

Carlton Shepherd
Konstantinos Markantonakis

Trusted Execution Environments

 Springer

Trusted Execution Environments

Carlton Shepherd • Konstantinos Markantonakis

Trusted Execution Environments

 Springer

Carlton Shepherd
School of Computing
Newcastle University
Newcastle upon Tyne, Tyne and Wear, UK

Konstantinos Markantonakis
Information Security Group
Royal Holloway, University of London
Egham, Surrey, UK

ISBN 978-3-031-55560-2 ISBN 978-3-031-55561-9 (eBook)
<https://doi.org/10.1007/978-3-031-55561-9>

© The Editor(s) (if applicable) and The Author(s), under exclusive license to Springer Nature Switzerland AG 2024

This work is subject to copyright. All rights are solely and exclusively licensed by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors, and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Paper in this product is recyclable.

For James, Henry, and Ellen.
— Carlton

*I would like to dedicate this book to Maria,
Eleni, and Georgios, and my grandparents
Areti and Kostas, Sofia and Eftyhios.*
— Kostas

Foreword

The need for Trusted Execution Environments (TEE) long predates the invention of computers, as there have always been sensitive activities that need carrying out reliably, protected from imposters, eavesdroppers, thieves, and disruptors. These activities may involve precious things and so protected storage and controlled access falls within the TEE concept.

A castle is a very old example of a TEE. Its strong walls would protect against physical attacks and meetings within an inner keep would resist eavesdropping, with trained guards alert to attackers trying to trick their way inside. Important documents of the time would carry an official wax seal for tamper detection and integrity protection and could be securely stored in a strong box with other valuables. Shared codes would allow ciphered communications for secure messaging beyond the castle. Of course, few people own castles, and they are not designed for our increasingly IT-centric society; but, fast forward to the 1990s, and we had developed smart card chips. These chips mimicked the castle protection on a microscopic and IT-enabled scale, with secure storage, execution, communication, and sophisticated hardware and software defences against physical, side-channel, and fault attacks. Standards also emerged that supported multi-vendor, multi-application compatibility, and internationally recognised security evaluations. Therefore, it might sound like the TEE requirement was satisfied in the 1990s, but sadly no. Real castles served the narrow interests of their owners, and smart cards did the same for the card Issuers. Large vendor organisations have also become so powerful that they adopt proprietary approaches when international standards do not suit them.

Rather than providing an open security foundation for a range of applications, access to security chips was closely guarded by their owners, with policies that, practically or financially, deterred third party access. The Trusted Platform Module (TPM) initiative made use of embedded smart-card style security chips, but they were not mandated and so there is no common hardware security anchor across all device types. If we consider mobile phones as the most ubiquitous user device, a range of hardware-enabled proprietary security solutions exist, which the vendors ask us to trust without disclosure of design and implementation. The vendors also control these solutions for security reasons, but also for business advantage.

So, how is it we have sensitive applications when there is no ubiquitous and accessible hardware-based TEE for developers to exploit? Developers just do their best, with increased error risk from multiple product versions and/or relying solely on software measures when there is no better alternative. Perhaps the risk is manageable for many of the things we do with IT, but what about for cyber-physical systems? Would you reduce your TEE requirements for your medical devices, your driverless car or your smart power-grid? Clearly the need for TEE is as strong as ever, but it has yet to be adequately realised, despite suitable technologies being available. If you are interested in this vital topic and how it is evolving, then I commend you to read this book from authors knowledgeable in the field.

Royal Holloway, University of London,
Egham, United Kingdom
May 2023

Prof. Keith Mayes

Acknowledgements

We would like to thank members of the Smart Card and Internet of Things Centre (SCC) at Royal Holloway, University of London, who painstakingly commented on and corrected early drafts of this book. A big “thank you” is due to Nicola Bates, Amir Rafi, Jan Kalbantner, Zhanyu Sha, and Darren Hurley-Smith for generously volunteering their time and energy. Moreover, we wish to extend our thanks to the SCC’s past members who influenced and developed the group’s research profile in secure and trusted execution platforms. Particular thanks goes to Raja Naeem Akram, Iakovos Gurulian, Gerhard Hancke, Damien Sauveron, and all the others who are too many to mention in full. We also wish to pay special thanks to Keith Mayes, who led the creation of the SCC and guided the group through good times and challenging ones. Keith has been instrumental in building the group’s international reputation in the area of smart cards, secure elements, and related technologies. Lastly, we wish to thank the team at Springer for their flexibility and assistance towards making this book a reality.

Contents

Part I Foundations

1 Introduction	3
1.1 Why OS Security Is Not Enough.....	4
1.2 Confidential Computing.....	6
1.3 Scope.....	7
1.4 Target Audience.....	8
References.....	9
2 Background Material	11
2.1 Central Processing Unit Basics.....	11
2.2 Memory Hierarchy.....	13
2.3 Protection Rings.....	15
2.3.1 X86-64.....	16
2.3.2 ARM Exception Model.....	17
2.3.3 RISC-V Privilege Modes.....	18
2.4 Modern CPU Optimisations.....	18
2.4.1 Pipelining.....	18
2.4.2 Out-of-Order Execution.....	21
2.4.3 Speculative Execution.....	22
2.5 Virtual Memory Management.....	23
2.6 Memory Protection.....	24
2.7 Motherboards and Memory Controllers.....	25
2.8 System-on-Chips.....	27
References.....	28

Part II Technologies for Secure and Trusted Execution

3 Operating System Controls	33
3.1 Overview.....	33
3.2 Users, Groups, and Permissions.....	34
3.3 Sandboxing.....	36

- 3.3.1 Linux System Calls and Secure Computing Mode..... 36
 - 3.3.2 OS-Level Virtualisation..... 38
 - 3.4 Mandatory Access Control..... 43
 - 3.4.1 SELinux..... 44
 - 3.4.2 AppArmor..... 46
 - 3.5 Case Study: Linux Containers..... 47
 - 3.6 Discussion..... 49
 - 3.7 Using Hardware to Protect Software: The Case of Secure Boot..... 50
 - References..... 52
- 4 Isolated Hardware Execution Platforms..... 55**
 - 4.1 Introduction..... 55
 - 4.2 Smart Cards..... 56
 - 4.2.1 Java Card..... 57
 - 4.2.2 Case Study: EMV Payment Transactions..... 61
 - 4.3 Certifying a Security Platform..... 64
 - 4.3.1 Common Criteria..... 65
 - 4.3.2 The Federal Information Processing Standards..... 67
 - 4.4 Secure Elements..... 68
 - 4.4.1 GlobalPlatform Secure Element Specifications..... 69
 - 4.4.2 Host-Based Card Emulation..... 71
 - 4.5 Google Titan M..... 74
 - 4.6 Apple Secure Enclave..... 75
 - References..... 77
- 5 Building Execution Environments from the Trusted Platform Module..... 79**
 - 5.1 Introduction..... 79
 - 5.2 Trusted Platform Module..... 80
 - 5.2.1 Measured Boot..... 83
 - 5.2.2 Remote Attestation..... 84
 - 5.2.3 Binding and Sealing for Secure Storage..... 86
 - 5.2.4 TCG TPM 1.2 and 2.0..... 86
 - 5.2.5 Additional TCG TPM Standards..... 87
 - 5.2.6 Dynamic Root of Trust..... 88
 - 5.3 Microsoft Palladium..... 88
 - 5.4 Intel Trusted Execution Technology..... 89
 - 5.5 AMD Secure Virtual Machine and Secure Encrypted Virtualisation..... 90
 - 5.6 Flicker..... 92
 - 5.7 TrustVisor..... 92
 - References..... 93
- 6 Trusted World Systems..... 97**
 - 6.1 Introduction..... 97
 - 6.2 GlobalPlatform TEE..... 99

- 6.2.1 System Architecture..... 100
- 6.2.2 Client and Internal Core API Concepts..... 103
- 6.2.3 Additional APIs..... 110
- 6.2.4 Deployment Model..... 112
- 6.2.5 Protection Profile..... 114
- 6.3 ARM TrustZone..... 117
 - 6.3.1 TrustZone for Cortex-A..... 118
 - 6.3.2 TrustZone for Cortex-M..... 123
- 6.4 ARM Confidential Compute Architecture..... 125
- 6.5 Case Study: Android Keystore..... 126
- 6.6 Case Study: Android Fingerprint Authentication..... 128
- References..... 131
- 7 Enclave Computing..... 133**
 - 7.1 Introduction..... 133
 - 7.2 Intel Software Guard Extensions..... 134
 - 7.2.1 Software Architecture..... 136
 - 7.2.2 Memory Organisation..... 137
 - 7.2.3 Intel Memory Encryption Engine..... 140
 - 7.2.4 Memory Access Control Enforcement..... 141
 - 7.2.5 Launch and Tear-down Procedures..... 142
 - 7.2.6 Attestation..... 143
 - 7.2.7 Sealing..... 148
 - 7.3 Intel Trust Domain Extensions..... 149
 - 7.3.1 Trust Domain Lifecycle..... 151
 - 7.3.2 Memory Protection and Encryption..... 154
 - 7.3.3 Physical Memory Organisation..... 155
 - 7.3.4 Remote Attestation..... 155
 - 7.4 Research Proposals..... 156
 - 7.4.1 Komodo..... 157
 - 7.4.2 Sanctum..... 158
 - 7.4.3 Keystone..... 159
 - References..... 161

Part III Ongoing and Future Considerations

- 8 Deployment Issues, Attacks, and Other Challenges..... 167**
 - 8.1 Introduction..... 167
 - 8.2 Deployment Issues..... 168
 - 8.2.1 Closed-Source Technology Development..... 168
 - 8.2.2 Large and Dynamic TCBS..... 169
 - 8.2.3 Fragmentation..... 169
 - 8.3 Attacks..... 171
 - 8.3.1 Implementation Errors and Improper Usage..... 171
 - 8.3.2 Software Side-Channel Attacks..... 173
 - 8.3.3 Physical Side-Channel Attacks..... 176

- 8.3.4 Fault Injections..... 177
- 8.4 Certification Shortcomings..... 179
- 8.5 Ethical Considerations..... 180
 - 8.5.1 Privacy and Surveillance..... 180
 - 8.5.2 Curtailing User Freedom..... 181
- References..... 182
- 9 Conclusion..... 185**
 - 9.1 The Journey So Far..... 185
 - 9.2 Major Takeaways..... 186
 - 9.2.1 Lack of Universality..... 186
 - 9.2.2 Attacks and Vulnerabilities Against TEEs: A Pragmatic Perspective..... 187
 - 9.2.3 Considering TEEs in a Wider Security Context..... 188
 - 9.3 Final Thoughts: The Road Ahead..... 189
 - References..... 190
- Index..... 193**

About the Authors

Dr. Carlton Shepherd (Ph.D., B.Sc.) is a Lecturer (equivalent to Assistant Professor) in Computing at Newcastle University, UK. His expertise lies in mobile and embedded systems security at the interface of hardware and software, including trusted execution environments, side-channel and fault injection attacks, smart cards, secure elements and their applications. He holds a Ph.D. in Information Security from Royal Holloway, University of London, and a B.Sc. in Computer Science from Newcastle University.

Prof. Konstantinos Markantonakis (B.Sc., M.Sc., M.B.A., Ph.D.) received his B.Sc. in Computer Science from Lancaster University in 1995, his M.Sc. in Information Security in 1996, and his Ph.D. in 2000 and his MBA in International Management in 2005 from Royal Holloway, University of London. He is currently a Professor of Information Security in the Information Security Group in Royal Holloway University of London. He is also the Director of the Information Security Group Smart Card and IoT Security Centre (SCC). His main research interests include smart card security and applications, IoTs, embedded system security and trusted execution environments, payment, automotive, and avionics system security. He has published more than 200 papers in international conferences and journals. His research has attracted funding from competitive UK/EU funding sources and from industry. He is also the Director of the “Transformative Digital Technologies, Security and Society” Catalyst, at Royal Holloway University of London, responsible for coordinating multidisciplinary research and impact activities. He is the chair of IFIP WG 11.2 Pervasive Systems Security, has experience in the commercialisation of cyber security research, and is an experienced consultant for several technology companies.

Acronyms

ABI	Application Binary Interface
AES	Advanced Encryption Standard
ALU	Arithmetic Logic Unit
AMBA	Advanced Microcontroller Bus Architecture
AP	Access Point
APB	Advanced Peripheral Bus
API	Application Programming Interface
ARQC	Authorisation Request Cryptogram
AXI	Advanced eXtensible Interface
BGA	Ball Grid Array
CC	Common Criteria
CCRA	Common Criteria Recognition Arrangement
CCTL	Common Criteria Testing Laboratory
CLK	Clock
CMVP	Cryptographic Module Validation Program
CPU	Central Processing Unit
CRTM	Core Root of Trust for Measurement
CSR	Certificate Signing Request
CVE	Common Vulnerabilities and Exposures
DAA	Direct Anonymous Attestation
DAC	Discretionary Access Control
DDA	Dynamic Data Authentication
DES	Data Encryption Standard
DMA	Direct Memory Access
DRAM	Dynamic Random Access Memory
DRM	Digital Rights Management
DRTM	Dynamic Root of Trust for Measurement
EAL	Evaluation Assurance Level
ECC	Elliptic Curve Cryptography
ECDH	Elliptic Curve Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm

EPROM	Erasable Programmable Read-Only Memory
EEPROM	Electrically Erasable Programmable Read-Only Memory
ELF	Executable and Linkable Format
eMMC	Embedded MultiMediaCard
EMV	Europay, MasterCard, and Visa
FEK	File Encryption Key
FIPS	Federal Information Processing Standards
FIQ	Fast Interrupt Request
GIC	Generic Interrupt Controller
GID	Group Identifier
GND	Ground
GPIO	General-Purpose Input/Output
GPU	Graphics Processing Unit
HCE	Host Card Emulation
HDD	Hard Disk Drive
HMAC	Hashed Message Authentication Code
HUK	Hardware Unique Key
I/O	Input/Output
I ² C	Inter-Integrated Circuit
IC	Integrated Circuit
ICOM	Inter-Communications
IDAU	Implementation-Defined Attribution Unit
IEC	International Electrotechnical Commission
IETF	Internet Engineering Task Force
IMEI	International Mobile Equipment Identity
IMSI	International Mobile Subscriber Identity
IoT	Internet of Things
IOMMU	Input/Output Memory Management Unit
IPC	Inter-Process Communication
IRQ	Interrupt Request
ISA	Instruction Set Architecture
ISO	International Organization for Standardization
JCRE	Java Card Runtime Environment
JCVM	Java Card Virtual Machine
JTAG	Joint Test Action Group
JVM	Java Virtual Machine
LLC	Last Level Cache
MAC	Message Authentication Code
MPU	Memory Protection Unit
MLS	Multi-Level Security
MOSFET	Metal-Oxide-Semiconductor Field-Effect Transistor
NAS	Network-Attached Storage
NFC	Near-Field Communication
NIST	National Institute of Standards and Technology
NS	Non-Secure

NSA	National Security Agency
NVIC	Nested Vectored Interrupt Controller
OEM	Original Equipment Manufacturer
OS	Operating System
OTP	One-Time Password
PCR	Platform Configuration Register
PIC	Programmable Interrupt Controller
PID	Process Identifier
PIN	Personal Identification Number
PKI	Public Key Infrastructure
PMP	Physical Memory Protection
PoP	Package-on-Package
PoS	Point of Sale
PP	Protection Profile
PRNG	Pseudorandom Number Generator
RA	Remote Attestation
RAM	Random Access Memory
RBAC	Role-Based Access Control
REE	Rich Execution Environment
RFC	Request for Comments
RFID	Radio Frequency Identification
RNG	Random Number Generator
ROM	Read Only Memory
RoT	Root of Trust
RPMB	Replay Protected Memory Block
RSA	Rivest-Shamir-Adleman
RTC	Real-Time Clock
RTM	Root of Trust for Measurement
RTOS	Real-Time Operating System
RTR	Ready to Run
RTS	Ready to Send
SAR	Security Assurance Requirement
SAU	Security Attribution Unit
SCI	System Configuration and Integration
SCP	Secure Channel Protocol
SCR	System Control Register
SCS	System Control Space
SD	Secure Digital
SE	Security Extension
SEP	Secure Enclave Processor
SFR	Security Functional Requirement
SGX	Software Guard Extensions
SHA	Secure Hash Algorithm
SHS	Secure Hash Standard
SiP	System in Package

SMC	Secure Monitor Call
SoC	System on Chip
SRAM	Static Random Access Memory
SRK	Storage Root Key
SSD	Solid State Drive
SSH	Secure Shell
SSK	Secure Storage Key
SVM	Secure Virtual Machine
TCG	Trusted Computing Group
TDX	Trusted Domain Extensions
TEE	Trusted Execution Environment
TLB	Translation Lookaside Buffer
TLS	Transport Layer Security
TMF	Trusted Management Framework
TOE	Target of Evaluation
TPM	Trusted Platform Module
TRNG	True Random Number Generator
TSK	TA Storage Key
TSM	Trusted Service Manager
TUI	Trusted User Interface
TXT	Trusted Execution Technology
TZASC	TrustZone Address Space Controller
TZMA	TrustZone Memory Adapter
TZPC	TrustZone Protection Controller
UEFI	Unified Extensible Firmware Interface
UID	User Identifier
USB	Universal Serial Bus
UTS	Unix Time-Sharing System
UUID	Universally Unique Identifier
VM	Virtual Machine
VMM	Virtual Machine Monitor

List of Figures

Fig. 2.1	High-level system diagram of a modern eight-core X86-64 CPU	13
Fig. 2.2	A conventional memory hierarchy	14
Fig. 2.3	Modern CPU privilege levels (colours show components with similar roles across different architectures).....	16
Fig. 2.4	Instruction pipeline flow beginning with a filled pipeline. (IF: Instruction fetch, DE: Decode, EX: Execute, MA: Memory access, WB: Write-back)	20
Fig. 2.5	32-bit RISC-V PMP layout (standalone numbers denote the bit widths of adjacent fields) [2]	25
Fig. 2.6	CPU, and northbridge and southbridge connections	26
Fig. 2.7	High-level system-on-chip production workflow	28
Fig. 3.1	An overview of the Linux system call interface (SCI)	37
Fig. 3.2	An example chroot jail applied to /test/ that contains emulated system directories. A process running in the jail will see / as the root directory when it is actually confined to /test/ on the host	41
Fig. 3.3	Security monitor example	43
Fig. 3.4	SELinux policy decision flow diagram.....	45
Fig. 3.5	High-level architectures of VMs and containers.....	48
Fig. 3.6	Generic secure boot execution flow	51
Fig. 4.1	Smart card physical characteristics	57
Fig. 4.2	Functional units of a typical contactless smart card	58
Fig. 4.3	Java Card platform architecture	60
Fig. 4.4	Simplified EMV transaction message flow. Notation: AC = Application cryptogram, AID = Application ID, ARQC = Authorisation request cryptogram, T = Transaction data, TC = Transaction certificate	63

Fig. 4.5 Overview of a GlobalPlatform Secure Element TOE. Green boxes: Covered by the GP SE PP [29]. Red: Not covered by the GP SE PP. Purple: Selected standards used by the TOE in the GP SE PP. White: Selected standards that are not specifically mentioned in the GP SE PP but are typically employed by the TOE..... 72

Fig. 4.6 Conducting NFC-based transactions using a secure element-enabled device (left) versus HCE (right) 73

Fig. 4.7 Titan M block diagram, showing its CPU (orange), memory (blue) and I/O units (red), and countermeasures (yellow) 75

Fig. 5.1 An attacker user is unable to directly access critical assets on a SIM, even if they have total control over the host device 80

Fig. 5.2 TPM 2.0 functional units. Green regions: Covered by the TPM Main Specifications [4]. Blue: Defined by the TCG TSS specifications [7]. Orange: Third-party tools that interact directly with the TCG TSS. Red: Software that wishes to use the TPM 82

Fig. 5.3 An example measured boot sequence on a modern platform. Green blocks: Inherently trusted components and part of the TPM. Purple: Inherently trusted components, but not part of the TPM. Yellow: Untrusted software components to be measured. Note that the TPM is part of the host machine 84

Fig. 5.4 A generic remote attestation protocol 85

Fig. 5.5 AMD Secure Virtual Machine’s boot-time execution flow. Red: Inherently trusted hardware. Purple: Measurement software. Green: Measured software 91

Fig. 6.1 The system architecture of a GlobalPlatform TEE, showing TAs (green) and untrusted (red) and trusted components (blue). The TEE contains *m* security domains managing different TAs. The world isolation boundary (solid green line) changes depending on whether the REE or TEE controls the shared hardware (dotted line) 100

Fig. 6.2 GlobalPlatform TEE secure boot process, showing trusted (green) and untrusted (red) components 102

Fig. 6.3 An overview encrypting and storing TA data into an eMMC with a RPMB using the REE. In some implementations, the TEE may read/write directly to the eMMC (dashed line) 109

Fig. 6.4 High-level GlobalPlatform TEE deployment model 113

Fig. 6.5 ARMv8 exception level model 118

Fig. 6.6 Block diagram of an example ARM TrustZone system-on-chip, showing components available to the secure (green), untrusted (red), or both worlds (green/red gradient). Note: EL2 is not shown 120

Fig. 6.7 ARM Confidential Compute Architecture, showing untrusted (red), trusted (green), and mutually distrusting (yellow) components 126

Fig. 6.8 System architecture of the Android Keystore (arrows indicate information flow) [44] 127

Fig. 6.9 Architecture of the Android fingerprint authentication system (arrows indicate information flow) 129

Fig. 7.1 Intel SGX software architecture (Note that more than one ECALL and OCALL may be defined for each application) 137

Fig. 7.2 Intel SGX’s high-level memory layout 138

Fig. 7.3 Intel SGX remote attestation process 147

Fig. 7.4 The Intel TDX trust model, showing untrusted (red) and trusted (green) system components comprising its trusted computing base 152

Fig. 7.5 Sanctum enables enclaves to effectively execute on a private core (green) in the presence of untrusted components (red). Partitioned components are also shown (green-red gradient). A dual-core system is illustrated in this example 159

Fig. 7.6 Keystone execution flow, showing accessible (green) and inaccessible (red) physical memory regions during transitions between the operating system (OS), secure monitor (SM), and a single enclave 160

Fig. 8.1 A generic fault injection attack setup 178

List of Tables

Table 1.1	Comparing confidential computing technologies using criteria adapted from [34].....	7
Table 2.1	IETF RFC 7228 [3] device classes with examples	12
Table 4.1	Common Criteria EALs in line with ISO/IEC 15408 [20]	66
Table 4.2	An overview of GlobalPlatform Secure Channel Protocols (SCPs) covered by the GlobalPlatform Secure Element specifications. A given GlobalPlatform-compliant SE may support one or more of these SCPs.....	70
Table 6.1	Security requirements of GlobalPlatform TEE assets	115
Table 6.2	GlobalPlatform TEE Protection Profile threats [8]	117
Table 7.1	Selected Intel SGX instructions	144
Table 7.2	Selected Intel TDX instructions	152