,000 measurements per message. This includes client and server times to generate and verify each message, and the round-trip time to account for network latency – comprising message generation, transmission, verification and acknowledgement. For all experiments, the boards were connected over IEEE 802.3 Ethernet to a router with no other attached devices in order to minimise network overhead. The boards were configured in a client-server architecture: one acting as TA_{RE} and the other as TA_{SD}. Both boards were also connected to a central laptop (Lenovo T460s) using UART-to-USB for debugging and error identification. Figure 4.4 depicts our test-bed environment.

We recognise that using Ethernet over a router is only one particular network medium. A multitude of protocols could be used in a given IoT deployment, such as IEEE 802.15.4-based mediums, e.g. ZigBee[3] and Thread[4]; Z-Wave[5], which is common in home automation systems operating on the 800-900MHz spectrum; cellular networks, e.g. UMTS/HSPA (3G) and LTE (4G); Low-Power Wide-Areas Network (LPWAN) protocols with substantially reduced bit rates (up to 50kbit/s), e.g. LoRa[6] and SigFox[7]; IEEE 802.11 Wi-Fi networks; and Bluetooth, including Bluetooth LE. Indeed, given their relatively early inception, some of these protocols remain direct technological competitors, such as LoRa and SigFox

[3]ZigBee Alliance: https://www.zigbee.org
[4]Thread: https://www.threadgroup.org
[5]Z-Wave: https://www.z-wave.com
[6]LoRa: https://www.lora-alliance.org/
[7]SigFox: https://www.sigfox.com/en

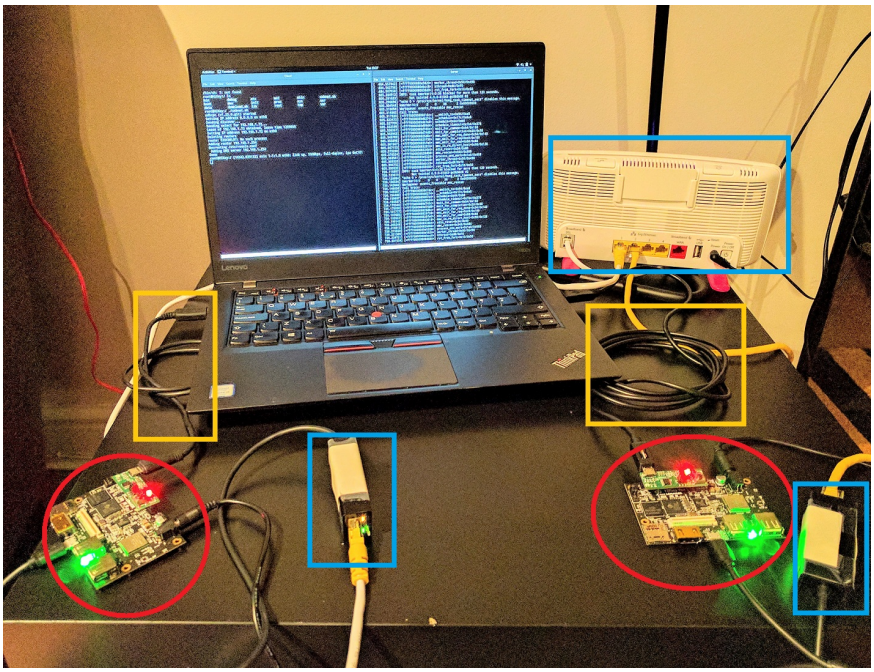


FIGURE 4.4: Test-bed environment: HiKey boards (circled) reachable over Ethernet via a LAN router (blue), and connected using UART-to-USB (yellow) to a Lenovo T460s for debugging.

in the LPWAN space. In light of this, we advise that our presented data below should be considered a baseline measurement.

4.7.1 Performance Comparison

The results are presented in Figure 4.5 (comparing BTP and UTP with TLS and SSL), Table 4.2 (BTP and UTP message-specific times), and Table 4.3 (total protocol round-trip time). As shown, our proposal performs consistently at approximately **1.7** seconds to execute in its entirety (round-trip) – yielding a ∼4x overheard versus TLS with ephemeral key Diffie-Hellman and RSA. This comprises the time to open/close TCP/IP sockets, receive and parse messages from the client/server, invoking the protocol TA functions and world context switches. This is in addition to performing all of the cryptographic algorithms including 2048-bit Diffie-Hellman key-pair generation, HMAC and signature generation and verification, including for quotes.

Some overhead was expected due to the results from related work: [70], described previously for mutual attestation of TPM platforms, also exhibited approximately between 3-5x overhead versus SSL and SSH on a Raspberry Pi B with software-based TPMs. Several attributes in particular were identified that we believe contributed to the exhibited overhead:

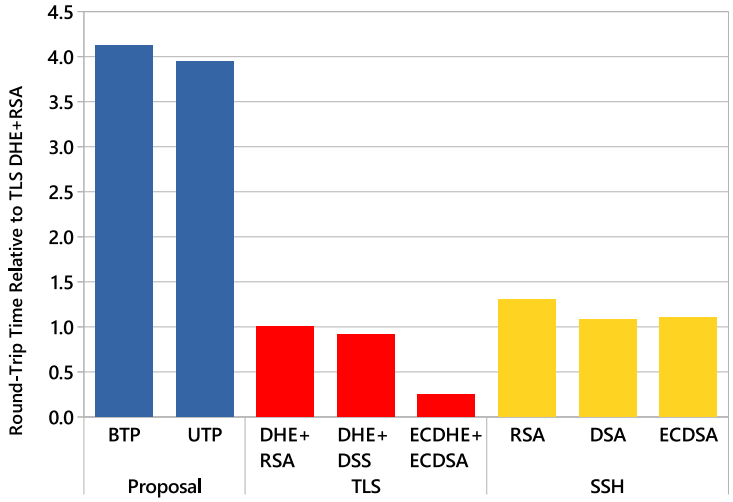


FIGURE 4.5: Relative protocol performance.

- Our test-bed implementation could benefit from optimisations typically found in widely-used protocol implementations, such as OpenSSH and OpenSSL, which have undergone years of peer-reviewed refinement from the C and X86-64 Assembly communities. An obvious potential optimisation is that the test-bed is single-threaded; OP-TEE does not support intra-TA multi-threading at the time of writing. On a supported TEE, one may consider parallelising the generation of $\sigma_{TA_{RE}}(X_{TA_{RE}})$ and $\sigma_{TA_{RE}}(V_{TA_{RE}})$, the latter of which includes quote generation, for message one (and for TA_{SD} in message three), rather than executing each operation sequentially. This could also consider parallelising the derived DH session key, i.e. $(g^{SD})^{RE}$, alongside $V_{TA_{SD/RE}}$ and $X_{TA_{SD/RE}}$.

- The penalty imposed by four and two world context switches between OP-TEE and Debian OS for retrieving and verifying messages on TA_{SD} and TA_{RE} respectively (see Figure 4.3). This also includes four context switches per device for the instantiation and destruction of each TA, preceding and following the protocol. Another factor is the time to allocate, initialise, execute and free dynamically-allocated, GlobalPlatform-specific objects and operations. The reader is referred to the GlobalPlatform Internal API specification [78] for full details regarding the preparation and structure of specific primitives.

  Related to this, each GlobalPlatform Internal API call is also several steps removed from the underlying implementation provided by LibTomCrypt in the OP-TEE call stack, shown in Listing 3. Beginning with a TA function in user mode, the GP Internal API functions, i.e. `TEE_*()` such as `TEE_CipherInit()`, are made available in the static library `libutee.a`,

Listing 3 OP-TEE GlobalPlatform Internal API call stack [208].

```
1: ta_function() {Arbitrary TA function – User space}
2: → TEE_*() {Provided in libutee.a}
3:    → utee_*() {System call interface}
4:       → tee_svc_*() {Kernel space}
5:          → crypto_*() {libtomcrypt.a and crypto.c}
6:           /* LibTomCrypt */ {libtomcrypt.a; algorithm implementations}
```

which principally performs parameter validation and error reporting. Next, `utee_*()` is invoked that transitions to the appropriate service in TEE kernel space using ARM supervisor calls via `SVC` instructions. `tee_svc*_()` is used to copy memory buffers between TEE user and kernel space, and to invoke a private abstraction layer, the internal Crypto API (`crypto_*()`), which implements the actual algorithms. The Crypto API uses implementations from LibTomCrypt, compiled as a static library `libtomcrypt.a`, and hardware-accelerated versions for supported platforms. (The HiKey boards used in the test-bed implementation provide hardware-accelerated AES and SHA-256 using the ARMv8-A Cryptographic Extensions instructions). While one expects a modern compiler to mitigate much of the incurred overhead, cryptographic GlobalPlatform functions are significantly abstracted from the *actual* implementations provided by LibTomCrypt.

- The difference in cryptographic implementations between SSH and TLS, both of which used OpenSSL, and LibTomCrypt used by OP-TEE. Expectedly, a significant portion of the protocol time occurs during Diffie-Helmann operations. This occurs once on the client-side in message one and three times in message two: twice server-side – once to compute g^{RE} and the other to compute the derived DH session key, i.e. $(g^{SD})^{RE}$ – and another on the client-side to compute its shared value after receiving g^{RE}, i.e. $(g^{RE})^{SD}$. Surprisingly, little work exists that systematically benchmarks security protocols and cryptographic primitives across ARM SoCs. While benchmarking cryptographic suites is outside the scope of this paper, a limited selection of work indicates that *"performance varies greatly"* [209] between cryptographic libraries and, indeed, with alternative compilers and compiler flags [210], [211].

Notwithstanding potential optimisations, the protocol executed in a round-trip time of 1692ms and 1612ms for BTP and UTP respectively. We stress that these timings should be considered a baseline measurement. As stated previously, we are currently unaware of any study that accurately benchmarks OP-TEE, or any other GlobalPlatform-compliant OS using ARM TrustZone, for reliably

identifying bottlenecks and discerning the cause of timing overheads imposed by TEEs. We see this as a valuable contribution in itself for future research.

4.7.2 Related Work Comparison

A criteria comparison is made with related protocols in Table 4.4. This table is an extension of the security criteria presented by Akram et al. in [69] and [70] in order to reflect the goals specified in Section 4.4. The table includes well-known transport-layer protocols (TLS [212] and SSH [213]) and their variants (DTLS [214]), and smart card-based protocols (the Markantonakis-Mayes [215], GP SCP81 [216] and Sirett-Mayes [217] protocols). The rationale behind incorporating smart cards is its similar goals relating to the communication with sensitive applications aboard a constrained device. Moreover, the smart card industry is similarly tightly controlled by card issuers and manufacturers and, like TEEs, must adhere to high standards established by Common Criteria. We also include other key exchange (Station-to-Station [218], Aziz-Diffie [219], ASPeCT [220] and Just Fast Keying or JFK [221]) and trusted channel protocols (Trusted TLS or T2LS by Gasmi et al. [71], Akram et al. (AMMBSC) [70], Greveler et al. (GJL) [66], Enhanced Privacy ID with SGX [160] and P-STCP [69]).

As shown in Table 4.4, no protocol satisfies the goals motivating this work regarding TEE mutual attestation. Some are closely related, such as AMMBSC [70], SGX+EPID [160] and GJL [66]. SGX+EPID is not considered due to the tightly-controlled nature in which Intel provisions PSKs during manufacturing and performs attestation (described in Section 4.4.3), which ultimately ties OEMs into a trusted relationship with Intel. Naturally, AMMBSC is avoided for its reliance on TPM, the issues of which were discussed in great detail in Chapter 2. GJL, which is similar to AMMBSC and uses TPMs, is avoided likewise. While used widely, (D)TLS and SSH are avoided due to the absence of trust provisions in addition to the difficulty of engineering TEE-based attestation in such protocols retroactively while remaining reasonably formally verifiable.

4.7.3 Limitations

TEEs offer their own benefits in trusted sensing, but avoiding TPMs relinquishes extensive hardware tamper-resistance. TEEs do not necessarily defend against sophisticated hardware attacks and side-channel analyses, but are designed primarily to resist software-based vectors. Consequently, we urge caution in using our work where complex hardware attacks are a reasonable possibility, e.g. military and government applications, where state-level adversaries comprise part of a threat model. Secondly, GlobalPlatform recently defined the Sockets API [105]

TABLE 4.4: Comparison with related protocols (adapted from Akram et al. [70]).

	Protocols															
Req.	STS [218]	AD [219]	ASPeCT [220]	JFK [221]	T2LS [71]	SCP81 [216]	MM [215]	SM [217]	P-STCP [179]	SSH [213]	TLS [212]	DTLS [214]	SGX [64]	GJL [66]	AMMBSC [70]	***Proposal***
S1	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
S2	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
S3	✓	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
S4	✗	✗	✗	✗	✓	(✓)	✗	✗	✓	✗	✗	✗	✓	✓	✓	✓
S5	–	–	–	–	✗	✗	–	–	✗	–	–	–	✗	✓	✓	✓
S6	✓	✓	✓	✓	✓	✓	✓	(✓)	✓	✓	✓	✓	(✓)	✓	✓	✓
S7	✓	✗	✓	✓	✓	✓	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓
S8	✗	✗	✗	✓	✓⋆	✗	✗	✗	✓	✗	✓⋆	✓⋆	✓	✓	✓	✓
F1	–	–	–	–	✗	✓	–	–	✗	–	–	–	•	✗	✗	✓
F2	–	–	–	–	–	–	–	–	–	–	–	–	✗	–	–	✓
F3	✗	✗	✗	✗	✓	✗	✗	✗	✓	✓⋆	✓	✓	✗	✗	✓	✓

✓— Supported. ✗— Unsupported. (✓) — Supported implicitly by the architecture. ✓⋆ — Supported after modifications. (–) — Not applicable, due to conflict with another requirement. • — Keys provisioned into hardware fuses during manufacturing.

for initiating TCP/IP connections from the trusted world. This provides an interesting avenue of research, which, in the context of our protocol, would remove a context switch between the secure and trusted world. Mature implementations of the Sockets API are not widely-available; OP-TEE, for example, does not fully support GP Sockets API at the time of writing. We aim to revisit this in the future when available in order to evaluate the overhead reduction, if any, in reducing world context switches. Finally, defending against availability attacks is currently an open challenge in TEE research [148]. TEEs rely implicitly upon co-operation between untrusted and trusted world, and an untrusted world adversary that simply drops messages from the TEE would immediately prevent protocol execution, thus raising availability concerns. This TEE-wide vulnerability was considered out-of-scope in this paper.

4.8 Conclusion

In this work, we investigated the application of TEEs to constrained sensing devices that wish to communicate remotely, and raised the challenge of secure and mutually trusted TEE intercommunication between remotely located devices. To this end, the development and evaluation of a secure and trusted channel protocol was presented that, unlike past work, provides one- or two-way remote attestation for trust assurance without trusting intermediary components. The proposal is applicable to sensitive deployments that would benefit from the advantages provided by TEEs without disclosing data to a potentially compromised REE on either end-point. In Sections 4.4 and 4.5, both the goals and protocols were formalised, along with the operational challenges and assumptions. In Section 4.6, we discussed their implementation using two widely-used development boards using ARM TrustZone, before evaluating its performance against untrusted world TLS and SSH in Section 4.7. Both protocols were subjected to formal analysis using Scyther, which found no attacks under the Dolev-Yao adversarial model. The scripts are included in Appendix A. We showed that our proposal executes within reasonable time (under 1.7 seconds on average) and exhibits an overhead similar to existing work of approximately 4x overhead versus TLS and SSH without optimisation.

4.8.1 Future Work

In future work, we aim to investigate the following lines of research:

- *Trusted multi-party protocol.* Generalising our protocol to create a trusted channel shared between N-parties in close proximity. This could be advantageous where security-sensitive, multi-party communications are used

frequently, such as authentication schemes using sensor data from a body area network (BAN), or a sensor-driven home automation system.

- *Performance comparison with TPMs and other TEEs.* We aim to conduct a broad performance evaluation of our protocol with TPM-based solutions and emerging TEEs, such as the AMD PSP and Trustonic Kinibi, on a range of device types. Furthermore, we plan a performance comparison with other cryptographic libraries.

- *DAA-based privacy preservation.* Our work, as is, allows the identification and mapping of devices during the session from the exchange in TM-signed messages using certified, device-specific signing keys. This was addressed in TPMs using DAA, but, in this work, our concentration was focussed on the baseline case before advancing to incorporating DAA-based schemes. We believe that this would be the most beneficial next step in enhancing our proposals in future work.

- *Applicability of elliptic curves.* Recent GP specifications include Elliptic Curve Cryptography (ECC), but this is only an optional extra for compliance. Moreover, only five NIST curves are supported currently, up to NIST P-521, but further curves are expected to be released in the future [78]. Once matured, we aim to revisit our protocol to provide elliptic curve-based Diffie-Hellman (ECDH) to reduce key sizes and computational overhead.

Chapter 5

EmLog: Tamper-Resistant Logging for Constrained Devices with TEEs

5.1 Introduction

The previous chapters introduced secure and trusted technologies, and motivated the development of the Trusted Execution Environment (TEE) as a low-footprint mechanism to protect the critical data upon which many sensitive security systems depend. This proceeded to conducting the first investigation into evaluating the application of TEEs to continuous authentication for preserving the confidentiality and integrity of its assets at rest and during execution (Chapter 3). The subsequent chapter identified, assessed and evaluated the issue of TEE intercommunication on two remote devices, whereby TEEs may communicate over a secure and mutually trusted channel without revealing TEE-resident data to either device's untrusted worlds.

This thesis continues its examination of establishing trust in modern embedded devices using TEEs in this chapter by investigating its applicability in protecting system logging data for facilitating audit and forensic investigations. Secure and trustworthy system logging is beneficial in these scenarios. Ideally, logs should be tamper-resistant both at rest and during execution, and should remain protected during the transmission process to a trusted remote verifier. This chapter proposes and evaluates EmLog – a novel secure logging mechanism using ARM TrustZone and the GlobalPlatform TEE as a root of trust for preserving logs on constrained devices.

5.1.1 Motivation

Recording and analysing system logs are a principal feature of audit and forensic investigations in event reconstruction, error detection (and subsequent remediation), and intrusion detection [222]. These logs record salient system features, such as user activity, resource consumption, peripheral use and error details. The primary aim is to construct an audit trail in order to understand system activity, enforce user accountability, and evidence malicious behaviour. Consequently,

logs are routinely targeted by adversaries to conceal evidence of wrongdoing. Because of this, both NIST [222] and ISO 27001:2013 (Annex A, A.12.4) [223] recommend secure log-keeping as an essential practice in preserving the auditability of a compromised system. Not only should logs be stored in a way that cryptographically preserves their confidentiality and integrity, but trusted computing primitives have been identified as desirable in existing proposals [224], [225]. Such technologies, primarily TPMs, have been used for tamper-resistant storage of logging keys, performing log-centric cryptographic operations, and providing evidence of platform integrity to third-party verifiers using remote attestation.

However, the advent of low-cost, mass-produced Internet of Things (IoT) devices complicates the use of trusted computing for tamper-resistant logging. As stated previously in this thesis, a multitude of proposals suggest using IoT devices in sensitive application domains, like health and social care remote monitoring [226], [227], identifying fires and gas leakages in the home [228], and monitoring the operational and environmental conditions of manufacturing spaces and equipment [18] – all of which are natural applications for tamper-resistant logging. The current body of related work focuses principally on the application of discrete hardware TPMs, which presents a variety of previously discussed challenges relating to constrained devices. The most pertinent drawback is the TPM's inability to directly host arbitrary applications, e.g. a secure logging application, without using additional software processes that concentrated on launching TPM-backed hypervisors [224], [229]. Earlier TPM-based TEEs suffered from well-known challenges surrounding the size of its Trusted Computing Base (TCB) – the minimal set of software and hardware components essential to its security – that widened the scope for introducing security and performance defects [84], [230], [231]. Over time, modern TEEs, typified by Intel SGX and ARM TrustZone, have converged towards enabling isolated execution using the same core execution hardware of the REE; the TCB, ultimately, is reduced to this hardware – the CPU (SGX) or SoC (ARM TZ) – and the software interface definitions exposed between the REE and TEE.

Existing TEE-based logging schemes have hitherto applied only server-side TEEs to protect logs transmitted from remote devices. The challenge of protecting logs on constrained devices with TEEs in particular remains, to the best of our knowledge, unaddressed in existing literature.

5.1.2 Chapter Structure

In this chapter, we present EmLog – a tamper-resistant logging system that leverages the GlobalPlatform TEE and ARM TrustZone for protecting logs *at source* on mobile and embedded systems. EmLog offers further security benefits over past work, including public verifiability of log origin, resilience to TEE key

compromise, and supports secure I/O with peripheral devices. After reviewing related work (Section 5.2), we formalise the security goals and threat model in Sections 5.3 and 5.4 respectively. EmLog is implemented on an off-the-shelf ARM development board hosting OP-TEE [152] – an open-source and GlobalPlatform-compliant TEE that uses TrustZone (Section 5.6) – and evaluated using three datasets in Section 5.7. Finally, we conclude our work in Section 5.8 and identify future areas of research.

5.1.3 Contributions

To our knowledge, this is the first attempt at preserving logs on constrained devices using TEEs as a root of trust. The contributions of this paper are:

- The proposal of a novel secure logging scheme for creating tamper-resistant logs with trust assurances based on the GlobalPlatform TEE. The system is tailored for ARM-based constrained devices, such as wearables and sensing platforms, for on-device protection of logs for potential use in audit and forensic investigations.
- A test-bed evaluation using a GlobalPlatform-compliant TEE instantiated by ARM TrustZone, with performance benchmarks across three datasets. The results indicate that EmLog has low run-time memory footprint, five-times persistent storage overhead, and 430–625 logs/sec throughput.

This chapter is based on our following publication:

- C. Shepherd, R. N. Akram, and K. Markantonakis. "EmLog: Tamper-Resistant System Logging for Constrained devices with TEEs," in *Proceedings of the 11th IFIP International Conference on Information Security Theory and Practice*, ser. WISTP '17, Springer, 2018.

5.2 Related Work

Existing proposals may be categorised as: **1)**, *secure untrusted system logging*, focusing on cryptographic methods for detecting tampered logs on untrusted platforms; and **2)**, *trusted logging*, for applying trusted hardware primitives for log preservation. The key proposals and their contributions are now examined.

5.2.1 Secure Untrusted System Logging

Secure untrusted system logging schemes seek to detect the tampering of logs on an untrusted machine, M, after some compromise at point t in time. M is assumed to possess the ability to collect and store system log entries to file(s)

and, upon request, transmit them to a verifier, V, over a secure channel who then inspects and verifies the contents. Before time t, M is not assumed to be compromised, and, for most schemes, M and V are assumed to possess a pre-shared key (PSK), which we denote the Root Logging Key, RLK. After t, little can be done to trust the contents of logs produced after that point, but these schemes do address protecting the logs produced prior to t.

Schneier and Kelsey [232] proposed the use of MACs within a linear one-way hash chain to protect log integrity. Each entry is of the form $L_j = (W_j, A_j, E_{K_j}(D_j), Y_j, Z_j)$, consisting of a log entry type W_j, an authentication key A_j, the encrypted log entry $E_{K_j}(D_j)$, an incremental hash entry $Y_j = h(Y_{j-1}, E_{K_j}(D_j), W_j)$, and a MAC entry $Z_j = MAC_{A_j}(Y_j)$. The scheme rests on a one-way hash function, h, used to incrementally key the MACs of Z_j, as each MAC key, A_j, is found by computing $A_j = h(A_{j-1})$, where A_0 is the RLK. The MAC key is also used in the derivation of the log encryption key by computing $K_j = h(W_j, A_j)$. It is also necessary to delete the incremental keys, A_j, at each stage in order to prevent an adversary from fabricating logs and reconstructing the chain retrospectively. The concept of a hash chain underpins the bulk of proposals in secure logging, with and without trusted hardware. Bellare and Yee [233], similarly, propose updating the secret key at regular time intervals (epochs) using a sequence of values from pseudo-random functions, implemented using the International Data Encryption Algorithm (IDEA), to update the log MAC keys in each epoch. They denote the infeasibility of deriving MAC keys in previous epochs as 'forward integrity'.

Both of these schemes are based inherently upon symmetric cryptography; the initial RLK must be distributed to any V wishing to verify the logs' integrity. This poses a key management challenge if V changes or is potentially one of several entities; large numbers of RLK keys must be securely managed across all interested parties. To address this, Holt [234] propose LogCrypt that uses public-key cryptography in conjunction with the scheme in [232] by replacing the MACs with signatures from regularly re-generated key-pairs. A hash-chained symmetric RLK is still used to to key the encryption of logs using an appropriate algorithm, e.g. AES. In accordance with the security principle of key separation, LogCrypt uses differing keys for encryption and verification, rather than using RLK for both tasks, as in [232]. Logcrypt also offers *public verifiability* such that third-parties can authenticate the origin of log entries without knowledge of a secret PSK (shortfalls of [232] and [233]).

Later, Ma et al. [235] introduced FssAgg, which uses an aggregated chain of signatures to achieve public verifiability and to thwart *truncation attacks* in which an attacker aims to delete a tail-end subset of log entries. Yavuz et al. [236] proposed LogFAS, which addresses both challenges with better storage and computational complexity than [234] and [235] using the Schnorr signature scheme.

Recently, however, Hartung [237] presented four attacks against LogFAS [236] and two variants of FssAgg [235], which enables secret key recovery and log forgery. Consequently, both schemes are dissuaded from use.

5.2.2 Secure Logging with Trusted Hardware

The other category of related work focusses on the application of trusted computing primitives in order to facilitate secure logging. This includes providing stronger guarantees for the security of the logging application (known also as a 'logger') both under execution and at rest, as well as the confidentiality and integrity of log entries. These schemes are complementary to previous schemes by taking a system security perspective rather than a wholly cryptographic one. Indeed, it is the case that many of these systems – described below – incorporate cryptographic principles from the previously described schemes.

In 2003, Chong et al. [238] produced the first work in applying tamper-resistant hardware for providing additional security assurances with respect to cryptographic logging schemes. The authors explore the use of the iButton[1] – a device containing 134kB NVRAM, a real-time clock (RTC) and a 32kHZ processor in a 17.35mm cylindrical stainless steel enclosure[2] – in order to protect the initial pre-shared key, RLK (or A_0), of the Schneier and Kelsey scheme [232]. The iButton is used to provide tamper-resistant timestamping of log entries, as well as encryption using 64-bit DES from SHA-1 keys.

Later, Sinha et al. [225] suggested a using a TPM with a novel forward integrity scheme based on branched key chaining and the Schneier and Kelsey [232] scheme. Here, logs are divided into epochs (blocks) with each block comprising an independent sequence of hash-chained log entries (sub-epochs). The purpose of this subdivision is to reduce the overhead of sealing each individual log entry to storage using the TPM. The root entries of each epoch are hash-chained with past epochs, thus creating a two-dimensional hash chain for protecting against *re-ordering attacks*. A re-ordering attack is one in which an adversary re-orders log blocks (arranged in time) in order to mislead investigators; that is, attempting to construe a series events that may be correct in their *contents* but not in their *chronology*. For each new epoch, the previous epoch's logs are securely stored using the TPM's seal functionality in the typical fashion, i.e. encrypts the logs with a TPM-bound key such that only that particular TPM can decrypt/'unseal' them.

Böck et al. [224] explored the use of AMD's Secure Virtual Machine (SVM) – an early inception of the TEE – for launching a `syslog` client daemon and

[1] iButton: https://www.maximintegrated.com/en/products/ibutton/ibuttons/index.cfm

[2] The iButton famously underpinned the Java Ring [239] in the late 1990s.

logging application from the TPM's secure boot chain. The logger executes with access to TPM-bound key-pairs for encrypting and signing individual log entries. Upon request, the logs are decrypted and transmitted to the verifying party over a secure channel. The TPM keys are certified for authenticating that signed logs originated from a particular AMD SVM instance; the TPM is used in convincing a remote verifier of the platform's expected operating state via remote attestation.

Nguyen et al. [240] proposed streaming medical logs to a server application executing within an Intel SGX enclave (see Section 2.4.6) that applies the tamper-resistance algorithm. In this scheme, logs are transmitted from healthcare devices to the enclave using TLS, which then computes a simple hash chain scheme in which each value is found using $a_i = h(a_{i-1}, s_i, h(c))$, where a_i is the current entry, s_i is the i^{th} sequence number, and c is the contents of the log message. Each log entry is signed using an key-pair stored within the enclave and subsequently stored to file using SGX's sealing mechanism.

Karande et al. [241] introducde SGX-Log, which protects server-side device logs received from arbitrary remote devices. SGX-Log, like [225], uses block-based hash chains and seals them to secure storage for persistent log integrity and confidentiality; the scheme, like [240], is also implemented within an SGX enclave. The authors note that continual sealing also provides resilience to attacks in which large volumes of logs in memory are lost due to a power loss. Remote attestation is also suggested to authenticate the server enclave before transmitting the logs. However, public-key cryptography is not used establishing public verifiability of origin, unlike [224] and [234], nor is it apparent how to protect logs from the devices from which they are sourced. The proposed scheme is evaluated using three datasets on an SGX-enabled Linux system with an eight-core Intel i7-6700 CPU (3.4GHz) and 64GB RAM, yielding a small (<7%) overhead versus a non-SGX implementation.

5.2.3 Discussion

Modern TPM- and TEE-based approaches [224], [225], [240], [241] still fall short of satisfying many desirable properties identified in past work. Public verifiability of origin, as in [224] and [234], has not been addressed in recent TEE loggers for directly authenticating system data from remote devices. Most notably, recent TEE-based schemes, i.e. [241] and [240], focus primarily on protecting logs *after* being received by a server-side log processing application. There has been little attention paid to protecting logs collected on *source* devices, which may themselves host a TEE. An adversary aboard a compromised source device may simply tamper the logs before reaching the server that applies some tamper-resistance algorithm. This is where the contribution of this work lies; the focus is on protecting logs *at source* on constrained devices.

To complicate matters, source devices are unlikely to transmit logs in real-time in order to minimise network and computational overhead, and so secure storage should be used to preserve unsent logs. Moreover, TEEs typically contain other security-critical applications, such as biometric authentication and payment tokenisation. As a result, a TEE-based logging mechanism should operate with reasonable resource consumption, particularly run-time memory, to preserve performance and limit the rise of Denial of Service (DoS) conditions.

5.3 Security Goals

To scope this work, we identify a series of security and functional goals for constructing a TEE-based system for protecting logs on constrained devices, which are drawn from the issues identified in related work in Section 5.2:

R1. *Isolated execution*: the system shall process logs in an environment isolated from a regular 'rich' OS, e.g. Android, to provide strong integrity assurances of the application and data under execution.

R2. *Forward integrity*: the integrity of a given block of logs shall not be affected by a key comprise of a previous block.

R3. *Log confidentiality*: on-device log confidentiality should be preserved to prevent the disclosure of potentially sensitive entries.

R4. *Remote attestation*: the proposal shall allow third-parties to verify the logging application's integrity post-deployment to provide assurances that logs were sourced from an integral and authentic platform.

R5. *Secure log retrieval*: remote, authorised third-parties shall be able to securely retrieve device logs with mutual trust assurances.

R6. *Public verifiability*: the system shall allow third-parties to authenticate the origin of log entries without access to private key information.

R7. *Truncation attack-resistant*: the system shall be resistant to attacks that aim to delete a contiguous subset of tail-end log entries.

R8. *Re-ordering attack-resistant*: the proposal shall resist attempts to change the order of entries in the log sequence.

R9. *Power-loss resilience*: the loss of tamper-resistant logs shall be minimised in the event of a device power-loss.

R10. *Suitable root of trust*: a root of trust for constrained device architectures shall be used, ideally without requiring additional security hardware.

5.4 Threat Model

Two types of adversary are addressed for when, firstly, the logs are collected and processed *on the device* and, secondly, when they are retrieved by a verifier *over a network*. These are described as follows:

- *On-device adversary*: a software-based attacker that compromises the system at time t and attempts to arbitrarily alter, forge or observe logs produced before t. This adversary can operate at any protection level in the untrusted world, i.e. Rings 0–3, including arbitrarily altering execution flow and accessing non-TEE kernel space services. For example, the attacker may aim to modify the contents of individual log entries to conceal any evidence of wrongdoing, or, like [241] and [225], perform re-ordering attacks in which blocks of log entries are re-ordered to construe a misleading sequence of events chronologically. As stated in Section 5.3, we also tackle the issue of truncation-resistance, identified in [235], where an attacker aims at deleting at sub-set of tail-end log entries. Additionally, like [241], we consider power-loss attacks in which an attacker disables power to the device with the aim of erasing significant numbers of logs kept in volatile RAM.

- *Network adversary*: an adversary that attempts to arbitrarily alter, forge, replay or observe logs between the source device and the verifier over a network channel, e.g. Bluetooth or WiFi/802.11. The attacker may also attempt to masquerade as a legitimate party to either end-point to collect logs illicitly. Here, we consider the principle goal of the attacker to modify or observe log entries as they are transmitted over-the-air to the log verifying authority.

The first case considers an *on-device* adversary similar to related work, namely Böck et al. [224], Karande et al. [241], Sinha et al. [225], and the secure untrusted system logging schemes identified in Section 5.2.1. In other words, we do not attempt to secure untrusted world logs *after* a compromise by the above adversary after time t, as a kernel-mode adversary may simply write directly to the kernel message buffer used to queue log entries (see Section 5.5.1). Related to this, we do not consider denial-of-service (DoS) attacks from an adversary after time t who attempts to flood the logging scheme with log entries at a rate faster than they are processed in the TEE. As per related work, the assurances are valid only before time t at which the device is compromised by the on-device adversary stated above. Principally, the aim is to protect system logs collected before time t against kernel-level adversaries. In the second case, for the *network* adversary, we do not cover DoS attacks in which the attacker impedes all network traffic

between device and the verifier, thus preventing log retrieval; physical retrieval would be one remedy in this situation.

We assume the presence of a GlobalPlatform-compliant TEE that satisfies the GlobalPlatform TEE Protection Profile, the scope of which was discussed in detail in Section 2.4.7 of Chapter 2. As such, threats that fall outside the protection remit of the GlobalPlatform TEE are beyond the scope of this work, such as developer-induced TA and TEE programming errors, supply chain attacks, and advanced hardware attacks requiring substantial time, capital, equipment and expertise, e.g. laser fault injections. Consequently, caution is urged in situations where such threats are considered. We refer the reader back to Sections 2.5.1, 2.5.2 and 2.6 for a discussion of the threats considered within the security remit of TEEs and other secure and trusted execution technologies generally. For clarity, we consider the issue of log entry *analysis* – the act of assessing system activity and detecting anomalous and malicious events, usually achieved using anomaly detection algorithms (see Xu et al. [242] and Fu et al. [243]) – to also be out-of-scope in this work.

5.5 EmLog Architecture Design

We assume the presence of a GlobalPlatform-compliant TEE, a service provider that provisions EmLog into the TEE before deployment, and a third-party wishing to retrieve all or a partial set of the device's logs in order to verify them. The GP TEE, which maintains two sets of applications for each world, necessitates two logging components: one that collects logs from untrusted world applications and transmits these to the TEE over the world boundary via the GP Client API [77], and another that applies the protection algorithm within the TEE and responds to retrieval requests. An extension to the hash matrix in [225] and [241] is proposed to apply the tamper-resistance scheme within the GP TEE, which is developed further in Section 5.5.2. Next, the log blocks are stored every n blocks, or at a time epoch t, using the secure storage functionality of the GP TEE. After receiving a retrieval request, the source TEE authenticates the remote verifier and vice-versa, after which the blocks are unsealed and transmitted over a secure and trusted channel between the TEEs (Section 5.5.3). We illustrate this process in Figure 5.1. Each stage is described in the following sub-sections.

5.5.1 Log Collection

A conventional Linux-based kernel uses an internal message ring buffer to store log messages, which is made available to user space monitoring applications, such as `dmesg` and `klogd`, using the `sys_syslog` syscall. For user-mode logging, `syslogd` listens on `/dev/log`, where logs are registered to using the

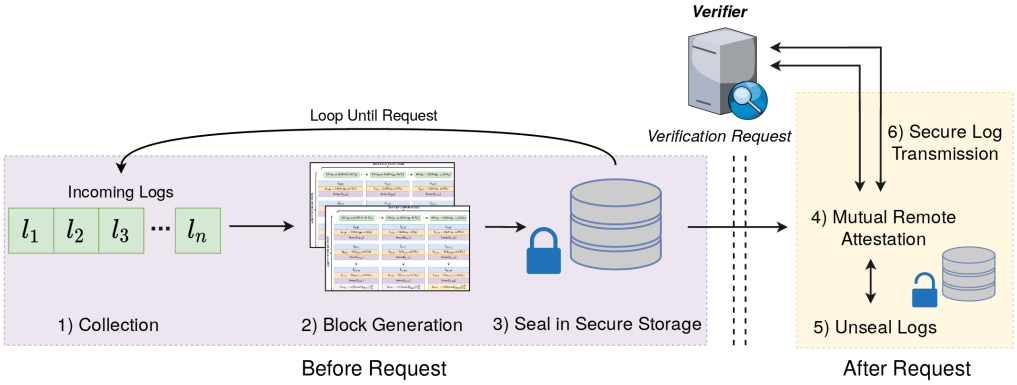


FIGURE 5.1: High-level TEE-based logging workflow.

`syslog` function from the C standard library. Logs are subsequently written to file or transmitted through the `syslog` protocol to a remote server over a User Datagram Protocol (UDP) connection. Some implementations, e.g. `syslog-ng`, provide further functionality like streaming logs over TCP with TLS. For collecting untrusted world logs, we suggest a `syslogd` variant that, ideally on a per-entry basis, transmits each system-wide log entry over the world boundary to the EmLog TA executing within the GP TEE via the GP Client API.

5.5.2 Block Generation

A variant of the hash matrix used in [241] and [225] is proposed for preserving the integrity of log entries and their sequence in time; we develop how this is expanded to enable public verifiability while limiting the exposure of a master Root Logging Key (RLK) using appropriate key derivation.

In such a structure, hash sequences are created in which each block key, bK, is derived using a one-way hash function, h, over the previous block key and current block ID, bID; that is, $bK_{bID} = h(bK_{bID-1}, bID)$. The initial block key ($bID = 0$) is derived from a device-specific RLK. Each block key is used to derive an individual message key, k, for keying an HMAC in a similarly chained fashion, i.e. $k_{(bID,mID)} = h(k_{\{bID,(mID-1)\}}, mID)$ for log entry mID in block bID, up to the block size m. Note that bK is used to derive k when $mID = 0$. A block-based approach provides power-loss resilience and truncation resistance (developed further in Section 5.5.3) while allowing the retrieval of subsets, i.e. blocks i to j, without transmitting all logs from the genesis block ($bID = 0$) to the remote verifier.

As it stands, this scheme is vulnerable to forgery attacks if just a single block key is compromised: an adversary can apply h on the leaked key with the next block ID to forge subsequent blocks and entries therein. Consequently, storing RLK and deriving keys within trusted hardware, e.g. an secure element (SE) or TPM, is desirable, but with the cost of additional hardware complexity.

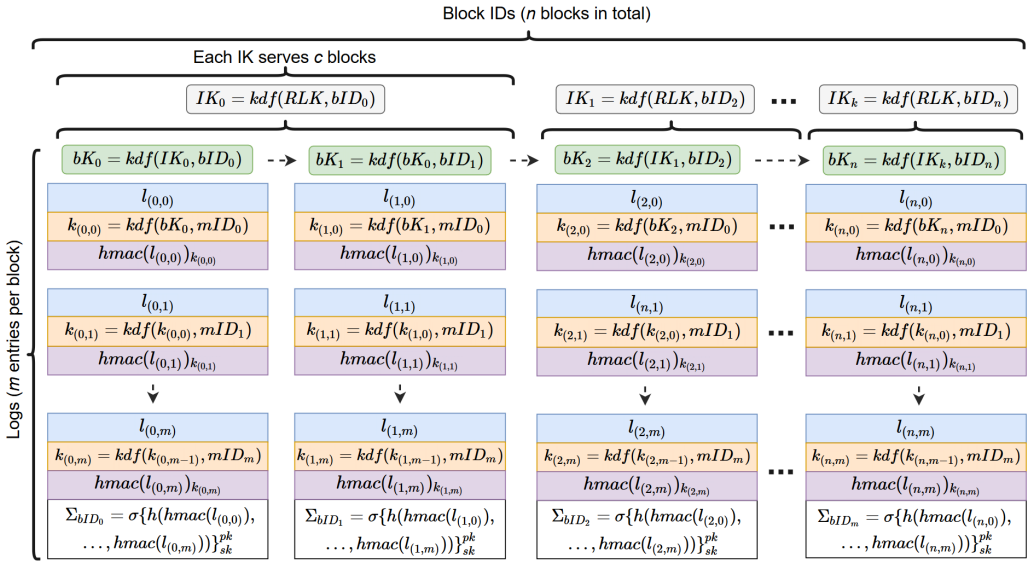


FIGURE 5.2: Proposed signature-based log matrix.

TEEs provide strong resilience to software attacks, but, unfortunately, are not invulnerable to developer-induced programming and API errors. The impact of RLK and block key divulgence, however, can be limited using a key derivation scheme – a simple one of which we describe below.

Key Derivation. We suggest a simple scheme as follows:

1. Intermediate keys (IKs) are derived from the RLK using a secure key derivation function. Each IK is subsequently used to derive keys for c blocks in that group.

2. Each IK derives an initial block key, bK_{bID}, for that block group; IK must then be erased from (run-time) memory.

3. This first bK_{bID} is then used to sequentially derive the message-specific keys within that block; each message-specific key should be erased from memory.

4. The next bK_{bID} is derived using $kdf\,(bK_{bID-1}, bID)$ for up to c blocks, after which another IK is generated. After generating bK_{bID}, then bK_{bID-1} should be erased from memory to limit unnecessary exposure. The scheme then repeats from step 2 with the newly generated IK.

In past proposals, the disclosure of a block key would necessitate the re-provisioning of the RLK – a device-specific, possibly hardware-infused key, which would affect the device in perpetuity without potentially costly intervention. Our approach (Figure 5.2) limits the damage wrought by a compromised block key by affecting only future blocks *in that group*. In the worst case, besides divulging RLK, the exposure of an IK can compromise only c blocks at most.

For the key derivation function, we suggest the HMAC-based extract-and-expand KDF (HKDF) by Krawczyk [244], [245] (RFC 5689). HKDF takes keying material and a non-secret salt as input, and repeatedly generates HMACs under the input to return cryptographically strong output key material. Unlike plain hash functions, used prevalently in past work, HKDF produces provably strong key material from as-strong or weaker input key material [244].

Log Integrity and Verifiability. For log message integrity, first compute $hmac(\ell_{(bID,mID)})$ for message ℓ with block ID, bID, and message ID, mID, under key $k_{(bID,mID)}$ – derived from the previous message key or, for $mID = 0$, the block key. As stated above, each k should be immediately deleted from memory to limit memory consumption and exposure. This does not prevent auditing log sequences, since message keys may be regenerated from the pre-shared RLK.

In current symmetric-only schemes [225], [232], [238], [241], public verification of log origin (R6 in Section 5.3) requires knowledge of block and message keys on all interested devices, which is derived ultimately from RLK. Evidently, revealing RLK is undesirable as it enables the malicious creation and manipulation of valid blocks. Rather, we propose signing each block with an efficient signature scheme, σ, such as ECDSA, and a device-specific signing key-pair (pk, sk) over the concatenation of the block message HMACs (Figure 5.2). This key-pair should be certified by a CA trusted by both the verifier, V, and host GP TEE device in order to provide data origin authentication. The RLK and key-pair should be accessible only to the TEE, which is achievable using the TEE's secure storage mechanism or, for hardware tamper-resistance (and its associated deployment overhead), using an external SE as suggested by GlobalPlatform [40]. In some circumstances, logs may contain sensitive data, in which case we suggest limiting verifiability to whitelisted entities. It is also observed that the block size, m, is inversely proportional to the number of signing operations; smaller block sizes will incur more signing operations for a given set of log entries (see Section 5.7 for this overhead).

5.5.3 Secure Storage and Remote Retrieval

Real-time log streaming is likely to be detrimental for power- and network-limited devices, and we suggest storing blocks prior to eventual transmission within the TEE's secure storage. Secure storage can be implemented in two ways according to GlobalPlatform:

1. Using the file-system and storage medium, e.g. flash drive, controlled by the untrusted world *"as long as suitable cryptographic protection is applied, which must be as strong as that used to protect the TEE code and data itself"* [78].

2. Using hardware controlled only by the TEE, such as an external SE.

In the first method, log blocks are typically sealed using authenticated encryption, e.g. AES in GCM mode [78], with a key derived specifically for the TA under execution from a separate, device-specific root storage key. This prevents other TAs or other entities from accessing secured data, thus providing on-device log confidentiality (R3), integrity and authenticity. While requiring additional hardware, the second method is resilient against adversaries that aim to delete encrypted records from an untrusted file-system. This deletion threat is otherwise difficult to address unless the storage medium is a so-called 'WORM' medium (Write Once Read Many), such as a single-write compact disc (CD-R).

Note that securely storing every completed block, i.e. in [241], may yield undesirable performance overhead for the devices targeted in this work. Rather, the parameters c (block group size) and m (block length) can control the number and size of blocks kept in RAM respectively. This satisfies power-loss resilience (R9), in addition to R3, by limiting the number of new blocks kept in memory (for sufficiently small values of c and m, which we evaluate in Section 5.7). This approach also satisfies truncation attack-resistance (R7), assuming the logs are stored on a medium accessible *only* to the TEE, e.g. secure element or separate replay-protected memory block and not, for example, encrypting logs to an untrusted filesystem under the potential control of the on-device adversary in Section 5.4.

In past work, log retrieval is proposed using TLS [240], or one-way remote attestation for authenticating the platform of the remote verifier [241]. (Many remote attestation protocols, such as EPID used in Intel SGX [160], enable secure channels to be bootstrapped, over which unsealed logs can be transmitted securely). However, the remote authority, which may itself process logs in its own TEE [240], [241], is likely to request reciprocal trust assurances from the source TEE; this is, two-way remote attestation to be performed for both the source *and* verifying entities. Rather than performing one-way attestation separately for both entities, one alternative is mutual TEE attestation as proposed in previous chapter (Chapter 4) in which both communicating TEEs are attested and authenticated within the protocol run. Using this, a secure channel can be bootstrapped between the TEE end-points over which unsealed logs can be transmitted securely without exposing them to untrusted world elements, thus meeting secure log retrieval (R5) in Section 5.3. In the forthcoming chapter, Section 6.8 of Chapter 6, we develop the idea of secure log retrieval with mutual attestation and describe its instantiation using the BTP protocol proposed previously in Chapter 4.

5.6 Implementation

We implemented EmLog using OP-TEE – an open-source, GlobalPlatform compliant TEE by Linaro [152] – with Debian OS (Linux-based) as the untrusted world OS. Two applications were developed in C – the only development language supported at the time of writing by OP-TEE – for the EmLog trusted application (TA) residing within the secure world, and a separate application in the untrusted world. The untrusted world application reads static log entries from file and sequentially transmits each log entry as a string to the EmLog TA residing in OP-TEE using the GP Client API [40]. OP-TEE abstracts low-level details of handling world context switches via secure monitor code using SMC calls; the switch to the TrustZone secure world is performed transparently to the target TA using the GP Client API via its universally unique identifier (UUID). Our implementation transferred each log message sequentially from file to the EmLog TA; that is, two world context switches per log entry to, firstly, transfer the entry contents where it is processed further, and, secondly, to return the success indicator to the untrusted application.

After the EmLog TA receives each entry from the untrusted world application, it is processed into a data structure comprising a 4-byte message ID, 32-byte HMAC tag, and 256-byte field for the entry text. The GP Internal API [78] was used to interface with the cryptographic methods; in OP-TEE, cryptographic methods are implemented using the `LibTomCrypt` library. More specifically, SHA-256 was used for log entry HMACs (`TEE_TYPE_HMAC_SHA256` in GP Internal API nomenclature) and also for the implementation of HKDF for message, block and intermediate key derivation[3]. Block keys are derived upon receiving the first message of a block, while message keys are derived sequentially when each successive log entry is received; IKs are derived upon the creation of the first block and then after every c blocks. Each completed message data structure is inserted into a separate data structure representing the message block containing a 4-byte block ID and a signature on a hash of the concatenated log HMACs using 256-bit ECDSA (NIST `secp256r1` curve via `TEE_ECC_CURVE_NIST_P256` from the GP Internal API).

Each block is then encrypted to secure storage and deleted from RAM before constructing the following block. Secure storage is performed transparently in OP-TEE when calling the Trusted Storage API functions from the GP Internal API; in particular, the `TEE_*PersistentObject()` function family from [78]. By default, data is encrypted to the untrusted world file-system using 128-bit AES in GCM mode (32GB eMMC flash memory was the default storage medium on our implementation platform: the same HiKey LeMaker development board

[3]Based on the python-hkdf implementation from https://github.com/casebeer/python-hkdf.

used in previous chapters). This is keyed under a TA-specific storage key (TSK) by OP-TEE, which is derived from its UUID and a root secure storage key (SSK) as follows: $TSK = HMAC(SSK, TA_{UUID})$ using HMAC-SHA256. Encrypted data is then stored by OP-TEE in the untrusted world's Linux file-system under the `/data/tee/` directory. The reader is referred to [246] for further details regarding OP-TEE's secure storage mechanism.

We note that the GP Internal API defines its own memory allocation functions, `TEE_Malloc` and `TEE_Free`, for dynamically (de-)allocating memory to regions accessible only to the TA, which were used frequently for memory-managing blocks and messages at run-time. Log blocks were subsequently deallocated once committed to secure storage using the GP Internal API to limit minimise consumption. As discussed in Chapter 3, memory consumption is an issue with OP-TEE when working with large datasets. The current OP-TEE release allocates, by default, 32MB RAM for the TEE kernel and all resident TAs, with the rest allocated to the untrusted world OS. For a standard TA, the Linaro Working Group stipulates a default stack and heap size of 1kB (stack) and 32kB (data) respectively, both of which can be increased up to a maximum recommended 1MB per TA. This can be adjusted up to the RAM limitations imposed by the deployed platform by adjusting OP-TEE's compilation flags; the 1MB recommendation was violated, to 2MB, for evaluating the largest group of blocks kept in memory prior to secure storage – discussed further in Section 5.7.

5.7 Evaluation

EmLog was evaluated using a HiKey LeMaker – an ARM development board with a Huawei HiSilicon Kirin 620 SoC with 2GB RAM and an ARM Cortex A53 CPU (eight-cores at 1.2 GHz with TrustZone extensions). This same platform has underpinned the implementations of the previous test-beds in this thesis; its specifications are typical of modern medium-to-high end IoT-type systems based on SBCs. The proposal was benchmarked using three log file datasets that are now described briefly:

1. *U.S. Securities and Exchange Commission (SEC) EDGAR*: Apache logs from access statistics to SEC.gov. We use the latest dataset[4] with over a million entries (192 MB). (Mean entry length: 115.1 characters; S.D.: 5.7).

[4]US SEC EDGAR: `http://www.sec.gov/dera/data/Public-EDGAR-log-file-data/2016/Qtr2/log20160630.zip`

TABLE 5.1: Mean key derivation and secure storage times (milliseconds; S.D. in brackets).

Key Derivation			Secure Storage Seal		Secure Storage Unseal	
IK	**Block Key**	**Message Key**	**IK**	**Block**	**IK**	**Block**
1.530 (0.067)	1.541 (0.062)	1.547 (0.088)	59.46 (3.78)	115.80 (5.36)	48.22 (2.73)	94.88 (2.80)

TABLE 5.2: Mean HMAC and ECDSA generation and verification times (milliseconds).

	HMAC (SHA-256)	**ECDSA (NIST P256)**
Generate	0.056 (0.020)	20.14 (1.29)
Verify	0.059 (0.014)	20.77 (1.33)

2. *Mid-Atlantic Collegiate Cyber Defense Competition (CDC)*: IDS logs from the U.S. National CyberWatch MACCDC event, with ∼166,000 (27 MB) of Snort fast alert logs[5]. (Mean entry length: 165.3 characters; S.D.: 38.2).

3. ***EmLogs***: Our dataset from OP-TEE OS boot, initialisation and GlobalPlatform test suite logs via the `xtest` command, and untrusted world logs from `dmesg`. Over 25,000 records (1.7MB). (Mean entry length: 94.1 characters; S.D.: 49.3).

The results are shown in Tables 5.1 to 5.4 and Figure 5.3. Table 5.1 shows the mean CPU time to derive 256-bit IKs, block and message keys from a pre-generated RLK using HKDF. These were measured over 1,000 iterations within the EmLog TA using the GlobalPlatform `TEE_Time` method for system time, which is implemented using the ARM Cortex `CNTFRQ` (CPU frequency) and `CNTPCT` (count) timing registers. Table 5.1 shows the mean time for sealing and unsealing IKs and blocks (for $m = 100$, averaged across all entries) via the GP Internal API. Table 5.2 lists the mean 256-bit ECDSA and HMAC-SHA256 times computed across all entries, while Table 5.3 shows the mean creation and verification times of message blocks for each dataset (for varying values of m, the number of entries per block), as well as block groups. In this context, verification encompasses the time to reconstruct the hash matrix in Figure 5.2 and to verify the block signatures and message HMACs. Group creation and verification time was measured for varying values of c (blocks per group), which included the time for sealing and unsealing blocks to secure storage respectively. Table 5.4 shows the mean persistent memory consumption of logs in secure storage, which was measured directly from the aforementioned `/data/tee` directory containing TA-sealed files. Lastly, Figure 5.3 shows the relative performance of secure storage, key derivation and block and group creation/verification times from Table 5.1.

[5]MACCDC dataset: `http://www.secrepo.com/maccdc2012/maccdc2012_fast_alert.7z`

TABLE 5.3: Mean block and group generation and verification times (milliseconds).

Dataset		Block (m =)10	Block 100	Block 250	Block 500	Group (c =)1	Group 10	Group 25	Group 50*
(1)	**Gen.**	40.1 (2.1)	70.5 (2.2)	115.6 (2.6)	198.7 (3.6)	229.4 (6.4)	1898 (30.2)	4622 (48.1)	9101 (80.2)
	Verify	41.2 (1.5)	71.9 (1.7)	114.8 (1.9)	204.3 (2.1)	213.4 (5.7)	1634 (23.7)	3967 (50.0)	8302 (78.3)
(2)	**Gen.**	39.8 (1.9)	68.3 (1.7)	117.2 (1.7)	202.3 (2.0)	231.7 (7.3)	1910 (28.1)	4687 (64.0)	9323 (81.5)
	Verify	40.1 (1.6)	66.2 (1.8)	119.9 (1.8)	199.0 (2.0)	215.2 (6.1)	1658 (23.8)	4046 (47.3)	8165 (73.9)
(3)	**Gen.**	42.1 (3.0)	69.1 (2.1)	118.6 (2.3)	201.9 (3.2)	230.0 (6.9)	1890 (27.0)	4621 (42.8)	9274 (79.0)
	Verify	40.0 (1.6)	69.3 (1.8)	120.4 (1.8)	200.4 (1.9)	217.3 (6.4)	1656 (25.2)	4132 (48.1)	8188 (75.5)

* Heap size set to 2MB to accommodate all data.
All other experiments recorded with the maximum recommendation of 1MB.

TABLE 5.4: Mean persistent memory consumption for per block secure storage across all datasets (kilobytes).

Block Sizes							
(m =)10	50	100	250	500	750	1000	2500
16.4 (1.4)	41.0 (1.8)	73.7 (1.9)	155.7 (4.3)	307.2 (10.9)	454.7 (11.1)	606.2 (11.8)	1495.0 (16.2)

5.7.1 Discussion

Little to our surprise, block generation and verification time scales linearly with message length, which, for large values of m, is influenced heavily by the key derivation operations (approximately 1.5ms per message, shown in Table 5.1). At smaller values, e.g. $m = 10$, this is dominated mostly by the ECDSA overhead ($\sim$20ms, as per Table 5.2). Figure 5.3 indicates that the relative timing overhead is $\sim$80–100% for every 100 message increase in the block length.

Group creation and verification times rise significantly with the number of blocks, c, kept in RAM before secure storage. This is driven significantly by the secure storage overhead, which is measured at approximately 115.8ms and 94.9ms for sealing and unsealing respectively (Table 5.1). Despite this, however, even the largest group sizes, $c = 25$ and $c = 50$ (2,500 and 5,000 entries in total), completed between 4.0 to 9.3 seconds, corresponding to a throughput of approximately 538 and 625 logs per second. At first, it seems attractive to maximise c to avoid the expense of secure storage operations, which caused the throughput to drop to $\sim$430 and 525 entries for the smallest groups ($c = 1$ and $c = 10$ blocks). Maintaining many blocks in RAM, however, increases the impact of a power-loss; systems that log infrequently may see significant data loss if large numbers of logs spread over a large period of time are lost. Consequently, c should be set based on the expected log and transmission frequencies.

For memory consumption, all experiments were conducted within the Linaro Working Group's run-time recommendations (1MB stack and heap), except for $c = 50$ blocks (5,000 entries), which required 2MB of each. Expectedly, persistent memory consumption of block secure storage (Figure 5.3) scales linearly with

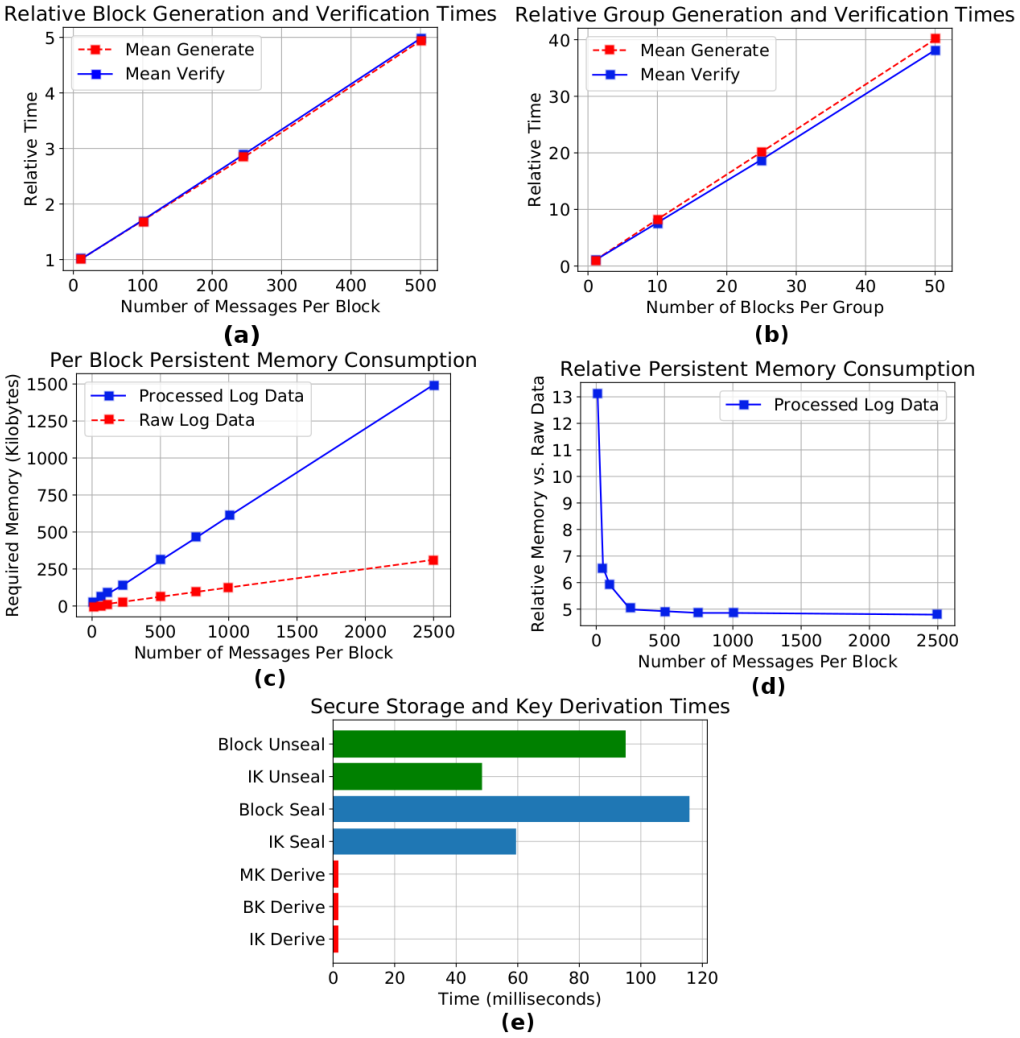


FIGURE 5.3: **(a)**, Relative block creation and verification times versus block length; **(b)**, relative group generation and verification for varying numbers of blocks; **(c)**, persistent memory consumption for per block secure storage; **(d)**, relative memory consumption for group secure storage; and **(e)**, raw key derivation and secure storage times.

TABLE 5.5: Security goal comparison with related work.

	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	Root of Trust
Untrusted World Schemes											
Schneier & Kelsey [232]	–	✓	✓	–	–	✗	✗	✗	–	–	–
Bellare & Yee [233]	–	✓	✓	–	–	✗	✗	✗	–	–	–
FssAgg [235]	–	✓	✗	–	–	✓	✓	✓	–	–	–
Logcrypt [234]	–	✓	✓	–	–	✓	✗	✗	–	–	–
LogFAS [236]	–	✓	✗	–	–	✓	✓	✓	–	–	–
Trusted Logging											
Chong et al. [238]	✓	✗	✓	✗	✗	✗	✗	✗	✗	✗	iButton
Sinha et al. [225]	✗	✓	✓	✓	✗	✗	✓	✓	✓	✗	TPM
Böck et al. [224]	✓	✗	¶	✓	✗	✓	✗	✗	✗	✗	AMD SVM
Nguyen et al. [240]	✓	✗	¶	✓	¶	✗	✓	✓	✓	✗	Intel SGX
SGX-Log [241]	✓	✗	✓	✓	¶	✗	✓	✓	✓	✗	Intel SGX
EmLog	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	GP TEE

✓– Supported; ✗– Not supported; ¶– Partially supported; (–) – N/A.

message size. Our test-bed uses a fixed 256-byte text field for each log entry, which accounts for the broadly similar performance across all datasets. We also calculated the persistent memory consumption compared with the mean size of raw logs; the relative consumption is large for small block sizes ($m < 250$), due likely to the fixed-size meta-data used by OP-TEE to manage cross-TA secure storage objects. For larger block sizes, this converges to slightly under five-times overhead versus raw logs; the absolute size of smaller block sizes remains low, however, at 16–155 kilobytes, according to Table 5.4.

5.7.2 Related Work Comparison

We compare the features of EmLog's with previous work in Table 5.5 using the goals defined in Section 5.3. Notably, the use of ARM TrustZone and the GlobalPlatform TEE makes it appropriate for mobile and embedded devices targeted in this work (R10), unlike SGX-based schemes, which are restricted to Intel CPUs associated with laptop, desktop and server machines. EmLog satisfies the features of related cryptographic and trust-based proposals, such as resistance to truncation (R7) and re-ordering (R8) attacks, and public verifiability of log origin (R6). We also offer forward integrity protection for compromised block keys (R2) using more sophisticated key derivation, thus moving the cost-reward ratio further away from an attacker. By avoiding TPMs, however, we relinquish strong hardware tamper-resistance, and we urge caution of our work in high-security domains, e.g. military and governmental use, where complex hardware and side-channel attacks are reasonable threats.

5.7.3 Limitations Discussion

A major limitation with TEE-based logging systems, as identified by Holt [234], is when a compromised untrusted world maliciously generates spurious log messages with the aim of causing a backlog within the TEE, thus giving rise to Denial of Service (DoS) conditions. That is, the untrusted world generates more spurious logs than the time taken by the TEE to apply a particular tamper-resistance algorithm. One potential solution would be to trigger an alarm in the presence of unusually large numbers of logs, cease the algorithm and alert a trusted authority over directly from the TEE, e.g. using TLS over the GP Sockets API [105].

A related attack is when a heavily compromised untrusted world drops all messages transmitted from the TEE. This could include parameters sent to a remote network entity to begin a DH-based key exchange for remote log retrieval, or simply prevents any log entries being transmitted to the TEE. Once again, this is difficult to defend against with modern TEEs, and we can only identify the solutions discussed previously in attempting to contact a remote authority and recording anomalous activity internally.

Thirdly, our implementation used two world context switches per untrusted world log message. One area of future research would be to buffer some number d of log messages kept in the untrusted world before initiating the world context switch to the TEE. This would provide yet another 'dial' with which to customise our proposed scheme; by adjusting d, the number of context switches would be reduced by a relationship of $\frac{1}{d}$, which could significantly increase throughput (depending on the time needed to perform that switch, which was out-of-scope in this work). At present, we made a context switch for each log entry from the untrusted world, i.e. $d = 1$; our experiments hence represent the 'worst case' in this respect.

Lastly, secure logging schemes suffer from the challenge of log deletion, i.e. an adversary attempts to delete a substantial subset of logs from the device, including those with tamper-resistance. This is generally seen to be difficult to defend against robustly without dedicated WORM (Write-Once, Read-Many) storage, e.g. CD-ROM [20], [225], [241]. One solution might be to replicate WORM devices in an attached SE with a trusted path to the TEE, whereby logs are transmitted to the SE for persistent storage, but naturally requires additional security hardware. Another solution would be to minimise the number of stored logs prior to transmission to a trusted authority, who can then identify and remedy infected devices using an out-of-band method, i.e. in person.

5.8 Conclusion

This chapter introduced EmLog – a tamper-resistant logging scheme for modern constrained device using the GlobalPlatform TEE and ARM TrustZone. We began with a two-part review of related work in Section 5.2 by summarising cryptographic proposals and those reliant upon trusted hardware for tamper-resistant logging, before formulating the security goals and the threat model in Sections 5.3 and 5.4 based on past work. After this, we introduced the architectural design and proposed an improved log preservation algorithm for providing public verifiability of log origin and key exposure resilience. We described the implementation of EmLog in Section 5.6 and presented indicative performance results using diverse datasets in Section 5.7. For the first time, our work brings secure, TEE-based logging to mobile and embedded devices, and protects against strong software-based untrusted world and network adversaries. Our evaluation shows that EmLog yields five-times persistent storage overhead versus raw logs for applying tamper-resistance; runs within reasonable run-time memory constraints for TEE applications, as stipulated by the Linaro Working Group; and has a throughput of up to 625 logs/sec. In future work, we aim to investigate the following:

- *Group logging schemes for multiple devices*. Extend EmLog to allow secure and efficient sharing of logs between multiple TEEs. This could be used in schemes that compute trust scores prior to making group decisions [247], e.g. authenticating users via contextual data from multiple wearable devices.

- *TEE performance comparison*. We hope to evaluate EmLog under other TEE instantiations, namely Intel SGX and other GP-compliant TEEs, such as TrustTonic's Kinibi, especially for microcontrollers on low-end IoT devices.

Chapter 6

Remote Credential Management with Mutual Attestation for Trusted Execution Environments

6.1 Introduction

The previous chapters proposed TEEs to provide additional trust and security assurances to a particular application and its data – continuous authentication in Chapter 3 and tamper-resistant system logging in Chapter 5 – both under execution and persistently. This also included the challenge of TEE-to-TEE communication with the proposal of a secure and trusted channel protocol with mutual trust assurances in Chapter 4. This chapter expands upon TEE-to-TEE communication with mutual attestation for addressing the challenges of remotely managing TEE credentials throughout their lifetimes. These credentials may be used to authenticate decisions and data originating from the device, such as sensor measurements collected and transformed by the TEE from on-board I/O peripherals. In this chapter, we focus on centralised IoT domains where the application of TEEs is both compelling and justified, such as smart cities and Industrial IoT (IIoT), where TEE credential management is likely to be prominent.

6.1.1 Motivation

IDC [8] projects that the largest IoT investments in 2016 originated from manufacturing ($178bn), focussing on field servicing and asset management; transportation ($78bn), for monitoring and tracking the conditions and locations of freight; and utilities ($69bn), for electricity and gas infrastructure monitoring. Industrial IoT (IIoT) is motivated by the desire for greater automation and more intelligent fault detection and reporting, and environmental monitoring, like optimising climate controls to protect goods and maximise energy efficiency. IIoT also aims to enhance the information provided to decision-makers from factory-level, 'on-the-floor' processes in order to maximise capital and labour allocation. This is dubbed the 'fourth industrial revolution' by the German Industrie 4.0

initiative [248]. Capgemini recently estimated that IIoT may add up to $1.5tn to the global economy by 2022 [249].

Moreover, global investment in 'smart city' programmes is anticipated to rise from $80bn to $135bn between 2018–2021 [250]. Smart cities are augmented urban environments that use measurements from *in situ* sensing platforms to improve operational efficiency and public safety. In 2014, the Chicago Department of Innovation and Technology [251] deployed a city-wide network of sensor-equipped microcontrollers mounted on street lights to monitor carbon monoxide, nitrogen dioxide and particulate measurements to track air quality and warn those with respiratory conditions. The State of Gujarat, India, recently trialled flood predication from measurements collected from river bank sensors, which alerted citizens of potential floods using automated SMSs, phone calls, and local sirens [252]. Smart cities also incorporate assisted living for vulnerable users, e.g. fall detection systems and transmission of pollen warnings [253]; parking systems that use occupancy sensors to inform local drivers of vacant spaces to limit congestion [254]; and transport management systems that use occupancy and location sensors to predict abnormal congestion levels and delays [255].

As we have seen previously, it is the effect of compromised devices used to monitor and reactively inform larger sensitive processes, such as in manufacturing, critical infrastructure and assistive technologies, which raise the gravest concerns [256]. A compromised sensing device that surreptitiously feeds falsified measurements, e.g. temperature and battery consumption, to a reactive control system could conceal abnormal operating conditions and potentially endanger personnel. Ultimately, this rests in whether devices can be *trusted* to inform critical decision-making where malicious data could yield damaging consequences.

Despite widespread availability, managing IoT TEE credentials throughout their life-cycle has received limited attention. Such credentials may be used to authenticate and securely report measurements – typically taking the form of a certificate, or in user-authentication of device handlers using biometrics or passwords. Challenges arise when credentials require migrating, revoking, updating or backed up in a secure and trusted manner, such as in safely migrating device credentials to a replacement model that supersedes it.

All of these issues are compounded by that large numbers of TEEs, potentially thousands in IIoT and smart city deployments, must be administered, thus limiting the feasibility of human intervention.Furthermore, IoT systems may contain heterogeneous TEEs: Intel SGX, as we have seen, is restricted to Intel CPUs on more powerful devices, while ARM TrustZone is better suited to constrained devices with ARM SoCs, such as microcontrollers and SBCs. To avoid disclosing credentials to potentially untrusted elements, managing credentials *between* TEE-enabled devices, such as in migration, ought to be conducted such that *both*

end-points are authenticated and attested by one another. This may also extend to backing-up or updating sensitive credentials to a remote sever-side TEE. It is important that credential management procedures are *generic* to a particular underlying TEE implementation, *mutually trusted* for operations between TEEs, and *verified* to assure its correctness. We develop the work from Chapter 4 to expand our proposed mutual attestation protocol to facilitate a range of remote credential management operations *between* TEEs.

6.1.2 Contributions

For the first time, we analyse and address four critical challenges when managing heterogeneous TEE credentials over its lifetime with bi-directional trust assurances for remote *migration* (Section 6.4), *revocation* (Section 6.5), *backups* (Section 6.7), and *updates* (Section 6.6). We also show how secure log retrieval – developing work from Chapter 5 – can be realised with mutual attestation in Section 6.8. We focus principally on centralised deployments comprising a Trusted Service Manager (TSM) – a trusted third-party that manages credentials, as per the GlobalPlatform specifications [107] – and where the application and cost of maintaining TEEs on constrained devices is compelling and justified, e.g. IIoT and smart cities. This paper presents the following contributions:

- A detailed analysis and examination of past literature is presented on four credential management challenges facing the deployment of TEEs in centralised IoT deployments.
- Five proposed protocols for facilitating IoT TEE credential management. The protocols are agnostic of the TEE, communication medium, and employ a Trusted Service Manager (TSM) [107].
- Each proposed protocol is subjected to formal symbolic verification using the Scyther analysis tool, which found no attacks. We freely release the protocol verification specifications for further research and analysis, which are available online at: `https://www.dropbox.com/s/uq0hftj6b6c1zux/remote-credential-scyther-scripts.zip`.

This chapter is based on the following on work:

- (In Press) C. Shepherd, R. N. Akram, and K. Markantonakis. "Remote Credential Management with Mutual Attestation for Trusted Execution Environments," Proceedings of the 12th IFIP International Conference on Information Security Theory and Practice, ser. WISTP '18, Springer, 2019.

6.2 Deploying TEE Credentials

In this section, we state the assumptions regarding the deployment of TEE-resident credentials. Section 2.4.7 of Chapter 2 described a proposed deployment model for TEEs within the GlobalPlatform TEE specifications. The reader is referred to briefly revisit this section in order inform subsequent discussions on credential management.

Credentials are typically programmed initially into a TEE during systems integration after the production and delivery of TEE hardware from a silicon vendor. After its deployment, a Trusted Service Manager (TSM) – incorporated into the device manufacturer or outsourced to a trusted third party – is responsible for maintaining the TEE, its TAs, and any on-board credentials thereafter. Section 2.4.7 of Chapter 2 described how, within the GP TEE model, Security Domains (SDs) are personalised with keys that authenticate remote TEE management operations and separate the control of applications under one stakeholder from those belonging to another. These TEE-specific details are abstracted for the purposes of this chapter, which aims principally to identify and present *generic* credential management protocols that are agnostic to the underlying TEE implementation and architecture.

At present, the GlobalPlatform management functions govern only the high-level operational lifecycles of TEE, SD and TA on the device. They do not concern the way credentials and other data held within TAs is retrieved or updated, nor how TEEs and TAs operate within specific application scenarios. Explicitly, the GP TEE TMF specifications state that *"an implementation can support several protocols to fulfil different security constraints, different business needs, or local rules"* [107].

6.2.1 TEE Credential and Profile Management

We define TEE credentials as *the set, C, of key material, certificates, and other authentication data contained in a TA*. Credentials may also comprise a key derived from a password-based key derivation function using a password from an operator, or one provisioned later over a secure channel to the TA. They may also be encapsulated in a model that maps contextual data, like sensor inputs, to authentication states using machine learning, such as Continuous Authentication (CA) and other biometrics. Ultimately, these credentials serve as the evidence for authenticating privileged operations to a remote party. Secure Elements (SEs), e.g. Universal Integrated Circuit Cards (UICCs), are another means for application and credential hosting on constrained devices. The reader is referred back to Chapter 2 in which a detailed comparison of secure and trusted execution

technologies was presented, including a feature comparison and their capability against various adversaries.

6.3 Protocol Design

The GP Trusted Management Framework (TMF) does not stipulate specific secure channel protocols, but only that the TSM and TEE should mutually authenticate over a channel that preserves the *"integrity and the confidentiality of the exchanges,"* and addresses replay attacks against a Dolev-Yao adversary [107]. These basic requirements omit desirable features identified in existing trusted computing literature [257], [258], such as trust assurances that the target TEE is authentic and integral. As discussed in Chapter 4, this is usually realised using remote attestation where system measurements, e.g. bootloader and TA binaries, are measured by a trusted entity. RA protocols authenticate the platform to a remote verifier using these measurements. In sensitive IoT deployments, mutually authenticating *both* end-points may be desirable, e.g. TEE-to-TEE communication for securing backups from a GP TEE to a cloud-based backup enclave using Intel SGX. Here, RA protocols can be conducted on each end-point or using a *mutual attestation* protocol, as proposed in Chapter 4.

In this work, we maintain trust assurances by only transmitting credentials *between* the TEEs and/or a TSM, without revealing them to any untrusted device elements over a network We also introduce an abstract controller, a generic IoT Controller (IC), with which the host edge device may communicate with the IoT deployment network, is issued management commands, performs firewalling, device blacklisting and routing with other devices e.g. on a local- (LAN), metropolitan (MAN) or wide-area network (WAN). We now explicitly state the protocol features we aim to achieve, which are sourced from related mutual attestation literature [257], [258] (see Section 4.4 of Chapter 4), and the requirements stipulated by the GlobalPlatform TMF specification [107]:

S1) *Mutual key establishment*: a shared secret key is established for communication between the two entities.

S2) *Forward secrecy*: the compromise of a particular session key should not affect previous protocol runs.

S3) *Trust assurance*: the proposal shall allow third-parties to verify the TEE's integrity post-deployment to provide assurances that credentials were sourced from an integral and authentic platform.

S4) *Mutual trust verification*: both end-points shall successfully attest the state of the other before permitting the establishment of a secure channel.

S5) *Mutual entity authentication*: each communicating end-point shall authenticate the other's identity to counter masquerading attempts.

S6) *Denial of Service (DoS) resilience*: resource allocation shall be minimised at both end-points to prevent DoS conditions from arising.

S7) *Key freshness*: the shared key shall be fresh to the session in order to prevent replay attacks.

F1) *Avoidance of additional trust hardware*: the protocol shall avoid the need for additional security hardware, e.g. TPMs and SEs, other than the TEE.

F2) *TEE-agnostic*: the protocols shall remain agnostic of the underlying TEE architecture to facilitate interoperability.

To be clear, we assume the same threat model defined in Section 4.4.1 of Chapter 4 to apply when conducting communications between TEEs on remotely located devices.

6.3.1 Setup Assumptions

A public-key infrastructure is assumed in which a trusted CA issues certificates to the TSM, TAs, and the backup (BA), revocation (RA) and maintenance (MA) authorities – developed in the forthcoming sections – for managing backups, revoking credentials and physically maintaining devices respectively. The initial TA certificates are assumed to be securely provisioned before deployment to the host device. The TEE itself is assumed to be trusted and to possess certified device-specific attestation and command keys for signing attestation responses.

The credentials are also assumed to be securely stored by the TEE, e.g. using the sealing abstraction with a device-specific storage root key (as stipulated by the GlobalPlatform TEE PP, see Section 2.4.7 in Chapter 2). Secure means of key generation, derivation and random number generation is also assumed. Note that many TEEs *cannot initiate network connections*, and must rely on the untrusted OS – the Rich Execution Environment (REE) – to implement such interfaces, e.g. TCP/IP sockets. GP TEEs may perform this via the GP Sockets API [105], but it is not generalisable to Intel SGX. We illustrate the REE throughout this paper in the proposed protocols, but it may be omitted for TEEs with direct networking capability.

6.4 Migration

TEE migration is the process of transferring and re-provisioning credentials from TA_A to TA_B between *distinct* TEEs. In IoT, migration is beneficial in preserving

credentials during a device replacement or relocation, where credentials can be securely transferred to other units without incurring reinitialisation costs. This is likely to occur if a device is replaced by a successor model, i.e. due to obsolescence, potentially in a separate physical location. Migrating credentials across TEEs has already attracted some attention in related literature [259], [260]. We summarise these schemes and their contributions.

6.4.1 Related Work

Arfaoui et al. [259] present a privacy-preserving protocol for migrating credentials between GlobalPlatform TEEs. A PKI-based protocol is proposed for authorising the credential transfer between the TEE, TSM and the TEE's service providers, each of whom controls a set of TAs in a Security Domain (SD). After authorisation, a second protocol is used to transfer data between the SDs by each service provider using a PKI- or password-based authenticated key exchange. Both protocols are subjected to formal verification using the AVISPA analysis tool. While the authors note the importance of remote attestation during TSM-TEE authorisation, it is not presented or verified as part of the protocol; it is also omitted during the credential transfer process between the TEEs. Moreover, *mutual* trust assurances between the TSM and the TEEs is not discussed.

Kostiainen et al. [260] tackle migration for TEE open credential platforms where service providers can provision arbitrary credentials for, for example, virtualised access control cards. They propose encrypting and backing-up credentials on a trusted server using a tokenised password known only to the user. The credentials are migrated by re-entering the password, which is re-tokenised on the receiver device, and transmitted and verified by the backup server that releases the encrypted credentials. Similar to [259], the proposal lacks trust assurances between the TSM and both TEEs.

6.4.2 Proposed High-level Migration Procedure

Credentials must be deleted on the device from which they are migrated, while transferring them over a secure channel with mutual trust assumptions between TA_A and TA_B, which is absent in existing literature. In Figure 6.1, we show how migration can be performed between two disparate TAs, accounting for the shortcomings in related work; the messages are described as follows:

1. A mutual remote attestation protocol, such as BTP proposed in Chapter 4, is executed between TSM and TA_A to bootstrap a secure and mutually trusted channel (STCP).

2. TSM transmits the begin migration command to TA_A.

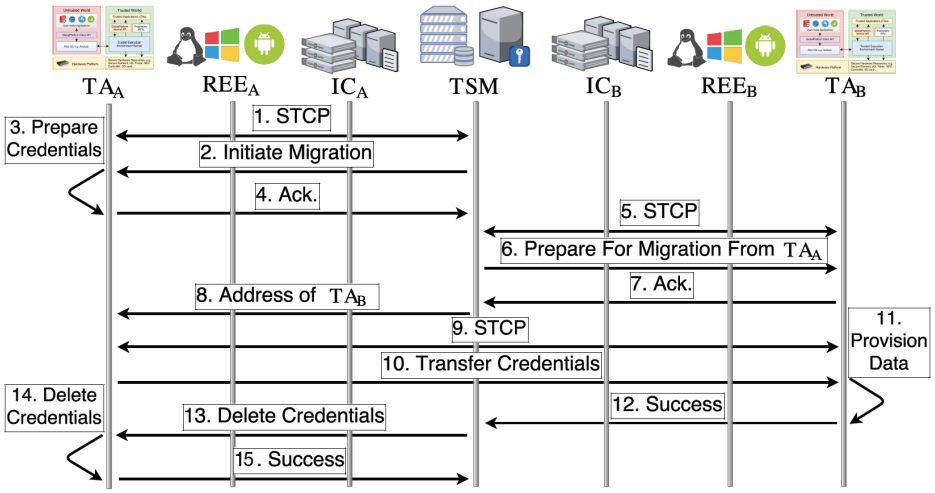


FIGURE 6.1: Proposed TEE credential migration procedure.

3. TA_A unseals the credentials from its secure storage for transmission.
4. TA_A acknowledges to TSM that the credentials were unsealed successfully.
5. A separate STCP instance is executed between (TSM, TA_B).
6. TSM sends a message to TA_B to prepare for credential/profile (re-)provisioning.
7. TA_B acknowledges to TSM that it is ready to receive credentials.
8. TSM transmits the ID of TA_B to TA_A to which to transmit its unsealed credentials.
9. An STCP is formed using mutual remote attestation between TA_A and TA_B.
10. The credential transfer occurs between TA_A and TA_B.
11. The transferred credential is provisioned into the secure storage of TA_B.
12. TA_B acknowledges its credential re-provisioning success to TSM.
13. TSM instructs TA_A to delete its credentials.

14-15. TA_A deletes the migrated credential and acknowledges its success TSM.

6.4.3 Discussion

The high-level procedure uses three secure and trusted channels (STCPs) with mutual attestation between (TSM, TA_A), (TSM, TA_B) and (TA_A, TA_B), thus addressing the absence of trust assurances in existing schemes. The proposal also avoids unnecessarily exposing credentials to the TSM by transmitting data

directly between the mutually authenticated TAs. Both ICs require no additional cryptographic operations and act only as switches for communicating the protocol messages to the correct IoT device in its intended. Implicitly, the protocol avoids specifying TEE-specific functionalities. Rather for F2 (TEE agnosticism), we abstract the protocol appropriately to allow migrations between heterogeneous TEEs by allowing either a GP TEE application or Intel SGX enclave to act as either TA. For TEE-specific implementation guidance, the reader is referred to existing work such as the GlobalPlatform TMF specifications [107], and the work by Arfaoui et al. [259] for managing and authorising SDs on the GP TEE. In Section 6.9, we specify the protocols and procedures formally, and detail an enhanced mutual attestation protocol for STCP for satisfying the remaining features listed in Section 6.3.

Atomicity

Credential migrations should be atomic to preclude the rise of data consistency issues, particularly to prevent credential loss and unintended duplication. For instance, if party A transmits its credential, c, and deletes it before being successfully re-provisioned by party B, e.g. due to a software failure or by an adversary who drops messages containing c before reaching B, then c becomes lost. A related issue occurs when A transmits c to B, who successfully re-provisions it, but A subsequently fails to delete c locally. In this case, c is now unintendedly replicated across two locations. A further is trusting whether A and B actually *did* delete and re-provision the credential respectively, and not just to have claimed so.

In the presented procedure, the deletion of c held by TA_A (stage 14 in Figure 6.1) is performed *iff* the credential was successfully transferred and subsequently re-provisioned by TA_B beforehand (stages 10-12). For the first two challenges, it is critical for TSM to be aware of the re-provisioning (stage 12) and deletion success messages (stage 15), and remedial action ought to be taken if either message is not received after an acceptable time limit. This includes voiding and attempting to repeat the procedure – valid up to stage 13, after which the initial credential may already be deleted by TA_A – or triggering manual intervention to investigate a potential networking or hardware failure on either device.

The case of verifying whether the parties do, indeed, delete and re-provision credentials is more difficult. In this work, we assume that the TEE and TA code would be subjected to rigorous testing and development practices to correctly delete and re-provision credential material at the appropriate stages of the protocol. Additionally, remote attestation is used by all parties to gain evidence about the state of the target TEE with whom it is communicating. Ideally, as noted by Kostiainen et al. [260], credentials should be managed on a trusted storage

medium accessible only to the TEE, i.e. replay-protected non-volatile memory or a secure element, which is the case we assume in this work.

6.5 Revocation

Credentials may be revoked if they reach the end of their predefined lifespan(s), as part of a key rotation policy; if the OEM discovers a vulnerability in the TA or TEE kernel code, and the credentials were potentially compromised; or the device is retired from service. As discussed in Section 6.2.1, compromised IoT credentials could be abused to authenticate malicious decisions and data; revocation is vital in preventing compromised devices from influencing larger critical processes.

6.5.1 Related Work

Revocation has been previously studied in TPM and smart card literature. Chen and Li [261] address credential revocation in Direct Anonymous Attestation (DAA), as used in TPM 2.0. DAA, like group signatures, allows the signer to demonstrate knowledge of its individual private key corresponding to the group's public key without revealing its identity. Revoking DAA credentials is challenging as the signer's identity is not revealed, even to the group manager with the group key. The authors review two solutions: *rekey-based*, where the issuer updates its public key (which may or may not include its corresponding secret key), and allows only legitimate non-revoked signers to update their credentials; and *Verifier-Local Revocation (VLR)*, where the verifier inputs a revocation list, RL, to the DAA's verification function and accepts only signatures from signers, $S \notin RL$.

Lueks et al. [262] address revoking attribute-based credentials (ABCs) for smart cards anonymously. A Revocation Authority (RA) possesses a revocation list (RL) of anonymous revocation values, $g^r_{\epsilon,v}$, submitted by the user or verifier (user- and system-instantiated revocation), where r is the revocation value in the user's credential. A revocation 'epoch', ϵ (corresponding to a time period) is used to provide unlinkability by re-computing and re-sending the new valid RLs to the verifiers at each epoch; that is, $RL_{\epsilon,V} = sort(\{g^r_{\epsilon,V} \mid r \in MRL\})$, where MRL is the master revocation list. Using bloom filters, this occupies only 4–8MB for 2^{21} revoked credentials depending on the probability tolerance.

Katzenbeisser et al. [263] propose revocation for TPM 1.2 using blacklisting and whitelisting. Blacklisting orders revoked keys into a hash chain, where the final hash value, h, is stored in a TPM register. When a key is revoked, a new hash entry is created and the TPM updates h accordingly. Before loading a key, the TPM tests whether it is in the blacklist before unsealing it. Whitelisting incrementally creates keyed hashes of each permitted key with the value of the

TPM's internal secure counter as the whitelist's 'version'. A key is valid *iff* the keyed hash counter value matches the TPM's internal counter. Revocation is performed by incrementing the TPM's counter and updating all non-revoked hashes with the new value.

6.5.2 Proposed High-level Revocation Procedure

Privacy-preserving credential schemes, e.g. DAA and anonymous credentials, are beneficial in verifying credentials without divulging users' identities. However, the focus of this work is on inherently centralised IoT deployments, such as smart cities and IIoT architectures, where the concern of violating credential privacy has far fewer consequences than government electronic ID cards or TPMs on consumer devices. Relaxing this constraint provides us with some headway to pursue simpler, PKI-based solutions for providing a baseline protocol for TEE credential revocation with mutual attestation. Moreover, we discuss two approaches for IoT based on blacklisting and whitelisting using a trusted RA. The procedure is illustrated in Figure 6.2 and described as follows:

1. TA and TSM form a STCP using mutual remote attestation to verify the platforms' integrity and authenticity.
2. TSM instructs TA to reveal the current credentials in use, C, which are then unsealed from storage, e.g. encrypted in untrusted storage or an SE.
3. C is transmitted to the TSM over the STCP.
4. The TSM forms a STCP with the revocation authority, RA, who maintains the master revocation list of whitelisted or blacklisted credentials.
5. TSM submits C to RA, who returns a list of the revoked credentials in C, i.e. $RC \subseteq C$, from its master revocation list (MRL).
6. RA returns the revoked credentials, RC, to TSM.

Next, if RC is not empty, the TSM informs IC of the credentials to disallow in future transactions. The TSM also instructs the TA to update the revocation status of $RC \in C$ internally to prevent further use. Note that a malfunctioning device may fail to update this status and reuse revoked credentials. The use of revoked credentials should be reported to MA responsible for decommissioning compromised devices. Like [262], delegating revocation list management to RA removes the burden of potentially multiple verifiers synchronising a single list. The TSM can submit a lookup request to the RA, who queries the blacklist or whitelist in $O(1)$ using an associative array. Traditional revocation schemes, e.g. PKI certificate revocation, use blacklisting. The RA maintains a master

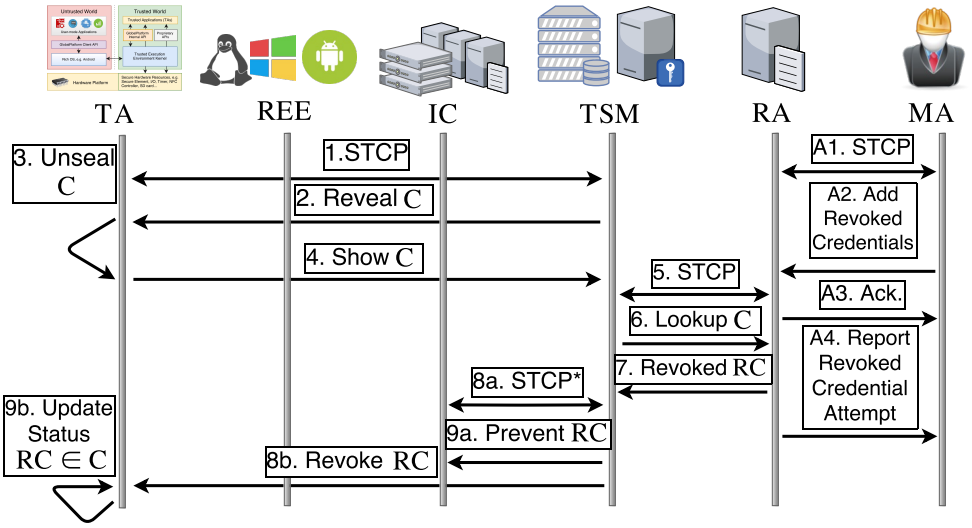


FIGURE 6.2: Proposed high-level credential revocation procedure. **8a. STCP between IC and TSM is unnecessary if the route is considered trustworthy.*

revocation list (MRL) of revoked credentials that should not be used in any IC transaction; the MA may submit credentials it wishes to revoke to RA (*maintainer-instantiated revocation*). RA checks the revocation status of C by verifying $C \notin MRL$. Whitelisting, conversely, comprises a list of only the permitted credentials. A credential is revoked by removing it from the whitelist and, if applicable, updating the list with its replacement. Revocation is tested by verifying $C \in MRL$. The use of blacklists and whitelists is context-specific; whitelisting may be preferred over blacklisting when the valid credential set is small and changes infrequently.

6.6 Updates

Remotely updating IoT credentials is beneficial during routine renewal schedules, e.g. X.509 certificates that reach their validity expiry date; replacing a revoked credential after, for example, the detection of a compromise; or the device is relocated and the organisational unit to which the credential is issued is no longer valid. Generally, credential update is the process of securely replacing an obsolete credential, c_i, with a newly issued c_i'. Once replaced, c_i should be revoked to prevent the re-use of obsolete credentials. Little work has been conducted academically for updating TEE credentials; this is likely due to the the simplicity of a TSM creating a secure channel and modifying the credential or, indeed, the similarity with restoring backups. Updates can be considered a variation of backup restoration where c_i' is retrieved from a trusted server, rather than c_i

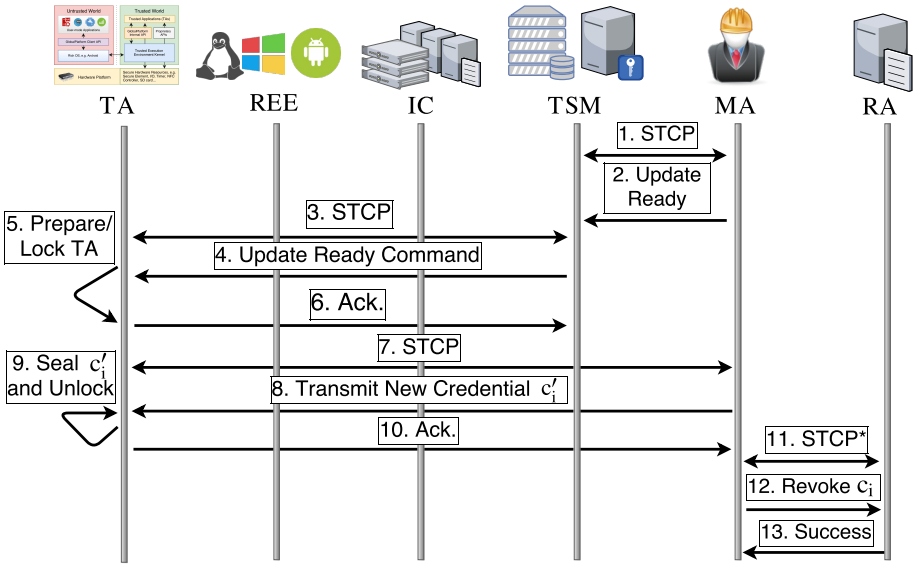


FIGURE 6.3: Proposed credential update procedure with mutual attestation.*11. (MA, RA) STCP is unnecessary if the route is trustworthy.

registered by the user. Revoking c_i, achievable using the process described in Section 6.5.

6.6.1 Proposed High-level Update Procedure

We reintroduce the maintenance authority (MA) from Section 6.5, which issues credential updates, e.g. as part of a rotation policy and/or recognising potentially comprised devices/credentials. If desired, the MA is also responsible for registering obsolete credentials with the revocation authority (RA). The high-level update mechanism is as follows:

1. TSM establishes an STCP with MA, who procures and provides the necessary credential updates.

2. MA notifies the TSM of an updated credential. This may include identities of which TEEs need updated or, indeed, all TEEs.

3-4) An STCP is conducted between (TSM, TA), and transmits an update preparation command to TA.

5) TA is locked, i.e. prevented from interacting with the REE, until the update is performed to prevent using outdated credentials.

6) TA acknowledges to TSM that it is ready to update.

7-8) TA establishes a STCP with MA to receive the update, and MA transmits the updated credential, c_i', to TA.

9) TA seals c_i' to its secure storage for future use; c_i should be deleted internally before unlocking.

10) TA acknowledges to MA that c_i' was initialised successfully.

11-13) (MA, RA) use an STCP to white/blacklist obsolete credential c_i.

6.7 Backup

Backup is the process of securely retrieving the set of credentials, C, belonging to a TA for remote storage. In standard security practice, backups often form the cornerstone of a disaster recovery plan – as stipulated by ISO 27001:2013 [223] – for recovering data from corruption and accidental deletion. Backups may also constitute part of a data retention policy, where device data is copied for local analysis and evidence of regulatory adherence. Like migration, the credential/profile backup is beneficial in IoT if the original was entered by hand (human input), e.g. password, or derived from a trained machine learning model, e.g. behavioural biometrics, which is time-consuming to retrain. The restoration of backups can be addressed using the Remote Update protocol proposed in Section 6.6. Next, we examine related work in the backup of remote credentials aboard secure and trusted execution technologies.

6.7.1 Related Work

Kostiainen et al. [264] address TEE credential backup, restoration and disabling on consumer mobile phones, and propose two solutions. The first uses a SE, a SIM card, where the TEE credentials are protected under a SIM-specific key provisioned by its provider. This allows the user to uninstall a familiar hardware element, i.e. the SIM, before releasing the device for repairs or to lend to an untrusted user. Upon reinserting the SIM, an on-board TEE credential manager is used to decrypt and re-initialise the encrypted credentials. The second solution involves the use of a removable microcontroller to counter an honest-but-curious remote server, RS. RS has a shared key K_s with the TEE, and stores the backups using a secure counter for rollback protection. To prevent RS reading the credentials, the TEE encrypts them under a separate key, K, derived from a local counter on the microcontroller, and re-encrypts them under K_s.

Akram et al. [265] examine credential restoration for multi-application smart cards on smartphone SEs. They enhance a Trusted Environment and Execution Manager (TEM), which dynamically enforces the smart card's run-time security policies, to implement credential backup and restoration mechanisms. A Backup and Restoration Manager (BRM) is added to the smart card software stack that interfaces with a TEM-resident backup token handler, which stores tokens issued

by application service providers. The user registers the BRM with a backup server (BS); when the user wants to backup, the BRM encrypts the token(s) and communicates them to BS. To restore data, e.g. to a new card, the user provides the BRM with his/her BS credentials to download the backed-up tokens. A secure channel is formed and the token(s) authenticate the credential restoration from the service provider(s).

6.7.2 Proposed High-level Backup Procedure

We introduce a trusted Backup Authority (BA) responsible for storing retrieved credentials. This may be a cloud-based storage provider or Hardware Security Module (HSM) possessed by the credential issuing authority. The precise means by which the BA securely stores credentials is considered out-of-scope in this work. The proposed procedure between the target TA and BA is shown in Figure 6.4 and described below:

1-4) TSM and BA establish a secure and trusted channel with mutual attestation, and TSM requests BA to prepare for backup.

5-6) TSM forms an STCP with TA, commanding it prepare the credentials for remote backup. TSM provides the identity of BA that performs the backup.

7-8) TA unseals the credentials to transmit to BA, and notifies TSM that they were unsealed successfully.

9-12) TA and BA form a STCP over which C is transmitted and stored securely by BA. While TSM is considered trusted, the direct connection between TA and BA mitigates the risk of unnecessary credential exposure to TSM.

13) BA notifies TSM that TA was backed-up successfully.

6.8 Secure Log Retrieval

This section develops work presented in the previous chapter, Chapter 5, on tamper-resistant system logging on constrained devices with TEEs. Recording system attributes, such as error details, user activity and potential security events, often underpin underpin audit and after-the-fact forensic investigations. As we seen, logs are also used to enforce user accountability, event reconstruction and intrusion detection, and are routinely targeted by attackers to conceal wrongdoing. However, while a wealth of literature exists for producing tamper-resistance records, little research examines the secure and trustworthy transmission of logs.

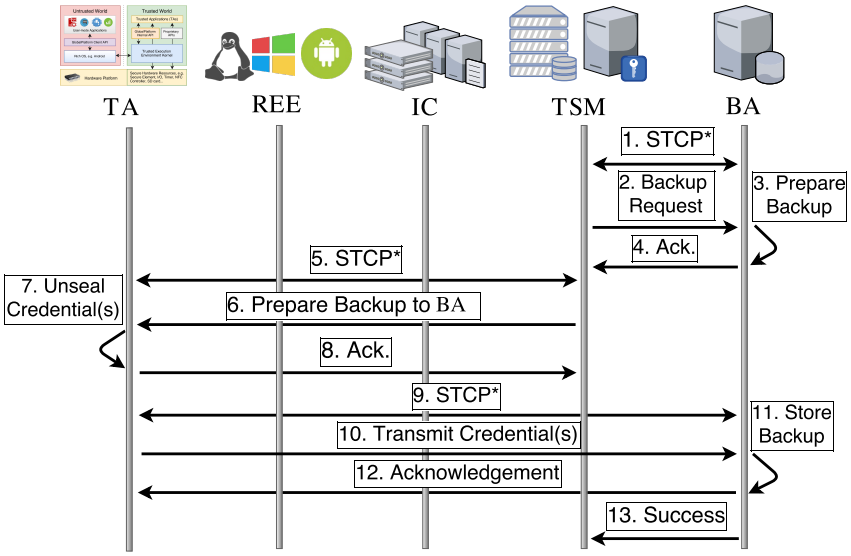


FIGURE 6.4: Proposed high-level remote credential backup procedure.

6.8.1 Proposed High-level Audit Log Transmission Procedure

Most schemes, e.g. Böck et al. [224] and Sinha et al. [225], stipulate the use of a secure channel, such as TLS, while Karande et al. [241] and our contribution in the previous chapter proposed one-way and mutual remote attestation channels respectively. However, we did not formalise nor verify this retrieval process explicitly. In this section of this chapter, we address this shortfall by proposing a log transmission protocol in the context of centralised IoT deployments. To this end, we propose a secure and mutually trusted log retrieval process, as shown in Figure 6.5. It is identical in function to the Remote Backup procedure described in Section 6.7 and Figure 6.4, but the role of the Backup Authority (BA) is replaced with an Auditing Authority (AA) with the capacity to store, process and analyse the transmitted logs. We refer the reader to Section 6.7.2 for a more detailed analysis of this procedure.

6.9 Procedure Analysis

We formalise the procedures from those presented in the previous sections, which are listed in Procedures 1 to 5 using the notation from Table 6.1. In Section 6.3.1, we outlined the set-up assumptions behind the proposals, such as key establishment, which we refer to frequently in Sections 6.9.1 and 6.9.2, which present high-level and formal protocol analyses using Scyther respectively.

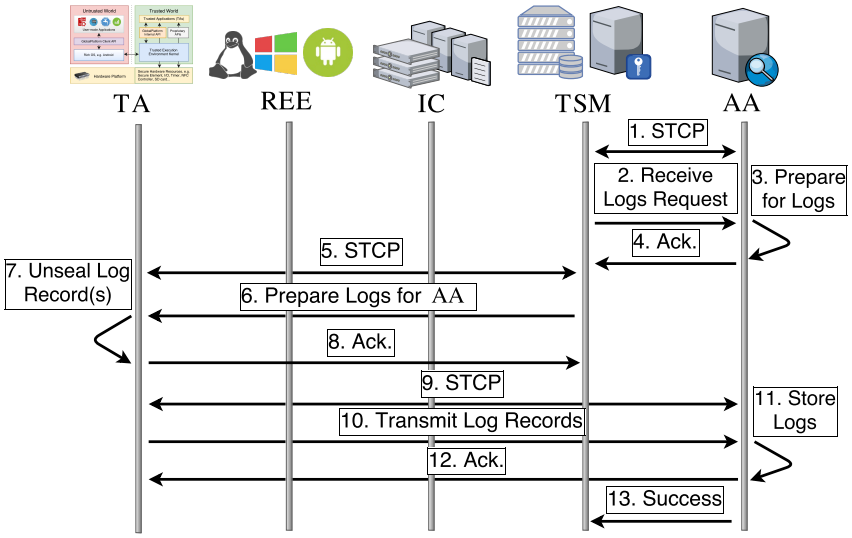


FIGURE 6.5: Proposed high-level audit record transmission procedure.

6.9.1 High-level Analysis

Each proposed procedure is underpinned by a variant of the mutual remote attestation protocol proposed in Chapter 4. This protocol (Protocol 6), which establishes a TEE-to-TEE secure channel, is modified to support authenticated encryption with associated data (AEAD). AEAD instantiations, namely as AES in GCM mode, exhibit a 2-3 times performance benefit over AES + HMAC-SHA constructions [266], [267]. This is opposed to our previous proposals in Chapter 4 that considered AES in a non-AEAD mode of operation, e.g. AES-CBC, with an additional HMAC tag for confidentiality, integrity and data authenticity. The theoretical security of AES-GCM is well-understood [268]. Practically, it also reduces the number of GP Internal API [78] calls substantially, from eight calls for HMAC-SHA+AES, to only four GP TEE API calls, thus reducing the protocol's implementation complexity.

This modified BTP forms the basis for establishing the trust relationships between the TEEs, which was outlined in the security goals in Section 6.3. The protocol is based on ephemeral Diffie-Hellman key agreement, thus achieving session forward secrecy (S2), mutual key establishment (S1) and key freshness (S7). Additionally, the protocol conducts mutual attestation using the quoting abstraction for verifying the target platform's integrity and authenticity, which satisfies S3 and S4 (mutual trust verification). The assumption of PKI and the provisioning of key pairs to sign attestation values, command instructions, e.g. $Prep\_Backup$ and $Revoke\_Success$, and the shared secret, provides mutual entity authentication (S5).

The protocols avoid the use of additional trusted hardware for constrained

Procedure 1 Proposed Migration Procedure with BTP (MP-BTP)

1: Execute BTP (TSM, TA_1)
2: $TSM \rightarrow TA_1 : [(Prep\_Migrate \,||\, X_{TSM})\sigma_{TSM}]_{AE_K}$
3: $TA_1 \rightarrow TSM : [(TA_1\_Ack \,||\, X_{TA1})\sigma_{TA1}]_{AE_K}$
4: Execute BTP (TSM, TA_2)
5: $TSM \rightarrow TA_2 : [(Prep\_Migrate \,||\, X'_{TSM})\sigma_{TSM}]_{AE_{K'}}$
6: $TA_2 \rightarrow TSM : [(TA_2\_Ack \,||\, X'_{TA2})\sigma_{TA2}]_{AE_{K'}}$
7: $TSM \rightarrow TA_1 : [(ID_{TA2} \,||\, X_{TSM})\sigma_{TSM}]_{AE_K}$
8: Execute BTP (TA_1, TA_2)
9: $TA_1 \rightarrow TA_2 : [(C \,||\, X''_{TA1})\sigma_{TA1}]_{AE_{K''}}$
10: $TA_2 \rightarrow TA_1 : [(Stored\_C\_Ack \,||\, X''_{TA2})\sigma_{TA2}]_{AE_{K''}}$
11: $TA_2 \rightarrow TSM : [(TA_2\_Done \,||\, X'_{TA2})\sigma_{TA2}]_{AE_{K'}}$
12: $TSM \rightarrow TA_1 : [(Delete\_Creds \,||\, X_{TSM})\sigma_{TSM}]_{AE_K}$
13: $TA_1 \rightarrow TSM : [(TA_1\_Done \,||\, X_{TA1})\sigma_{TA1}]_{AE_K}$

Procedure 2 Proposed Backup Procedure with BTP (BP-BTP)

1: Execute BTP (TSM, BA)
2: $TSM \rightarrow BA : [(Prep\_Backup \,||\, X_{TSM})\sigma_{TSM}]_{AE_K}$
3: $BA \rightarrow TSM : [(BA\_Ack \,||\, X_{BA})\sigma_{BA}]_{AE_K}$
4: Execute BTP (TSM, TA)
5: $TSM \rightarrow TA : [(Prep\_Backup \,||\, X'_{TSM})\sigma_{TSM}]_{AE_{K'}}$
6: $TA \rightarrow TSM : [(TA\_Ack \,||\, X'_{TA})\sigma_{TA}]_{AE_{K'}}$
7: $TSM \rightarrow TA : [(ID_{BA} \,||\, X_{TSM})\sigma_{TSM}]_{AE_K}$
8: Execute BTP (TA, BA)
9: $TA \rightarrow BA : [(C \,||\, X''_{TA})\sigma_{TA}]_{AE_{K''}}$
10: $BA \rightarrow TA : [(Backup\_Ack \,||\, X''_{BA})\sigma_{BA}]_{AE_{K''}}$
11: $BA \rightarrow TSM : [(Backup\_Done \,||\, X'_{BA})\sigma_{BA}]_{AE_{K'}}$

Procedure 3 Proposed Revocation Lookup Procedure with BTP (RL-BTP)

1: Execute BTP (TSM, TA)
2: $TSM \rightarrow TA : [(Reveal\_Creds \,||\, X_{TSM})\sigma_{TSM}]_{AE_K}$
3: $TA \rightarrow TSM : [(C \,||\, X_{TA})\sigma_{TA}]_{AE_K}$
4: Execute BTP (TSM, RA)
5: $TSM \rightarrow RA : [(Lookup \,||\, C \,||\, X'_{TSM})\sigma_{TSM}]_{AE_{K'}}$
6: $RA \rightarrow TSM : [(Revoked \,||\, RC \,||\, X'_{RA})\sigma_{RA}]_{AE_{K'}}$
If $RC \neq \emptyset$:
7: $TSM \rightarrow TA : [(Revoke \,||\, RC \,||\, X_{TSM})\sigma_{TSM}]_{AE_K}$
8: $TA \rightarrow TSM : [(Revoke\_Ack \,||\, X_{TA})\sigma_{TA}]_{AE_K}$
9: (Optional) Execute BTP (TSM, IC)
10: $TSM \rightarrow IC : [(Revoke \,||\, RC \,||\, X''_{TSM})\sigma_{TSM}]_{AE_{K''}}$
11: $IC \rightarrow TSM : [(Revoke\_Ack \,||\, X''_{IC})\sigma_{IC}]_{AE_{K''}}$

Procedure 4 Proposed Revocation Reporting Procedure with BTP (RR-BTP)

1: Execute BTP (MA, RA)
2: $MA \rightarrow RA : [(Revoke \,||\, C \,||\, X_{MA})\sigma_{MA}]_{AE_K}$
3: $RA \rightarrow MA : [(Revoke\_Ack \,||\, X_{RA})\sigma_{RA}]_{AE_K}$
4: If reported credentials ($RepC \neq \emptyset$):
$RA \rightarrow MA : [(Report \,||\, RepC \,||\, X_{RA})\sigma_{RA}]_{AE_K}$

Procedure 5 Proposed Update Procedure with BTP (UP-BTP)

1: Execute BTP (MA, TSM)
2: $MA \rightarrow TSM$: $[(Update\_Ready \mid\mid X_{MA})\sigma_{MA}]_{AE_K}$
3: $TSM \rightarrow MA$: $[(TSM\_Ack \mid\mid X_{TSM})\sigma_{TSM}]_{AE_K}$
4: Execute BTP (TSM, TA)
5: $TSM \rightarrow TA$: $[(Prep\_Update \mid\mid X'_{TSM})\sigma_{TSM}]_{AE_{K'}}$
6: $TA \rightarrow TSM$: $[(TA\_Ack \mid\mid X'_{TA})\sigma_{TA}]_{AE_{K'}}$
7: Execute BTP (TA, MA)
8: $TA \rightarrow MA$: $[(Get\_Update \mid\mid X''_{TA})\sigma_{TA}]_{AE_{K''}}$
9: $MA \rightarrow TA$: $[(new\_c_j \mid\mid X''_{MA})\sigma_{MA}]_{AE_{K''}}$
10: $TA \rightarrow MA$: $[(New\_Cred\_Ack \mid\mid X''_{TA})\sigma_{TA}]_{AE_{K''}}$
11: Execute BTP (MA, RA)
12: $MA \rightarrow RA$: $[(Revoke \mid\mid c_i \mid\mid X'''_{MA})\sigma_{MA}]_{AE_{K'''}}$
13: $RA \rightarrow MA$: $[(Revoke\_Success \mid\mid X'''_{RA})\sigma_{RA}]_{AE_{K'''}}$

Protocol 6 Modified Bi-directional Trust Protocol (BTP) from Chapter 4

1: $A \rightarrow B : ID_A \mid\mid ID_B \mid\mid n_A \mid\mid g^A \mid\mid AR_B$
2: $B \rightarrow A : ID_B \mid\mid ID_A \mid\mid n_B \mid\mid g^B \mid\mid \left[(X_B)\sigma_B \mid\mid (V_B)\sigma_B\right]_{AE_K} \mid\mid AR_A$
$X_B = H(ID_A \mid\mid ID_B \mid\mid g^A \mid\mid g^B \mid\mid n_A \mid\mid n_B)$
$V_B = Q_B \mid\mid n_B \mid\mid n_A$
3: $A \rightarrow B : [\,(X_A)\sigma_A \mid\mid (V_A)\sigma_A\,]_{AE_K}$
$X_A = H(ID_A \mid\mid ID_B \mid\mid g^A \mid\mid g^B \mid\mid n_A \mid\mid n_B)$
$V_A = Q_A \mid\mid n_A \mid\mid n_B$

TABLE 6.1: Protocol notation.

Notation	Description
TSM	TEE trusted service manager.
RA	Revocation authority.
MA	Device maintenance authority.
TA_X	TEE trusted application on device X.
IC_X	Abstract IoT controller X.
n_X	Secure random nonce generated by X.
$H(D)$	Secure one-way hash function, H, on D.
$X \rightarrow Y$	Message transmission from X to Y.
ID_X	Identity of X.
$A \mid\mid B$	Concatenation of A and B.
g^X	Diffie-Hellman exponentiation of X.
AR_X	Attestation request on target entity X.
Q_X	Attestation quote from TEE X.
$(A)\sigma_X$	Signed message A from X under a private-public key-pair (K, P).
$[m]_{AE_K}$	Message m is encrypted using authenticated encryption under session key K derived from the protocol's shared secret.
D'	Data specific to a separate session to D.

IoT devices, such as TPMs, secure elements and smart cards by focusing on TEEs, which may be implemented using the GlobalPlatform TEE and ARM TrustZone on all Cortex-A and M chipsets (F1). The protocols are designed to incorporate abstract TAs, which are verified using the quoting abstraction, whether they be Intel SGX enclaves of GP TEE TAs, thus providing TEE agnosticism (F2). Note, however, that this abstracts away the precision of related work, e.g. Arfaoui et al. [259], which addresses migration specifically in the context of the GP TEE. Such work incorporates GP TEE-specific entites, such as security domains (SDs) and root SDs, which do not exist on Intel SGX or earlier TPM-based TEEs, like Intel TXT [269]. As such, users of this work should be aware of the implementation specifics when deploying these protocols. We refer readers to [259] and [107] for guidance for GP TEEs, and [110] for Intel SGX.

6.9.2 Symbolic Verification

As we introduced in Chapter 4, Scyther is a protocol verification tool presented by Cremers [29] for assuring correctness using symbolic analysis. This was re-employed to verify the protocols proposed in this chapter; we refer the reader back to Section 4.5.4 of Chapter 4 for a description of Scyther. We analyse all protocols using the default Dolev-Yao adversarial model, between TA1, TA2 and the TSM for migration; TA, TSM and BA for backup; TA, TSM and RA for revocation lookup and MA and RA for revocation reporting; TA, MA, RA and TSM for updates; and re-verifying the modified BTP with AEAD between TA1 and TA2. The protocols are modelled under the multi-protocol setting offered by Scyther for evaluating procedures comprising sub-protocols. We see, for instance, that BP-BTP (Protocol 2) contains BTPs executed between (TSM, BA), (TSM, TA) and (TA, BA). Multi-protocol analysis in Scyther accounts for attacks that, for example, replay messages between (TSM, BA) to (TA, BA) and vice-versa, across all sub-protocols of a given procedure.

To be clear, we assume a uniform threat model between entities in the sub-protocols, i.e. the interactions between (TA_1, TA_2) for credential migration is analysed under the same adversarial model as, for example, (TSM, BA) in the backup procedure. In other words, the same Dolev-Yao adversarial model is used to analyse all sub-protocols; however, in reality, the roles of BA, RA, TSM and MA are likely be served by one or more trusted organisations over a network subjected to more rigorous security controls, such as enterprise firewalls and frequent inspection. This is opposed to two devices communicating in the field over a more uncontrolled network topology, as we assume with (TA_1, TA_2). In short, we assume the worst case that the network between all communicating parties is untrusted.

Like Chapter 4, we used Scyther to test the secrecy of transmitted quotes from each party in each procedure and corresponding sub-protocols, e.g. (`Secret, qta1`) and credentials (`Secret, c`); aliveness (`Alive`); replay protection, i.e. non-injective agreement (`Niagree`) and non-injective synchronisation (`Nisynch`), as defined in [29]; session key secrecy (`SKR, K`); and the reachability of all entities, e.g. (`Reachable, TA`). Scyther found no attacks on any procedure under a Dolev-Yao adversary. However, we urge the reader to be mindful of the limitations of verification tools, as set out in Section 4.5.4 of Chapter 4.

The Scyther specifications for each protocol are also released for further scrutiny[1]. Note that the entities tested in our procedures are considered to be trusted, i.e. not compromised. TEEs are inherently designed against complex, potentially kernel-level adversaries – for a detailed review of the threats that a TEE defends against, the reader is referred back to Section 2.4.7 in Chapter 2 – which also includes the TEEs we assume to be in operation by the backup, revocation and maintenance authorities, and the TSM.

6.10 Conclusion

In this chapter, we presented the first investigation into secure and trusted remote TEE credential management on constrained devices using mutual attestation. This resulted in the proposal of a suite of protocols for supporting secure remote *migration*, *revocation*, *backups*, and credential *updates*. This work is a development of the ideas and challenges of mutual attestation (Chapter 4), and we demonstrated in Section 6.8 how it can be applied to secure and mutually trusted log retrieval from work in Chapter 5. After summarising TEE credential deployment in Section 6.2, we formalised the threat model, security goals and assumptions for a typical centralised IoT TEE credential deployment in Section 6.3. For each management challenge, we reviewed the state-of-the-art before proposing procedures and protocols for securely realising these notions using mutual attestation. The protocols were subjected to formal symbolic verification using Scyther, which found no attacks under the Dolev-Yao adversarial model. We publicly release the verification scripts for further research and scrutiny (see Section 6.1.2).

6.10.1 Future Work

While we evaluated the performance of the core protocol in Chapter 4 for bootstrapping a secure and mutually trusted channel for underpinning the remaining proposed procedures, it would be worthwhile to implement and evaluate the procedures in this chapter in an emulated IoT environment with multiple devices.

[1]Protocol specifications available online at: https://www.dropbox.com/s/uq0hftj6b6c1zux/remote-credential-scyther-scripts.zip

A particularly useful endeavour in future research would be to evaluate each protocol on a range on various platforms, from MCUs to higher-end SBCs, using heterogeneous TEEs. A related task would be to implement and evaluate alternative wireless mediums common in IoT deployments, such as ZigBee, LoRa, Bluetooth and 802.11 Wi-Fi, for identifying any potential latency challenges.

Chapter 7

On the Effectiveness of Sensor-based Proximity and Relay Attack Detection Mechanisms for NFC Transactions

The previous chapters principally considered applications of TEEs and proposed a number of protocols, procedures and systems that support those applications. In this chapter, we explore the application of TEEs in protecting the credentials used in NFC-based contactless transactions and their vulnerability to *relay attacks*. Relay attacks stem from the absence of proximity assurances in the NFC specifications, i.e. whether two devices really *are* within the intended operating distance (less than 5cm). We examine the current state-of-the-art of relay attacks and re-evaluate the efficacy of proposed sensor-based mechanisms for detecting them under the conditions actually stipulated by industry.

7.1 Introduction

Near-Field Communication (NFC) [270] and Host Card Emulation (HCE) [111] – discussed in Section 2.2.5 in Chapter 2 – have opened mobile platforms to application domains that were previously instantiated using smart cards. This has led to the development of the use of smartphones in a range of services, such as payments, transportation and access control – exemplified by Google Pay[1], Apple Pay[2] and Samsung Pay[3] as three widely-deployed systems. Deloitte estimated that 5% of the 600-650 million NFC-enabled mobile phones were used at least once a month to make a contactless payment globally in 2015 [271]. In the same year, 12.7% of smartphone users in the USA were actively using contactless mobile payments according to Statista, while the value of such transactions is projected to grow by two-thirds in the next three years alone: from $114 billion (USD) in 2018 to $190 billion (USD) in 2021 [272]. It is projected that 2018 will see approximately 166 million NFC mobile payment users

[1]Google Pay: https://pay.google.com/
[2]Apple Pay: https://www.apple.com/uk/apple-pay/
[3]Samsung Pay: https://www.samsung.com/uk/samsung-pay/

worldwide, corresponding to an approximately 200% increase from previous estimations made in 2015 (53.9m) [273]. Similar trends are expected to follow in transportation and access control for where mobiles are used to deliver smart card-type services [274]. This is before considering alternative devices besides smartphones, such as smartwatches, that may participate in such transactions.

One of the use cases for TEEs on modern mobile devices is the storage of payment credentials used to authenticate users during NFC-based contactless transactions. TEEs and SEs form the cornerstone for protecting the integrity of such credentials against software adversaries from untrusted world components and a selection of hardware-based adversaries (see Section 2.4.7 of Chapter 2). TEEs and SEs may be used to store credentials directly, such as certified device-specific key-pairs used for device authentication. TEEs and SEs are also used to store limited use tokens via the process of *tokenisation*, described further in Section 7.2.1, particularly in the context of payment transactions. The repercussions of unauthorised credential use are somewhat obvious, such as illicit payments billed to the victim's account, or using physical access control credentials to enter a restricted area. Relay attacks are a method by which even TEE- and SE-bound credentials can be abused by exploiting an inherent weakness in the NFC specifications over which a TEE or SE communicates. In this chapter, the focus is on the challenge of proximity and relay attack detection (PRAD) in contactless transactions between a mobile handset and a terminal or Point of Sale (PoS) within restricted time-frames.

7.1.1 Motivation

In a relay attack [275], [276], the aim of the adversary is to extend the physical distance of the communication channel between the victim's mobile phone and the transaction terminal, where each message is relayed across this extended distance. A multitude of Proximity/Relay Attack Detection (PRAD) mechanisms have been proposed that rely on collecting measurements of the ambient environment surrounding the transaction instrument and terminal. These proposals collect measurements from mobile sensors, such as temperature, location and motion sensors, which subsequently undergoes a similarity comparison to assure that the transaction devices are genuinely in proximity, and not subject to a potential relay attack.

However, the proposals presented in existing literature are not compliant with industry-imposed constraints that stipulate maximum transaction times. Mobile payments and transportation are two major domains expected to benefit from NFC contactless transactions where controls exist regarding the maximum transaction times. These are governed by the EMV specifications and ITSO respectively. In this chapter, we question whether *ambient sensing on mobile devices*

is an effective PRAD method under the time conditions stipulated by industry, and present an empirical evaluation to this end. In this first part of this chapter, we consider the notion of *proximity detection* in which we attempt to detect whether two co-located legitimate devices (at NFC distance) can be detected from sensor information alone. This is extended in the second part of the chapter to cover *relay attack detection*, where we use data collected from an emulated relay attack setup from a device that streams sensor data 1.5m away; this represents a 'worst case' close-quarters relay attack.

Generally, this study aims at evaluating the following null and alternative hypotheses. *Null hypothesis*: short transaction durations (under 500ms) have no effect on the error rates of ambient sensing-based proximity and relay attack mechanisms. *Alternative hypothesis*: short transaction durations capture insufficient information to provide acceptable error rates for ambient sensing-based proximity and relay attack detection mechanisms.

7.1.2 Contributions

This chapter presents the following contributions:

- We present a two-fold evaluation, employing both similarity- and machine learning-based analyses, demonstrating that ambient sensing-based PRAD mechanisms perform poorly under the transaction duration requirements stipulated by EMV and ITSO.
- We evaluate the effectiveness of 17 widely-deployed ambient sensors available through the Android SDK as a PRAD method for time-restricted contactless transactions. The evaluation was conducted using data collected from a mock relay attack set-up with consumer mobile devices.
- This two-part study presents results for both *proximity* and *relay attack* detection, and questions the applicability of proposed methods in related work to time-critical transactions.
- A data-set and test-bed environment[4] for reproducing the results of this work and facilitating future research.

The contributions in this chapter are based on our following publications:

- C. Shepherd, I. Gurulian, E. Frank, K. Markantonakis, R. N. Akram, K. Mayes, and E. Panaousis. "The Applicability of Ambient Sensors as Proximity Evidence for NFC Transactions", in *Mobile Security Technologies*, ser. IEEE Security & Privacy Workshops, IEEE, 2017.

[4] Available at: https://github.com/AmbientSensorsEvaluation/

- I. Gurulian, C. Shepherd, E. Frank, K. Markantonakis, R. N. Akram, and K. Mayes. "On the Effectiveness of Ambient Sensing for NFC-based Proximity Detection by Applying Relay Attack Data", in *Proceedings of the 16th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, ser. IEEE TrustCom '17, IEEE, 2017.

- **(Invited Paper)** I. Gurulian, K. Markantonakis, C. Shepherd, E. Frank, and R. N. Akram. "Proximity Assurances Based on Natural and Artificial Ambient Environments", in *Proceedings of the 10th International Conference on Innovative Security Solutions for Information Technology and Communications*, ser. SECITC '17, Springer, 2017.

7.2 Background and Related Work

In this section, we discuss the role of TEE in the domain of payments more specifically, and discuss a number of generic deployment models for deploying proximity- and transaction time-sensitive applications using ambient sensing. This is followed by an examination of related work that propose sensor-based mechanisms for proximity and relay attack detection.

7.2.1 TEEs in NFC Contactless Transactions

Mobile TEEs typically use secure I/O with an NFC controller in order to conduct sensitive NFC-based contactless transactions without relying upon the co-operation of entities in the untrusted world (Section 2.4.4 in Chapter 2 showed how this is instantiated on TrustZone SoCs) [277]–[279]. We briefly describe how this is realised in payments using Samsung Pay [277].

Firstly, each device TEE hosts a set of TAs developed by authorised payment networks, such as Visa, MasterCard and American Express, which are provisioned with immutable, TA- and device-specific certified key-pairs for authenticating itself with the respective payment network. EMV[5] [280] also stipulates the use of *tokenisation* in mobile payments, which is described below in the context of TEEs. Stage one is *card enrollment*, where the user registers card details to the device over a trusted UI path with the payment TA. These details are submitted by the payment TA to the corresponding payment network or Token Service Provider Provder (TSP), e.g. Visa, who, if the details are valid, return a set of dynamic Limited Use Keys (LUKs) with which to authorise future transactions. One LUK is consumed each time Samsung Pay is used to make a card transaction, and LUKs are replenished based on the payment network logic. These payment

[5]EMV is a financial body comprising American Express, Discover, JCB, MasterCard, Union-Pay, and Visa for developing secure payment standards worldwide

credentials are provisioned after the state of the device is successfully verified by the Samsung Pay server using remote attestation. The payment network TA is responsible for retrieving additional LUKs from the payment network remotely. For tokenisation, card enrollment also involves the TSP replacing the 16-digit card Personal Account Number (PAN) with a 16-digit substitute, or Digitised PAN (DPAN). Whenever the device is used to make a *purchase*, Samsung Pay transmits the token DPAN along with a cryptogram generated from the LUK within the TEE to the merchant's PoS via NFC. This process is illustrated in Figure 7.1.The merchant then transmits this to the merchant bank, where the payment network converts the token back into the PAN in order to process the transaction and debit the user's account.

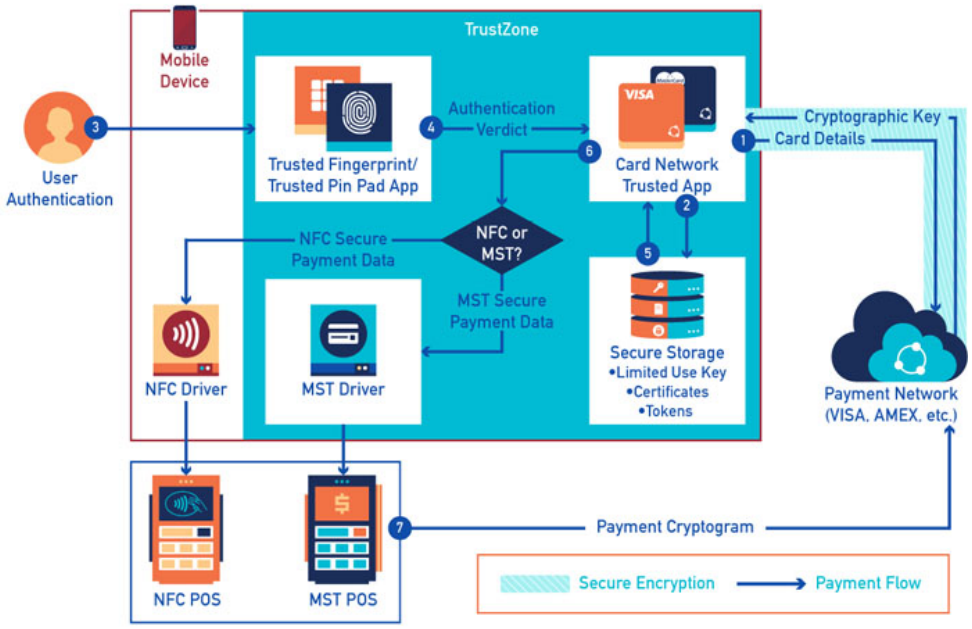


FIGURE 7.1: TEE-based NFC Payments with Samsung Pay [277].

Payments are only one application of TEE-based NFC transactions. For transport applications, Ekberg and Tamrakar [281] describe a generic architecture for gated ticketing with TEEs in two phases. Phase one presumes the user's transport TA is provisioned with a certified key-pair. After receiving a challenge from the gate terminal, the TA transmits the public component along with the PAN over NFC, where the terminal (connected to the transport system's back-end) returns n short-term ticketing credentials mapped to that public key and the user's PAN. In phase two (payment), assuming the user PAN was not blacklisted, the transport TA transmits an unused short-term credential from phase one to the gate terminal, which passes it to the back-end for fare calculation and payment from a pre-paid account.

Other systems, such as London Buses, charge the user akin to typical payments using the global payment infrastructure under a pay-as-you-go (PAYG) model. Some, such as the London Underground, use an aggregated model in which multiple transactions are backlogged before performing fare calculation, e.g. between multiple towns in a single day, also known as the Aggregated PAYG model [282]. In general, NFC acts only as the *communication medium* over which credentials are passed, while the TEE is used to host them in a tamper-resistant environment.

7.2.2 Relay Attacks

NFC is a set of communication protocols based on Radio Frequency Identification (RFID) that was developed by NXP and Sony in 2002. It operates wirelessly on the 13.56 MHz Radio Frequency (RF) band using electromagnetic induction between two loop antennas on two NFC-enabled devices. NFC has a maximum operating distance of 10cm and a bit-rate of 424kbps, and is standardised in ISO/IEC 18092 [283] and ISO/IEC 21481 [284]. Nowadays, many modern handsets contain NFC transceivers, such as the Apple iPhone, Samsung Galaxy S9, and the Google Pixel. However, NFC contains a long-standing issue of containing no security mechanisms for proximity detection in order to ensure that the distance of communicating devices is, indeed, the intended distance [281]. Typical NFC services simply *assume* that the devices are located together at a short-range operating distance, i.e. 1-5cm. It is this assumption that allows relay attacks to be conducted successfully.

A relay attack [275], [276] is a passive man-in-the-middle attack in which the aim of the malicious actor is to extend the physical distance of the communication channel between the victim's mobile phone and the transaction terminal. Each message is then 'relayed' across this extended distance. The attacker extends this distance using equipment that masquerades as legitimate devices to both the terminal and victim device, as shown in Figure 7.2. The attacker has the potential to gain access to services using the victim's account if the application-layer messages are relayed successfully without detection. Relay attacks are an instantiation of the Grand Master Chess problem in network security in which a naïve player, A, poses as a grandmaster player to two other grandmasters, B and C. B and C begin playing as if they are facing a grandmaster, A, who is actually relaying the messages between B and C as a man-in-the-middle, without actually possessing any requisite knowledge.

At present, additional user-authentication mechanisms, namely a fingerprint or Personal Identity Number (PIN) entry input, may or may not be required in order to authorise a contactless mobile transaction, depending on the deployment scenario. This adds a layer of defence against a relay attack in which the attacker

FIGURE 7.2: Overview of a relay attack.

must convince the victim to complete the challenge before the transaction is authorised. However, a second factor of authentication is often not required; for example, it is not mandatory in NFC payments below a pre-set transaction limit (currently £30 in the UK) [285]. Moreover, the the use of PINs and biometrics cannot prevent relay attacks if the victim interacts with a malicious terminal believing it to be genuine – also known as the Mafia Fraud attack [286]. Whatever the situation, a relay attack allows an attacker to circumvent the protections offered by TEEs by, instead, exploiting the proximity vulnerability in the NFC specifications in order to abuse the credentials transmitted by a TEE over NFC.

7.2.3 Relay Attack Countermeasures

For contactless smart cards, relay attacks can be countered using distance-bounding protocols [287], [288]. Here, at its most a basic, a challenger, V, measures the time taken to receive a response from the prover, P [288] (also known as time-of-flight). If this time exceeds a pre-determined threshold, t, then the presence of a relay attack is assumed to have occurred based on the added network latency introduced between the two intermediary devices controlled by the attacker. Smart card-based distance-bounding protocols remains an active research domain, with new attacks and countermeasures emerging regularly [286], [289], [290]. Hardware-based distance-bounding products have also begun to emerge: 3DB-Access[6] offers distance-bounding based on the time-of-flight of radio packets transmitted on the 6-8GHz spectrum. It operates up to an approximate range of 120 meters, with a stated accuracy of 10cm, 100% bounding success rate, and low power consumption ($2\mu J$ per measurement); its marketed applications include thwarting relay attacks on contactless car keys, and assuring co-location of IoT devices and assets within industrial environments.

Notably, however, these approaches are not easily generalisable to NFC-enabled phones because of the variance in hardware-software configurations on mobile platforms that renders it difficult to establish reliable and reproducible timings [55], [270], [291]. Alternative methods have been proposed to provide

[6]3DB-Access: https://www.3db-access.com/

proximity detection using environmental and motion sensors present on modern mobile handsets [291]–[296]. In Section 7.2.4, we discuss how ambient sensors have been proposed to counter relay attacks in NFC-based mobile contactless transactions.

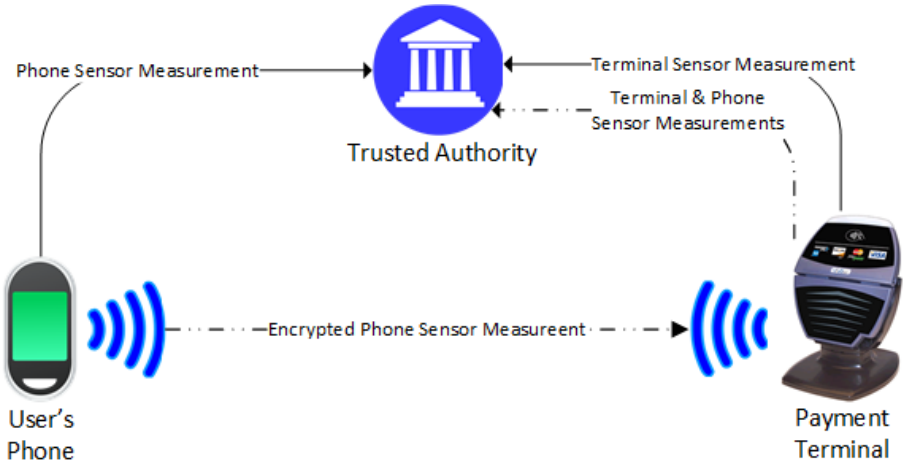


FIGURE 7.3: Generic deployment of mobile sensing for proximity detection.

An ambient sensor measures a particular environmental property of its immediate surroundings, such as temperature, light, humidity and sound; a wealth of such sensors are deployed within modern smartphone and tablets (see Appendix B). Figure 7.3 illustrates a general approach for using ambient sensing as a PRAD mechanism for contactless transactions with the following variations:

1. **Independent Reporting**. Both the transaction instrument, e.g. user's phone, and the terminal/PoS collect sensor measurements independently, who each transmits them to a trusted authority, TrA (solid lines in Figure 7.3). The authority compares the sensor measurements, based on some predefined comparison algorithm with set margins of error (threshold), and decides whether the two devices are within proximity to each another.

2. **Payment Terminal Dependent Reporting**. The smartphone encrypts its collected measurements with a shared key between itself and the TrA, and transmits the encrypted message to the payment terminal (shown as double-dot-dash lines in Figure 7.3). The payment terminal sends its own measurements and the smartphone's encrypted measurements to the TrA for comparison.

3. **Payment Terminal (Localised) Evaluation**. The smartphone transmits its measurement to the payment terminal, which compares them with its own measurements locally. It is the payment terminal that decides whether the two are in proximity.

7.2.4 Proposed Sensing-based Countermeasures

In this section, we summarise the key pieces of related work that suggest the use of ambient sensing as a proximity and relay attack detection mechanism for secure transactions.

Ma et al. [292] propose the use of GPS (Global Positioning System) location data for determining the proximity of two mobile phones. A ten-second recording window is used in which GPS data points are collected each second on two devices, which subsequently undergo a similarity comparison. The authors note that, when the devices were placed 2m apart, the GPS values actual recorded by each device corresponded to distances between 1.78m–6.21m. Using this maximum error (6.21m), the work evaluates relay attack success rates at varying distances (in meters): 100% at 1m apart, 92.5% at 2m, 85% at 3m, and 67.5% at 5m, before dropping significantly to 0% at 20m and beyond. While this shows promise at larger distances, i.e. >20m, even the best case scenarios (1–3m) are far beyond the intended operating distance of NFC (1-5cm). That is, the scheme performs poorly against attacks at close-quarters (<3m), like in queues at a store PoS or ticketing gate.

Halevi et al. [291] demonstrate the use of ambient sound and light for proximity detection. The authors analyse sensor measurements – collected for a duration of 2 and 30 seconds for light and audio respectively – using a range of similarity comparison metrics (see Section 7.3.3). The two devices were separated by a distance of 3-12 inches, and 200 transactions were captured at five unique locations, including a coffee shop, concert hall, library (at two different locations), and McDonalds. The evaluation results yielded an attack detection rate of 82.5%–100% depending on location. However, the sample durations (2–30 seconds) significantly exceed the timing restrictions stipulated by EMV and ITSO for transaction durations.

Varshavsky et al. [296] use the shared radio environment of devices – the presence of WiFi access points and associated signal strengths – as a proximity detection mechanism for secure device pairing. The approach is evaluated with genuine co-located devices at 5cm apart, while illegitimate transactions are conducted at 1m, 3m, 5m and 10m away. The work also evaluates a range of total transaction times between 1–30 seconds, producing a worst-case false positive rate of approximately 0.7–0.9 for an attacker located 1m away depending on the sampling period between 1–30s. However, this drops significantly to under 0.1 for attacker located further away (3–10m). This corresponds with previous work that shows distance has a significant effect on error rates. The approach focusses on the challenge of device-pairing, rather than relay attacks on NFC-based mobile transactions, and does not consider the time-critical durations targeted in this work.

Urien et al. [295] propose a mutually authenticated secure channel protocol based on elliptic curve cryptography with ambient sensing in order to counter relay attacks. The proposal incorporates ambient temperature in conjunction with timing-based distance bounding to determine whether two devices are co-located before allowing the creation of the secure channel. The work, however, was not implemented and has no experimental data to evaluate its effectiveness.

Mehrnezhad et al. [297] propose the use of an accelerometer to provide proximity assurances of a mobile device with a payment terminal. The scheme requires the user to tap the terminal twice in succession, before comparing the sensor data from the device and the terminal for similarity using a variety of statistical measures (Section 7.3.3). The authors evaluate a recording time range of 0.6–1.5 seconds and report an equal error rate (EER) of between 17.65% and 30.22%, depending on the similarity metric, after a two-tap interaction.

Karapanos et al. [124] propose Sound-Proof that uses the ambient sound recorded by the user's computer and mobile device as a second factor of authentication during log-in attempts. After a recording period of 3 seconds on both devices, cross-correlation is used to compare the similarity of measurements from the devices to determine whether they are, indeed, co-located or not. The total time to perform the distance-bounding is 4677ms and 4944ms over WiFi and cellular connections respectively (including the recording time), with an exhibited EER of 0.2%. We note that, while Sound-Proof is meant as a second-factor for log-ins in conjunction with a password, and not NFC-based transactions such as for payments and transportion, its methods are still interesting to this work.

Truong et al. [294] evaluated four different sensors – WiFi, Bluetooth, GPS and Audio – for providing relay attack resistance. The authors record data over a total sampling period of two minutes, and estimate the accuracy of the proposal across the average of 5–15 second segments contained therein. Features are extracted from each segment, such as cross correlation of audio signals and the intersection of shared WiFi ESSIDs, which are used as input to a supervised learning classifier using decisions trees. The scheme reports a false negative rate of between 8.95% (5s) and 1.49% (15s), and a false positive rate of 7.14% (5s) to 2.14% (15s). While the results are promising, the relatively long sampling duration renders it unsuitable for time-critical NFC-based mobile transactions.

Jin et al. [298] demonstrate the use of a smartphone's magnetometer to establish proximity assurance for secure device pairing against relay attacks. The authors use statistical similarity measures, including average cross correlation and Pearson's correlation coefficient, within a secure channel protocol to assure both end-points that they are communicating with the intended nearby device. The work evaluates three different handsets using the proposed measures under varying sampling durations between 2–5 seconds, against an attacker device

located 10cm away. The results indicate that, in a user study of untrained users, a 90–100% success rate for attack detection is achievable on average within 4.5s, falling to 80% at 2s.

Shrestha et al. [293] evaluated four ambient sensor modalities – ambient temperature, precision gas, humidity, and altitude – for proximity detection using specialised hardware, known as Sensordrone. The authors collect 207 samples at 21 different locations, which is used as input to a Multiboost classifier with Random Forests as the weak learner. The results yield a false negative rate of between 2.96%–23.47%, and a false positive rate ranging between 5.81%–32.40%. The authors claim that these particular sensors require only one sample to achieve these error rates.

TABLE 7.1: Sensor-based PRAD mechanisms from related work.

Paper	**Sensors**	**Sample Required**	**Contactless Suitability**
Ma et al. [292]	GPS Location	10 seconds	Unlikely
Halevi et al. [291]	Ambient Audio	30 seconds	Unlikely
	Light	2 seconds	Likely
Varshavsky et al. [296]	WiFi	1–30 seconds	Likely
Urien et al. [295]	Temperature	N/A*	N/A
Mehrnezhad et al. [297]	Accelerometer	0.6–1.5 seconds	Likely
Karapanos et al. [124]	Ambient Audio	3 seconds	Likely
Truong et al. [294]	GPS Location	5–15 seconds	Unlikely
	WiFi	—	—
	Ambient Audio	—	—
	Bluetooth	—	—
Jin et al. [298]	Magnetometer	2–5 seconds	Likely
Shrestha et al. [293]	Temperature (T)	Per sample	Likely
	Precision Gas (G)	—	—
	Humidity (H)	—	—
	Altitude (A)	—	—
	HA	—	—
	HGA	—	—
	THGA	—	—

* Neither implemented nor evaluated. — Repeat entry. N/A Not applicable.

Table 7.1 summarises past work, using sensor sampling durations required to achieve their desired accuracy in order to determine their suitability for protecting time-critical NFC-based mobile phone transactions. 'Unlikely' proposals have sample durations so large that they vastly exceed a reasonable time in order to

conduct mobile-based services, while those with reasonably short durations are labelled 'Likely'. However, even schemes denoted 'Likely' may not be suitable; no proposal was evaluated under the specific time constraints stipulated by the banking and transport sectors. Precisely, *EMV stipulates a maximum transaction duration of* ***500ms*** [285], [299], [300], while, *for transport-related transactions, it must complete between* ***300ms–500ms*** [301], [302]. In these sectors, minimising transaction time is critical in order to maximise customer throughput for reducing waiting times and congestion. In mass transit, aiming to process 30 travellers per minute is a widely-adopted guideline by transport authorities [302]. In light of this, we see that no scheme in Table 7.1 evaluates sampling durations within this limit, which is the primary objective of this work.

7.2.5 General Approaches

In previous work, two general approaches have been used for implementing sensing-based PRAD mechanisms:

- *Threshold-based Similarity*: the use of time and frequency domain similarity metrics, such as Mean Absolute Error (MAE), Pearson's Correlation Coefficient and Coherence. A single threshold, t, is found that aims to separate all legitimate transactions from illegitimate ones using a given similarity metric. The transaction is accepted if the metric result falls within this pre-set t of the maximum allowed dissimilarity.

- *Machine Learning*: the use of supervised classification algorithms, such as Naïve Bayes, Support Vector Machines (SVMs) and Random Forests, to produce more complex models for mapping input feature vectors to labels. The classifier is trained on a set of feature vectors with corresponding binary labels (legitimate or relayed transaction), which are collected beforehand. The trained model is used to classify subsequent transaction data streams as legitimate or relayed.

Standard binary classification evaluation metrics have been applied to measure the effectiveness of a particular scheme, namely classification accuracy [291], f-scores [293], [294] and Equal Error Rate (EER) [297]. F-scores and EERs involve the computation of *false positives/acceptances* (the number of relayed transactions accepted erroneously) and *false negatives/rejections* (the number of legal transactions rejected). F-scores account for *precision*, the correct positive results divided by the number of all positive results, and *recall*, the number of correct positive results as proportion of the number of positive results that should have been identified. The EER is found by calculating the False Acceptance Rate (FAR) and False Rejection Rate (FRR), shown in Eq. 7.1, over a range of thresholds and finding the rate at which $FAR = FRR$. Alternatively, some authors have opted

to present the FAR and FRR results alone [296]. Finally, accuracy represents the correct identification of positive and negative transactions in the test set (Eq. 7.2), but does not clearly illustrate the number of false positives and negatives.

F-scores and accuracy have been used to primarily evaluate machine learning-based relay attack detection, e.g. [293], [294], while EERs have typically been employed for threshold-based similarity approaches [297] to find an acceptance threshold that, broadly, balances usability (false rejection rate) with security (false acceptance rate). We use the EER as a common evaluation metric for assessing the performance of machine learning and threshold-based approaches across a variety of similarity metrics.

$$FAR = \frac{FP}{FP + TN} \qquad FRR = \frac{FN}{FN + TP} \tag{7.1}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{7.2}$$

7.3 Effectiveness for Proximity Detection

In the first step of our study, we evaluate the effectiveness of ambient sensing for proximity detection alone between two co-located devices at NFC distance. Detecting the presence of a relay attack is an extension of this, which we study later in Section 7.4.

7.3.1 Test-bed Construction

Proximity detection is the notion that the sensor measurements taken from two devices, TT and TI, are sufficiently similar to belong to the same transaction, while remaining sufficiently distinct to separate it from measurements taken from other device pairs, i.e. (TT', TI'). Two applications were developed and installed on a pair of Android devices: one emulating a transaction terminal (TT) and the other acting as the transaction instrument (TI). In this study, we evaluated 17 sensors available through the Android SDK that were also available through our available devices: a Google Nexus 9 tablet, and Google Nexus 5, Samsung SGS5 Mini, and Samsung Galaxy S4 smartphones. The choice of which device acted as TI and TT depended on sensor availability. Table 7.2 states the availability of each sensor on each handset.

When the devices come sufficiently close, an NFC connection is established and both begin recording data using a specified sensor. After collecting measurements for 500ms, in line with the transaction requirements stipulated by EMV and transport, each device stored the recorded data in a local database for off-line analysis. Figure 7.4 illustrates the recording process.

TABLE 7.2: Device sensor availability.

Sensors	Nexus 9	Galaxy S4	Nexus 5	SGS5 Mini
TT-TI Pair: Nexus 9 → Nexus 5				
Accelerometer	✓	✓	✓	✓
Bluetooth	✓	✓	✓	✓
GPS	✓	✓	✓	✓
Gyroscope	✓	✓	✓	✓
Magnetic Field	✓	✓	✓	✓
Network Location	✓	✓	✓	✓
Pressure	✓	✗	✓	✗
Rotation Vector	✓	✓	✓	✓
Sound	✓	✓	✓	∗
WiFi	✓	✓	✓	✓
TT-TI Pair: Nexus 9 → SGS5 Mini				
GRV†	✓	✗	∗	✓
TT-TI Pair: SGS5 mini → Nexus 5				
Gravity	∗	✓	✓	✓
Light	∗	✓	✓	✓
Linear Acceleration	✓	∗	✓	✓
Proximity	✗	✓	✓	✓
TT-TI Pair: Galaxy S4 → Galaxy S4				
Relative Humidity	✗	✓	✗	✗
Ambient Temperature	✗	✓	✗	✗

✓: Present on device. ✗: Not present. ∗: Known technical issues on this handset. † GRV: Geomagnetic Rotation Vector.

7.3.2 Data Collection

Next, a field trial was conducted in which sensor data from 1,000 transactions per sensor was collected from 252 users at four different locations on the university campus: 1,000 at the *library*, *cafe*, *dining hall*, and *computing lab* – approximately 250 at each. During the trial, the TT device was fixed, thus replicating a terminal/PoS, while TI was free of any restrictions. However, prior to these trials, the first stage was investigating whether *any* values could be captured in such a small time-frame for each sensor. We collected 50-100 initial transactions in the computing lab and eliminated sensing modalities that exhibited *sensor failures* in which the sensor returned no values within 500ms. For completeness, we also recorded the number of *transaction failures*, in which a transaction was unexpectedly terminated due to, for example, the user unintentionally moving the device away mid-transaction and the protocol failing to complete. Table 7.3 displays the transaction and sensor failure rates for each sensing modality, including those transactions collected during field trials.

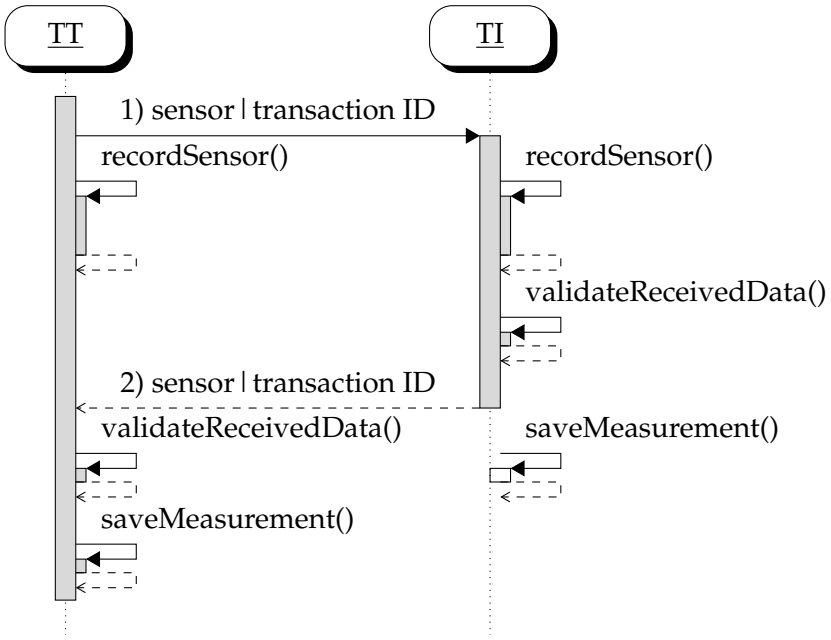


FIGURE 7.4: Measurement recording overview.

Six sensors notably failed to capture any measurements in the 500ms timeframe for the vast majority of transactions: Bluetooth, GPS, Network Location, WiFi, ambient temperature and humidity. All of these modalities recorded a sensor failure rate between 91.17–100%, which would evidently cause problems in practice, and so we omitted these sensors from subsequent analysis. The field trials involved collecting 1,000 transactions for each sensor modality, as stated previously. We requested 252 users to present TI to TT as many times as they preferred. We established walk-in counters at the four aforementioned locations around the university campus; students walking nearby were invited to assist us in the trial. A walk-in counter established at the university library is shown in Figure 7.5.

7.3.3 Analysis Approach

After collecting approximately 1,000 transactions per sensor, the data was retrieved from each of the device's local databases converted to CSV in preparation for either similarity- or machine learning-based analysis. For each sensor, the EER were computed using six time- and frequency-domain similarity measures, including those used in previous work. We list these forthwith. *Time domain metrics*: Mean Absolute Error (MAE), Eq. 7.3; Pearson's correlation coefficient [297], Eq. 7.4; maximum cross-correlation [124], [291], [294], Eq. 7.5; and Euclidean distance [294], Eq. 7.6. *Frequency domain*: coherence [297], Eq. 7.7. *Both domains*: time-frequency distance [291], [294], Eq. 7.8. Each metric was applied directly onto the sensor data collected during the field trials. For machine learning, the

TABLE 7.3: Sensor and transaction reliability.

Sensors	Total Transactions	Transaction Failures	Sensor Failures
Accelerometer	1025	13 (1.26%)	0 (0%)
Bluetooth	101	1 (0.99%)	99 (99.1%)
GRV	1019	8 (0.78%)	0 (0%)
GPS	101	1 (0.99%)	100 (99.10%)
Gyroscope	1022	11 (1.07%)	0 (0%)
Magnetic Field	1027	17 (1.65%)	0 (0%)
Network Location	1053	15 (1.42%)	960 (91.17%)
Pressure	1018	10 (0.98%)	0 (0%)
Rotation Vector	1023	14 (1.36%)	0 (0%)
Sound	1047	4 (0.38%)	0 (0%)
WiFi	100	0 (0%)	100 (100%)
Gravity	1165	143 (12.27%)	0 (0%)
Light	1057	37 (3.50%)	0 (0%)
Linear Acceleration	1175	159 (13.53%)	3 (0.26%)
Proximity	1071	58 (5.41%)	0 (0%)
Ambient Temperature	50	0 (0%)	47 (94%)
Humidity	50	0 (0%)	47 (94%)

Weka package was employed, while a Python application was developed for similarity-based similarity learning using the Numpy, Scipy, Matplotlib and Pandas Python packages for metric implementations, CSV data processing, and result computation.

$$MAE(A,B) = \frac{1}{N}\sum_{i=1}^{N}|A_i - B_i| \tag{7.3}$$

$$corr(A,B) = \frac{\sum_{i=1}^{N}((A_i-\mu_A)(B_i-\mu_B))}{\sqrt{\sum_{i=1}^{N}(A_i-\mu_A)^2\sum_{i=1}^{N}(B_i-\mu_B)^2}} \tag{7.4}$$

Where μ_A represents the arithmetic mean of A.

$$M_{corr}(A,B) = \max(cross\_correlation(A,B)) \tag{7.5}$$

$$d(A,B) = \sqrt{\sum_{i=1}^{N}(B_i - A_i)^2} \tag{7.6}$$

$$C_{AB}(f) = \frac{|G_{AB}(f)|^2}{G_{AA}(f)\cdot G_{BB}(f)} \qquad F_{AB} = \sum_f C_{AB}(f) \tag{7.7}$$

Where G_{AA} is the auto-spectral density of A, and G_{AB} is the cross-spectral density of signals A and B (left, Eq. 7.7). The similarity is found by the sum of

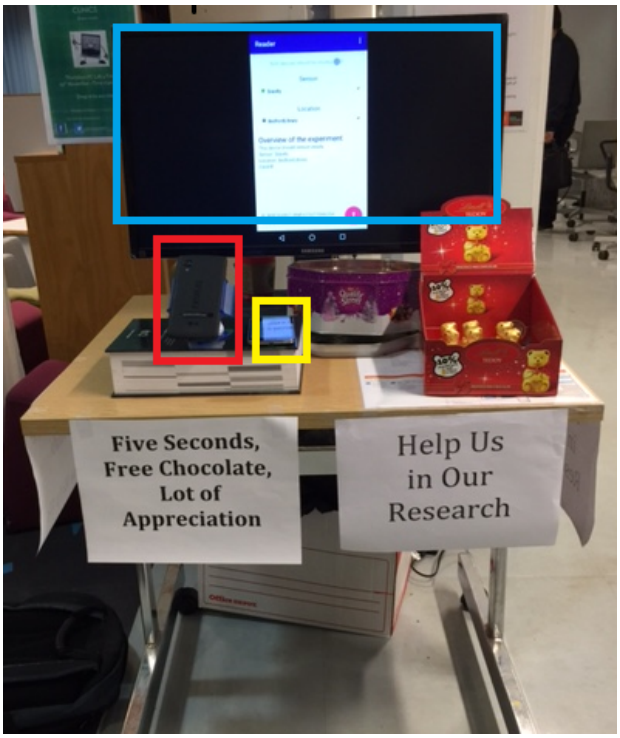


FIGURE 7.5: Walk-in booth located at the university library, showing TT (red rectangle), TI (yellow), and a display stream of TT showing current sensor progress (blue).

the magnitudes of coherence values at all frequencies (right).

$$Diff(A,B) = \sqrt{D_{time}(A,B)^2 + D_{freq}(A,B)^2} \tag{7.8}$$

Where $D_{time}(A,B) = 1 - M_{corr}(A,B)$ and $D_{freq}(A,B) = ||FFT(A) - FFT(B)||$, in which $||FFT(A) - FFT(B)||$ is the Euclidean norm of the FFTs of signals A and B.

The set of all transactions was produced after retrieving the databases from TT and TI. Each transaction, T_i can be represented as a tuple of TT and TI values, i.e. $T_i = (TT_i, TI_i)$, where each TT and TI contains a set of sensor measurements for transaction i. The set of legitimately co-located samples, C, was the set of all transactions for each sensor from the TT and TI collected during field trials, i.e. $C = \{(TT_1, TI_1), (TT_2, TI_2), \ldots, (TT_n, TI_n)\}$, where n was the number of successful transactions shared between the devices. The set of unauthorised pairs, which were not co-located, was formed by exhaustively pairing the measurements of TT_i with every TI_j belonging to another transaction ($i \neq j$). This pairs the measurements of TT_i with TI_j measurements taken from another physical location or at a different point in time in the same location. (Measurements are assumed to be independent between transactions; the field trials collected measurements over the course of a week across all locations). For

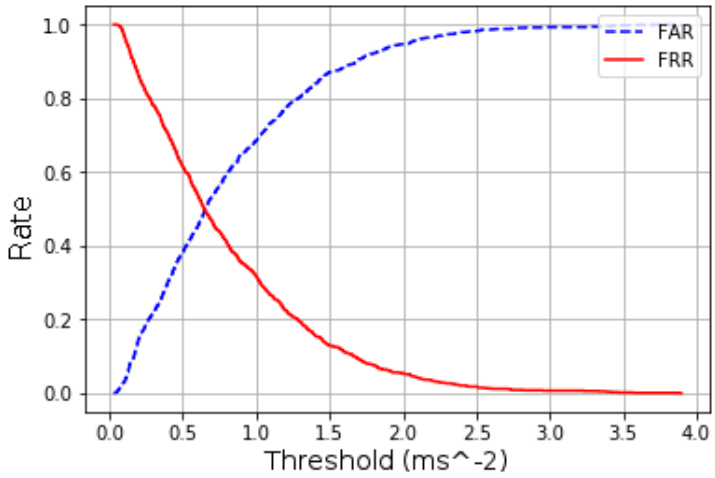


FIGURE 7.6: FAR and FRR curves for the accelerometer sensor with MAE similarity metric.

effective proximity detection, a similarity- or machine learning-based model must be able to distinguish only legitimate sensor values taken at the same location (1-5cm apart) and time. Other sensor pairs, taken at a different location or time period, ought to be rejected.

Having formed the positive and negative transaction sets, we computed the FAR and FRR rates, and the point at which they intersect (EER), for each of the observed thresholds for each of the aforementioned similarity metrics: Mean Absolute Error (MAE), Pearson's Correlation Coefficient (PCC), Cross Correlation (C-Corr), Euclidean Distance (ED), Coherence (Coh), and Time-Frequency Distance (T-FD). An example curve for the accelerometer sensor using MAE is found in Figure 7.6. For machine learning, we conducted 10-fold cross-validation 10 times using six supervised classification algorithms proposed in related literature and beyond: Random Forest, Naïve Bayes, Logistic Regression, Decision Tree (C4.5), Support Vector Machine (SVM) and Multilayer Perceptron.

7.3.4 Analysis Equipment

The similarity-based analysis was conducted on a Linux (Fedora) machine with a quad-core Intel i5-4690k (3.7GHz) and 16GB of RAM. The analysis application was developed in Python, using the Pandas [303], NumPy [304] and SciPy [305] libraries for data loading and numerical computation, including implementations of the aforementioned statistical measures. For the analysis by machine learning, the Weka toolkit, implemented in Java, was employed on a cluster of 10 Ubuntu Linux computers with quad-core Intel i7-2600 CPUs (3.8GHz) and 16 GB of RAM each. Multi-threading was used for training random forests and performing parameter optimisation for SVMs, and parallelising the computation of each similarity-based metric.

TABLE 7.4: Similarity-based EERs for each sensor with Mean Absolute Error (MAE), Pearson's Correlation Coefficient (PCC), Maximum Cross-Correlation (C-Corr), Euclidean Distance (ED), Coherence (Coh) and Time-Frequency Distance (T-FD). Best result for each sensor shown in bold.

Sensor	MAE	PCC	C-Corr	ED	Coh	T-FD
Accelerometer	0.495	**0.493**	0.501	0.498	0.542	0.501
GRV†	0.442	0.486	0.500	**0.442**	0.524	0.498
Gravity	0.500	**0.496**	0.498	0.501	0.506	0.498
Gyroscope	0.498	**0.454**	0.493	0.498	0.548	0.499
Light	0.501	0.500	0.545	0.502	**0.471**	0.546
Linear Acceleration	0.504	**0.481**	0.494	0.507	0.507	0.500
Magnetic Field	**0.323**	0.574	0.537	0.337	0.568	0.536
Pressure	**0.284**	0.634	0.601	0.284	0.503	0.601
Rotation Vector	0.498	0.466	0.501	0.278	0.500	**0.273**
Sound	0.343	0.555	0.481	**0.338**	0.517	0.481

Proximity excluded due to insufficient unique values.
† GRV: Geomagnetic Rotation Vector sensor.

7.3.5 Results

The results for the threshold-based and machine learning analyses are presented in Tables 7.4 and 7.5 respectively. Note that the proximity sensor was excluded from the analysis. We discovered that only on a small selection of Android devices with proximity sensors return the precise distance at which an object is located from the sensor. The majority of others return a binary value for whether an object is close to or far from the sensor (within 5cm) [306]. Our test-bed devices returned only binary values and, as a result every transaction contained 'far' values, as the devices were tapped back-to-back and the sensor was located on the front of the devices. Consequently, this returned identical values in almost all cases when applying the similarity metrics described previously, e.g. $MAE = 0$, which prevented threshold-finding and effective machine learning classification. Readers should be aware of this technical peculiarity when considering proximity-based sensors on mobile handsets.

Other technical challenges existed elsewhere: the Rotation Vector sensor, for example, returned significant numbers of zero values on the test-bed devices, which likely distorted the results of our analysis. This is due to the fact that the previous sensor is returned unless the device is significantly perturbed in order to minimise energy consumption [307]. The GRV sensor, which uses the magnetometer for computing rotations, was able to capture values as intended. Lastly, we noticed that sound was capable of capturing values for only half of the permitted 500ms time-frame, due to the latency introduced in initialising the microphone and recording the sample. (Note that we did not consider any 'pre-recording' measures before the transaction as a potential remedy. The

TABLE 7.5: Estimated EERs for machine learning algorithms, obtained by repeating stratified 10-fold cross-validation 10 times. Best result for each sensor shown in bold.

Sensor	Random Forest	Naive Bayes	Logistic Regression	Decision Tree	Support Vector Machine	Multilayer Perceptron
Accelerometer	0.626±0.024	0.509±0.026	0.526±0.023	0.500± 0.0	**0.498**±0.025	0.551±0.025
GRV	**0.435**±0.021	0.447±0.024	0.474±0.031	0.500± 0.0	0.489±0.036	0.450±0.026
Gravity	0.874±0.018	0.579±0.020	0.579±0.024	**0.500**± 0.0	**0.500**±0.026	0.746±0.112
Gyroscope	0.683±0.027	**0.499**±0.024	0.543±0.024	0.500± 0.0	0.511±0.025	0.514±0.025
Light	0.576±0.026	0.515±0.024	0.533±0.025	**0.500**± 0.0	0.508±0.024	0.513±0.028
Linear Acceleration	0.603±0.025	0.507±0.027	0.543±0.023	**0.500**± 0.0	**0.500**±0.021	0.554±0.028
Magnetic Field	**0.292**±0.021	0.319±0.020	0.322±0.020	0.415±0.015	0.398±0.046	0.329±0.026
Pressure	0.103±0.010	0.107±0.010	0.287±0.013	**0.092**±0.054	0.319±0.045	0.114±0.019
Rotation Vector	**0.276**±0.046	0.563±0.243	0.596±0.233	0.500± 0.0	0.513±0.243	0.488±0.245
Sound	**0.288**±0.019	0.314±0.022	0.310±0.021	0.347±0.136	0.411±0.041	0.306±0.020

effectiveness of pre-recording, particularly in relation to battery consumption and ideal pre-recording times, is considered out-of-scope in this study).

7.3.6 Discussion

The results indicate that no sensor in either analysis can satisfactorily distinguish between proximate and non-proximate device data pairs. Some sensors provide virtually no discriminatory power and perform similarly to a random classifier (50%), such as the accelerometer (49.3–49.8% EER), linear acceleration (48.1–50.0%), GRV (48.6–50.0%), light (47.1–50.0%), gyroscope (45.4–49.9%), and gravity sensors (49.6–50.0%). Other sensors evidently provide significantly better discrimination, such as magnetic field (29.2–32.3%), pressure (9.2–27.0%) and rotation vector (27.3–27.6%). In the best case, with the pressure sensor using the Decision Tree classifier, the EER was 9.2%.

By definition of the EER, this implies that approximately 9.2% of both legitimate and illegitimate transactions would be rejected and accepted respectively. Rejecting almost 1-in-10 legitimate transactions in a high throughput scenario is likely to cause user annoyance in practice, such as mobile ticketing in a subway system. For some high security deployments, such as physical access control, these error rates may, indeed, be an acceptable trade-off in order to reduce relay attack detection to only 1-in-10 illegitimate transactions being accepted. Generally, however, for payments and transportation applications that require high throughput and convenience, it is difficult to recommend any single sensor in our analysis as an effective proximity detection method in time-critical deployments.

7.4 Relay Attack Detection

The previous study focused only on proximity detection rather than using data from relay attacks. Relay attack detection is an extension of proximity detection:

an effective relay attack detection mechanism must thwart attackers located nearby *and* allow genuinely proximate devices using data collected from an attack setup. This is opposed to the previous study that did not incorporate data from actual attacking devices; rather, the non-proximate pairs were estimated by pairing data from other transactions. We resolve this in this study by conducting further field trials and collecting data from two devices that were genuinely in proximity and a third device – the attacker – located 1.5m (5ft) away. This replicated a relay attack in which an adversary launches the attack on a nearby victim, such as in a store queue, without detection. We aimed to determine whether sensor measurements from a illegal transaction device pair – the terminal and the device 5ft away – could be distinguished from a legitimate pair, i.e. the terminal and the device in correct proximity.

Like the previous section, we present an empirical study that evaluates mobile sensors as a potential relay attack detection mechanism under the practical time constraints stipulated by EMV. An effective sensor should be able to reject sensor values that were recorded on distant devices, while still accepting legitimate transactions. We conducted another two-fold evaluation based on similarity-based threshold analysis and machine learning.

7.4.1 Test-bed Design

A test-bed environment was designed and developed that captures both legitimate and illegitimate pairs of sensor measurement under real-world conditions. A genuine pair of measurements is from two devices that are physically in close proximity to each other (<3cm), while an illegitimate pair is from two devices that are not physically in close proximity, which we consider to be 5ft (1.5m). To achieve this, we develop a test-bed, illustrated in Figure 7.2, that records sensor measurements on both the legitimate terminal and device, and an emulated victim phone at distance. (All three devices measure the ambient sensor values concurrently in order to avoid any discrepancies in the measurements due to differences in time). The relay pair comprised a transaction terminal (TT) and a transaction instrument (TI′), while the legitimate pair consisted of a relay transaction terminal (TT′), and a genuine transaction instrument (TI). Devices TI′ and TI were tapped simultaneously against devices TT and TT′ respectively.

Figure 7.7 illustrates the overview of the test-bed. The Transaction Terminal (TT) is a static device, and we use this as a reference point for our two pairs. The Transaction Instrument (TI′) is a mobile phone in close proximity to the TT. Another mobile phone, at a 1.5m distance from the TT, is referred to as the Transaction Instrument (TI), and is co-located with the Transaction Terminal (TT′). At a point in time a user taps TI′ to TT; at approximately this point, TI is also tapped against TT′ and initiates the ambient sensor measurements.

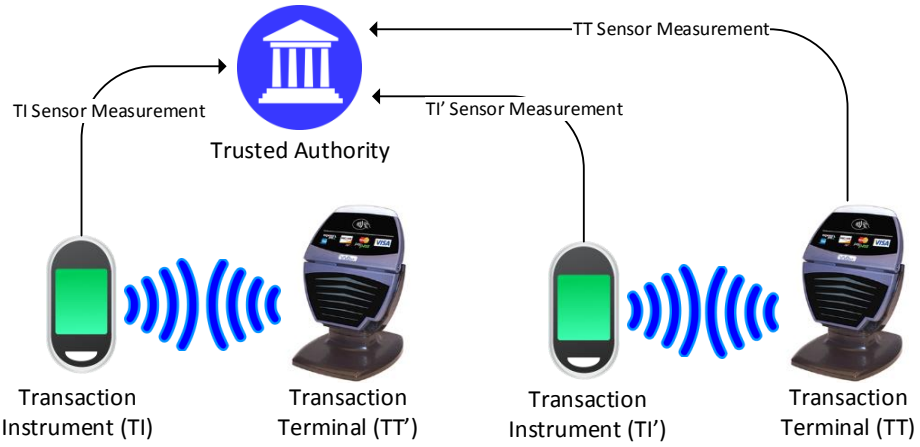


FIGURE 7.7: Test-bed overview. (TT′ and TI′ are the adversarial devices used as the intermediaries in a relay attack).

Hence, at approximately the same time, we have three separate ambient sensor measurements for the three devices (TT′ does not record any sensor values). Overall, for an ambient sensor to be effective, the Trusted Authority should be able to distinguish the genuine pair from the illegitimate pair. To evaluate each ambient sensor's effectiveness for proximity detection and to detect relay attacks, we analysed the collected data using threshold and machine learning based analyses.

7.4.2 Test-bed Construction

The devices TT and TI′ were placed at a distance of 5ft from the devices TT′ and TI. When TI′ was brought in close proximity to TT, an NFC connection between the two devices would be established, initiated by TT, indicating the beginning of a transaction. According to the EMV standard, TT and TI′ should be in proximity, i.e. within the intended 5cm distance of NFC [285]. During the analysis process, the pair TT–TI′ represented the genuine devices, where no relay attack was involved. The pair TT–TI represented the genuine devices, where a relay attack was active.

Three Samsung Galaxy S4 (GT-I9500) Android devices running Android 5.0.1 were used in the experimental phase for data collection[7]. The reader is referred back to Table 7.2 in Section 7.3.1 for the availability of sensors aboard this handset. Additionally, a Nexus 5 Android device was used as TT′, which was not collecting sensor data. As we saw in Table 7.3 in Section 7.3.2, many sensors could not

[7]This selection differs from the previous study in Section 7.3 as the two studies were conducted approximately 6 months apart and we no longer had access to the SGS5 Mini or Nexus 9.

capture any values in the 500ms timeframe in our initial tests. Based on this, we again eliminated the sensors exhibiting over 90% failure rates from subsequent analysis, namely Bluetooth, GPS, Network Location and WiFi sensors. We also eliminated Ambient Temperature and Relative Humidity sensors, due to their rarity on mobile handsets at the time of writing[8]. Another issue we encountered was also raised in the previous section. The proximity sensor on Samsung Galaxy S4 returns only a true or false value when the sensor, located in the front of the device, is covered or uncovered. As such, it could not be used effectively in the experimental phase. Lastly, the sound sensor, i.e. the microphone, was also tested; we found that this could not initiate and record values within 500ms for the Galaxy S4 handsets. Consequently, we also removed this sensor from subsequent analysis; however, we note that on *some* devices, this is possible, namely the Nexus 9 and Nexus 5 evaluated previously. We omitted this sensor for the purposes of this evaluation, but we stress that while it *could* form an effective relay attack detection mechanism, the previous evaluation provided evidence to the contrary. It is unknown whether this issue is prevalent across a wide range of handsets. This process of elimination led to the selection of seven sensors from which to collect data during field trials: *accelerometer*, *gyroscope*, *magnetic field*, *rotation vector*, *gravity*, *light*, and the *linear acceleration* sensor.

A total of 400 transactions were collected for each sensor, distributed over two distinct locations on the university campus. For each transaction, a sensor capture lasting 500ms was initiated upon the touching of TT with TI′, and TT′with TI on both sides using NFC. Upon completion, the devices TT, TI′, and TI stored the collected sensor data locally for off-line analysis. The complete implementation of the test-bed, data analysis and collection data sets for this second analysis are available at: `https://github.com/AmbientSensorsEvaluation` for further research in the domain of relay attack detection under time-critical contactless NFC transactions.

Transaction Synchronisation

The first challenge is designing a framework in which three devices can collect sensor data simultaneously using a fourth device as an intermediary relay, while remedying the issue of the devices recording at significantly different times (or not at all). This is more difficult than the previous study, which relied on data from two communicating devices and can be implemented with a simpler protocol with only one NFC-based connection. To this end, three Android applications were developed in total. Firstly, devices TI′ and TI were running the same

[8]While evaluating the availability of sensors across the mobile market was considered out-of-scope, informal analyses by mobile enthusiasts conclude that only four of today's devices include these sensors [308].

application, while a second application, for device TT, included two connection interfaces. The first interface was used for the NFC connection with TI′ in which TT was set to NFC reader mode, allowing it to interact with discovered NFC device. The second interface for TT was used for connecting with TT′ over WiFi in which messages were relayed broadcast a UDP packet in the local network. For both interfaces, the message transmitted on the NFC or wireless channel included the sensor to be measured in that transaction, and a transaction ID. In real-world scenarios, device TI′ would act as the device communicating with TT′. However, since the scope of this paper is to evaluate the effectiveness of the ambient environment as an anti-relay mechanism, device TT was responsible for sending the information across a WiFi (via the hotspot functionality) for the aforementioned synchronisation. Devices TT and TT′ were connected to the same wireless hotspot, created for the requirements of the experiment. Lastly, the third application running on device TT′ contained a broadcast listener for UDP packets from TT. Upon receiving a packet from device TT, device TT′ would be able to initiate a transaction with device TI, upon tapping the latter to the former. After the initiation of a transaction, devices TT, TI′, and TI would start recording data using a predefined sensor for 500ms. The use of this synchronisation set-up was for the purposes of off-line transaction analysis, and was independent of the measurement collection itself, i.e. the purpose of this study. Figure 7.8 presents an overview of the recording process.

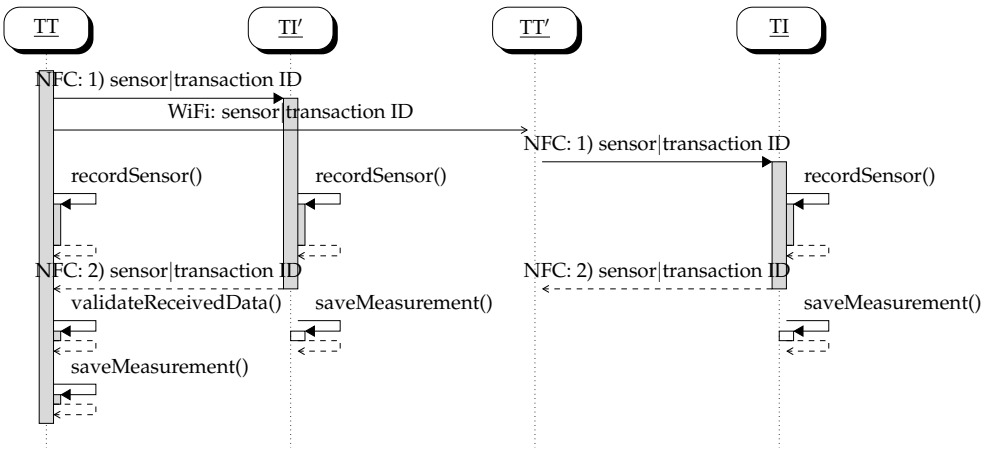


FIGURE 7.8: Measurement recording process.

Device Data Collection

On the Android operating system, data captured by a sensor is returned to an application in time intervals set by the application. The rate at which data was polled from the sensors was set at the highest available common across all three devices.Following the recording time period, devices TI′ and TI would send

a response message, containing the transaction ID and sensor used to TT and TT′, respectively. Device TT would then validate the received data. The three devices would store the recorded data in a local SQLite database, along with the transaction ID, sequence number, timestamp and the pre-defined location in which the recording took place. During the data analysis phase, only transactions that existed in all three databases, based on their transaction ID, were considered. This is also shown in Figure 7.8.

7.4.3 Data Analysis

As mentioned previously, we limited our sensor selection to the best performing sensors from our evaluation on proximity detection (Section 7.3), i.e. those which successfully and consistently captured values within the 500ms time limit over 1,000 transactions. The same analysis techniques were used as the previous work for proximity detection – threshold-based analysis with the same similarity measures, and machine learning – using the EER evaluation metric. However, this time the *illegitimate* dataset, **I**, was created from all measurements taken between TT and TI, as TI was located 1.5m away, i.e. $\mathbf{I} = \{(TT_1, TI_1), (TT_2, TI_2), \ldots, (TT_n, TI_n)\}$. The *legitimate* transaction set, **L**, was created using the sensor measurements between TT and TI′, i.e. $\mathbf{L} = \{(TT_1, TI'_1), (TT_2, TI'_2), \ldots, (TT_n, TI'_n)\}$. At first, it makes little sense to collect data from an adversary device, TI′. However, since TI′ is in a 'legitimate' position according to the terminal/PoS, i.e. NFC distance (<5cm), this pair was used as the legitimate dataset for convenience and time-saving during field trials.

7.4.4 Results Discussion

The results for the threshold-based and machine learning analyses of this study can be found in Tables 7.6 and 7.7 respectively. Similar to our previous study (Section 7.3), some sensors provided some but poor discriminatory power: the magnetic field sensor gave EERs of 36.1–43.3% in the analyses, linear acceleration yielded EERs of 30.7–44.3%, rotation vector (28.5–32.7%), light (29.3–36.7%; the EER of light is illustrated in Figure 7.9), and accelerometer (27.7–46.8%). The remaining two sensors provided greater discriminatory power, namely the gyroscope (17.9% EER with Random Forest) and the rotation vector sensor (27.7%, also with Random Forest).

Even in the best case, using gyroscope with random forest, it still implies that 17.9% of legitimate and illegitimate transactions would be respectively denied and accepted incorrectly. An error rate of approximately one-in-five is even poorer than our results in the previous study presented in the last section. A likely reason is that the first study used measurements taken at wildly different

TABLE 7.6: Threshold-based EERs (using the metric abbreviations from Table 7.4).

Sensor	MAE	PCC	C-Corr	ED	Coh	T-FD
Accelerometer	0.494	0.477	0.590	**0.468**	0.507	0.590
Gyroscope	0.521	**0.455**	0.535	0.495	0.528	0.489
Magnetic Field	0.444	0.473	0.470	**0.433**	0.487	0.470
Rotation Vector	0.330	0.472	**0.327**	0.670	0.534	0.509
Gravity	0.521	0.490	0.401	**0.289**	0.503	0.362
Light	**0.367**	0.488	0.444	0.372	0.505	0.437
Linear Acceleration	0.482	0.536	0.503	0.506	**0.443**	0.493

TABLE 7.7: Estimated EER for machine learning algorithms, obtained by repeating 10-fold cross-validation 10 times.

Sensor	Random Forest	Naive Bayes	Decision Tree	Logistic Regression	Support Vector Machine
Accelerometer	**0.277**±0.052	0.474±0.047	0.358±0.059	0.483±0.050	0.454±0.126
Gyroscope	**0.179**±0.041	0.354±0.059	0.228±0.049	0.356±0.055	0.288±0.045
Magnetic Field	**0.361**±0.055	0.400±0.053	0.389±0.063	0.421±0.061	0.385±0.053
Rotation Vector	**0.285**±0.052	0.327±0.055	0.317±0.073	0.353±0.050	0.325±0.050
Gravity	0.499±0.046	0.488±0.043	0.494±0.057	**0.484**±0.043	0.486±0.156
Light	0.361±0.059	0.369±0.058	**0.293**±0.149	0.407±0.054	0.351±0.054
Linear Acceleration	**0.307**±0.050	0.484±0.048	0.392±0.057	0.502±0.049	0.397±0.058

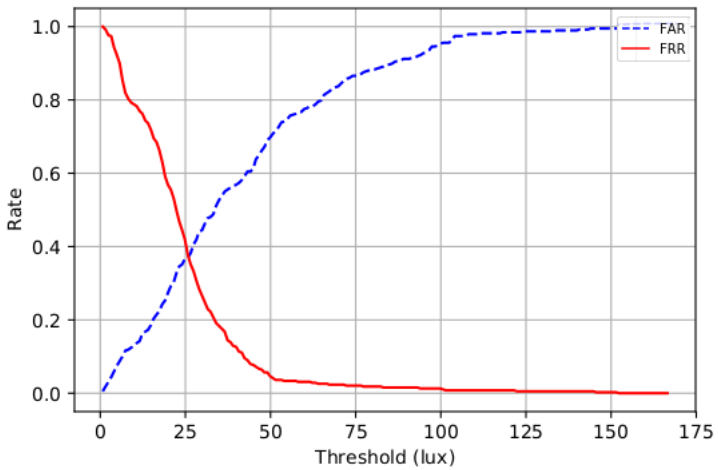


FIGURE 7.9: FAR-FRR curves for the light sensor using the MAE similarity metric.

locations in the formation of the illegitimate transaction set. This is relevant for relay attacks that occur at great distances, e.g. 100m–200m (the physical distances between the university locations in the previous study). However, this investigation aimed to evaluate 'worst case' relay attacks that take place in a store queue or other close-quarters environment. From this analysis, it is difficult to recommend any of the sensors for a single-sensor deployment for high security applications, such as banking. Such sensors might be appropriate for low-security access control, but we recommend that a thorough analysis of the sensors and their performance in the chosen domain is performed prior to deployment.

7.5 Conclusion

Proximity and Relay-Attack Detection (PRAD) is an important element for many contactless and wireless technologies. In this work, we illustrated that the viability of PRAD mechanisms can be largely dependent on the time constraints mandated by industry requirements. Contactless payment transactions for example – whether smart card- or smartphone-based – must adhere to <3cm for proximity and <500ms for transaction duration, as stipulated by the EMV specifications. We evaluated the claim that natural ambient environments can provide a robust PRAD, as stated by some previous literature, under industry-specified time constraints. This was evaluated for both proximity detection (Section 7.3) and as a relay attack detection mechanism using a test-bed that reflected an actual attack (Section 7.4). We presented the results of a two-part evaluation using six similarity metrics used previously and several widely-used machine learning classifiers. In all cases, the results were far from what was claimed in past literature; our initial results indicate that natural ambient environments perform poorly in time-critical domains like banking and transport.

One reason that past work achieved different results is the significantly larger sampling durations (see Section 7.2.3), which ranged up to 30 seconds [296]. The sampling duration imposed in our experiments was in line with the performance requirements of an EMV application, i.e. 500 milliseconds. Transportation is another major application for contactless smart cards; in this domain, the recommended transaction duration is far lower, between 300-500 milliseconds. The 500 millisecond limit in our experiment was thus an upper bound of the recommendations of two significant application areas where contactless mobile phones may be utilised. Some work, such as Shrestha et al. [293], fuse sensing modalities in order to improve accuracy. In these past two studies, we were interested in addressing the base case of ruling out the use of *individual* sensors, as per Jin et al. [298], Ma et al. [292], Mehrnezhad et al. [297], Halevi et al. [291], and Urien et al. [295]. We stress that evaluating multiple sensor combinations would be

the first avenue for future work in evaluating the efficacy of ambient sensors for proximity and relay attack detection, which has been suggested in the remaining works, e.g. Shrestha et al. [293].

Notwithstanding, our results cast doubt over the efficacy of the previous work in this area under the industry-specific time constraints mandated by EMV and ITSO. Having evaluated a set of sensors available through the Android API, we discovered that the proposed methods in related work do not necessarily scale when sampling durations are reduced to these times. As a result, we are inclined to reject the null hypothesis set out in Section 7.1.1 in the case of using single sensors. This is based on the substantially high error rates (9.2% in our best case for proximity detection using the pressure sensor with decision tree classifier; 17.9% EER for relay attack detection using the gyroscope with random forest) versus those exhibited in existing work, e.g. up to 100% attack detection success rates in [298] and [292], which use longer transaction durations. However, as discussed, evaluating multiple sensors would be a logical next step in providing stronger evidence towards rejecting the null hypothesis. Additionally, our results provide evidence towards accepting the alternative hypothesis, also set out in Section 7.1.1. Notably, in 7.3 of Section 7.3.2, we found that some modalities used in existing work, such as WiFi [296], Bluetooth [294] and GPS location [292], [294], failed to collect any data samples under the transaction duration limit. Once again, evaluating multiple sensors would provide further evidence to this end: work by Shrestha et al. [293] already shows that multiple inputs significantly improves error rates using longer sampling durations. It remains to be seen, however, whether sufficient information is captured by multiple modalities to lower error rates to an acceptable level within the duration considered in this work. In short, we strongly recommend that any sensor-based PRAD proposal should be evaluated *in situ* based on the operating restrictions of the intended deployment domain.

7.5.1 Future Research

In future research, we hope to investigate the following:

- *Effectiveness of sensor fusion techniques with multiple input modalities.* Simultaneously measuring multiple sensors within the transaction time duration, and applying sensor fusion techniques in order to robustly assess whether it does, indeed, reduce the risk associated with using sensors individually.

- *Evaluating a greater diversity of devices.* Re-evaluating each sensor over a greater selection of devices. Our initial experiments in Sections 7.3.1 and 7.4.2 showed that devices behave significantly different when initiating and collecting data from sensors over small time-frames. The Galaxy S4

failed to initialise and record any data samples in the 500ms time-frame, while the Nexus 5 and 9 devices managed to collect data for approximately half of the allotted time. It is our suspicion that this is due to latencies introduced in the OEM's HAL for interfacing between the device software and SoC hardware sensors, which is device-dependent. This investigation may eventually lead to the examination of real-time operating systems (RTOSs) for minimising data collection latencies.

- *Alternative relay mediums.* Our second study employed WiFi as the relay link in each transaction. In future work, we aim to evaluate alternative relay communication mediums, particularly Bluetooth, that an attacker may deploy in order to mount an attack. Wireless typically differ with respect to possible bit-rates, including their potential depreciation through objects, such as concrete and paper. It would be interesting to evaluate the effectiveness of various mediums in conducting successful relay attacks.

Chapter 8

Concluding Remarks

8.1 Summary

This thesis has shown that TEEs can be utilised to achieve a multitude of security and trust assurances for protecting critical applications and their data aboard constrained embedded devices. The primary benefits of TEEs include hardware-assisted protection from user- and kernel-mode adversaries that aim to compromise data confidentiality and integrity; a trusted path between I/O peripherals and the TEE without trusting intermediate components; and tight integration with the underlying execution hardware used by a conventional OS, thus maximising performance and eliminating the need for additional security hardware modules, such as a dTPMs and SEs. However, given their relative infancy, TEEs lack the maturity and the wealth of literature that exists, particularly in the TPM world, regarding the management, performance, under a number of disparate applications. Consequently, many existing solutions were simply incompatible to TEEs without extensive modifications and re-framing, and omitted the additional benefits provided by TEEs, as well as their unique challenges. This is further complicated by the challenges in addressing the divergence in competing TEEs that feature disparate architectural and manufacturing assumptions.

The contributions presented in this thesis have often involved referencing and reconciling previous trusted computing literature in order to account for the differences, challenges and benefits provided by TEEs. Ultimately, this has resulted in contributions that go above and beyond the security, trust and performance assurances offered by previous technologies, such as SEs, TPMs and smart cards, for protecting the assets aboard constrained embedded devices. This thesis also aimed to bridge theory and practice by backing its contributions using performance evaluations with actual TEEs on commercial platforms. This provided a source of quantitative measurements to evaluate the practical viability of the proposed schemes. Moreover, we verified any proposed protocols under known adversarial models using formal verification tools to assure correctness; however, we must stress that the reader must be aware of the limitations of such

tools, as discussed in Section 4.5.4 of Chapter 4. Additionally, we strived to make the contributions agnostic to the underlying architecture in order to maximise their applicability to any TEE.

We began in Chapter 2 by describing the system architectures of modern mobile and embedded devices in order to inform subsequent discussions of secure and trusted execution technologies. This included reviewing microcontrollers (MCUs), single-board computers (SBCs), and modern system assembly practices for minimising PCB sizes while retaining performance, such as system-on-chips (SoCs), system-in-packages (SiPs) and package-on-packages (PoPs). After this, a detailed review was presented of the evolution of secure and trusted execution technologies, both past and present, for preserving sensitive applications and their data. This included smart cards; Secure Elements (SEs); the Trusted Platform Module (TPM); early TEEs, such as Intel TXT and AMD SVM, which were constructed upon TPMs, Nokia ObC, Microsoft Palladium and TI M-Shield; and the most recent developments in TEEs, such as Intel SGX and the GlobalPlatform TEE specifications. We also discussed the Common Criteria framework in evaluating these platforms in relation to their claimed security guarantees. At the end of this chapter, a security comparison was presented in relation to the relative capabilities of these technologies, and their ability in defending against the on- and off-device adversaries described in the GlobalPlatform specifications. We seen how TEEs are a promising candidate – albeit with trade-offs, particularly with respect to hardware tamper-resistance – for protecting sensitive applications of embedded devices used in many suggested IoT deployments.

The first contribution of this thesis, presented in Chapter 3, was an investigation of applicability of TEEs in protecting the assets of continuous authentication (CA) schemes – an emerging paradigm for addressing the shortfalls of existing schemes, such as PINs, hardware tokens and fingerprint recognition. In this chapter, we analysed the potential assets and threats facing the implementation of CA schemes in a standard OS environment. To counter these, we proposed a TEE-based solution for protecting the scheme under execution and its behavioural model, which we evaluated using a publicly-available dataset on both Intel SGX, using an off-the-shelf Lenovo T460s device, and a Hikey LeMaker board with ARM TrustZone. The results found that TEE-based CA is both practical and worthwhile for protecting the run-time states of CA schemes from kernel-level adversaries, with a modest performance overhead.

This thesis' second contribution focussed on the construction of mutually trusted channels between TEEs on remotely located devices that wish communicate in a secure fashion without trusting intermediary components. In Chapter 4, we proposed two protocols for providing bi-directional trust, where both devices each host TEEs, and uni-directional trust for supporting situations where only

one device hosts a TEE, akin to traditional remote attestation. The proposals were evaluated on two ARM development boards reflecting the typical specifications of SBC-based IoT devices. Our practical implementation on two Hikey Lemaker boards exhibited approximately four-times overhead versus TLS with DHE and RSA, with a total round-trip time of 1.7 seconds without optimisation. Moreover, both protocols are subjected to formal symbolic analysis using Scyther against a Dolev-Yao adversary, which found no attacks.

The third contribution addressed the challenges of tamper-resistant logging for protecting data aboard constrained devices using TEEs for use in auditing and forensics. The state-of-the-art of wholly cryptographic and TEE-based schemes for preserving the confidentiality, integrity and authenticity of log records was explored. This was followed by identifying several shortfalls in existing work and formulating a list of the security and functional goals we aimed to tackle with respect to constrained device logging. In light of this, we presented the design, implementation and evaluation of a novel system, EmLog, for such protecting logs on these devices against complex software-based adversaries in the untrusted world. The proposed scheme also offers several additional properties over past proposals, such as public verifiability of logs and key compromise resilience. EmLog is evaluated on three log datasets using an off-the-shelf ARM development board with an open-source, GlobalPlatform-compliant TEE. On average, EmLog runs with low run-time memory overhead (1MB heap and stack), 430–625 logs/second throughput, and five-times persistent storage over-head versus unprotected logs alone.

The focus then shifted to the problem of secure and trusted remote management of TEE-bound credentials aboard constrained devices applied to centralised use-cases, like industrial IoT (IIoT) and smart cities. This served as the fourth contribution of this thesis. In this chapter, we posed five challenges for deploying IoT TEEs: remote credential backups, updates, migration and revocation, as well as extending the work in the previous chapter for retrieving logs of IoT TEEs in a secure and trusted fashion. To this end, we showed how the contributions from Chapter 4 can be extended to provide a series of secure and trusted procedures for performing credential management and audit log retrieval using mutual attestation. The protocols are developed using the requirements and threat model relevant to IoT TEEs, and subjected to formal verification using Scyther, which found no attacks.

The fifth contribution of this thesis, presented in Chapter 7, segued into investigating and evaluating the suitability of mobile sensors in protecting NFC-based mobile transactions from relay attacks. Such attacks are a pertinent vector for defeating the protections afforded by TEEs and misusing the credentials stored therein during contactless transactions, e.g. payments and transport ticketing. We

evaluated the state-of-the-art for preventing relay attacks using methods based on detecting the environment of the communicating devices. Notably, related work has been conducted largely independently of the time-constraints imposed by industry, namely 500ms for EMV and 300-500ms for transport applications. Next, a two-part study was conducted for assessing the efficacy of 17 ambient sensors available through the Android API for *proximity* and *relay attack* detection under a time restriction of 500ms. Our analysis, comprising both similarity- and machine learning-based methods proposed in related work, produced poor error rates with respect to the usability and security of even the best-performing sensors. We subsequently call into question the applicability of these schemes with respect to the operational timing restraints imposed in practice by industry.

It is hoped that the contributions presented throughout this thesis provide useful set of proposals, recommendations and further ideas for establishing greater security and trust assurances in TEEs on embedded devices. The contributions were inspired towards protecting the most sensitive applications and data envisaged in the IoT sphere, such as in assistive technologies, logistics, industrial IoT, and healthcare.

8.2 Future Directions and Challenges

This section highlights potential research directions and challenges, technical and otherwise, in the deployment of TEEs on constrained devices.

8.2.1 Group TEE Attestation

In the contributions involving remote attestation, primarily Chapters 4 and 6, we focussed on basic static attestation between two communicating entities. A worthwhile endeavour would be the exploration of alternative attestation mechanisms, such as dynamic attestation (see Section 4.2.2 of Chapter 4), which aims to protect run-time states. Moreover, we did not generalise the proposed protocols this to $n > 2$ parties. This could be potentially useful in the construction of multi-party secure and trusted channels for, for example, sharing authentication scores across a Body Area Network (BAN) for continuous authentication, as proposed by Hintze et al. [309]. The generalisation of key exchange protocols, such as (EC-)Diffie-Hellman is well-studied in related literature; however, it is less studied how group attestation scales, besides previously mentioned work by Asokan et al. [73] for swarm attestation, which primarily aggregates attestation values, and not a shared secure channel between all nodes.

8.2.2 Post-Quantum TEEs (PQ-TEEs)

Another potential research direction is the application of post-quantum cryptography (PQC) constructs to TEEs and their supporting mechanisms. PQC is borne from the vulnerability of the hardness assumptions of traditional cryptosystems, such as integer factorisation (RSA), discrete logarithm problem (Diffie-Hellman) or the elliptic-curve discrete logarithm problem (ECC), to quantum computers, principally using Shor's algorithm. This has led to the development of lattice- (NTRU, BLISS), hash- (Merkle signature scheme), code- (McEliece), and isogeny-based schemes, which rely upon hardness assumptions believed to be quantum-hard [310]–[314]. The solutions presented throughout this thesis were not designed with quantum adversaries in mind. This also extends to commercial solutions: Intel SGX's remote attestation mechanism, for example, is based on ECDH and thus vulnerable to a quantum adversary. Quantum-resistant TEE mechanisms, however, have so far received little attention from the community. We note that TPM attestation *has* began to account for these threats: both Ando et al. [315] and Liu et al. [316] propose hash-based signatures using the Merkel signature scheme for quantum-resistant TPM remote attestation quotes, with approximately five-times overhead versus classical-based ECDSA reported in [316]. However, the literature remains sparse in this area. We are currently unaware of any work that has began to explore quantum-resistant cryptographic algorithms for TEEs, such as for attestation, secure provisioning and storage.

8.2.3 Evaluating ARM TrustZone-M

The implementations described in Chapters 3, 4 and 5 were evaluated using a development board based on an ARM Cortex-A53, which instantiates a TEE using TrustZone for the ARMv8-A architecture (TZ-A). Section 2.4.4, however, described TrustZone for ARMv8-M (TrustZone-M, or TZ-M), which provides the same high-level features but differs significantly in how TrustZone is realised for more constrained, Cortex-M based platforms, like embedded microcontrollers. Notably, TZ-M aims to minimise the overhead of secure/non-secure world switching through, for example, sharing virtually all general-purpose registers between the worlds and avoiding the use of a secure monitor exception handler to mediate world switches. However, a lack of access to development kits precluded an evaluation of TZ-M in this thesis. As such, in future work, it would be useful to re-visit the implementations in the above chapters to empirically evaluate the performance of TZ-A versus TZ-M. In particular, assessing the energy consumption and latency benefits yielded during world context switches would be useful.

8.2.4 Ethics

The application of TEEs also has challenges of an ethical dimension. It is easy to see how TEEs can be used to securely protect user credentials, for payments and biometrics, and other data beneficially. However, it is similarly conceivable how TEEs could be misused to implement technologies that significantly impede on users' privacy, with virtually no control offered to end-users under a closed-access model. An extreme example being a tracking TA that harvests location data from a GPS module over a trusted path in order to inform targeted advertising. Another example is a 'secure' voice or messaging that service that encrypts messages under a TEE-bound key accessible to the authorities of an intrusive organisation or even government. TEE-enforced DRM, such as Widevine used by Netflix and Amazon Prime Video, edges towards this dilemma. DRM has long been criticised for removing freedom of choice for users, but also as a necessary step for protecting intellectual property. This dilemma, with respect to trusted computing, has long-standing roots dating back to the early 2000s on the development of Palladium [172]. Some privacy and free software activists, such as Richard Stallman, have even deemed trusted computing as 'treacherous' for removing control away to users to the benefit of private organisations [317]. This thesis avoided potentially contentious applications, focussing on ones generally beneficial applications to both users and enterprises. However, it is important for TEEs researchers and developers to remain ethically aware of proposing and implementing solutions that infringe on users' liberty.

8.2.5 TEE Intrusion Tolerance

While TEEs undoubtedly exhibit greater assurances relative to standard OS environments (REEs), they are not evaluated to the same rigour as Secure Elements (SEs) and smart cards. Modern TEEs are typically written in memory-unsafe languages, i.e. C/C++, and evaluated to a lower assurance level versus smart cards and secure elements: CC EAL2 for TEEs versus CC EAL4+ for smart cards and SEs. This implies TEEs are assured with less rigour regarding their stated security goals compared with SEs and smart cards. Moreover, TEEs consider many threats out of scope: errors in inter-world API logic and function definitions, e.g. that contain buffer overflow vulnerabilities from unchecked array bounds, could undermine a TEE-based security solution.

A final future research direction might be the evaluation of approaches that aim to detect and limit the damage caused by TA errors, such as local dynamic attestation for internal TAs, or TEE OS IDSs that continually monitor TA states for detecting unexpected behaviour. This is likely to lead to an examination of literature in field of intrusion tolerance – well-explored for aircraft and industrial

control systems, and safety-critical systems generally – for providing guarantees regarding security properties even when one or more TEE components are compromised. This includes backup mechanisms, for immediately backing data to more trusted elements for forensic analysis in the detection of a potential compromise; redundancy mechanisms that provide multiple replicated sources of input, e.g. sensor inputs, for detecting whether values have been corrupting using statistical measures; and forwards error recovery, in which the system detects a compromise and moves to a more trusted element, such as a SE, for performing subsequent operations [318].

Bibliography

[1] A. Pantelopoulos and N. G. Bourbakis, "A survey on wearable sensor-based systems for health monitoring and prognosis", *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 40, no. 1, pp. 1–12, 2010.

[2] D.-M. Han and J.-H. Lim, "Smart home energy management system using IEEE 802.15.4 and ZigBee", *IEEE Transactions on Consumer Electronics*, vol. 56, no. 3, 2010.

[3] N. Noury, T. Hervé, V. Rialle, G. Virone, E. Mercier, G. Morey, A. Moro, and T. Porcheron, "Monitoring behavior in home using a smart fall sensor and position sensors", in *1st Annual International Conference on Microtechnologies in Medicine and Biology0*, IEEE, 2000, pp. 607–610.

[4] M. M. Aung and Y. S. Chang, "Temperature management for the quality assurance of a perishable food supply chain", *Food Control*, vol. 40, pp. 198–207, 2014, Elsevier.

[5] E. S. Nadimi, R. N. Jørgensen, V. Blanes-Vidal, and S. Christensen, "Monitoring and classifying animal behavior using ZigBee-based mobile ad hoc wireless sensor networks and artificial neural networks", *Computers and Electronics in Agriculture*, vol. 82, pp. 44–54, 2012, Elsevier.

[6] Gartner, *8.4 Billion Connected 'Things' Will Be in Use in 2017, Up 31 Percent From 2016*, http://www.gartner.com/newsroom/id/3598917, 2017.

[7] Statista, *Internet of Things (IoT) connected devices installed base worldwide from 2015 to 2025*, https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/, 2018.

[8] IDC, *Internet of Things spending forecasts*, https://www.idc.com/getdoc.jsp?containerId=prUS42209117, 2017.

[9] PwC, *Industrial Internet of Things*, https://pwc.com/id/en/publications/assets/ticepublications/industrial-internet-of-things.pdf, 2016.

[10] Bain & Company, *Choosing the right platform for the Industrial IoT*, http://www.bain.com/publications/articles/choosing-the-right-platform-for-the-industrial-iot.aspx, 2017.

[11] A. Nordrum, "Popular Internet of Things forecast of 50 billion devices by 2020 is outdated", *IEEE Spectrum*, vol. 18, 2016, https://spectrum.ieee.org/tech-talk/telecom/internet/popular-internet-of-things-forecast-of-50-billion-devices-by-2020-is-outdated.

[12] E. Bertino and N. Islam, "Botnets and Internet of Things security", *IEEE Computer*, vol. 50, no. 2, pp. 76–79, 2017, ISSN: 0018-9162. DOI: 10.1109/MC.2017.62.

[13] McAfee, *Threat Report: March 2017*, https://mcafee.com/us/resources/misc/infographic-threats-report-mar-2017.pdf, 2017.

[14] B. Krebs, "Did the Mirai botnet really take Liberia offline?", *Krebs on Security*, vol. 4, 2016, https://krebsonsecurity.com/2016/11/did-the-mirai-botnet-really-take-liberia-offline/.

[15] U. Lindqvist and P. G. Neumann, "The future of the Internet of Things", *Communications of the ACM*, vol. 60, no. 2, pp. 26–30, 2017.

[16] J. Wurm, K. Hoang, O. Arias, A.-R. Sadeghi, and Y. Jin, "Security analysis on consumer and industrial IoT devices", in *21st IEEE Asia and South Pacific Design Automation Conference*, IEEE, 2016, pp. 519–524.

[17] M. C. Domingo, "An overview of the Internet of Things for people with disabilities", *Journal of Network and Computer Applications*, vol. 35, no. 2, pp. 584 –596, 2012.

[18] L. Da Xu, W. He, and S. Li, "Internet of Things in industries: A survey", *IEEE Transactions on Industrial Informatics (TII)*, vol. 10, no. 4, pp. 2233–2243, 2014.

[19] C. Shepherd, F. A. P. Petitcolas, R. N. Akram, and K. Markantonakis, "An Exploratory Analysis of the Security Risks of the Internet of Things in Finance", in *Proceedings of the 14th International Conference on Trust and Privacy in Digital Business*, ser. TrustBus '17, Springer, 2017, pp. 164–179.

[20] B. Schneier, *Security economics of the Internet of Things*, https://www.schneier.com/blog/archives/2016/10/security_econom_1.html, 2016.

[21] F-Secure, *Pinning down the IoT*, https://www.fsecureconsumer.files.wordpress.com/2018/01/f-secure_pinning-down-the-iot_final.pdf, 2018.

[22] UK Government Office for Science, *The Internet of Things: Making the most of the Second Digital Revolution*, https://gov.uk/government/uploads/system/uploads/attachment_data/file/409774/14-1230-internet-of-things-review.pdf, 2014.

[23] S. Sicari, A. Rizzardi, L. A. Grieco, and A. Coen-Porisini, "Security, privacy and trust in Internet of Things: the road ahead", *Computer Networks*, vol. 76, pp. 146–164, 2015, Elsevier.

[24] Z. Yan, P. Zhang, and A. V. Vasilakos, "A survey on trust management for Internet of Things", *Journal of network and computer applications*, vol. 42, pp. 120–134, 2014.

[25] S. R. Islam, D. Kwak, M. H. Kabir, M. Hossain, and K.-S. Kwak, "The Internet of Things for health care: a comprehensive survey", *IEEE Access*, vol. 3, pp. 678–708, 2015.

[26] K. T. Nguyen, M. Laurent, and N. Oualha, "Survey on secure communication protocols for the Internet of Things", *Ad Hoc Networks*, vol. 32, pp. 17–31, 2015.

[27] US National Security Agency (NSA), *Security-Enhanced Linux*, https://www.nsa.gov/what-we-do/research/selinux/, 2008.

[28] Android Open Source Project, *SELinux for Android 8.0*, https://source.android.com/security/selinux/images/SELinux_Treble.pdf, 2018.

[29] C. J. Cremers, "The Scyther tool: Verification, falsification, and analysis of security protocols", in *International Conference on Computer Aided Verification*, Springer, 2008, pp. 414–418.

[30] S. B. Furber, *ARM System-on-Chip Architecture*. Pearson, 2000.

[31] Wikipedia, *Example ARM-based system-on-chip (Image)*, https://en.wikipedia.org/wiki/System_on_a_chip, 2018.

[32] IEEE, *1149.1-1990 – Standard Test Access Port and Boundary-Scan Architecture*, https://standards.ieee.org/findstds/standard/1149.1-1990.html, 1990.

[33] A. Fontanelli, "System-in-package technology: Opportunities and challenges", in *9th International Symposium on Quality Electronic Design*, IEEE, 2008, pp. 589–593.

[34] S. F. Barrett and D. J. Pack, "Atmel AVR microctonroller primer: programming and interfacing", *Synthesis Lectures on Digital Circuits and Systems*, vol. 7, no. 2, pp. 1–244, 2012.

[35] K. Sung, "Recent video game console technologies", *Computer*, vol. 44, no. 2, pp. 91–93, 2011, IEEE.

[36] J. Liénard, A. Vogs, D. Gatziolis, and N. Strigul, "Embedded, real-time UAV control for improved, image-based 3D scene reconstruction", *Measurement*, vol. 81, pp. 264–269, 2016, Elsevier.

[37] . Boards, *HiKey LeMaker Specifications*, http://www.lemaker.org/product-hikey-specification.html, 2018.

[38] International Organisation for Standardisation, *ISO/IEC 15408: Information technology – Security techniques – Evaluation criteria for IT security*, https://www.iso.org/standard/50341.html, 2009.

[39] R. J. Anderson, *Security Engineering: A Guide to Building Dependable Distributed Systems*. John Wiley & Sons, 2010.

[40] GlobalPlatform, *TEE Protection Profile (v1.2)*, 2014.

[41] ——, *TEE Secure Element API (v1.0)*, 2015.

[42] United States Computer Emergency Readiness Team (US-CERT), *The Common Criteria*, https://www.us-cert.gov/bsi/articles/best-practices/requirements-engineering/the-common-criteria, 2013.

[43] S. Arora, "National e-id card schemes: A european overview", *Information Security Technical Report*, vol. 13, no. 2, pp. 46–53, 2008.

[44] International Organisation for Standardisation, *ISO/IEC 14443: Identification cards – Contactless integrated circuit cards*, https://www.iso.org/standard/70170.html, 2016.

[45] ——, *ISO/IEC 7816: Identification cards – Integrated circuit cards*, https://www.iso.org/standard/54089.html, 2011.

[46] W. Rankl and W. Effing, *Smart Card Handbook*. John Wiley & Sons, 2004.

[47] G+D Group, *Smart Health Cards*, https://www.gi-de.com/mobile-security/industries/public-sector/smart-health-cards/, 2018.

[48] A. Umar, "On the security and performance of mobile devices in transport ticketing", PhD thesis, Royal Holloway, University of London, 2018.

[49] B. Vibert, C. Rosenberger, and A. Ninassi, "Security and performance evaluation platform of biometric match on card", in *World Congress on Computer and Information Technology (WCCIT)*, IEEE, 2013, pp. 1–6.

[50] GlobalPlatform, *Card Financial Configuration (v1.0)*, 2018.

[51] ——, *GlobalPlatform UICC Configuration (v2.0)*, 2015.

[52] ——, *GlobalPlatform Card Common Implementation Configuration (v2.0)*, 2015.

[53] ——, *GlobalPlatform Card Contactless Extension Configuration (v2.0)*, 2015.

[54] Oracle, *Java Card Technology*, http://www.oracle.com/technetwork/java/embedded/javacard, 2018.

[55] A. Umar, K. Mayes, and K. Markantonakis, "Performance variation in host-based card emulation compared to a hardware security element", in *1st IEEE Conference on Mobile and Secure Services*, IEEE, 2015, pp. 1–6.

[56] Consult Hyperion and GSMA, *HCE and tokenisation for payment services*, https://www.gsma.com/digitalcommerce/wp-content/uploads/2014/11/GSMA-HCE-and-Tokenisation-for-Payment-Services-paper_WEB.pdf, 2014.

[57] Trusted Computing Group (TCG), *TCG Glossary 1.1, Rev. 1.00*, 2017.

[58] International Standards Organisation, *ISO/IEC 11889 – Information technology – Trusted platform module library*, 2015.

[59] S. Balfe, E. Gallery, C. J. Mitchell, and K. G. Paterson, "Challenges for trusted computing", *IEEE Security & Privacy*, vol. 6, no. 6, pp. 60–66, 2008.

[60] A.-R. Sadeghi, "Trusted computing—special aspects and challenges", in *International Conference on Current Trends in Theory and Practice of Computer Science*, Springer, 2008, pp. 98–117.

[61] W. Arthur, D. Challener, and K. Goldman, *A Practical Guide to TPM 2.0*, https://link.springer.com/chapter/10.1007/978-1-4302-6584-9_12, 2015.

[62] T. Abera, N. Asokan, L. Davi, F. Koushanfar, A. Paverd, A. R. Sadeghi, and G. Tsudik, "Things, trouble, trust: on building trust in IoT systems", in *53rd ACM/EDAC/IEEE Design Automation Conference (DAC)*, IEEE, 2016, pp. 1–6.

[63] L. Chen, D. Page, and N. P. Smart, "On the design and implementation of an efficient DAA scheme", in *International Conference on Smart Card Research and Advanced Applications*, Springer, 2010, pp. 223–237.

[64] Intel, *Intel Software Guard Extensions: EPID Provisioning and Attestation Services*, https://software.intel.com/en-us/blogs/2016/03/09/intel-sgx-epid-provisioning-and-attestation-services, 2016.

[65] M. Nauman, S. Khan, X. Zhang, and J.-P. Seifert, "Beyond kernel-level integrity measurement: Enabling remote attestation for the Android platform", in *International Conference on Trust and Trustworthy Computing*, Springer, 2010, pp. 1–15.

[66] U. Greveler, B. Justus, and D. Loehr, "Mutual remote attestation: Enabling system cloning for TPM-based platforms", in *International Workshop on Security and Trust Management*, Springer, 2011, pp. 193–206.

[67] T. Abera, N Asokan, L. Davi, J.-E. Ekberg, T. Nyman, A. Paverd, A.-R. Sadeghi, and G. Tsudik, "C-FLAT: Control-flow attestation for embedded systems software", in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ACM, 2016, pp. 743–754.

[68] G. Dessouky, S. Zeitouni, T. Nyman, A. Paverd, L. Davi, P. Koeberl, N Asokan, and A.-R. Sadeghi, "LO-FAT: Low-Overhead Control Flow ATtestation in hardware", in *54th ACM/EDAC/IEEE Design Automation Conference (DAC)*, IEEE, 2017, pp. 1–6.

[69] R. N. Akram, K. Markantonakis, and K. Mayes, "Remote attestation mechanism for user centric smart cards using pseudorandom number generators", in *International Conference on Information and Communications Security*, Springer, 2013, pp. 151–166.

[70] R. N. Akram, K. Markantonakis, K. Mayes, P.-F. Bonnefoi, D. Sauveron, and S. Chaumette, "An efficient, secure and trusted channel protocol for avionics wireless networks", in *IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*, IEEE, 2016, pp. 1–10.

[71] Y. Gasmi, A.-R. Sadeghi, P. Stewin, M. Unger, and N Asokan, "Beyond secure channels", in *ACM Workshop on Scalable Trusted Computing*, ACM, 2007, pp. 30–40.

[72] A.-R. Sadeghi and S. Schulz, "Extending IPsec for efficient remote attestation", in *International Conference on Financial Cryptography and Data Security*, Springer, 2010, pp. 150–165.

[73] N Asokan, F. Brasser, A. Ibrahim, A.-R. Sadeghi, M. Schunter, G. Tsudik, and C. Wachsmann, "Seda: Scalable embedded device attestation", in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, ACM, 2015, pp. 964–975.

[74] Trusted Computing Group (TCG), *TCG Software Stack (TSS) specification*, https://www.trustedcomputinggroup.org/tcg-software-stack-tss-specification/, 2009.

[75] M. Ryan, "Introduction to the TPM 1.2", Trusted Computing Group, http://courses.cs.vt.edu/cs5204/fall10-kafura-BB/Papers/TPM/Intro-TPM.pdf, 2009.

[76] GlobalPlatform, *TEE System Architecture (v1.1)*, 2017.

[77] ——, *TEE Client API (v1.0)*, 2010.

[78] ——, *TEE Internal Core API (v1.1.1)*, 2016.

[79] A. Sallam, *XenDesktop and the Evolution of Hardware-Assisted Server Technologies*, Citrix, https://goo.gl/UgNgUa, 2016.

[80] Intel, *Intel Trusted Execution Technology (White Paper)*, http://www.intel.com/content/www/us/en/architecture-and-technology/trusted-execution-technology/trusted-execution-technology-security-paper.html, 2010.

[81] Intel, *Intel Virtualization Technology (Intel-VT)*, https://www.intel.com/content/www/us/en/virtualization/virtualization-technology/intel-virtualization-technology.html, 2018.

[82] AMD, *AMD64 Secure Virtual Machine Architecture Reference Manual*, https://support.amd.com/TechDocs/24593.pdf, 2005.

[83] J. M. McCune, B. J. Parno, A. Perrig, M. K. Reiter, and H. Isozaki, "Flicker: An execution infrastructure for TCB minimization", in *ACM SIGOPS Operating Systems Review*, ACM, vol. 42, 2008, pp. 315–328.

[84] J. M. McCune, Y. Li, N. Qu, Z. Zhou, A. Datta, V. Gligor, and A. Perrig, "Trustvisor: Efficient TCB reduction and attestation", in *IEEE Symposium on Security and Privacy*, IEEE, 2010, pp. 143–158.

[85] A. Carroll, M. Juarez, J. Polk, and T. Leininger, "Microsoft Palladium: A business overview", *Microsoft*, pp. 1–9, 2002.

[86] M. Reynolds, *Palladium Security's Brave New World*, http://www.bus.umich.edu/kresgepublic/journals/gartner/research/111300/111331/111331.pdf, 2002.

[87] A.-R. Sadeghi and C. Stüble, "Taming 'trusted platforms' by operating system design", in *International Workshop on Information Security Applications*, Springer, 2003, pp. 286–302.

[88] Microsoft, *At WinHEC, Microsoft Discusses Details of Next-Generation Secure Computing Base*, https://news.microsoft.com/2003/05/07/at-winhec-microsoft-discusses-details-of-next-generation-secure-computing-base/, 2003.

[89] A. Vasudevan, E. Owusu, Z. Zhou, J. Newsome, and J. M. McCune, "Trustworthy execution on mobile devices: What security properties can my mobile platform give me?", in *International Conference on Trust and Trustworthy Computing*, Springer, 2012, pp. 159–178.

[90] Genode Operating System Framework, *An exploration of ARM TrustZone technology*, https://genode.org/documentation/articles/trustzone, 2018.

[91] ARM Holdings, *ARM Security Technology: Building a Secure System using TrustZone Technology*, http://infocenter.arm.com/help/topic/com.arm.doc.prd29-genc-009492c/PRD29-GENC-009492C_trustzone_security_whitepaper.pdf, 2009.

[92] S. G. Kelly, *Root-proof smartphones and other myths and legends*, CanSecWest, https://www.slideplayer.com/slide/3754258/, 2012.

[93] ARM Holdings, *TrustZone technology for Armv8-M Architecture Version 2.1*, https://developer.arm.com/products/architecture/cpu-architecture/m-profile/docs/100690/latest/arm-trustzone-technology, 2018.

[94] T. Nyman, J.-E. Ekberg, L. Davi, and N Asokan, "CFI CaRE: Hardware-Supported Call and Return Enforcement for Commercial Microcontrollers", in *International Symposium on Research in Attacks, Intrusions, and Defenses*, Springer, 2017, pp. 259–284.

[95] AMD, *AMD Secure Technology*, https://www.amd.com/en/technologies/security, 2018.

[96] L. Van Doorn, *Secure Hardware and the Creation of an Open Trusted Ecosystem*, https://classic.regonline.com/custImages/360000/369552/TCC%20PPTs/TCC2013_VanDoorn.pdf, 2013.

[97] M. Dorjmyagmar, M. Kim, and H. Kim, "Security analysis of Samsung KNOX", in *19th International Conference on Advanced Communication Technology (ICACT)*, IEEE, 2017, pp. 550–553.

[98] A. Atamli-Reineh, R. Borgaonkar, R. A. Balisane, G. Petracca, and A. Martin, "Analysis of trusted execution environment usage in Samsung KNOX", in *Proceedings of the 1st Workshop on System Software for Trusted Execution*, ser. SysTEX '16, Trento, Italy: ACM, 2016, 7:1–7:6. DOI: 10.1145/3007788.3007795.

[99] Trustonic, *Galaxy S8 – a focus on security*, https://www.trustonic.com/news/blog/galaxy-s8-a-focus-on-security/, 2017.

[100] N Asokan, J.-E. Ekberg, and K. Kostiainen, "The untapped potential of trusted execution environments on mobile devices", in *International Conference on Financial Cryptography and Data Security*, Springer, 2013, pp. 293–294.

[101] K. Kostiainen, "On-board Credentials: An open credential platform for mobile devices", PhD thesis, Aalto University, 2012.

[102] J.-E. Ekberg, "Securing software architectures for trusted processor environments", PhD thesis, Aalto University, 2013.

[103] M. Schwarz, S. Weiser, D. Gruss, C. Maurice, and S. Mangard, "Malware guard extension: Using SGX to conceal cache attacks", in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, Springer, 2017, pp. 3–24.

[104] I. Anati, S. Gueron, S. Johnson, and V. Scarlata, "Innovative technology for CPU based attestation and sealing", in *Proceedings of the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy*, New York, NY, USA, vol. 13, ACM, 2013.

[105] GlobalPlatform, *TEE Sockets API (v1.0)*, 2017.

[106] ——, *TEE TA Debug Specification (v1.0.1)*, 2016.

[107] ——, *TEE Management Framework (TMF), v1.0*, 2016.

[108] Trusted Computing Group (TCG), *Trusted Platform Module (TPM) Main Specifications*, https://trustedcomputinggroup.org/tpm-main-specification/, 2018.

[109] Texas Instruments, *M-Shield Mobile Security – White Paper*, www.ti.com/pdfs/wtbu/ti_mshield_whitepaper.pdf, 2008.

[110] V. Costan and S. Devadas, "Intel SGX explained", *IACR Cryptology ePrint Archive*, vol. 2016, p. 86, 2016, https://eprint.iacr.org/2016/086.pdf.

[111] Smart Card Alliance, *Host Card Emulation (HCE) 101 – a Smart Card Alliance Mobile and NFC Council White Paper*, http://www.iqdevices.com/pdfFiles/HCE-101-WP-FINAL-081114-clean.pdf, 2014.

[112] P. Kocher, D. Genkin, D. Gruss, W. Haas, M. Hamburg, M. Lipp, S. Mangard, T. Prescher, M. Schwarz, and Y. Yarom, "Spectre attacks: Exploiting speculative execution", *ArXiv preprint arXiv:1801.01203*, 2018.

[113] M. Lipp, M. Schwarz, D. Gruss, T. Prescher, W. Haas, A. Fogh, J. Horn, S. Mangard, P. Kocher, D. Genkin, *et al.*, "Meltdown: Reading kernel memory from user space", in *USENIX Security Symposium*, 2018, pp. 973–990.

[114] J. Van Bulck, M. Minkin, O. Weisse, D. Genkin, B. Kasikci, F. Piessens, M. Silberstein, T. F. Wenisch, Y. Yarom, and R. Strackx, "Foreshadow: Extracting the keys to the Intel SGX kingdom with transient out-of-order execution", in *USENIX Security Symposium*, 2018, pp. 991–1008.

[115] Intel, *Resources and response to side channel L1 terminal fault*, https://www.intel.com/content/www/us/en/architecture-and-technology/l1tf.html, 2018.

[116] M. Harbach, E. Von Zezschwitz, A. Fichtner, A. De Luca, and M. Smith, "It'sa hard lock life: A field study of smartphone (un) locking behavior and risk perception", in *Symposium on Usable Privacy and Security*, ACM, 2014, pp. 213–230.

[117] N. Micallef, M. Just, L. Baillie, M. Halvey, and H. G. Kayacik, "Why aren't users using protection? investigating the usability of smartphone locking", in *Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services*, ACM, 2015, pp. 284–294.

[118] E. Hayashi, O. Riva, K. Strauss, A. Brush, and S. Schechter, "Goldilocks and the two mobile devices: Going beyond all-or-nothing access to a device's applications", in *Symposium on Usable Privacy and Security*, ACM, 2012, p. 2.

[119] E. Hayashi, S. Das, S. Amini, J. Hong, and I. Oakley, "CASA: Context-Aware Scalable Authentication", in *Symposium on Usable Privacy and Security*, ACM, 2013, p. 3.

[120] F. Stajano, "Pico: No more passwords!", in *International Workshop on Security Protocols*, Springer, 2011, pp. 49–81.

[121] A. De Luca and J. Lindqvist, "Is secure and usable smartphone authentication asking too much?", *IEEE Computer*, vol. 48, no. 5, pp. 64–68, 2015.

[122] F. Monrose and A. D. Rubin, "Keystroke dynamics as a biometric for authentication", *Future Generation Computer Systems*, vol. 16, no. 4, pp. 351–359, 2000, Elsevier.

[123] M. O. Derawi, C. Nickel, P. Bours, and C. Busch, "Unobtrusive user-authentication on mobile phones using biometric gait recognition", in *6th International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, IEEE, 2010, pp. 306–311.

[124] N. Karapanos, C. Marforio, C. Soriente, and S. Capkun, "Sound-proof: Usable two-factor authentication based on ambient sound.", in *USENIX Security Symposium*, 2015, pp. 483–498.

[125] M. Frank, R. Biedert, E. Ma, I. Martinovic, and D. Song, "Touchalytics: On the applicability of touchscreen input as a behavioral biometric for continuous authentication", *IEEE Transactions on Information Forensics and Security (TIFS)*, vol. 8, no. 1, pp. 136–148, 2013.

[126] O. Riva, C. Qin, K. Strauss, and D. Lymberopoulos, "Progressive authentication: Deciding when to authenticate on mobile phones.", in *USENIX Security Symposium*, USENIX Association, 2012, pp. 301–316.

[127] M. Jakobsson, E. Shi, P. Golle, and R. Chow, "Implicit authentication for mobile devices", in *Proceedings of the 4th USENIX Conference on Hot Topics in Security*, USENIX Association, 2009, pp. 9–9.

[128] L. Li, X. Zhao, and G. Xue, "Unobservable re-authentication for smartphones", in *Network and Distributed System Security Symposium*, 2013, pp. 1–16.

[129] W. Shi, J. Yang, Y. Jiang, F. Yang, and Y. Xiong, "Senguard: Passive user identification on smartphones using multiple sensors", in *7th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, IEEE, 2011, pp. 141–148.

[130] M. Miettinen, S. Heuser, W. Kronz, A.-R. Sadeghi, and N Asokan, "ConXsense: Automated context classification for context-aware access control", in *Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security (ASIA CCS)*, ACM, 2014, pp. 293–304.

[131] H. Saevanee, N. L. Clarke, and S. M. Furnell, "Multi-modal behavioural biometric authentication for mobile devices", in *IFIP International Information Security Conference*, Springer, 2012, pp. 465–474.

[132] H. Ketabdar, M. Roshandel, and D. Skripko, "Towards implicit enhancement of security and user authentication in mobile devices based on movement and audio analysis", *4th International Conference on Advances in Computer-Human Interactions*, 2011.

[133] L. Fridman, A. Stolerman, S. Acharya, P. Brennan, P. Juola, R. Greenstadt, and M. Kam, "Multi-modal decision fusion for continuous authentication", *Computers & Electrical Engineering*, vol. 41, pp. 142–156, 2015, Elsevier.

[134] X. Wang, T. Yu, O. Mengshoel, and P. Tague, "Towards continuous and passive authentication across mobile devices: An empirical study", in *Proceedings of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, ACM, 2017, pp. 35–45.

[135] H. G. Kayacik, M. Just, L. Baillie, D. Aspinall, and N. Micallef, "Data driven authentication: On the effectiveness of user behaviour modelling with mobile device sensors", *ArXiv preprint arXiv:1410.7743*, 2014.

[136] P. Verlinde and G Cholet, "Comparing decision fusion paradigms using k-nn based classifiers, decision trees and logistic regression in a multi-modal identity verification application", in *Proceedings of the International Conference on Audio and Video-Based Biometric Person Authentication (AVBPA)*, Citeseer, 1999, pp. 188–193.

[137] C. Bo, L. Zhang, T. Jung, J. Han, X.-Y. Li, and Y. Wang, "Continuous user identification via touch and movement behavioral biometrics", in *International Performance Computing and Communications Conference (IPCCC)*, IEEE, 2014, pp. 1–8.

[138] D. P. Coutinho, A. L. Fred, and M. A. Figueiredo, "ECG-based continuous authentication system using adaptive string matching", in *Biosignals*, Elsevier, 2011, pp. 354–359.

[139] A. Mosenia, S. Sur-Kolay, A. Raghunathan, and N. K. Jha, "CABA: Continuous authentication based on bio-aura", *IEEE Transactions on Computers*, vol. 66, no. 5, pp. 759–772, 2017.

[140] C. Camara, P. Peris-Lopez, L. Gonzalez-Manzano, and J. Tapiador, "Real-time electrocardiogram streams for continuous authentication", *Applied Soft Computing*, 2017.

[141] Android Open Source Project, *Fingerprint HAL*, https://source.android.com/security/authentication/fingerprint-hal, 2018.

[142] Apple, Inc., *iOS 9.3 Security Guide*, https://www.apple.com/business/docs/iOS_Security_Guide.pdf, 2016.

[143] H. Liu, S. Saroiu, A. Wolman, and H. Raj, "Software abstractions for trusted sensors", in *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*, ACM, 2012, pp. 365–378.

[144] N. A. Safa, R. Safavi-Naini, and S. F. Shahandashti, "Privacy-preserving implicit authentication", in *IFIP International Information Security Conference*, Springer, 2014, pp. 471–484.

[145] J. Domingo-Ferrer, Q. Wu, and A. Blanco-Justicia, "Flexible and robust privacy-preserving implicit authentication", in *IFIP International Information Security Conference*, Springer, 2015, pp. 18–34.

[146] J. Šeděnka, S. Govindarajan, P. Gasti, and K. S. Balagani, "Secure outsourced biometric authentication with performance evaluation on smartphones", *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 2, pp. 384–396, 2015.

[147] B. Shrestha, N. Saxena, H. T. T. Truong, and N. Asokan, "Sensor-based proximity detection in the face of active adversaries", *IEEE Transactions on Mobile Computing*, 2018.

[148] M. Sabt, M. Achemlal, and A. Bouabdallah, "Trusted execution environment: What it is, and what it is not", in *14th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, IEEE, vol. 1, 2015, pp. 57–64.

[149] N. Micallef, H. G. Kayacık, M. Just, L. Baillie, and D. Aspinall, "Sensor use and usefulness: Trade-offs for data-driven authentication on mobile devices", in *Pervasive Computing and Communications (PerCom)*, IEEE, 2015, pp. 189–197.

[150] E. Shi, Y. Niu, M. Jakobsson, and R. Chow, "Implicit authentication through learning user behavior", in *Information Security*, Springer, 2011, pp. 99–113.

[151] A. Ng, *CS229 Lecture Notes: Supervised Learning*, http://cs229.stanford.edu/notes/cs229-notes1.pdf, 2017.

[152] Linaro, *OP-TEE: Open Portable Trusted Execution Environment*, https://www.op-tee.org/, 2017.

[153] Intel, Inc., *C/C++ Trusted Libraries*, https://software.intel.com/en-us/node/709038, 2018.

[154] ——, *Integrated Performance Primitives (IPP)*, https://software.intel.com/en-us/intel-ipp, 2016.

[155] H. Liu, S. Saroiu, A. Wolman, and H. Raj, "Software abstractions for trusted sensors", in *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*, ACM, 2012, pp. 365–378.

[156] P. Gilbert, L. P. Cox, J. Jung, and D. Wetherall, "Toward trustworthy mobile sensing", in *Proceedings of the 11th Workshop on Mobile Computing Systems & Applications*, ACM, 2010, pp. 31–36.

[157] S. Saroiu and A. Wolman, "I am a sensor, and I approve this message", in *Proceedings of the 11th Workshop on Mobile Computing Systems & Applications*, ACM, 2010, pp. 37–42.

[158] R. V. Steiner and E. Lupu, "Attestation in wireless sensor networks: A survey", *ACM Computing Surveys (CSUR)*, vol. 49, no. 3, p. 51, 2016.

[159] T. Abera, N Asokan, L. Davi, F. Koushanfar, A. Paverd, A.-R. Sadeghi, and G. Tsudik, "Things, trouble, trust: On building trust in IoT systems", in *53rd Annual Design Automation Conference*, ACM, 2016.

[160] E. Brickell and J. Li, "Enhanced privacy ID from bilinear pairing for hardware authentication and attestation", *International Journal of Information Privacy, Security and Integrity*, vol. 1, no. 1, pp. 3–33, 2011.

[161] H. Krawczyk, "SIGMA: The 'SIGn-and-MAc' approach to authenticated Diffie-Hellman and its use in the IKE protocols", in *Annual International Cryptology Conference*, Springer, 2003, pp. 400–425.

[162] M. Shaneck, K. Mahadevan, V. Kher, and Y. Kim, "Remote software-based attestation for wireless sensors", in *European Workshop on Security in Ad-hoc and Sensor Networks*, Springer, 2005, pp. 27–41.

[163] A Seshadri, A Perrig, L van Doorn, and P Khosla, "SWATT: Software-based attestation for embedded devices", in *IEEE Symposium on Security and Privacy*, 2004, pp. 272–282.

[164] A. Seshadri, M. Luk, E. Shi, A. Perrig, L. van Doorn, and P. Khosla, "Pioneer: Verifying code integrity and enforcing untampered code execution on legacy systems", in *ACM SIGOPS Operating Systems Review*, ACM, vol. 39, 2005, pp. 1–16.

[165] C. Castelluccia, A. Francillon, D. Perito, and C. Soriente, "On the difficulty of software-based attestation of embedded devices", in *16th ACM Conference on Computer and Communications Security*, ACM, 2009, pp. 400–409.

[166] K. Eldefrawy, G. Tsudik, A. Francillon, and D. Perito, "SMART: Secure and minimal architecture for (establishing dynamic) root of trust.", in *Network and Distributed System Security Symposium*, 2012, pp. 1–15.

[167] P. Koeberl, S. Schulz, A.-R. Sadeghi, and V. Varadharajan, "TrustLite: A security architecture for tiny embedded devices", in *Proceedings of the Ninth European Conference on Computer Systems*, ACM, 2014, p. 10.

[168] F. Brasser, B. El Mahjoub, A.-R. Sadeghi, C. Wachsmann, and P. Koeberl, "TyTAN: Tiny trust anchor for tiny devices", in *Proceedings of the 52nd Annual Design Automation Conference*, ACM, 2015, p. 34.

[169] C. Kil, E. C. Sezer, A. M. Azab, P. Ning, and X. Zhang, "Remote attestation to dynamic system properties: Towards providing complete system integrity evidence", in *IEEE/IFIP International Conference on Dependable Systems & Networks*, IEEE, 2009, pp. 115–124.

[170] L. Davi, A.-R. Sadeghi, and M. Winandy, "Dynamic integrity measurement and attestation: Towards defense against return-oriented programming attacks", in *Proceedings of the ACM Workshop on Scalable Trusted Computing*, ACM, 2009, pp. 49–54.

[171] A.-R. Sadeghi and C. Stüble, "Property-based attestation for computing platforms: Caring about properties, not mechanisms", in *Proceedings of the Workshop on New Security Paradigms*, ser. NSPW '04, Nova Scotia, Canada: ACM, 2004, pp. 67–77, ISBN: 1-59593-076-0.

[172] R. Anderson, "Cryptography and competition policy: Issues with 'trusted computing'", in *Proceedings of the 22nd Annual Symposium on Principles of Distributed Computing*, ACM, 2003, pp. 3–10.

[173] J. Poritz, M. Schunter, E. Van Herreweghen, and M. Waidner, "Property attestation – scalable and privacy-friendly security assessment of peer computers", 2004, https://domino.research.ibm.com/library/cyberdig.nsf/papers/215E33CB2B4F7FA485256E97002A0D6C/$File/rz3548.pdf.

[174] L. Chen, R. Landfermann, H. Löhr, M. Rohe, A.-R. Sadeghi, and C. Stüble, "A protocol for property-based attestation", in *Proceedings of the 1st ACM Workshop on Scalable Trusted Computing*, ACM, 2006, pp. 7–16.

[175] A.-R. Sadeghi, C. Stüble, and M. Winandy, "Property-based TPM virtualization", in *International Conference on Information Security*, Springer, 2008, pp. 1–16.

[176] J. Li, Y. Li, Y. Hu, H. Wang, and W. Liu, "An improved protocol for property-based attestation", in *Proceedings of the 32nd Chinese Control Conference*, IEEE, 2013, pp. 6343–6348.

[177] L. Chen, H. Löhr, M. Manulis, and A.-R. Sadeghi, "Property-based attestation without a trusted third party", in *Information Security*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 31–46.

[178] A. Lee-Thorp, "Attestation in trusted computing: Challenges and potential solutions", Royal Holloway, University of London, Tech. Rep., 2010, https://www.ma.rhul.ac.uk/static/techrep/2010/RHUL-MA-2010-09.pdf.

[179] R. N. Akram, K. Markantonakis, and K. Mayes, "A privacy preserving application acquisition protocol", in *11th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, IEEE, 2012, pp. 383–392.

[180] G. Lowe, "Casper: A compiler for the analysis of security protocols", *Journal of Computer Security*, vol. 6, no. 1-2, pp. 53–84, Jan. 1998. [Online]. Available: http://dl.acm.org/citation.cfm?id=353677.353680.

[181] A. Ibrahim, A.-R. Sadeghi, and S. Zeitouni, "Seed: Secure non-interactive attestation for embedded devices", in *Proceedings of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, ACM, 2017, pp. 64–74.

[182] S. Katzenbeisser, Ü. Kocabaş, V. Rožić, A.-R. Sadeghi, I. Verbauwhede, and C. Wachsmann, "PUFs: Myth, fact or busted? A security evaluation of physically unclonable functions (PUFs) cast in silicon", in *International Workshop on Cryptographic Hardware and Embedded Systems*, Springer, 2012, pp. 283–301.

[183] P. H. Nguyen, D. P. Sahoo, R. S. Chakraborty, and D. Mukhopadhyay, "Efficient attacks on robust ring oscillator puf with enhanced challenge-response set", in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, IEEE, 2015, pp. 641–646.

[184] D. P. Sahoo, P. H. Nguyen, D. Mukhopadhyay, and R. S. Chakraborty, "A case of lightweight PUF constructions: cryptanalysis and machine learning attacks", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 8, pp. 1334–1343, 2015.

[185] G. Hospodar, R. Maes, and I. Verbauwhede, "Machine learning attacks on 65nm Arbiter PUFs: Accurate modeling poses strict bounds on usability", in *IEEE International Workshop on Information Forensics and Security (WIFS)*, IEEE, 2012, pp. 37–42.

[186] U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, and J. Schmidhuber, "Modeling attacks on physical unclonable functions", in *Proceedings of the 17th ACM Conference on Computer and Communications Security (CCS)*, ACM, 2010, pp. 237–249.

[187] X. Xu and W. Burleson, "Hybrid side-channel/machine-learning attacks on pufs: A new threat?", in *Design, Automation and Test in Europe Conference and Exhibition (DATE)*, IEEE, 2014, pp. 1–6.

[188] Samsung Electronics, *An Overview of the Samsung KNOX Platform*, http://www.samsung.com/global/business/business-images/resource/white-paper/2013/06/Samsung_KNOX_whitepaper_June-0.pdf, 2015.

[189] H. Raj, S. Saroiu, A. Wolman, R. Aigner, J. Cox, P. England, C. Fenner, K. Kinshumann, J. Loeser, D. Mattoon, M. Nystrom, D. Robinson, R. Spiger, S. Thom, and D. Wooten, "fTPM: A software-only implementation of a TPM chip", in *Proceedings of the 25th USENIX Security Symposium*, Austin, TX: USENIX Association, Aug. 2016, pp. 841–856.

[190] P. Lafourcade and M. Puys, "Performance evaluations of cryptographic protocols verification tools dealing with algebraic properties", in *International Symposium on Foundations and Practice of Security*, Springer, 2015, pp. 137–155.

[191] S. Scott, "The design and analysis of real-world cryptographic protocols", PhD thesis, Royal Holloway, University of London, 2017.

[192] B. Blanchet, "Security protocol verification: Symbolic and computational models", in *Proceedings of the First international conference on Principles of Security and Trust*, Springer-Verlag, 2012, pp. 3–29.

[193] ——, "Automatic verification of security protocols in the symbolic model: The verifier proverif", in *Foundations of Security Analysis and Design*, Springer, 2014, pp. 54–87.

[194] S. Meier, B. Schmidt, C. Cremers, and D. Basin, "The TAMARIN prover for the symbolic analysis of security protocols", in *International Conference on Computer Aided Verification*, Springer, 2013, pp. 696–701.

[195] A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuéllar, P. H. Drielsma, P.-C. Héam, O. Kouchnarenko, J. Mantovani, *et al.*, "The avispa tool for the automated validation of internet security protocols and applications", in *International conference on computer aided verification*, Springer, 2005, pp. 281–285.

[196] B. Blanchet, B. Smyth, V. Cheval, and M. Sylvestre, "Proverif 2.00: Automatic cryptographic protocol verifier, user manual and tutorial", 2018, http://prosecco.gforge.inria.fr/personal/bblanche/proverif/.

[197] D. Dolev and A. Yao, "On the security of public key protocols", *IEEE Transactions on Information Theory*, vol. 29, no. 2, pp. 198–208, 1983.

[198] C. J. Cremers, P. Lafourcade, and P. Nadeau, "Comparing state spaces in automatic security protocol analysis", in *Formal to Practical Security*, Springer, 2009, pp. 70–94.

[199] C. J. Cremers, "Unbounded verification, falsification, and characterization of security protocols by pattern refinement", in *Proceedings of the 15th ACM Conference on Computer and Communications Security*, ser. CCS, ACM, 2008, pp. 119–128.

[200] G. Lowe, "A hierarchy of authentication specifications", in *Computer Security Foundations Workshop*, IEEE, 1997, pp. 31–43.

[201] C. J. Cremers, S. Mauw, and E. P. De Vink, "Injective synchronisation: An extension of the authentication hierarchy", *Theoretical Computer Science*, vol. 367, no. 1-2, pp. 139–161, 2006.

[202] M. Abadi and B. Blanchet, "Analyzing security protocols with secrecy types and logic programs", *ACM SIGPLAN Notices*, vol. 37, no. 1, pp. 33–44, 2002.

[203] C. Cremers, "Key exchange in IPsec revisited: Formal analysis of IKEv1 and IKEv2", in *European Symposium on Research in Computer Security (ESORICS)*, Springer, 2011, pp. 315–334.

[204] D. Basin, C. Cremers, and S. Meier, "Provably repairing the ISO/IEC 9798 standard for entity authentication", *Journal of Computer Security*, vol. 21, no. 6, pp. 817–846, 2013.

[205] C. Cremers and M. Horvat, "Improving the ISO/IEC 11770 standard for key management techniques", in *International Conference on Research in Security Standardisation*, Springer, 2014, pp. 215–235.

[206] ARM, *ARM Cortex Programmer's Guide for ARMv8-A. Version 1.0*, http://infocenter.arm.com/help/topic/com.arm.doc.den0024a/DEN0024A_v8_architecture_PG.pdf, 2015.

[207] NIST, *Recommendations for Key Management*, Special Publication 800-57 Part 1 Rev. 4, 2016.

[208] Linary Group, *OP-TEE crypto implementation (Documentation)*, https://github.com/OP-TEE/optee_os/blob/master/documentation/crypto.md, 2018.

[209] T. Bingmann, *Speedtest and comparsion of open-source cryptography libraries and compiler flags*, https://panthema.net/2008/0714-cryptography-speedtest-comparison/, 2008.

[210] P. A. Peterson, "Cryptkeeper: Improving security with encrypted RAM", in *IEEE International Conference on Technologies for Homeland Security (HST)*, IEEE, 2010, pp. 120–126.

[211] S. Müller, "Security trade-offs in cloud-storage systems", https://www.depositonce.tu-berlin.de/bitstream/11303/6786/4/mueller_steffen.pdf, PhD thesis, Elektrotechnik und Informatik der Technischen Universität Berlin (TU-Berlin), 2017.

[212] T. Dierks and E. Rescorla, *RFC 5246 – Transport Layer Security (TLS) Protocol Version 1.2*, https://tools.ietf.org/html/rfc5246, 2008.

[213] T. Ylonen and C. Lonvick, *RFC 4253 – The Secure Shell (SSH) Transport Layer Protocol*, https://tools.ietf.org/html/rfc4253, 2006.

[214] E. Rescorla and N. Modadugu, *RFC 6347 – Datagram Transport Layer Security (DTLS) Version 1.2*, https://tools.ietf.org/html/rfc6347, 2012.

[215] K. Markantonakis and K. Mayes, "A secure channel protocol for multi-application smart cards based on public key cryptography", in *Communications and Multimedia Security*, Springer, 2005, pp. 79–95.

[216] GlobalPlatform, *Card Remote Application Management over HTTP*, 2015.

[217] W. G. Sirett, J. A. MacDonald, K. Mayes, and K. Markantonakis, "Design, installation and execution of a security agent for mobile stations", in *International Conference on Smart Card Research and Advanced Applications*, Springer, 2006, pp. 1–15.

[218] W. Diffie, P. C. Van Oorschot, and M. J. Wiener, "Authentication and authenticated key exchanges", *Designs, Codes and Cryptography*, vol. 2, no. 2, pp. 107–125, 1992.

[219] A. Aziz and W. Diffie, "Privacy and authentication for wireless local area networks", *IEEE Personal Communications*, vol. 1, no. 1, pp. 25–31, 1994, IEEE.

[220] G. Horn and B. Preneel, "Authentication and payment in future mobile systems", *Journal of Computer Security*, vol. 8, no. 2,3, pp. 183–207, Aug. 2000. [Online]. Available: http://dl.acm.org/citation.cfm?id=1297828.1297832.

[221] W. Aiello, S. M. Bellovin, M. Blaze, R. Canetti, J. Ioannidis, A. D. Keromytis, and O. Reingold, "Just fast keying: Key agreement in a hostile internet", *ACM Transactions on Information and System Security (TISSEC)*, vol. 7, no. 2, pp. 242–273, 2004.

[222] K. Kent and M. Souppaya, "Guide to Computer Security Log Management", *NIST Special Publication*, vol. 92, 2006.

[223] International Standards Organisation, *ISO 27001:2013 – Information Security Management*, https://www.iso.org/standard/54534.html, 2013.

[224] B. Böck, D. Huemer, and A. M. Tjoa, "Towards more trustable log files for digital forensics by means of trusted computing", in *24th International Conference on Advanced Information Networking and Applications*, IEEE, 2010, pp. 1020–1027. DOI: 10.1109/AINA.2010.26.

[225] A. Sinha, L. Jia, P. England, and J. R. Lorch, "Continuous tamper-proof logging using TPM 2.0", in *7th International Conference on Trust and Trustworthy Computing*, NY, USA: Springer-Verlag, 2014, pp. 19–36. DOI: 10.1007/978-3-319-08593-7_2.

[226] S. Patel, H. Park, P. Bonato, L. Chan, and M. Rodgers, "A review of wearable sensors and systems with applications in rehabilitation", *Journal of Neuro-engineering and Rehabilitation*, vol. 9, no. 1, p. 21, 2012. [Online]. Available: http://dx.doi.org/10.1186/1743-0003-9-21.

[227] P. Rashidi and A. Mihailidis, "A survey on ambient-assisted living tools for older adults", *IEEE Journal of Biomedical and Health Informatics*, vol. 17, no. 3, pp. 579–590, 2013, ISSN: 2168-2194. DOI: 10.1109/JBHI.2012.2234129.

[228] D. Chen and M. Wang, "A home security ZigBee network for remote monitoring applications", in *International Conference on Wireless, Mobile and Multimedia Networks*, IET, 2006, pp. 1–4.

[229] R. Perez, R. Sailer, L. van Doorn, *et al.*, "vTPM: Virtualizing the trusted platform module", in *Proceedings of the 15th USENIX Security Symposium*, 2006, pp. 305–320.

[230] J. M. McCune, B. Parno, A. Perrig, M. K. Reiter, and A. Seshadri, "Minimal tcb code execution", in *IEEE Symposium on Security and Privacy*, IEEE, 2007, pp. 267–272.

[231] L. Singaravelu, C. Pu, H. Härtig, and C. Helmuth, "Reducing TCB complexity for security-sensitive applications: Three case studies", in *ACM SIGOPS Operating Systems Review*, ACM, vol. 40, 2006, pp. 161–174.

[232] B. Schneier and J. Kelsey, "Secure audit logs to support computer forensics", *ACM Transactions on Information and System Security (TISSEC)*, vol. 2, no. 2, pp. 159–176, 1999.

[233] M. Bellare and B. Yee, "Forward integrity for secure audit logs", Computer Science and Engineering Department, University of California at San Diego, Tech. Rep., 1997.

[234] J. E. Holt, "Logcrypt: Forward security and public verification for secure audit logs", in *Proceedings of the Australasian Workshops on Grid Computing and E-research*, Australian Computer Society, Inc., 2006, pp. 203–211.

[235] D. Ma and G. Tsudik, "A new approach to secure logging", *ACM Transactions on Storage*, vol. 5, no. 1, p. 2, 2009.

[236] A. A. Yavuz, P. Ning, and M. K. Reiter, "Efficient, compromise-resilient and append-only cryptographic schemes for secure audit logging", in *International Conference on Financial Cryptography and Data Security*, Springer, 2012, pp. 148–163.

[237] G. Hartung, "Attacks on secure logging schemes", *IACR Cryptology ePrint Archive*, vol. 2017, p. 95, 2017, https://eprint.iacr.org/2017/095.pdf.

[238] C. N. Chong, Z. Peng, and P. H. Hartel, "Secure audit logging with tamper-resistant hardware", in *IFIP TC11 18th International Conference on Information Security: Security and Privacy in the Age of Uncertainty*, Springer, 2003, pp. 73–84.

[239] S. Curry, "An introduction to the Java Ring", *Java World*, 1998, https://www.javaworld.com/javaworld/jw-04-1998/jw-04-javadev.html.

[240] H. Nguyen, B. Acharya, R. Ivanov, A. Haeberlen, L. T. X. Phan, O. Sokolsky, J. Walker, J. Weimer, W. Hanson, and I. Lee, "Cloud-based secure logger for medical devices", in *1st International Conference on Connected Health: Applications, Systems and Engineering Technologies*, IEEE, 2016, pp. 89–94. DOI: 10.1109/CHASE.2016.48.

[241] V. Karande, E. Bauman, Z. Lin, and L. Khan, "SGX-Log: Securing system logs with SGX", in *Proceedings of the Asia Conference on Computer and Communications Security (ASIA CCS)*, ACM, 2017, pp. 19–30, ISBN: 978-1-4503-4944-4. DOI: 10.1145/3052973.3053034.

[242] W. Xu, L. Huang, A. Fox, D. Patterson, and M. I. Jordan, "Detecting large-scale system problems by mining console logs", in *Proceedings of the ACM 22nd Symposium on Operating Systems Principles*, ACM, 2009, pp. 117–132.

[243] Q. Fu, J.-G. Lou, Y. Wang, and J. Li, "Execution anomaly detection in distributed systems through unstructured log analysis", in *9th IEEE International Conference on Data Mining*, IEEE, 2009, pp. 149–158.

[244] H. Krawczyk, "Cryptographic extraction and key derivation: The HKDF scheme", in *Advances in Cryptology – 30th Annual Cryptology Conference (CRYPTO)*, Springer, 2010, pp. 631–648. DOI: 10.1007/978-3-642-14623-7_34.

[245] H. Krawczyk and P. Eronen, *RFC 5869 – HMAC-based Extract-and-expand Key Derivation Function (HKDF)*, https://tools.ietf.org/html/rfc5869, 2010.

[246] Linaro, *Secure storage in OP-TEE*, https://github.com/OP-TEE/optee_os/blob/master/documentation/secure_storage.md, 2018.

[247] F. Bao and I.-R. Chen, "Dynamic trust management for Internet of Things applications", in *International Workshop on Self-aware Internet of Things*, ACM, 2012, pp. 1–6.

[248] H. Kagermann, J. Helbig, A. Hellinger, and W. Wahlster, *Recommendations for implementing the strategic intiative 'Industrie 4.0': Securing the future of german manufacturing industry*. Forschungsunion, 2013.

[249] Capgemini, *Smart factories: How can manufacturers realize the potential of digital industrial revolution*, https://goo.gl/3zR1Ed, 2017.

[250] International Data Corporation (IDC), *Worldwide smart cities spending guide*, https://www.idc.com/tracker/showproductinfo.jsp?prod_id=1843, 2017.

[251] C. A. Cardno, "Chicago tracks city streets 'fitness'", *Civil Engineering Magazine Archive*, vol. 86, no. 11, pp. 36–37, 2016.

[252] Smart Cities Council of India, *Predicting floods with the 'early flood warning system'*, https://goo.gl/jRUwnE, 2017.

[253] A. Solanas *et al.*, "Smart health: A context-aware health paradigm within smart cities", *IEEE Communications Magazine*, vol. 52, no. 8, pp. 74–81, 2014.

[254] IBM, *Parking in the smart city*, https://www.ibm.com/blogs/internet-of-things/iot-smart-parking/, 2017.

[255] Z. Xiong, H. Sheng, W. Rong, and D. E. Cooper, "Intelligent transportation systems for smart cities: A progress review", *Science China Information Sciences*, vol. 55, no. 12, pp. 2908–2914, 2012.

[256] A.-R. Sadeghi, C. Wachsmann, and M. Waidner, "Security and privacy challenges in industrial internet of things", in *52nd Annual Design Automation Conference*, ACM, 2015, p. 54.

[257] R. Akram, K. Markantonakis, K. Mayes, P.-F. Bonnefoi, D. Sauveron, and S. Chaumette, "An efficient, secure and trusted channel protocol for avionics wireless networks", in *35th Digital Avionics Systems Conference*, IEEE, Sep. 2016.

[258] U. Greveler, B. Justus, and D. Loehr, "Mutual remote attestation: Enabling system cloning for TPM-based platforms", in *International Workshop on Security and Trust Management*, Springer, 2011, pp. 193–206.

[259] G. Arfaoui, S. Gharout, J.-F. Lalande, and J. Traoré, "Practical and privacy-preserving TEE migration", in *IFIP International Conference on Information Security Theory and Practice*, ser. WISTP, Springer, 2015, pp. 153–168.

[260] K. Kostiainen, N Asokan, and A. Afanasyeva, "Towards user-friendly credential transfer on open credential platforms", in *Applied Cryptography and Network Security*, Springer, 2011, pp. 395–412.

[261] L. Chen and J. Li, "Revocation of Direct Anonymous Attestation", in *International Conference on Trusted Systems*, Springer, 2011, pp. 128–147.

[262] W. Lueks, G. Alpár, J.-H. Hoepman, and P. Vullers, "Fast revocation of attribute-based credentials for both users and verifiers", *Computers & Security*, vol. 67, pp. 308–323, 2017.

[263] S. Katzenbeisser, K. Kursawe, and F. Stumpf, "Revocation of TPM keys", in *International Conference on Trusted Computing*, Springer, 2009, pp. 120–132.

[264] K. Kostiainen, N Asokan, and J.-E. Ekberg, "Credential disabling from trusted execution environments", in *15th Nordic Conference on Secure IT Systems*, Springer, 2010, pp. 171–186.

[265] R. N. Akram, K. Markantonakis, and K. Mayes, "Recovering from a lost digital wallet", in *10th IEEE International Conference on Embedded and Ubiquitous Computing*, IEEE, 2013, pp. 1615–1621.

[266] S. Gueron, "AES-GCM for efficient authenticated encryption – Ending the reign of HMAC-SHA-1", *Real-World Cryptography*, 2013.

[267] ——, "AES-GCM software performance on the current high end CPUs as a performance baseline for CAESAR competition", *Directions in Authenticated Ciphers (DIAC)*, 2013.

[268] D. A. McGrew and J. Viega, "The security and performance of the Galois/Counter Mode (GCM) of operation", in *International Conference on Cryptology in India*, Springer, 2004, pp. 343–355.

[269] J. Greene, "Intel Trusted eXecution Technology (TXT)", *Intel White Papers*, 2012, https://intel.com/content/dam/www/public/us/en/documents/white-papers/trusted-execution-technology-security-paper.pdf.

[270] V. Coskun, B. Ozdenizci, and K. Ok, "A survey on near field communication (NFC) technology", *Wireless Personal Communications*, vol. 71, no. 3, pp. 2259–2294, 2013, Springer. DOI: 10.1007/s11277-012-0935-5.

[271] Deloitte, *Contactless mobile payments (finally) gain momentum*, https://goo.gl/Tfmx1a, 2015.

[272] Statista, *Proximity mobile payment transaction value in the United States from 2015 to 2021*, http://www.statista.com/statistics/244475/proximity-mobile-payment-transaction-value-in-the-united-states/, 2018.

[273] ——, *NFC mobile payment users worldwide from 2012 to 2018*, https://www.statista.com/statistics/461512/nfc-mobile-payment-users-worldwide/, 2018.

[274] VeriFone, *A cashless future on the horizon*, http://www.verifone.co.uk/media/1420610/VeriFone_Cashless_Future_Contactless.pdf, 2010.

[275] L. Francis, G. P. Hancke, K. Mayes, and K. Markantonakis, "Practical Relay Attack on Contactless Transactions by Using NFC Mobile Phones.", *IACR Cryptology ePrint Archive*, vol. 2011, p. 618, 2011.

[276] R. Verdult and F. Kooman, "Practical attacks on NFC enabled cell phones", in *3rd International Workshop on Near Field Communication (NFC)*, IEEE, 2011, pp. 77–82. DOI: 10.1109/NFC.2011.16.

[277] Samsung, *Device-side security: Samsung Pay, TrustZone, and the TEE*, http://www.developer.samsung.com/tech-insights/pay/device-side-security, 2018.

[278] Z. Ahmad, L. Francis, T. Ahmed, C. Lobodzinski, D. Audsin, and P. Jiang, "Enhancing the security of mobile applications by using TEE and (U)SIM", in *10th International Conference on Ubiquitous Intelligence and Computing*, IEEE, 2013, pp. 575–582.

[279] Android Open Source Project, *Trusty TEE*, https://source.android.com/security/trusty/, 2017.

[280] "EMV Payment Tokenisation Specification Standard", EMVCo., LLC, Specification Version 2.0, 2016.

[281] J.-E. Ekberg and S. Tamrakar, "Mass transit ticketing with NFC mobile phones", in *International Conference on Trusted Systems*, Springer, 2011, pp. 48–65.

[282] A. Umar, I. Gurulian, K. Mayes, and K. Markantonakis, "Tokenisation blacklisting using linkable group signatures", in *Security and Privacy in Communication Networks*, Springer International Publishing, 2017, pp. 182–198, ISBN: 978-3-319-59608-2.

[283] International Organisation for Standardisation, "ISO/IEC 18092: Information technology – Telecommunications and information exchange between systems – Near Field Communication – Interface and Protocol (NFCIP-1)", vol. 18092, 2004.

[284] ——, "ISO/IEC 21481: Information technology – Telecommunications and information exchange between systems – Near Field Communication – Interface and Protocol (NFCIP-2)", *ISO/IEC*, vol. 21481, 2004.

[285] "EMV Contactless Specifications for Payment Systems: Book D - EMV Contactless Communication Protocol Specification", EMVCo, LLC, Specification Version 2.6, 2016.

[286] C. Cremers, K. Rasmussen, B. Schmidt, and S. Capkun, "Distance hijacking attacks on distance bounding protocols", in *IEEE Symposium on Security and Privacy*, IEEE, 2012, pp. 113–127. DOI: 10.1109/SP.2012.17.

[287] K. B. Rasmussen and S. Capkun, "Realization of RF distance bounding.", in *USENIX Security Symposium*, USENIX Association, 2010, pp. 389–402.

[288] G. Hancke, K. Mayes, and K. Markantonakis, "Confidence in smart token proximity: Relay attacks revisited", *Computers & Security*, vol. 28, no. 7, pp. 615 –627, 2009, Elsevier. DOI: 10.1016/j.cose.2009.06.001.

[289] G. P. Hancke and M. G. Kuhn, "Attacks on time-of-flight distance bounding channels", in *Proceedings of the 1st ACM Conference on Wireless Network Security*, ser. WiSec '08, Alexandria, VA, USA: ACM, 2008, pp. 194–202, ISBN: 978-1-59593-814-5. DOI: 10.1145/1352533.1352566.

[290] I. Boureanu, A. Mitrokotsa, and S. Vaudenay, "Towards secure distance bounding", in *Fast Software Encryption*, Springer, 2014, pp. 55–67.

[291] T. Halevi, D. Ma, N. Saxena, and T. Xiang, "Secure proximity detection for NFC devices based on ambient sensor data", English, in *European Symposium on Computer Security*, ser. LNCS, S. Foresti, M. Yung, and F. Martinelli, Eds., Springer, 2012, ISBN: 978-3-642-33166-4.

[292] D. Ma, N. Saxena, T. Xiang, and Y. Zhu, "Location-aware and safer cards: Enhancing RFID security and privacy via location sensing", *IEEE Transactions on Dependable and Secure Computing*, vol. 10, no. 2, pp. 57–69, 2013, ISSN: 1545-5971.

[293] B. Shrestha, N. Saxena, H. T. T. Truong, and N Asokan, "Drone to the rescue: Relay-resilient authentication using ambient multi-sensing", in *International Conference on Financial Cryptography and Data Security*, Springer, 2014, pp. 349–364.

[294] H. T. T. Truong, X. Gao, B. Shrestha, N. Saxena, N Asokan, and P. Nurmi, "Comparing and fusing different sensor modalities for relay attack resistance in zero-interaction authentication", in *IEEE International Conference on Pervasive Computing and Communications*, IEEE, 2014, pp. 163–171.

[295] P. Urien and S. Piramuthu, "Elliptic curve-based RFID/NFC authentication with temperature sensor input for relay attacks", *Decision Support Systems*, 2014, Elsevier. DOI: 10.1016/j.dss.2013.10.003.

[296] A. Varshavsky, A. Scannell, A. LaMarca, and E. de Lara, "Amigo: Proximity-based authentication of mobile devices", in *Ubiquitous Computing*, ser. LNCS, Springer, 2007, pp. 253–270.

[297] M. Mehrnezhad, F. Hao, and S. F. Shahandashti, "Tap-Tap and Pay (TTP): Preventing man-in-the-middle attacks in NFC payments using mobile sensors", in *2nd International Conference on Research in Security Standardisation*, Springer, 2014.

[298] R. Jin, L. Shi, K. Zeng, A. Pande, and P. Mohapatra, "MagPairing: Pairing smartphones in close proximity using magnetometers", *IEEE Transactions on Information Forensics and Security (TIFS)*, vol. 11, no. 6, pp. 1306–1320, 2016, ISSN: 1556-6013. DOI: 10.1109/TIFS.2015.2505626.

[299] "MasterCard Contactless Performance Requirements", MasterCard, Specification, 2014, https://goo.gl/DLwgVP.

[300] VISA, "Transactions Acceptance Device Guide (TADG)", Specification Version 3.0, 2015, https://technologypartner.visa.com/download.aspx?id=32.

[301] Smart Card Alliance Transportation Council, "Transit and contactless open payments: An emerging approach for fare collection", White Paper, 2011, https://www.securetechalliance.org/resources/pdf/Open_Payments_WP_110811.pdf.

[302] VISA, "The future of ticketing: Paying for public transport journeys using visa cards in the 21st century", White Paper, 2013.

[303] W. McKinney, "Data Structures for Statistical Computing in Python", in *Proceedings of the 9th Python in Science Conference*, S. van der Walt and J. Millman, Eds., 2010, pp. 51 –56.

[304] S. van der Walt, S. C. Colbert, and G. Varoquaux, "The NumPy array: A structure for efficient numerical computation", *Computing in Science Engineering*, vol. 13, no. 2, pp. 22–30, 2011, IEEE, ISSN: 1521-9615. DOI: 10.1109/MCSE.2011.37.

[305] E. Jones, T. Oliphant, P. Peterson, *et al.*, *SciPy: Open source scientific tools for Python*, http://www.scipy.org/, 2001.

[306] Android Open Source Project, *Position Sensors – Android Developers*, https://developer.android.com/guide/topics/sensors/sensors_position.html#sensors-pos-prox, 2018.

[307] ——, *Android API Reference Documentation: Sensors Overview*, https://goo.gl/5fCiuG, 2018.

[308] Webcusp, *A Few Android Phones That Have Temperature Sensors*, http://webcusp.com/a-few-android-phones-that-have-temperature-sensor/, 2018.

[309] D. Hintze, R. D. Findling, M. Muaaz, E. Koch, and R. Mayrhofer, "Cormorant: Towards continuous risk-aware multi-modal cross-device authentication", in *Adjunct Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the ACM International Symposium on Wearable Computers*, ACM, 2015, pp. 169–172.

[310] D. J. Bernstein, "Introduction to post-quantum cryptography", in *Post-quantum cryptography*, Springer, 2009, pp. 1–14.

[311] J. Hoffstein, J. Pipher, and J. H. Silverman, "Ntru: A ring-based public key cryptosystem", in *International Algorithmic Number Theory Symposium*, Springer, 1998, pp. 267–288.

[312] D. J. Bernstein, T. Lange, and C. Peters, "Attacking and defending the McEliece cryptosystem", in *International Workshop on Post-Quantum Cryptography*, Springer, 2008, pp. 31–46.

[313] C. Peikert, "Public-key cryptosystems from the worst-case shortest vector problem", in *Proceedings of the 41st Annual ACM Symposium on Theory of Computing*, ACM, 2009, pp. 333–342.

[314] L. Chen, L. Chen, S. Jordan, Y.-K. Liu, D. Moody, R. Peralta, R. Perlner, and D. Smith-Tone, *Report on post-quantum cryptography*. US Department of Commerce, National Institute of Standards and Technology (NIST), 2016.

[315] M. Ando, J. D. Guttman, A. R. Papaleo, and J. Scire, "Hash-based TPM signatures for the quantum world", in *International Conference on Applied Cryptography and Network Security*, Springer, 2016, pp. 77–94.

[316] X. Liu, R. Misoczki, and M. R. Sastry, "Remote attestation for low-end prover devices with post-quantum capabilities", in *Proceedings of the 8th ACM Conference on Data and Application Security and Privacy*, ACM, 2018, pp. 84–94.

[317] R. Stallman, *Can You Trust Your Computer?*, https://gnu.org/philosophy/can-you-trust.en.html, 2015.

[318] Y. Deswarte, L. Blain, and J.-C. Fabre, "Intrusion tolerance in distributed computing systems", in *IEEE Computer Society Symposium on Research in Security and Privacy*, IEEE, 1991, pp. 110–121.

Appendix A

Protocol Supplements for Chapter 4

A.1 BTP Protocol Scyther Source

```
hashfunction h;
// Attestation Request
const AttReq: Function;
// Attestation response (quote)
usertype Quote;
usertype SessionKey;
// Diffie-Hellman components
usertype DH;

protocol btp(TASD,TARE)
{
  macro Scookie = h(gSD,nsd,TASD,TARE);
  role TASD
  {
    fresh nsd: Nonce;
    var nre: Nonce;
    fresh qsd: Quote;
    var qre: Quote;
    var K: SessionKey;
    fresh gSD: DH;
    var gRE: DH;

    send_1(TASD, TARE, nsd, gSD, AttReq, Scookie);

    recv_2(TARE, TASD, nre, gRE,
       {{TARE,TASD,nre,nsd,gSD,gRE}sk(TARE),
       {qre, nre, nsd}sk(TARE)}K,
       AttReq);

    send_3(TASD,TARE,
       {{h(TASD,TARE,gRE,gSD,nre,nsd)}sk(TASD),
       {qsd,nre,nsd}sk(TASD)}K,
       Scookie);

    claim_a1(TASD, Alive);
    claim_a2(TASD, Niagree);
    claim_a3(TASD, Nisynch);
    claim_a4(TASD, Secret, qsd);
    claim_a5(TASD, Reachable);
    claim_a6(TASD, SKR, K);
  }

  role TARE
```

```
  {
    fresh nre: Nonce;
    var nsd: Nonce;
    fresh qre: Quote;
    var qsd: Quote;
    fresh K: SessionKey;
    fresh gRE: DH;
    var gSD: DH;

    recv_1(TASD, TARE, nsd, gSD, AttReq, Scookie);

    send_2(TARE, TASD, nre, gRE,
       {{TARE,TASD,nre,nsd,gSD,gRE}sk(TARE),
       {qre, nre, nsd}sk(TARE)}K,
       AttReq);

    recv_3(TASD,TARE,
       {{h(TASD,TARE,gRE, gSD, nre,nsd)}sk(TASD),
       {qsd,nre,nsd}sk(TASD)}K,
       Scookie);

    claim_b1(TARE, Alive);
    claim_b2(TARE, Niagree);
    claim_b3(TARE, Nisynch);
    claim_b4(TARE, Secret, qre);
    claim_b5(TARE, Reachable);
    claim_b6(TARE, SKR, K);
  }
}
```

A.2 UTP Protocol Scyther Source

```
protocol utp(UASD,TARE)
{
  hashfunction h;
  // Attestation Request
  const AttReq: Function;
  // Attestation response (quote)
  usertype Quote;
  usertype SessionKey;
  // Diffie-Hellman components
  usertype DH;
  macro Scookie = h(gSD,nsd,UASD,TARE);
  role UASD
  {
    fresh nsd: Nonce;
    var nre: Nonce;
    var qre: Quote;
    var K: SessionKey;
    fresh gSD: DH;
    var gRE: DH;

    send_1(UASD,TARE, nsd, gSD, AttReq, Scookie);

    recv_2(TARE, UASD, nre, gRE,
       {{TARE,UASD,nre,nsd,gSD,gRE}sk(TARE),
       {qre,nre,nsd}sk(TARE)}K);
```

```
26
27    send_3(UASD,TARE,
28       {{h(UASD,TARE,gRE,gSD,nre,nsd)}sk(UASD)}K,
29       Scookie);
30
31    claim_a1(UASD, Alive);
32    claim_a2(UASD, Niagree);
33    claim_a3(UASD, Nisynch);
34    claim_a5(UASD, Reachable);
35    claim_a6(UASD, SKR, K);
36   }
37
38   role TARE
39   {
40    fresh nre: Nonce;
41    var nsd: Nonce;
42    fresh qre: Quote;
43    fresh K: SessionKey;
44    fresh gRE: DH;
45    var gSD: DH;
46
47    recv_1(UASD, TARE, nsd, gSD, AttReq, Scookie);
48
49    send_2(TARE, UASD, nre, gRE,
50       {{TARE,UASD,nre,nsd,gSD,gRE}sk(TARE),
51       {qre,nre,nsd}sk(TARE)}K);
52
53    recv_3(UASD,TARE,
54       {{h(UASD,TARE,gRE, gSD,nre,nsd)}sk(UASD)}K,
55       Scookie);
56
57    claim_b1(TARE, Alive);
58    claim_b2(TARE, Niagree);
59    claim_b3(TARE, Nisynch);
60    claim_b4(TARE, Secret, qre);
61    claim_b5(TARE, Reachable);
62    claim_b6(TARE, SKR, K);
63   }
64  }
```

Appendix B

Sensor Descriptions

This appendix provides a short description of the functionality of widely-deployed mobile and embedded sensors. This selection is drawn from those available through the Android Software Development Kit (SDK) [307].

- **Accelerometer**. The accelerometer sensor – deployed in most modern smartphones – measures the acceleration applied to the device on the x, y and z axes; its units are metres per second per second (ms^{-2}).
- **Ambient Temperature**. The ambient temperature sensor returns the room temperature in degrees Celsius (°C).
- **Bluetooth**. Bluetooth is a technology that facilitates wireless communication and operates in the ISM band centred at 2.4 gigahertz. As a proximity sensor, Bluetooth can be used to collect and measure the identity of Bluetooth devices in the vicinity, including their advertised device names and MAC addresses.
- **Geomagnetic Rotation Vector (GRV)**. The GRV sensor measures the rotation of the device using the device's magnetometer and accelerometer; it returns a vector containing the angles that the device is rotated in the x, y and z axes.
- **Global Positioning System (GPS)**. GPS is based on satellite-based global positioning and velocity measurement. A latitude and longitude pair is returned, representing a geographical location on Earth.
- **Gravity**. The gravity sensor on mobile handsets measures the effect of Earth's gravity on the device. It is measured in metres per second per second (ms^{-2}).
- **Gyroscope**. The gyroscope measures the rate of rotation of the device about the x, y and z axes; its units are radians per second ($rads^{-1}$).
- **Relative Humidity**. The relative humidity sensor returns the percentage of the relative ambient humidity in the air.

- **Light**. The light sensor measures the surrounding lighting conditions. Android measures this quantity in lux.
- **Linear Acceleration**. The linear acceleration sensor measures the effect of a device's movement on itself; its units are metres per second per second (ms^{-2}).
- **Magnetic Field**. The magnetic field sensor detects the Earth's magnetic field along three perpendicular axes x, y and z. Android measures these values in microteslas (μT).
- **Network Location**. A latitude and longitude pair is returned, representing a geographical location on Earth with less fidelity compared to GPS using cell tower and Wi-Fi signals. It is advised to use this method in situations requiring less battery power, fast repsonses and in indoors environments.
- **Pressure**. The pressure sensor measures the atmospheric pressure surrounding the mobile handset. It is measured in hectopascals (hPa).
- **Proximity**. The proximity sensor detects and measures an object placed within its detection range. It measures the distance to that object in centimetres. On many devices, this sensor returns only a boolean value, declaring whether something is in close proximity to the device or not.
- **Rotation Vector**. Rotation vector is a software sensor, similar to the GRV, but also incorporates the gyroscope. The returned values represent the angles which the device has rotated through the x, y and z axes.
- **Sound**. A waveform captured over some time period t and at some sampling rate using the device's on-board microphone.
- **WiFi**. Traditional IEEE 802.11 WiFi can be used as a sensor to detect the identifies of the networks in the vicinity of the mobile device, including their MAC addresses and ESSIDs.