Aalto University

UNIVERSITY OF WATERLOO

Hardware-assisted Trusted Execution Environments

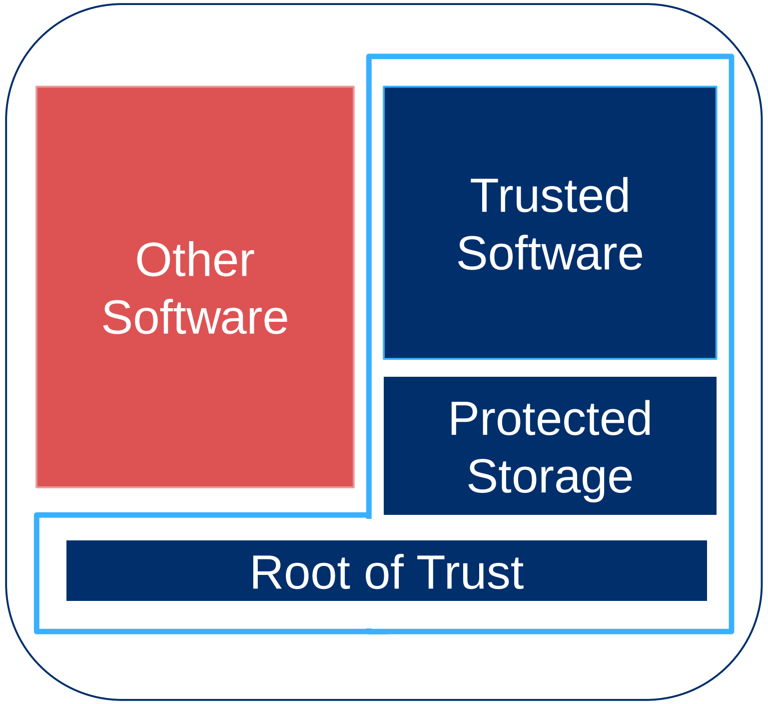
Look Back, Look Ahead

N. Asokan

https://asokan.org/asokan/

@nasokan

Hardware-assisted TEEs are pervasive



Hardware support for

- **Isolated execution: Isolated Execution Environment**
- **Protected storage: Sealing**
- **Ability to convince remote verifiers: (Remote) Attestation**

Trusted Execution Environments (TEEs)

Operating in parallel with "rich execution environments" (REEs)

Cryptocards

https://www.ibm.com/security/cryptocards/

Trusted Platform Modules

https://www.infineon.com/tpm

ARM TrustZone

arm

https://www.arm.com/products/security-on-arm/trustzone

Intel Software Guard Extensions

intel

https://software.intel.com/en-us/sgx

[A+14] "Mobile Trusted Computing", Proceedings of the IEEE, 102(8) (2014)
[EKA14] "Untapped potential of trusted execution environments", IEEE S&P Magazine, 12:04 (2014)

Concerns with TEEs: flaws

TPM Reset Attack

50,012 views

Evan Sparks
Published on Jun 18, 2007

A demonstration of a vulnerability in the TCG archi
running TPM without restarting the platform.

http://www.cs.dartmouth.edu/~pkilab/sparks/ (2007)

CLKSCREW: Exposing the Perils of Security-Oblivious Energy Management

Authors:
Adrian Tang, Simha Sethumadhavan, and Salvatore Stolfo, *Columbia University*
Distinguished Paper Award Winner!

https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/tang (2017)

Foreshadow (security vulnerability)

From Wikipedia, the free encyclopedia

This article is about the security vulnerability. For other uses, see Foreshadow (disambiguation).

Foreshadow is a vulnerability that affects modern microprocessors that was first discovered by two independent teams of researchers in January 2018, but was first disclosed to the public on 14 August 2018.[1][2][3][4][5][6][7][8][9][10][11][12][13][14][15][16] The vulnerability is a speculative execution attack on Intel processors that may result in the loss of sensitive information stored in personal computers, or third party clouds.[1] There are two versions: the first version (original/Foreshadow) (CVE-2018-3615) targets data from SGX enclaves; and the second version (next-generation/Foreshadow-NG) (CVE-2018-3620 and CVE-2018-3646) targets Virtual Machines (VMs), hypervisors (VMM), operating system (OS) kernel memory, and System Management Mode (SMM) memory.[1] Intel considers the entire class of speculative execution side channel vulnerabilities as "L1 Terminal Fault" (L1TF).[1] A listing of affected Intel hardware has been posted.[10][11]

Foreshadow is similar to the Spectre security vulnerabilities discovered earlier to affect Intel and AMD chips, and the Meltdown vulnerability that also affected Intel.[6] However, AMD products, according to AMD, are not affected by the Foreshadow security flaws.[6] According to one expert, "[Foreshadow] lets malicious software break into secure areas that even the Spectre and Meltdown flaws couldn't crack".[15] Nonetheless, one of the variants of Foreshadow goes beyond Intel chips with SGX technology, and affects "all [Intel] Core processors built over the last seven years".[2]

Foreshadow may be very difficult to exploit,[2][6] and there seems to be no evidence to date (15 August 2018) of any serious hacking involving the Foreshadow vulnerabilities.[2][6] Nevertheless, applying software patches may help alleviate some concern(s), although the balance between security and performance may be a worthy consideration.[5] Companies performing cloud computing may see a significant decrease in their overall computing power; individuals, however, may not likely see any performance impact, according to researchers.[9] The real fix, according to Intel, is by replacing today's processors.[5] Intel further states, "These changes begin with our next-generation Intel Xeon Scalable processors (code-

FORESHADOW
A logo created for the vulnerability, featuring a lock with a shadow

https://en.wikipedia.org/wiki/Foreshadow_(security_vulnerability) (2018)

(CCS 2019)

...

Concerns with TEEs: suspicions of motives

Software

MS Palladium protects IT vendors, not you – paper

Anderson gives us the FAQs

By John Lettice 28 Jun 2002 at 10:27 SHARE ▼

https://www.theregister.co.uk/2002/06/28/ms_palladium_protects_it_vendors/ (2002)

Problem: Third-party uncertainty about your software environment is normally a feature, not a bug

https://www.eff.org/wp/trusted-computing-promise-and-risk (2003)

Trusting Intel – Next Generation of Backdooring?

We have seen that SGX offers a number of attractive functionality that could potentially make our digital systems more secure and 3rd party servers more trusted. But does it really?

The obvious question, especially in the light of recent revelations about NSA backdooring everything and the kitchen sink, is whether Intel will have backdoors allowing "privileged entities" to bypass SGX protections?

http://theinvisiblethings.blogspot.fi/2013/09/thoughts-on-intels-upcoming-software.html (2013)

Outline

A Look Back: How did TEEs start?

What are some (useful) applications for TEEs?

What are the downsides of relying on hardware-assisted TEEs?

(How) can we deal with these downsides?

What are other examples of hardware-assisted security?

Look Back

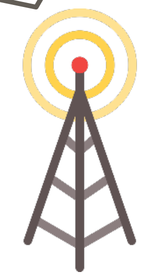
Platform security for mobile devices

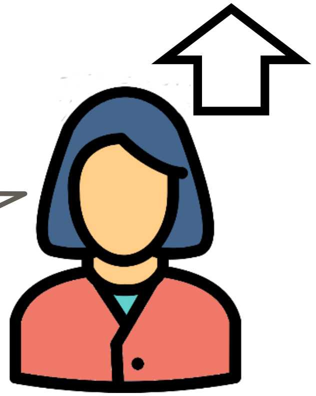
Mobile network operators:

1. Subsidy locks → immutable ID
2. Copy protection → device authentication, app. separation
3. …

Regulators:

1. RF type approval → secure storage
2. Theft deterrence → immutable ID
3. …





End users:

1. Reliability → app. separation
2. Theft deterrence → immutable ID
3. Privacy → app. separation
4. …

Closed → Open

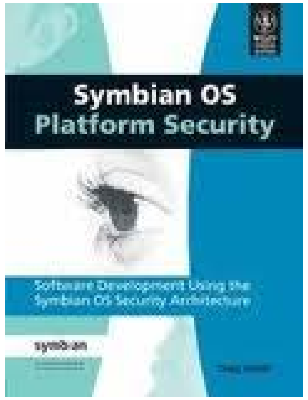
Different Expectations than for PCs!

Early adoption of software platform security

~2001

J2ME

~2004



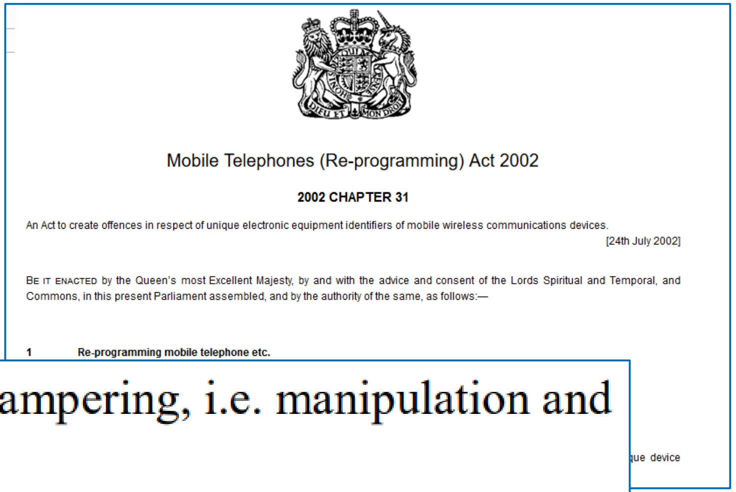
~2008

iOS

Mobile software platform security is now widely deployed

[KREA11] "Old, new borrowed, blue: a perspective on the evolution of mobile platform security architectures", ACM CODASPY (2011)

Example: regulatory compliance


Mobile Telephones (Re-programming) Act 2002

2002 CHAPTER 31

An Act to create offences in respect of unique electronic equipment identifiers of mobile wireless communications devices.

[24th July 2002]

Be it enacted by the Queen's most Excellent Majesty, by and with the advice and consent of the Lords Spiritual and Temporal, and Commons, in this present Parliament assembled, and by the authority of the same, as follows:—

1 Re-programming mobile telephone etc.

The IMEI shall not be changed after the ME's final production process. It shall resist tampering, i.e. manipulation and change, by any means (e.g. physical, electrical and software).

NOTE: This requirement is valid for new GSM Phase 2 and Release 96, 97, 98 and 99 MEs type approved after 1st June 2002.

3GPP TS 42.009, 2001

Secure storage of RF configuration parameters

Early TEEs for mobile phones

Nokia Radio Application Processor (RAP), ca. 2001

TrustZone®
Security Foundation by ARM®

Saara Matala & Thomas Nyman, "Historical insight into the development of Mobile TEEs", Aalto SSG research group blog (2019)

Mobile TEEs: Motivation

https://www.wsj.com/articles/SB893268045342680500


THE WALL STREET JOURNAL.

U.S. Edition ▾ | September 16, 2019 | Print Edition | Video

Home World U.S. Politics Economy Business Tech Markets Opinion Life & Arts Real Estate

Motorola Announces Plans To Offer Smart-Card Phone

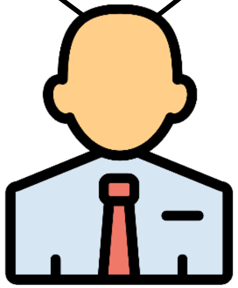
By Kimberley A. StrasselStaff Reporter of The Wall Street Journal

Updated April 23, 1998 12:01 am ET

LONDON -- Hoping to cash in on one of Europe's hottest technology markets, Motorola Corp. announced a new mobile telephone that features a slot for smart cards.

Business requirements:

- mobile payment
- subsidy lock



Regulatory requirements:

- tamper-resistant IMEIs
- secure storage for RF

Engineering constraints:

Cost of discrete security chip too high on bill of materials!

New approach: "*processor secure environments*"

Generic low-cost enabler emerged as skunkworks project within Nokia (rather than point solutions for particular use cases)

Mobile TEEs: Development

1982 — 1996 — 2002 — 2003 — 2004 — 2008 — 2015 — 2016 — 2019

2004 ARM TrustZone
[AF04] "TrustZone: Integrated Hardware and Software Security", Information Quarterly (2004)

2003 Texas Instruments OMAP 161x and 73x processors
[H03] "OMAP Platform Security Features ", Whitepaper (2003, updated 2008)

2002 Nokia, Kiiveri and Paatero US9111097B2
"Secure execution architecture"

1996 Intertrust, Ginter et al US 5892900A
"Systems And Methods For Secure Transaction Management And Electronic Rights Protection"

1982 Texas Instruments, Guttag US4521853A
"Secure microprocessor/microcomputer with secured memory"

1982 Texas Instruments, Guttag and Nussarallah US4521853A
"Security bit for designating the security status of information stored in a nonvolatile memory"

Mobile TEEs: Deployment

1982 — 1996 — 2002 — 2003 — 2004 — 2008 — 2015 — 2016 — 2019

TrustZone®
Security Foundation by ARM®

intel®

arm Cortex-M

First deployment: Nokia 6630 ("Charlie")

- first 3G phone with TI OMAP 1710 processor (June 2004)

ARM TrustZone currently widely deployed

- TrustZone-M for Cortex-M class microcontrollers (2016)

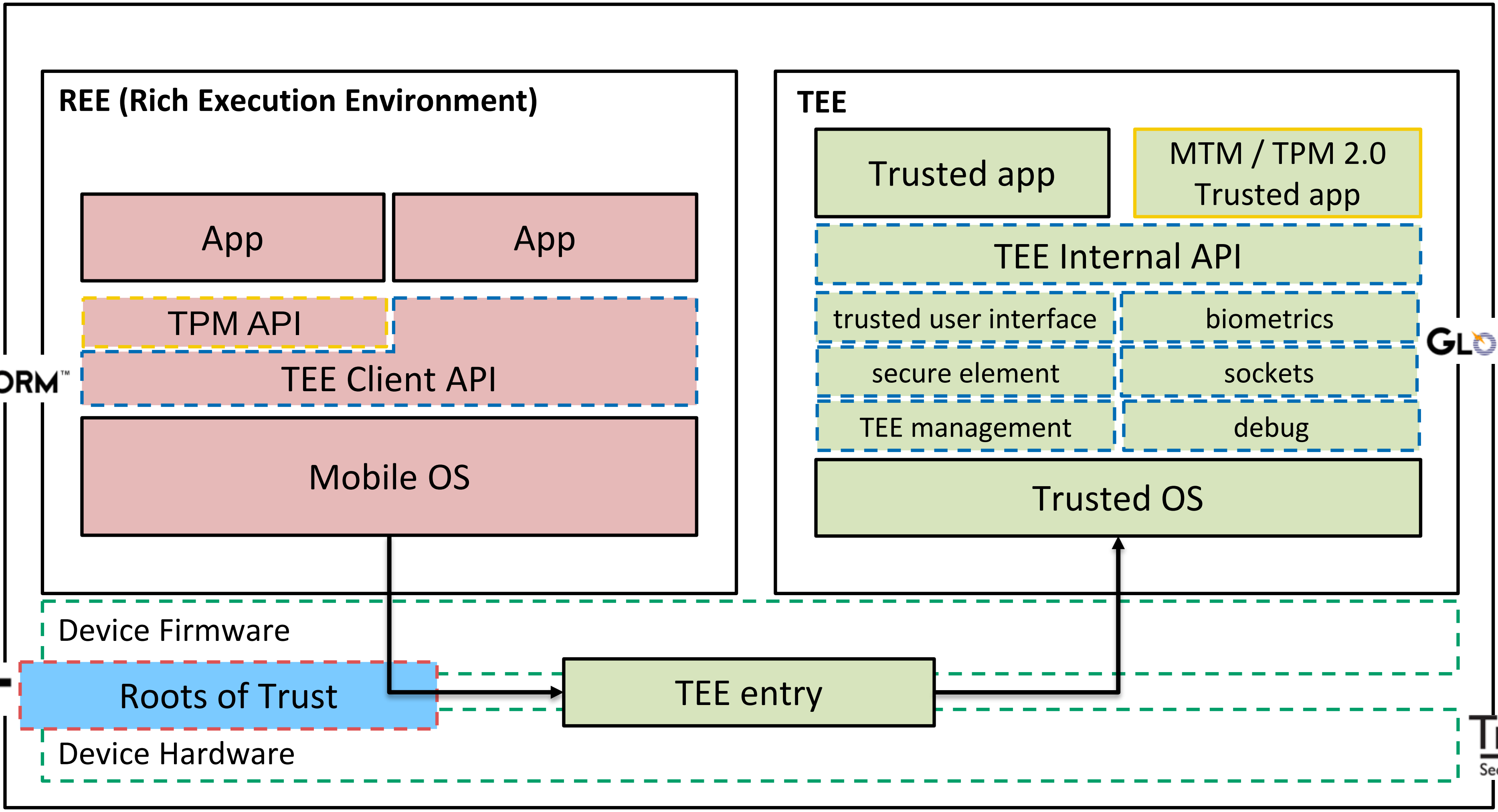
Ca. 2008, TEE unheard of in academic circles

- first papers in FC 2008, ASIACCS 2009
  [AE08] A Platform for OnBoard Credentials, Financial Cryptography and Data Security (2008)
  [KEAR09] On-board credentials with open provisioning, ACM ASIACCS (2009)

Intel SGX

- SkyLake (2015); wide availability of SDK "democratized" TEE research

Mobile TEEs: Standardization



Using TEEs

Original motivations (for mobile TEEs)

Tamper-resistant device identifiers (IMEIs)

for various use cases including theft protection, subsidy lock, and DRM

Sealed storage

for secure storage of RFID configuration data

Mobile payments

Boot integrity

TEE applications: academic literature (1/2)

Private membership test for malware scanning, private contact discovery,..

[TLPEPA17] "The Circle Game: Scalable Private Membership Test Using Trusted Hardware", ACM ASIACCS (2017)

[KLSAP17] "Private Set Intersection for Unequal Set Sizes with Mobile Applications", PETS (2017)

Signal private contact discovery, Sep 2017

> This is much faster. The above code still iterates across the entire set of registered users, but it only does so once for the entire collection of submitted client contacts. By keeping one big linear scan over the registered user data set, access to unencrypted RAM remains "oblivious," since the OS will simply see the enclave touch every item once for each contact discovery request.
>
> The full linear scan is fairly high latency, but by batching many pending client requests together, it can be high throughput.

https://signal.org/blog/private-contact-discovery

Protection of password-based web authentication

[KKPMA18] "SafeKeeper: Protecting Web Passwords using Trusted Execution Environments", WWW (WebConf) (2018)

Secure accounting for function-as-a-service (FaaS) settings

[AAKPS18] "S-FaaS: Trustworthy and Accountable Function-as-a-Service using Intel SGX", ACM CCSW (2019)

Scalable consensus for blockchains and cryptocurrencies

[LLKA19] "Scalable Byzantine Consensus via Hardware-Assisted Secret Sharing", IEEE Trans. Comp. 68(1) (2018)

[GLVA19] "Making Speculative BFT Resilient with Trusted Monotonic Counters", IEEE SRDS (2019)

Examples only, not a complete list

TEE applications: academic literature (2/2)

Private neural network evaluation

[TB19] "Slalom: Fast, Verifiable and Private Execution of Neural Networks in Trusted Hardware", ICLR (2019)

High-performance remote ORAM

[SGF18] "ZeroTrace: Oblivious Memory Primitives from Intel SGX", NDSS (2018)

[SS19] "ConsenSGX: Scaling Anonymous Communications Networks with Trusted Execution Environments", PETS (2019)

[HOJY19] "Hardware-Supported ORAM in Effect: Practical Oblivious Search and Update on Very Large Dataset", PETS (2019)

Verifiable computation

[TZLHJS17] "Sealed-Glass Proofs: Using Transparent Enclaves to Prove and Sell Knowledge", IEEE EuroS&P (2017)

Authenticated data feeds

[ZCCJS16] "Town Crier: An Authenticated Data Feed for Smart Contracts", ACM CCS (2016)

Examples only, not a complete list

TEE applications: commercial deployments

Digital rights management (e.g. Widevine L1 & L2 content decryption)

Widevine DRM Architecture Overview https://www.androidauthority.com/widevine-explained-821935/

Runtime integrity (e.g. OS kernel integrity monitoring)

[A+14] Samsung TIMA, ACM CCS (2016) https://doi.org/10.1145/2660267.2660350

Local user authentication (e.g. password authentication, biometrics)

Android Gatekeeper https://source.android.com/security/authentication/gatekeeper
Android Fingerprint HAL https://source.android.com/security/authentication/fingerprint-hal
Windows Hello https://docs.microsoft.com/en-us/windows/security/information-protection/tpm/how-windows-uses-the-tpm

Property Attestation (e.g. proof that cryptographic credential is protected by TEE)

Android Key and ID Attestation https://source.android.com/security/keystore/attestation
MirrorLink Content Attestation https://www.etsi.org/deliver/etsi_ts/103500_103599/10354404/01.03.00_60/ts_10354404v010300p.pdf
Also see [KAE11] Practical Property-Based Attestation on Mobile Devices, TRUST (2011)

Examples only, not a complete list

Downsides of TEEs

Downsides of TEE-based solutions

Difficulty of developer access

Risk of TEE compromise

Difficulty of developer access

TEEs were closed systems

Tools for TEE software development were cumbersome and/or expensive

Device or TEE vendor controls what applications allowed to executed in the TEE

- Ordinary developers cannot deploy TEE apps without vendor approval

Risk of TEE compromise

Software attacks

- Many trusted applications are written in unsafe languages
- Correct trusted code can be vulnerable to confused-deputy attacks
- Difficult even for hardware security module vendors [CC19]

Side-channel attacks

- Timing
- Memory access
- Electromagnetic emanations

[CC19] “Everybody be Cool, This is a Robbery!”, Black Hat (2019)

Dealing with downsides

Challenge: easy development

GlobalPlatform standards for TEE interfaces
http://www.globalplatform.org/specificationsdevice.asp

Open-source tools for TEE app development available:

- **OP-TEE** https://www.op-tee.org/
- **Open-TEE** https://open-tee.github.io/
- **OpenEnclave** https://openenclave.io

Developing TEE applications is no longer cumbersome or expensive

[MDNA15] "Open-TEE – An Open Virtual Trusted Execution Environment", TRUST (2015)

Challenge: open deployment

On-board Credentials (ObC) (2006-2009)

- open credential platform leveraging TEE functionality
- allows any developer to write/use TEE apps
- deployed in Nokia and Windows smartphones (2009-2012)
- applications:
  - RSA SecurID, mobile ticketing (trialed at NY MTA LIRR in 2011), even "soft SIM“

Other efforts to address the deployment hurdle:

- “User centric provisioning” work from Royal Holloway
  - E.g., “A Paradigm Shift in Smart Card Ownership Model” (2010)
- GlobalPlatform white paper
  - *“A New Model: The Consumer-Centric Model and How It Applies to the Mobile Ecosystem” Whitepaper (2012)*

[KEAR09] “On-board credentials with open provisioning”, ACM ASIACCS (2009)

Challenge: Dealing with TEE compromise

Hardware attacks pose a serious threat

No longer reasonable to assume hardware security to be inviolable

Abandon hardware-assisted TEEs altogether?

Instead rely only on cryptographic techniques like MPC?

TEEs still hold the promise of efficient solutions

Hardware-assistance and cryptography are not mutually exclusive!

Defense-in-depth is desirable

Novel approaches for dealing with TEE compromise may be feasible

Challenge: Dealing with TEE compromise

Prevent

- Formal verification
- Minimal coupling between TEE and REE

Tolerate

- Replication / redundancy
- Application-specific mitigation

Formal Verification

Formal verification can prevent software vulnerabilities

Good track record with protocol specs and implementations

- TLS 1.3 [CMSv16]
- miTLS, a formally-verified TLS implementation [BFKPS13]

Applicable to platform security

- seL4, a formally-verified microkernel [K+09]
- ProvenCore, ProvenCore-M, commercial formally-verified kernels [L15]

Caveat: Formal analysis is only as good as the underlying model

- seL4 needed to be patched for Meltdown like everything else

[CMSv16] "Automated Analysis and Verification of TLS 1.3: 0-RTT, Resumption and Delayed Authentication", IEEE S&P (2016)
[BFKPS13] "Implementing TLS with Verified Cryptographic Security", IEEE S&P (2013)
[K+09] "seL4: Formal Verification of an OS Kernel", ACM SOSP (2009)
[L15] "ProvenCore: Towards a Verified Isolation Micro-Kernel", MILS (2015)

Recall: TEE system architecture

Minimal Coupling

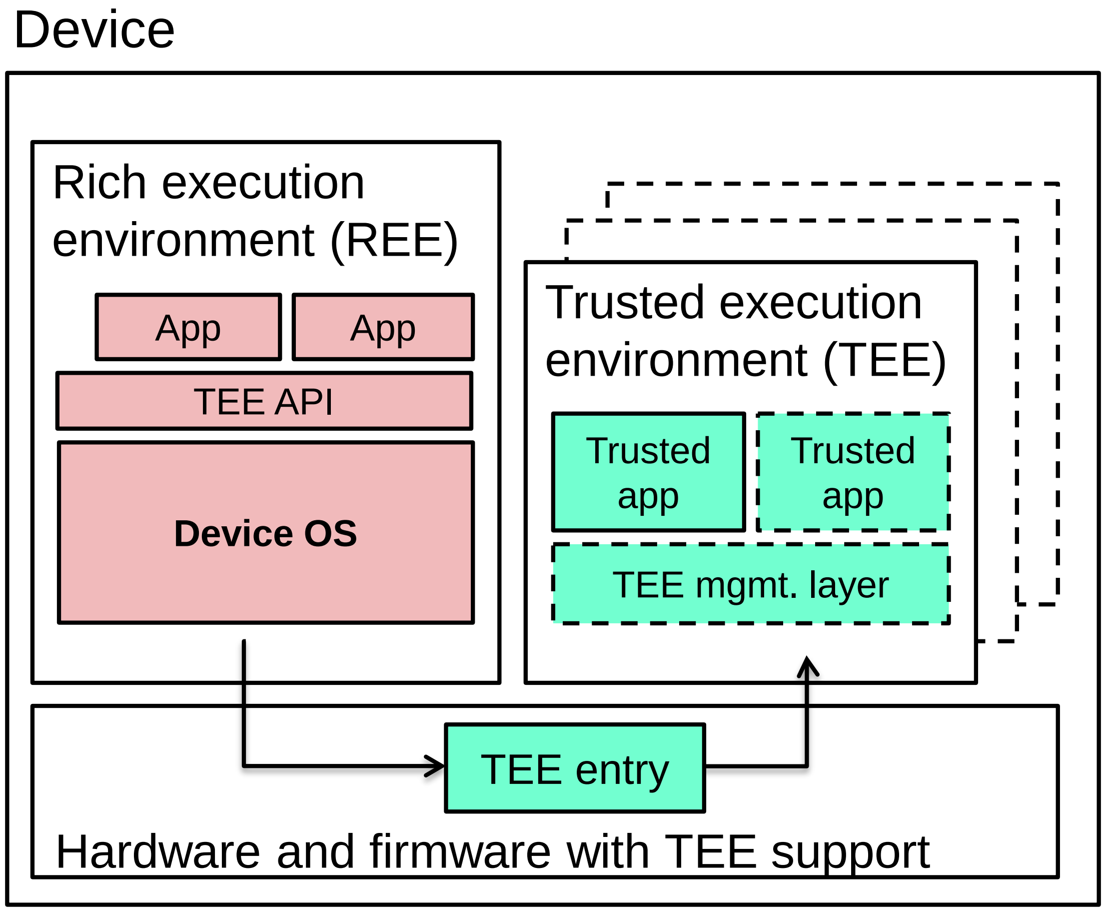
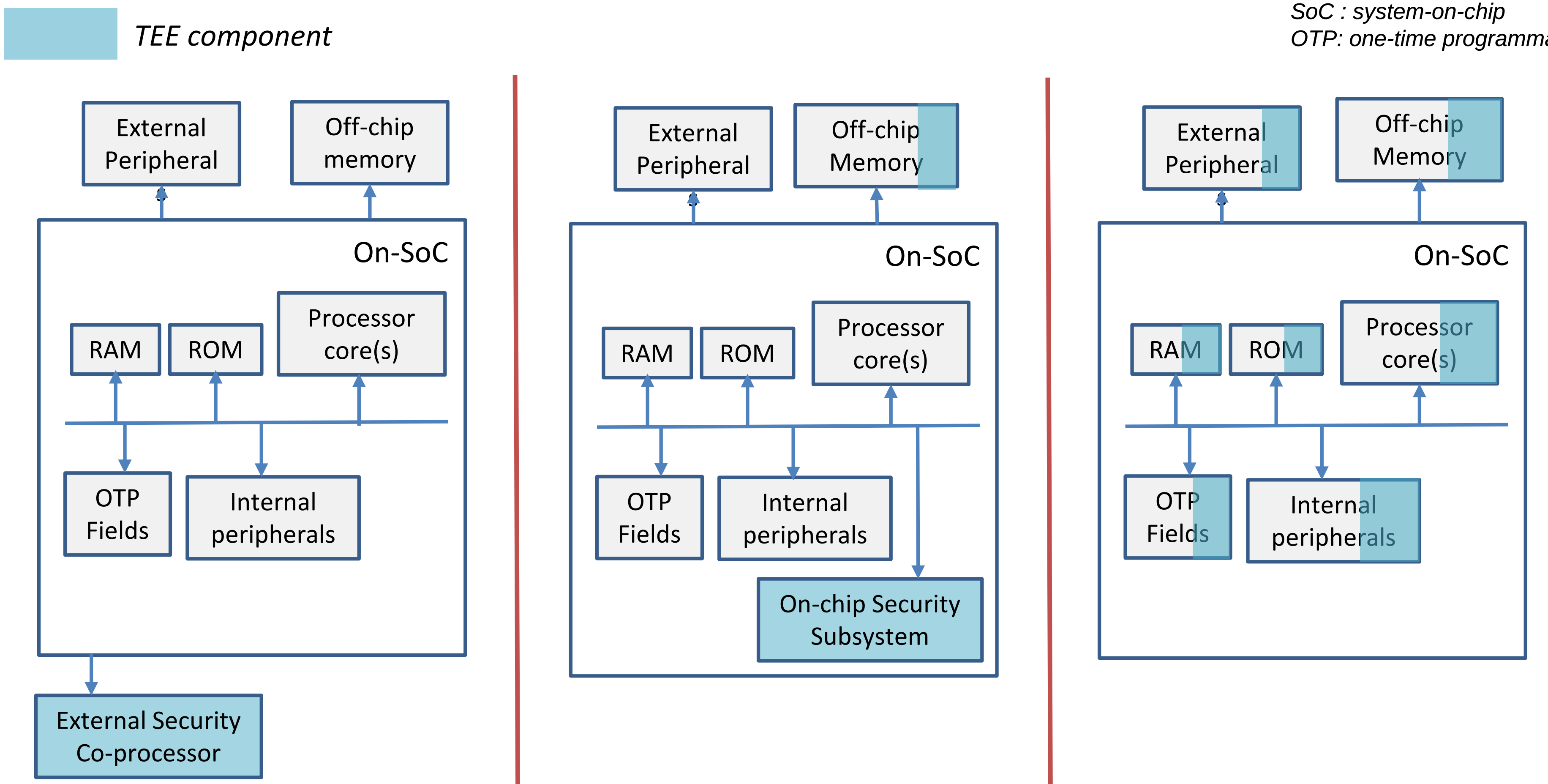


Figure adapted from: Global Platform. [TEE system architecture](). 2011.

TEE hardware realization alternatives

Minimal Coupling

Legend:
SoC : system-on-chip
OTP: one-time programmable



External Secure Element (TPM, smartcards)

Embedded Secure Element

Processor Secure Environment (SGX, TrustZone, M-Shield)

Figure adapted from: Global Platform. TEE system architecture. 2011.

Minimal coupling

TEE closely coupled to REE

Intel SGX

ARM Trustzone

TPM

TEE more isolated from REE

Larger attack surface

Greater sharing of resources (OS services, cache, memory, processor)

Smaller attack surface

Minimal coupling in the real world

Discrete security processors in modern smartphones

- Apple Secure Enclave Processor (SEP)
  Apple, iOS Security Whitepaper, May 2019
- Google Titan M
  Google Device Security Group, Building a Titan, Android Developers Blog, October 2018

Physical isolation mitigates against entire classes of hardware-level exploits

- Processor, caches, memory, and persistent storage are not shared with main OS

Layered defense using multiple TEEs

Minimal Coupling

Replication/Redundancy

Enables division of tasks (and secrets) between two (or more) elements

Improved security for stored secrets

- Android Strongbox Keymaster https://developer.android.com/training/articles/keystore#HardwareSecurityModule
- SEP Secure Key Store https://support.apple.com/en-us/HT209632

Sensitive peripheral management (e.g. camera LED indicator, microphone disconnect)

- SEP camera/microphone hardware control https://www.apple.com/mac/docs/Apple_T2_Security_Chip_Overview.pdf

Trusted path (e.g. isolated circuit to side buttons)

- Android Protected Confirmation https://android-developers.googleblog.com/2018/10/android-protected-confirmation.html

Insider attack resistance (e.g. firmware updates require device owner’s cooperation)

- Android Insider attack Resistance https://android-developers.googleblog.com/2018/05/insider-attack-resistance.html

Application-specific mitigation

Premise: Exploiting a hardware compromise may leave tell-tale signs

Approach: Use application-specific domain knowledge for detection or mitigation of the effects of hardware compromise

Example: Proof of Elapsed Time (PoET)

PoET is a replacement for Proof of Work in Bitcoin-like blockchains

Proof of Work:

First miner to solve puzzle wins (gets to proposes next block)

Work ~ Exp (difficulty)

Proposals can be made at a rate proportional to computational power

Proof of Elapsed Time:

TEE issues attestation after waiting (idly) for a while; First miner to get the attestation wins

Idle wait time ~ Exp (difficulty)

Proposals can be made at a rate proportional to the number of idle CPUs

Intel, [Hyperledger Sawtooth Documentation](), 2015

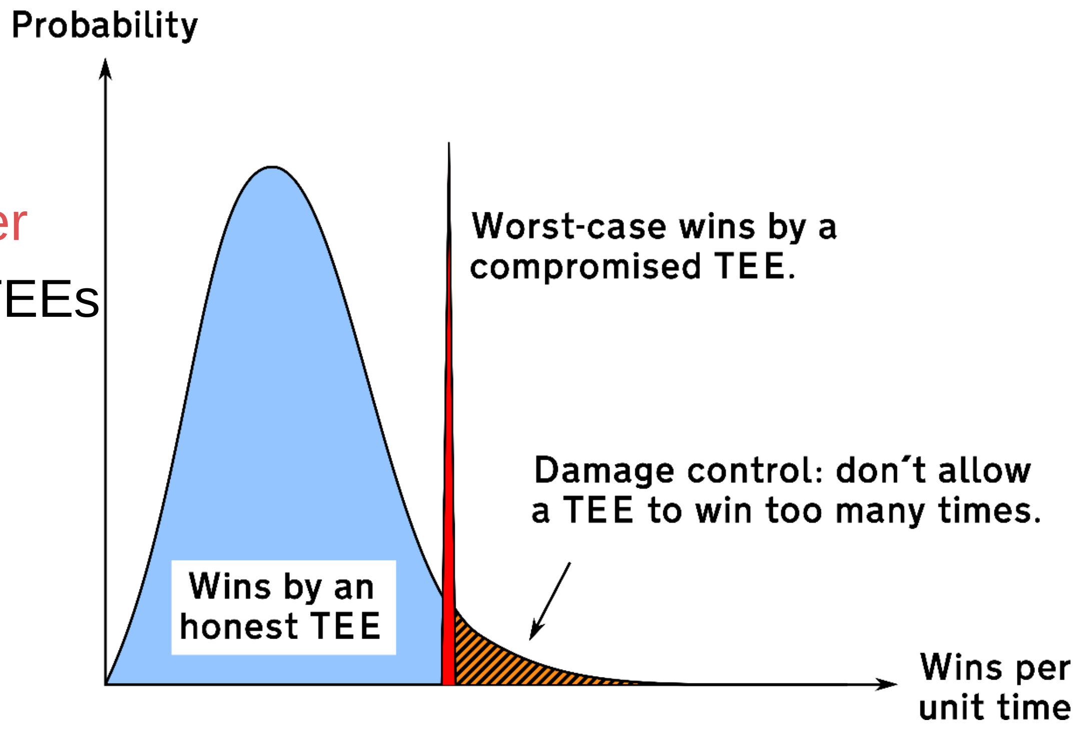
App-specific mitigation

Example: Dealing with TEE compromise

Problem: A compromised TEE can win every block

Statistical solution: refuse blocks from machines that have won too many times

- Before: compromised TEEs give attacker unlimited power
- After: attacker power proportional to # of compromised TEEs



"Design for Failure"

Open question: How can TEE-using applications detect/mitigate effects of TEE-compromise?

[Intel15] "Hyperledger Sawtooth Documentation" (2015)
[C+174] "On Security Analysis of Proof-of-Elapsed-Time (PoET)", SSS (2017)

Cross-layer design for security

App-specific mitigation

- Frank Piessens (2019)

Hennessy and Patterson on cross-layer design (for performance):

- “Achieving significant gains through such approaches will require a vertically integrated design team that understands applications, domain-specific languages and related compiler technology, computer architecture and organization, and the underlying implementation technology”
- “In this new era, vertical integration has become more important, and teams that can examine and make complex trade-offs and optimizations will be advantaged”

Such cross-layer design can have similar benefits for security

- Not surprising, as there are often significant trade-offs between security and performance

App-specific mitigation

Carrying security information across layers

- Frank Piessens (2019)

Applications may have precise information about what data in the program is confidential

In state-of-practice compilation, this information is lost

By preserving this information during compilation, we can use it to selectively close micro-architectural channels used in transient execution attacks

[P+15] "Secure compilation to protected module architectures", ACM TOPLAS (2015)
[ASJP12] "Secure compilation to modern processors", CSF (2012)
[N+19] "HardScope: Hardening Embedded Systems Against Data-Oriented Attacks", ACM DAC (2019)
[LNWPEA19] PAC it up: Towards Pointer Integrity using ARM Pointer Authentication. USENIX Security (2019)

Other types of hardware-assisted security

Software Attacks against TEEs

ATTACKING YOUR TRUSTED CORE: EXPLOITING TRUSTZONE ON ANDROID

PRESENTED BY

Di Shen

For years fingerprint scanning has been supported in many Android devices. Fingerprint scanning on ARM always needs an implementation of TrustZone. While we enjoy unlocking devices and paying by fingerprint, we also figure out these new features bring out some new attack surfaces. Attacking the kernel of Android or the secure world of TrustZone may be not impossible.

https://www.blackhat.com/us-15/briefings.html#attacking-your-trusted-core-exploiting-trustzone-on-android (2015)

02/05/2016

QSEE privilege escalation vulnerability and exploit (CVE-2015-6639)

In this blog post we'll discover and exploit a vulnerability which will allow us to gain code execution within Qualcomm's Secure Execution Environment (QSEE). I've responsibly disclosed this vulnerability to Google and it has been fixed - for the exact timeline, see the "Timeline" section below.

https://bits-please.blogspot.com/2016/05/qsee-privilege-escalation-vulnerability.html (2016)

30/06/2016

Extracting Qualcomm's KeyMaster Keys - Breaking Android Full Disk Encryption

After covering a TrustZone kernel vulnerability and exploit in the previous blog post, I thought this time it might be interesting to explore some of the implications of code-execution within the TrustZone kernel. In this blog post, I'll demonstrate how TrustZone kernel code-execution can be used to effectively break Android's **F**ull **D**isk **E**ncryption (**FDE**) scheme. We'll also see some of the inherent issues stemming from the design of Android's FDE scheme, even without any TrustZone vulnerability.

https://bits-please.blogspot.com/2016/06/extracting-qualcomms-keymaster-keys.html (2016)

Protection against software attacks

Novel hardware architectures

CHERI, fat pointers, …

Hardware extensions rolled out by processor vendors

x86**:**

- Memory Protection Keys (MPK)
- Memory Protection eXtensions (MPX)

ARM:

- Pointer Authentication (PA)
- Memory Tagging Extensions (MTE)
- Branch Target Indication (BTI)

How to utilize hardware-security primitives?

New hardware primitives are being rolled out

- Can we efficiently combine them to achieve new security properties?

How do different techniques compare?

- e.g., ARM PA and ShadowStack achieve similar security for return-address protection?

Understanding can help hardware vendors to choose which mechanisms to deploy

Intel Intel® 64 and IA-32 Architectures Software Developer Manuals (2019)
ARM Armv8.5-A Memory Tagging Extension (2019)

Forthcoming IEEE SP Special Issue

https://computer.org/digital-library/magazines/sp/call-for-papers-special-issue-on-hardware-assisted-security

IEEE Security and Privacy Magazine
Special Issue on Hardware-assisted Security
mid-to-late 2020

Submissions: Dec 22, 2019

Takeaways

TEEs have been around for more than two decades

Dominant design choice informed by cost and usability considerations

Unconditional trust in hardware-TEEs is no longer acceptable

TEEs are still useful: defense-in-depth, novel mitigations for TEE failure possible?

Other hardware-assisted security mechanisms to harden software are emerging

https://asokan.org/asokan/

@nasokan

Thanks to my students:
Lachlan J. Gunn, Hans Liljestrand, Thomas Nyman

Acknowledgments

Icons on Platform security for mobile devices and Dealing with TEE compromise made by those-icons, freepik, and Good Ware from www.flaticon.com licensed by CC 3.0 BY

Icons on Mobile TEEs: Motivation made by Good Ware from www.flaticon.com licensed by CC 3.0 BY

Nokia 6630 image by JotWu from en.wikipedia.org licensed by CC 3.0 BY

Web icon on title slide from material.io licensed by Apache 2.0