Hardening application security with SGX

●●●

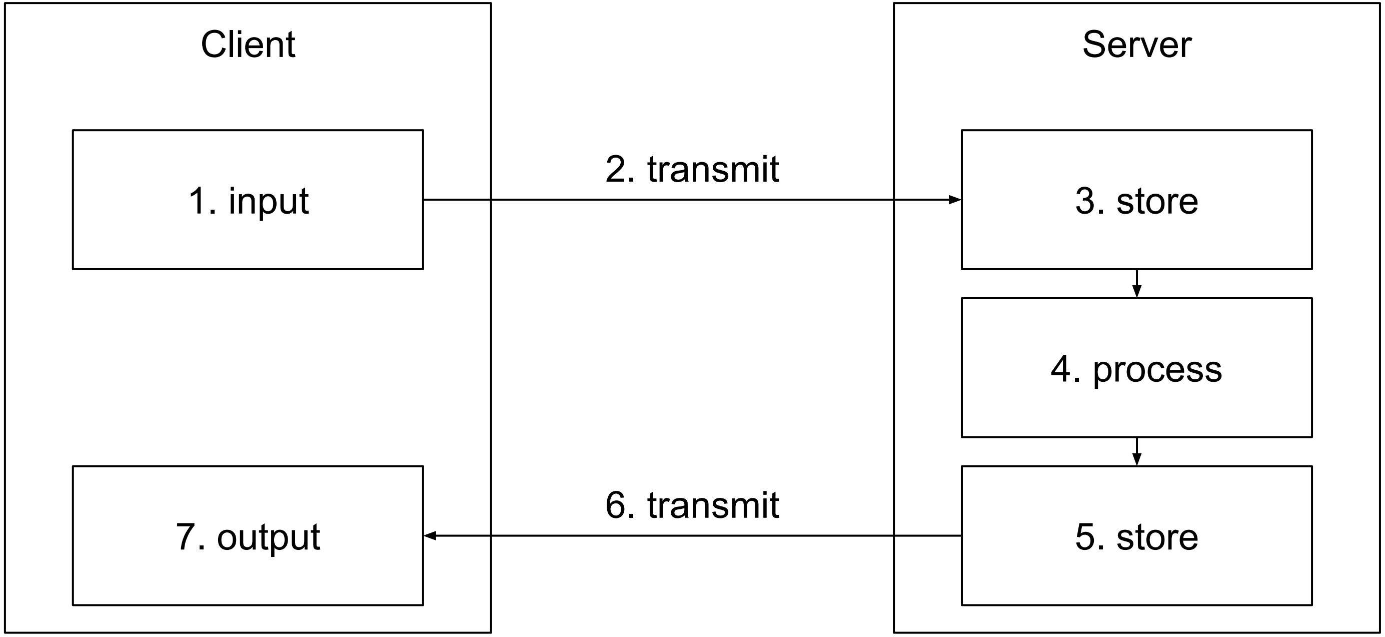
Introductory Presentation, Master’s Thesis

Operating Systems and Middleware Group, HPI

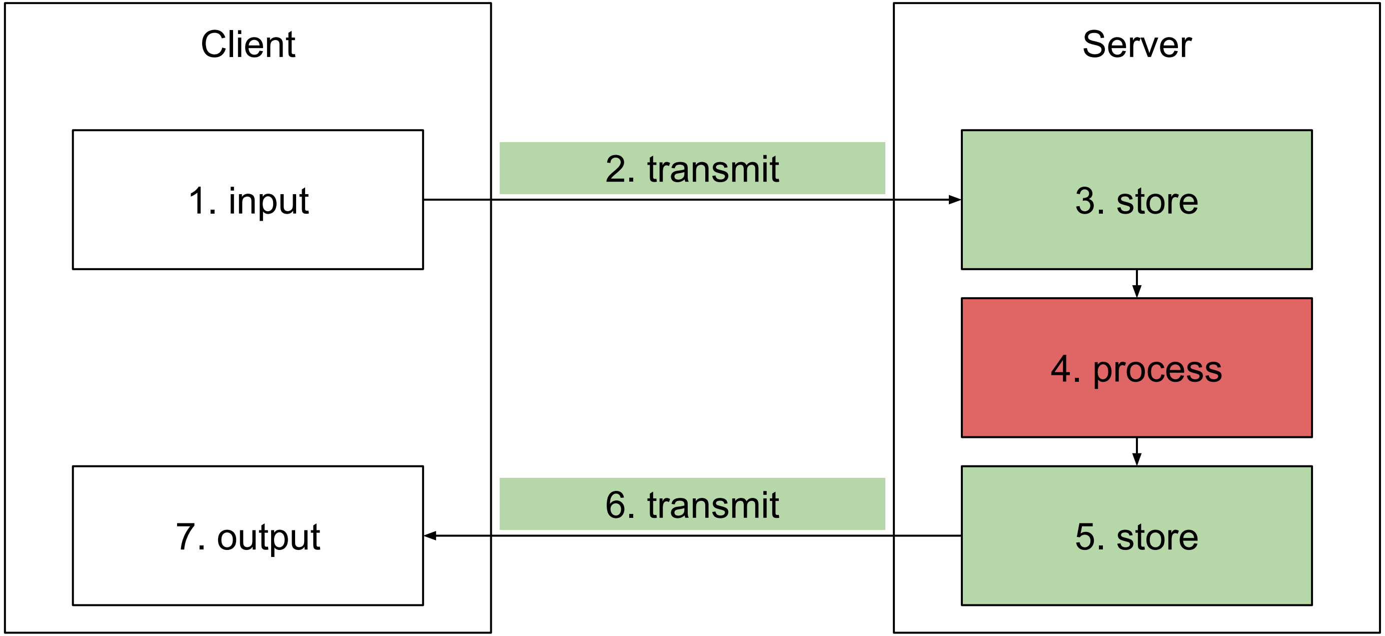
09.05.2017

Fredrik Teschke

Introduction



Introduction



Client

1. input

2. transmit

Server

3. store

4. process

5. store

6. transmit

7. output

Overview

Background

Related Work

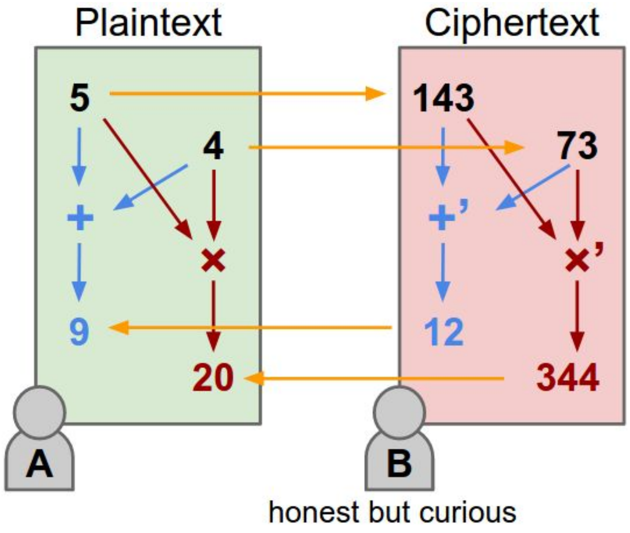
My Thesis

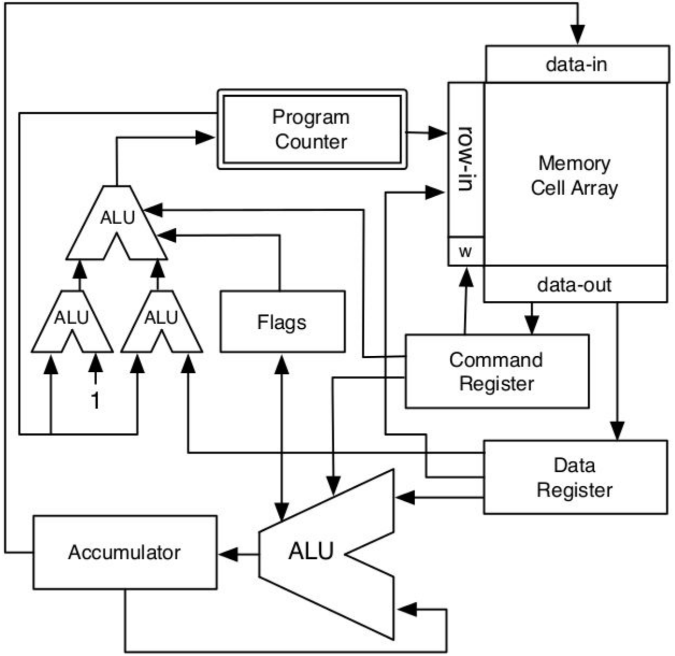
- scope of thesis
- approaches
- case studies

Michael Brenner, Jan Wiebelitz et al. „A smart-gentry based software system for secret program execution“.

Secure Remote Computing

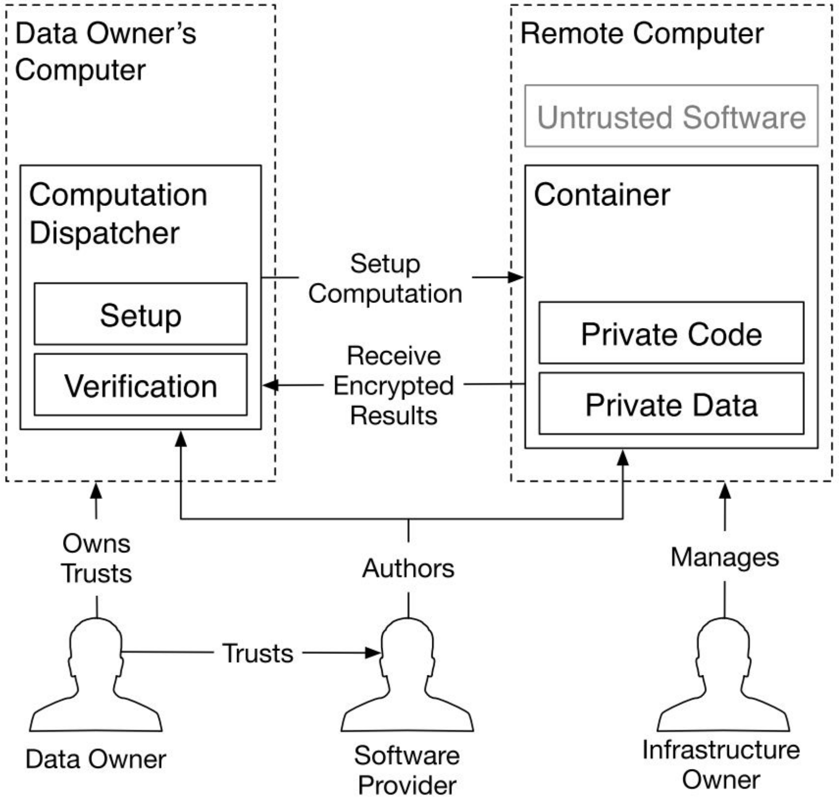
Fully Homomorphic Encryption





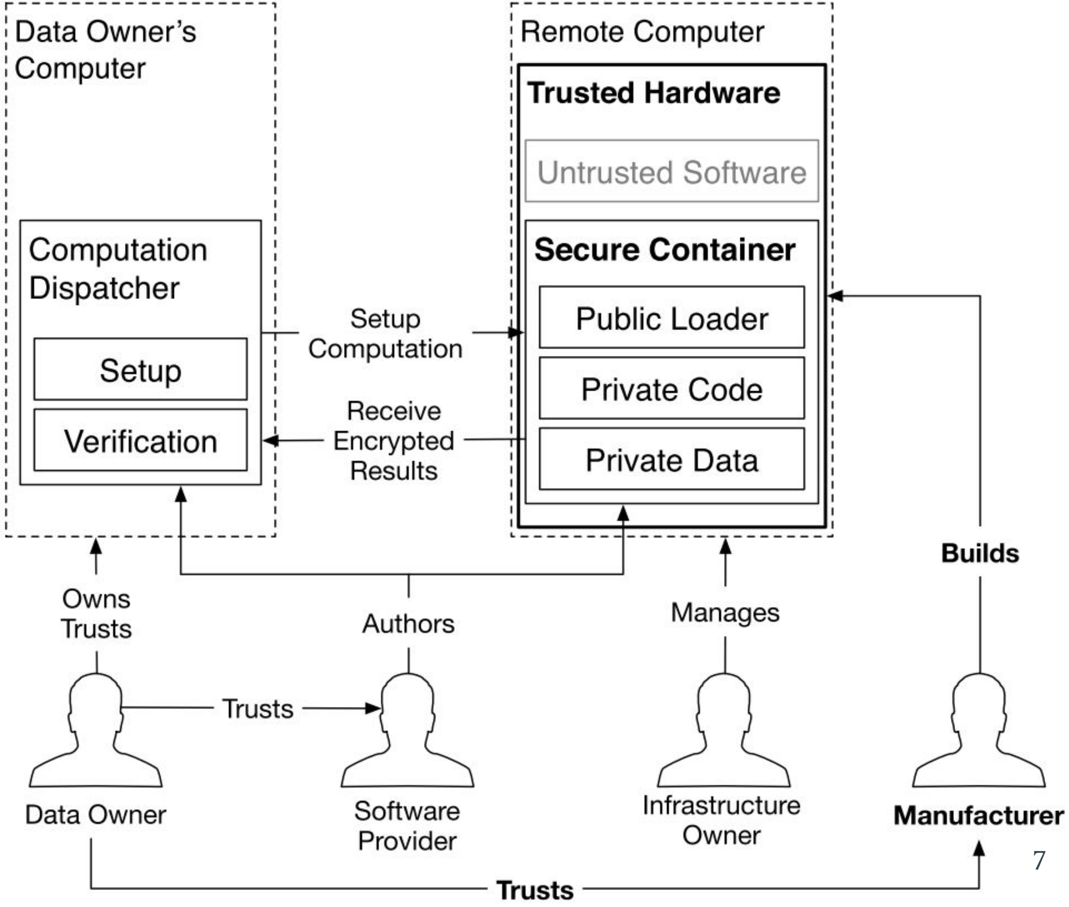
Victor Costan and Srinivas Devadas. „Intel SGX Explained“.

Secure Remote Computing



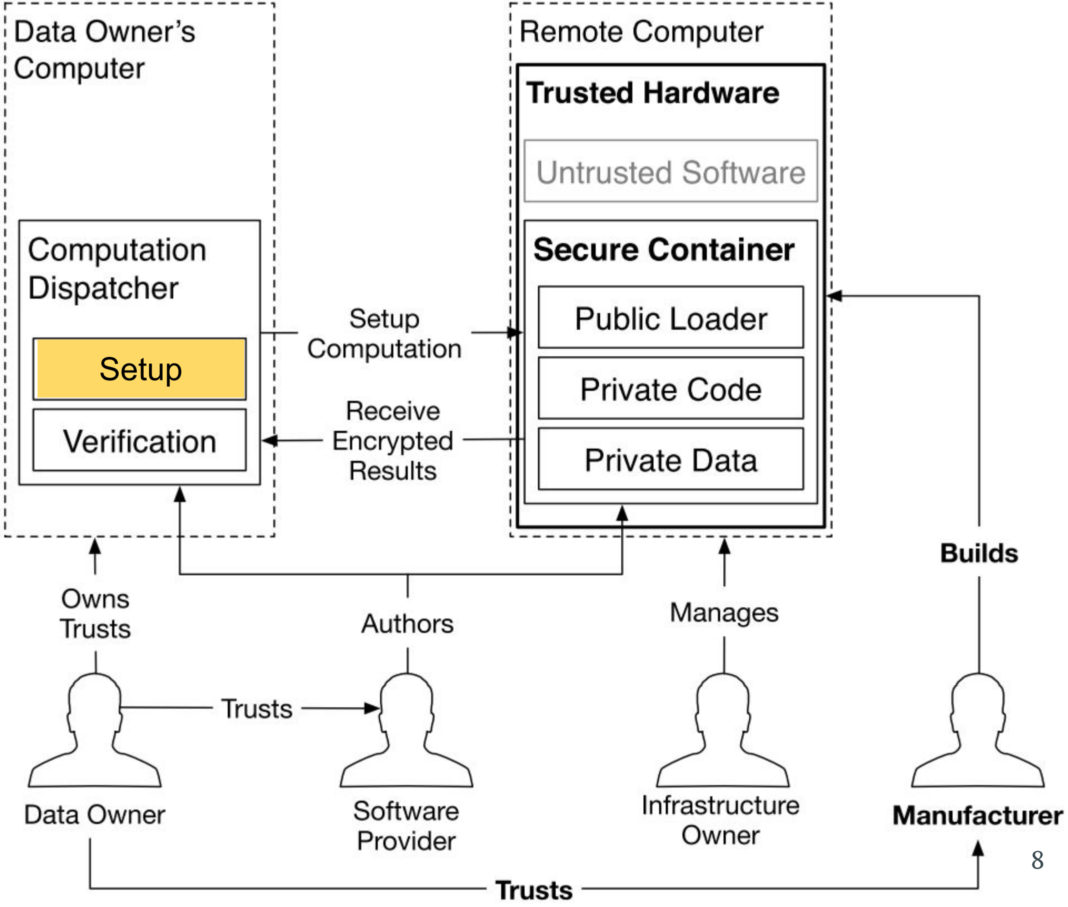
Victor Costan and Srinivas Devadas. „Intel SGX Explained“.

Trusted Computing



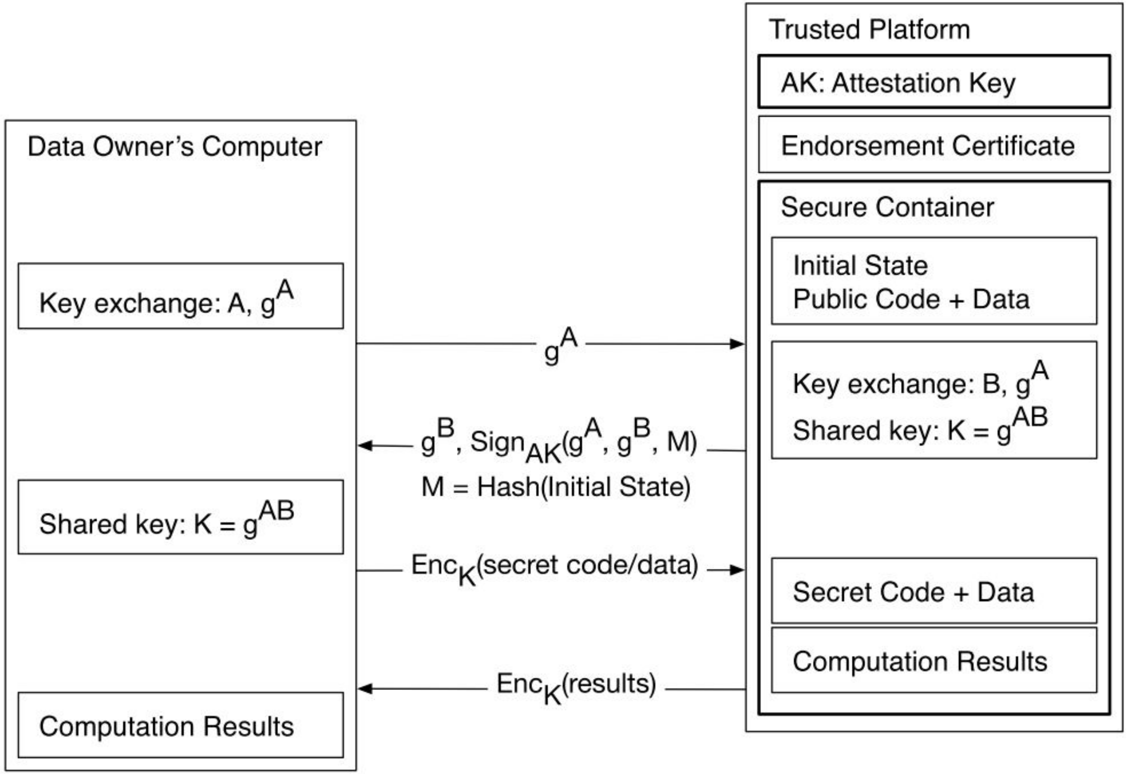
Victor Costan and Srinivas Devadas. „Intel SGX Explained“.

Trusted Computing

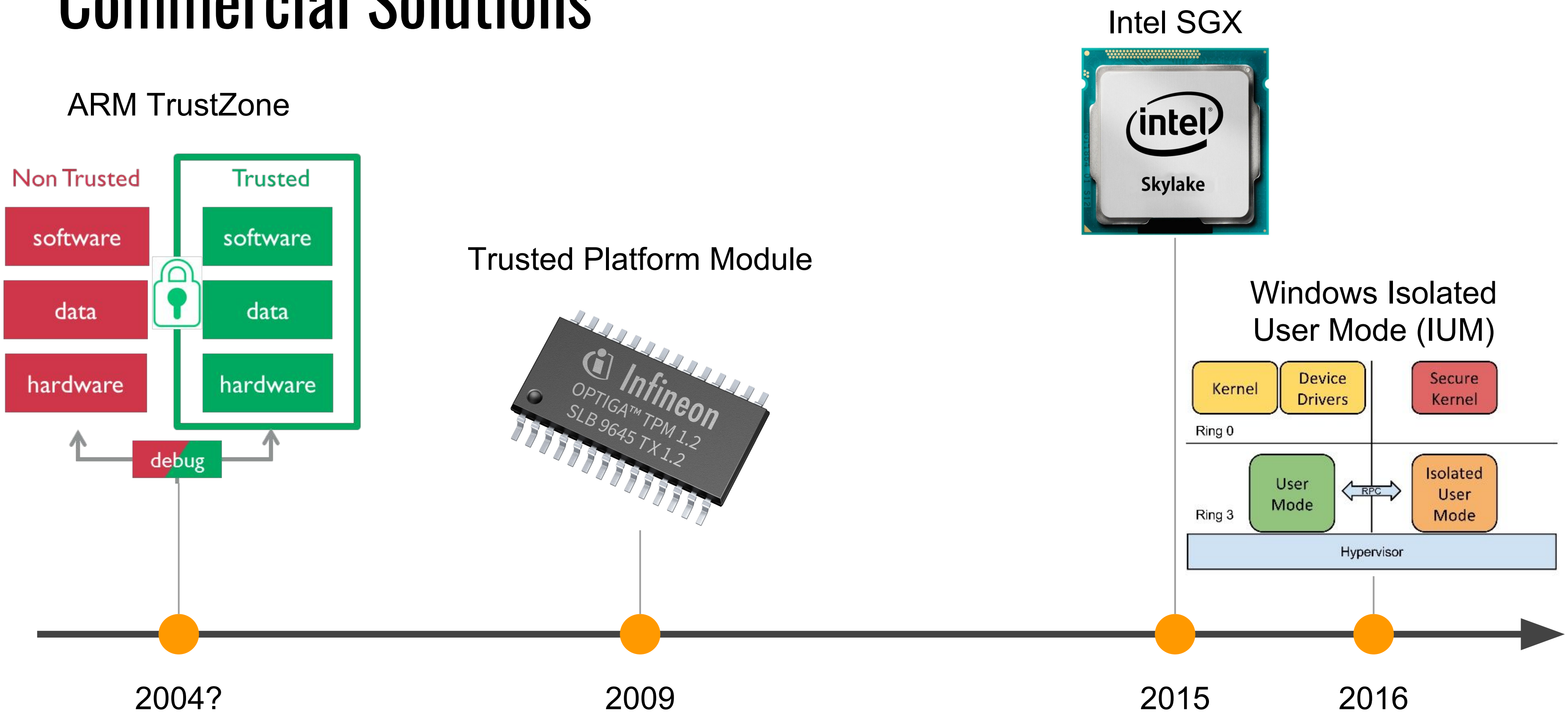


Victor Costan and Srinivas Devadas. „Intel SGX Explained“.

Software Attestation



Commercial Solutions



ARM TrustZone

Non Trusted

software

data

hardware

Trusted

software

data

hardware

debug

Trusted Platform Module

Intel SGX

Windows Isolated User Mode (IUM)

Kernel

Device Drivers

Secure Kernel

Ring 0

User Mode

RPC

Isolated User Mode

Ring 3

Hypervisor

2004?

2009

2015

2016

Commercial Solutions

tech-nology	concept	granularity	TCB	limitations	security	# of TEEs
TPM	security module (coprocessor) as root of trust for measured launch	entire software stack (BIOS, OS, all apps)	entire software stack + hardware	TPM is slow, entire software stack measured (delicate)	system is in guaranteed state (remote attestation)	1
Intel TXT, AMD SVM	late loading of trusted app (requires TPM)	trusted app (typically VMM)	trusted app + hardware (+ loaded VMs)	late load is “expensive”, TPM is slow, (entire VM still in TCB)	trusted app is in guaranteed state (remote attestation)	1
TrustZone	system split into normal and secure worlds via bus signal (incl. memory, peripherals)	entire secure world software stack	secure world software + world switching SW + hardware	single secure world	normal world cannot access secure world, secure world boots first	1
Windows IUM	software (hypervisor) based version of TrustZone	applications	hypervisor + secure Kernel	limited to Windows, Hypervisor based, isolation of secure apps	no attestation, hypervisor enforces separation	1
SGX	secure memory enclaves within process	enclave (security sensitive part of application)	enclave code + processor package	licensing, security issues	enclave is in guaranteed state (remote attestation), enclave memory is protected	n

Research Landscape: Enclaves (like SGX)

	HW[1]	VMM[2]-based	attestation	data sealing	trusted I/O	parallelism	comments
SGX[a]	SGX		X	X		X	CPU extensions, PAL memory encrypted in DRAM
Oasis[b]	CPU		X	X			like SGX, but without DRAM encryption use cache-as-ram for secrets
TLR[c]	TZ[3]		[4]	X		X	.NET containers in secure world no trusted I/O: don't want drivers in TCB
Flicker[d]	TPM		X	X			PAL runs in secure execution mode (Intel TXT/AMD SVM way to dynamically establish secure environment) -> slow/limited
TrustVisor[e]	TPM	X	X	X		X	PALs and legacy each in own virtual guest memory provides virtual TPM to each PAL
Fides[f]	TPM	X					secure kernel runs PALs (shared memory between PALs) like a software version of TrustZone

[1] hardware foundation

[2] virtual machine monitor / hypervisor

[3] TrustZone

[4] possible, but not implemented

Research Landscape: Applications (like IUM)

	HW[1]	VMM[2]-based	attestation	data sealing	trusted I/O	parallelism	comments
GP TEE	-			X	X	X	reference model, maps well to TrustZone implementations exist for Android (Trustonic)
Haven	SGX		X	X		X	unmodified applications with library OS in enclave protects from Iago attacks by OS
InkTag	-	X				X	trusted hypervisor monitors OS
MiniBox	TPM	X	X	X		X	TrustVisor for entire application (also provides virtual TPM) adds sandbox: OS protected as well

[1] hardware foundation

[2] virtual machine monitor / hypervisor

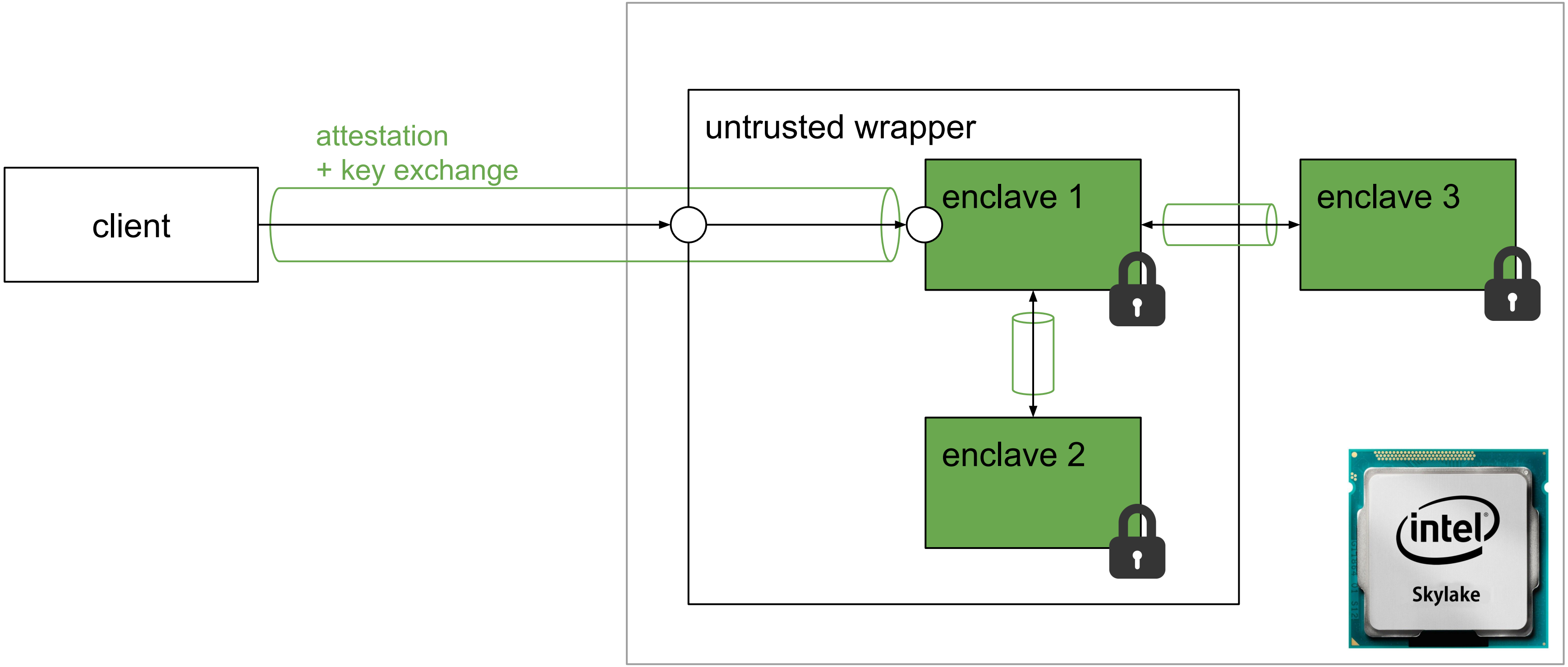
[3] GlobalPlatform reference model for Trusted Execution Environments: describes TEE internal and external API for building trusted applications

Research Landscape: Virtual Machines (like TPM/TXT)

	HW[1]	VMM[2]-based	attestation	data sealing	trusted I/O	parallelism	comments
CloudVisor	TPM	X	(X)[3]			X	small monitor underneath VMM (nested virtualization) enforces isolation
Nova		X				X	µ-hypervisor built from scratch decomposed like kernel, principle of least privilege
NoHype	[4]	(X)			(X)	X	static resource allocation: 1 VM per core, static memory slice no hypervisor interaction while executing needs hardware features that no product offers
vTPM	TPM	X	X	X		X	provide virtual TPM to each VM vTPM can be stored and migrated

[1] hardware foundation
[2] virtual machine monitor / hypervisor
[3] attestation only of CloudVisor monitor, not of individual VMs
[4] hardware virtualization support by CPU (extended page tables, VMM ring -1), devices with virtualization support

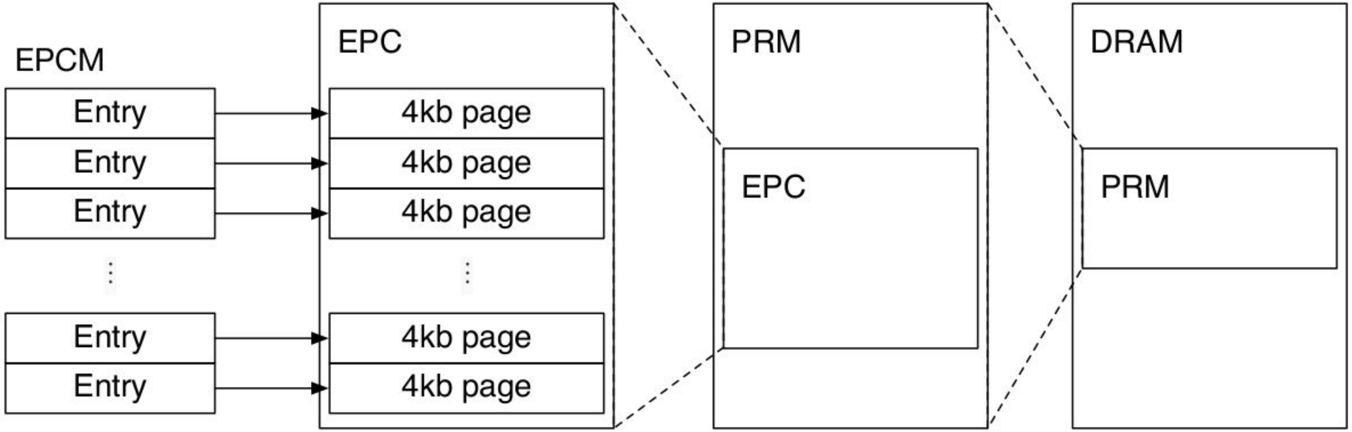
Intel SGX Programming Model



Victor Costan and Srinivas Devadas. „Intel SGX Explained“.

Intel SGX

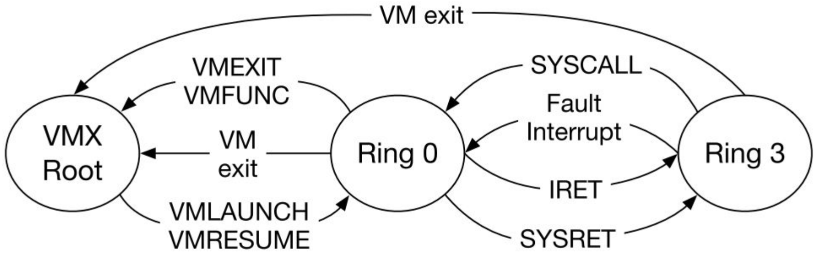
- instruction set extension (mostly microcode)
- OS schedules resources

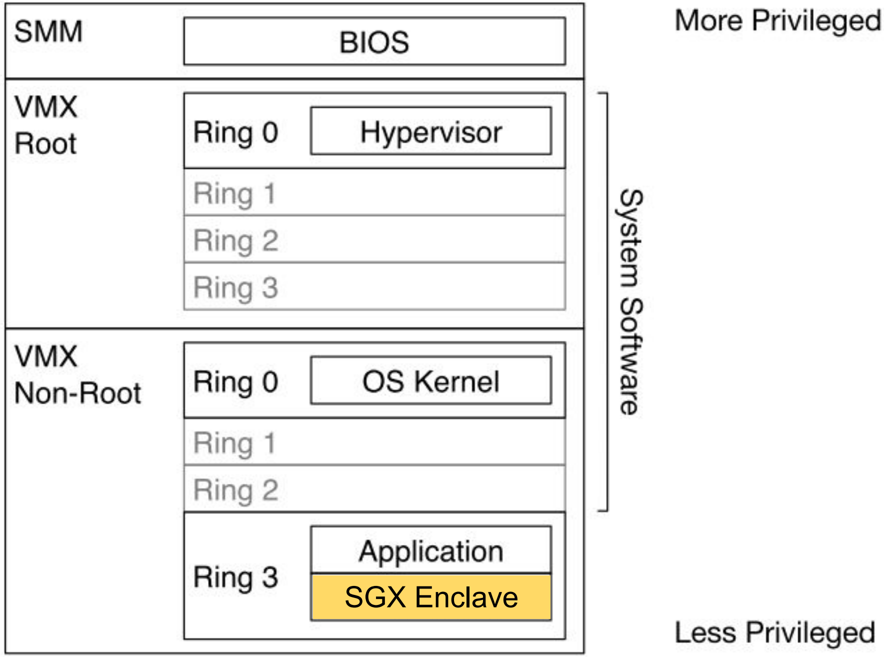


Victor Costan and Srinivas Devadas. „Intel SGX Explained“.

Intel SGX

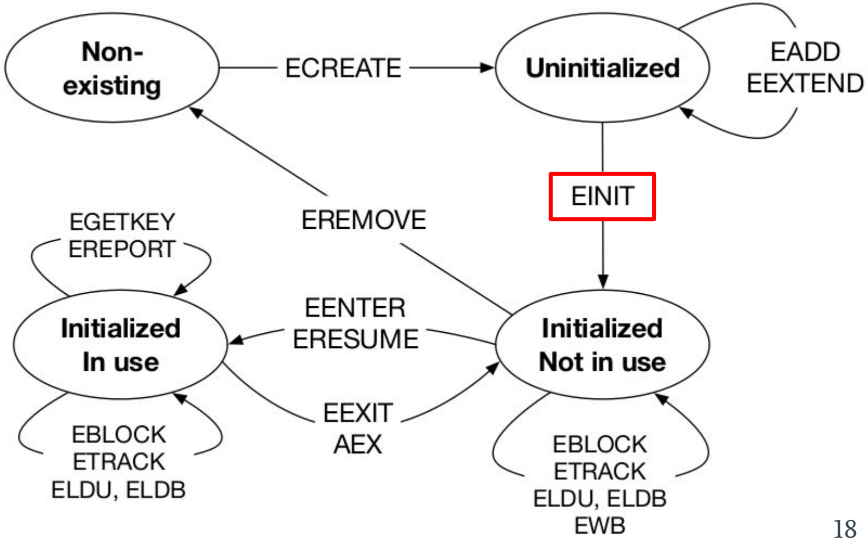
- instruction set extension (mostly microcode)
- OS schedules resources
- orthogonal to existing virtualization
    - x86 privilege levels
    - virtual memory

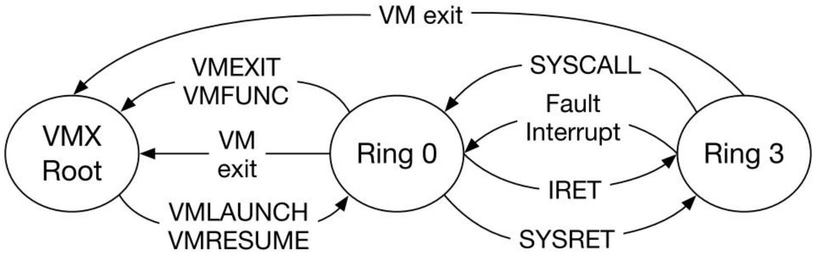




Intel SGX

- instruction set extension (mostly microcode)
- OS schedules resources
- orthogonal to existing virtualization
  - x86 privilege levels
  - virtual memory





Intel SGX SDK

- C, C++
- interface: ecalls, ocalls
    - .edl -> stub (pointer handling)
- no syscalls inside enclave

```
// demo.edl
enclave {
     trusted {
          void get_secret([out] secret_t* secret);
          void get_secret([user_check] secret_t* secret);
     };
     untrusted {
          void dump_secret([in] const secret_t* secret);
     };
};
```

Related Work

- Secure Databases
- Applications secured with SGX
- Application Partitioning Design Space

Arvind Arasu, Ken Eguro et al. „Querying Encrypted Data (Tutorial)“

Secure Databases Design Space

		encryption scheme		
		non homomorphic	**partially homomorphic**	**fully homomorphic**
secure location	**-**		CryptDB	?
	client	Arx	Monomi	
	co-processor		TrustedDB	
	FPGA		Cipherbase	
	SGX	?	?	

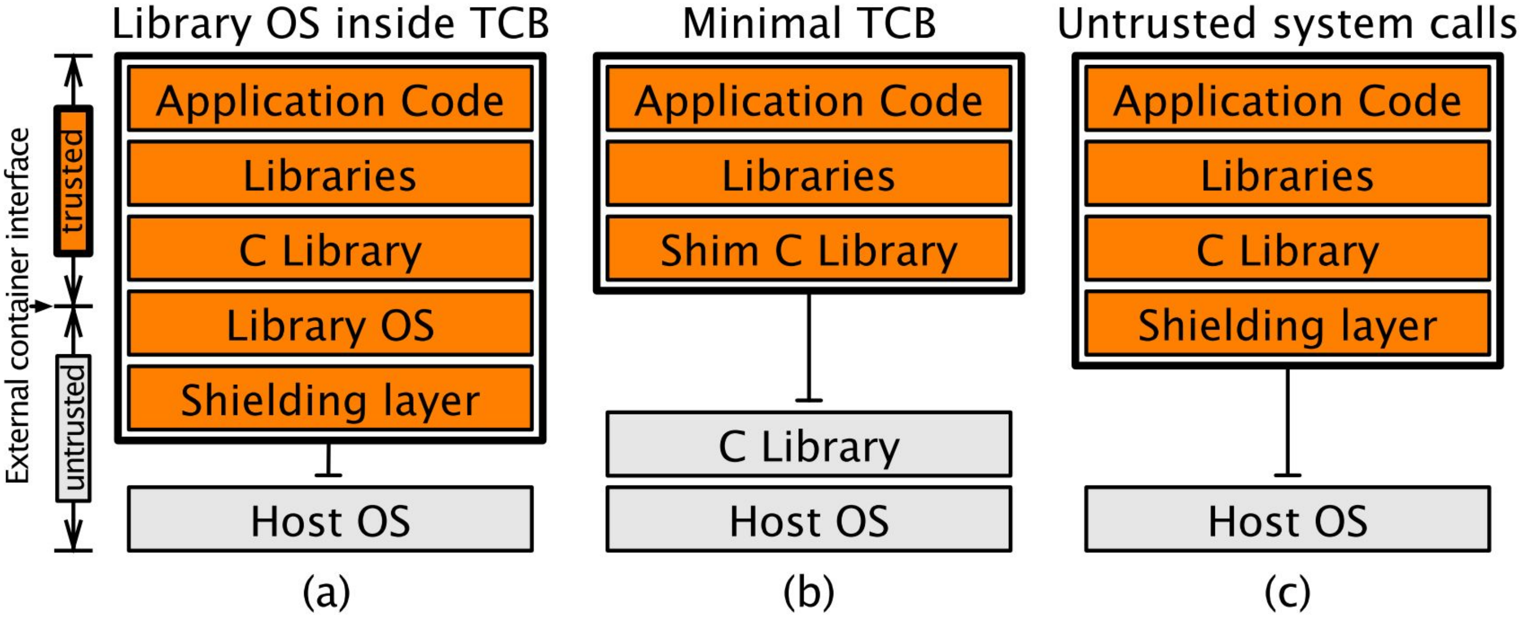
https://intelledger.github.io/introduction.html#proof-of-elapsed-time-poet
Felix Schuster, Manuel Costa et al. „VC3: Trustworthy data analytics in the cloud using SGX“.
Stefan Brenner, Colin Wulf et al. „SecureKeeper: Confidential ZooKeeper using Intel SGX“.
Sergei Arnautov, Bohdan Trach et al. „SCONE: Secure linux containers with Intel SGX“.

Intel SGX Applications

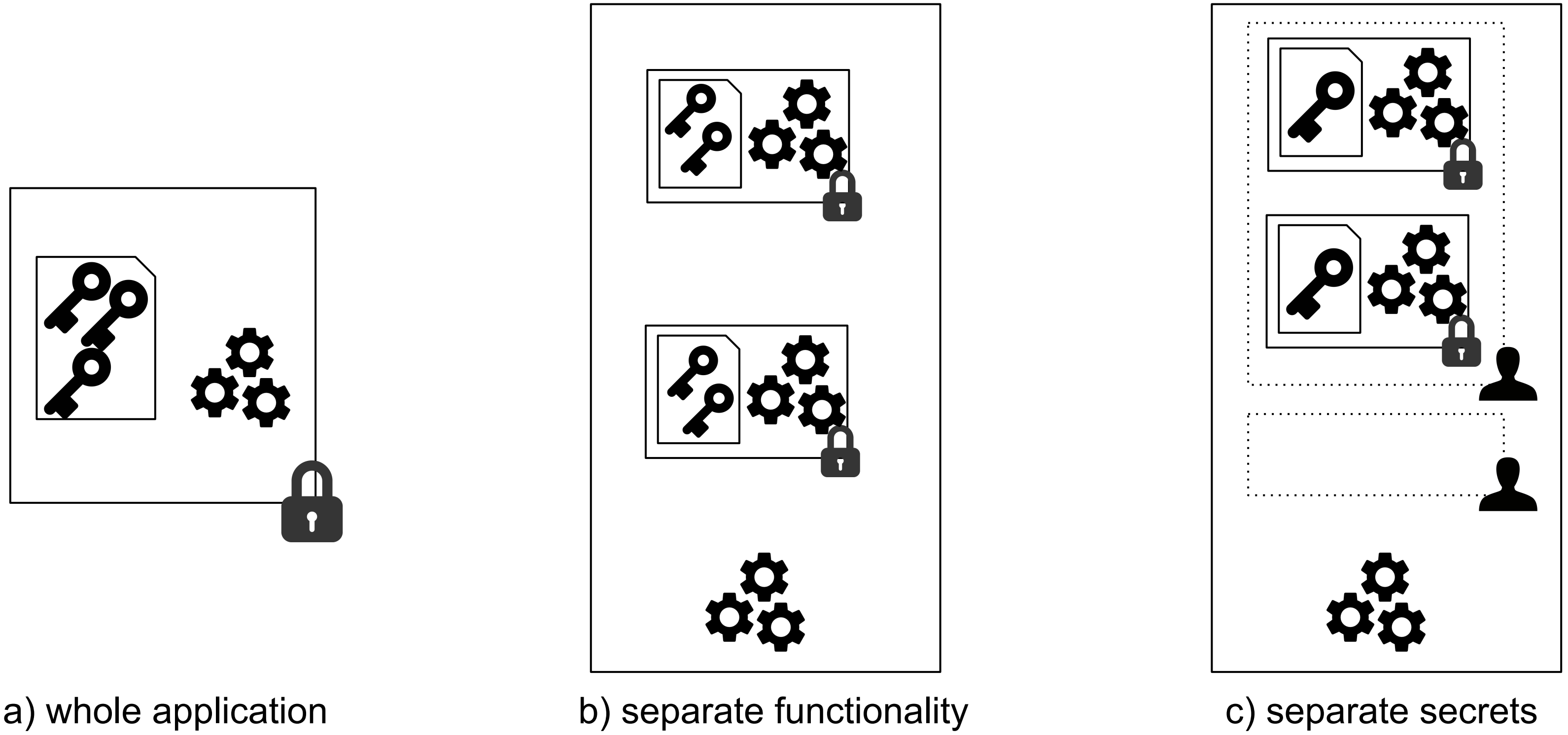
- Proof of elapsed Time
  - Blockchain
- Microsoft VC3
  - Verifiable Confidential Cloud Computing
  - in-band encrypted MapReduce
- Secure Zookeeper
- SCONE: Secure Linux Containers
  - user-level threading, syscall service workers
  - musl libc, transparent shielding

Sergei Arnautov, Bohdan Trach et al. „SCONE: Secure linux containers with Intel SGX“.

Enclave Interface Design Space



Application Separation Design Space



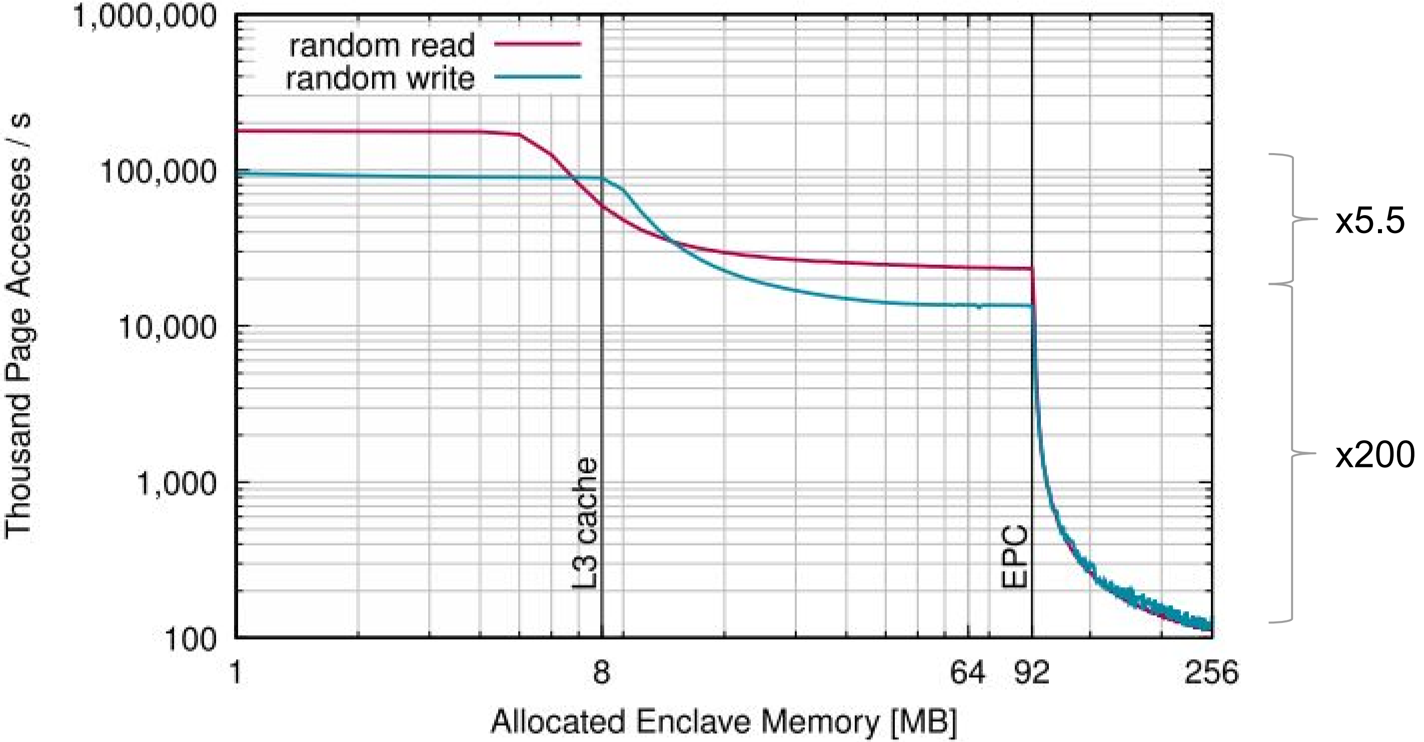
a) whole application

b) separate functionality

c) separate secrets

Stefan Brenner, Colin Wulf et al. „SecureKeeper: Confidential ZooKeeper using Intel SGX“.

Application Memory Management



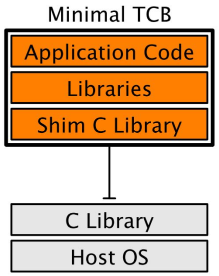
Scope

- comparison of available solutions (qualitative)
- design decisions (SGX)
- case study (DBMS)

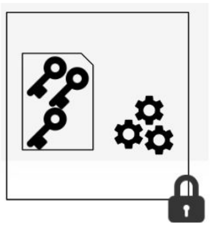
Case Study: KissDB

- key value store
  - vanilla C
  - ~400 SLOC
  - test
- use case for SGX Lib https://github.com/ftes/sgx-lib
  - rapid migration helper (libc shim)
- missing
  - attestation + secure communication
  - extract only security critical functionality into enclave?

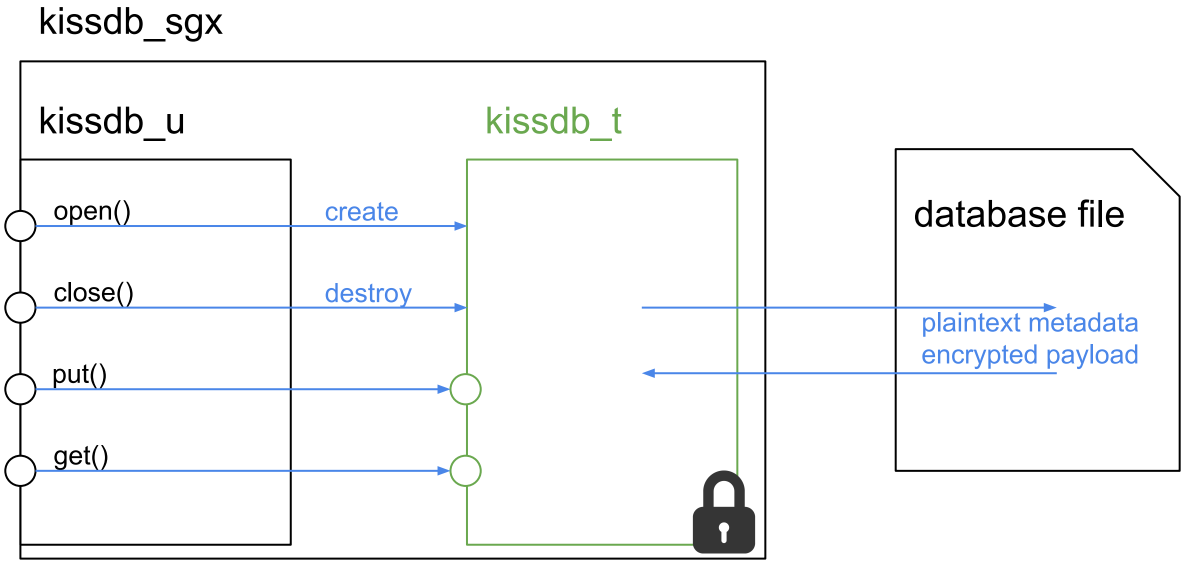
Case Study: KissDB



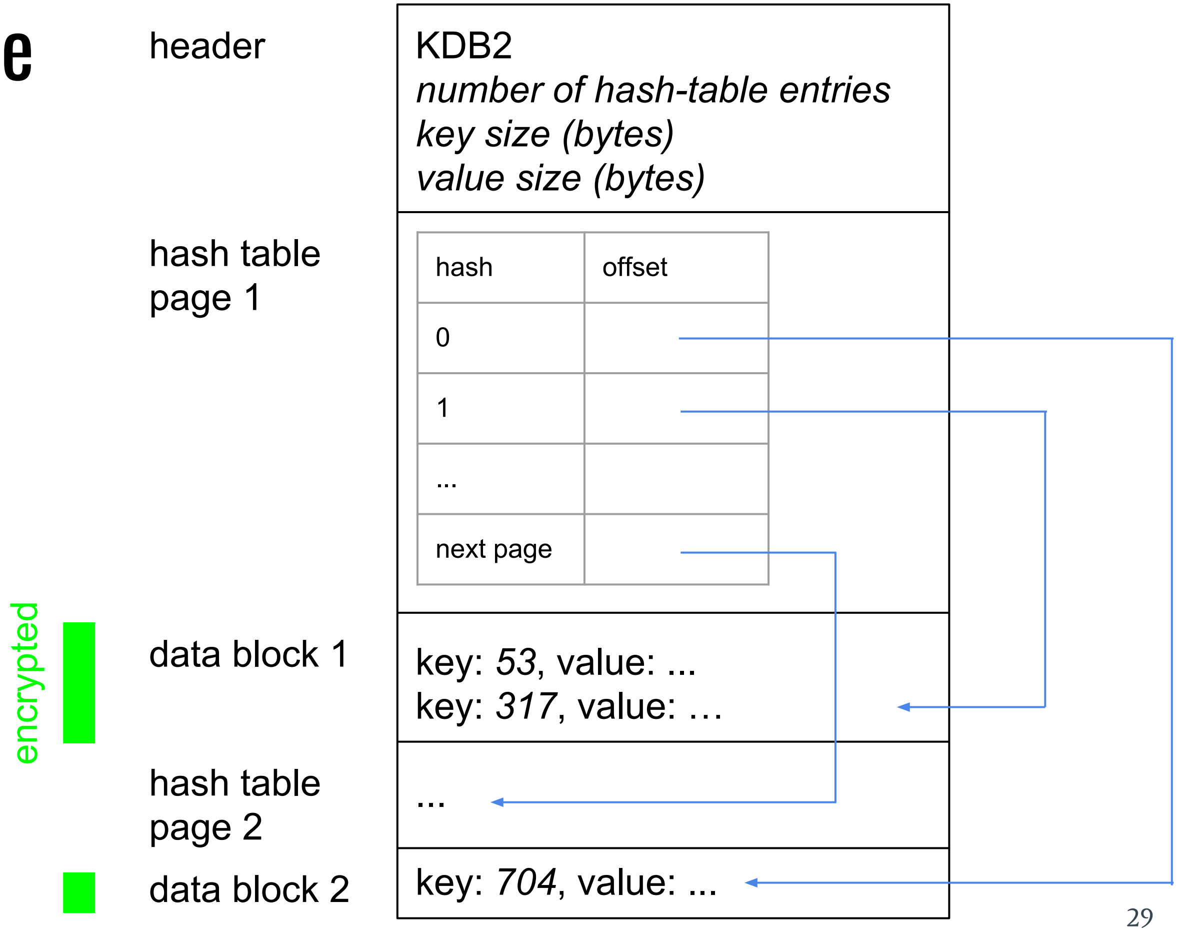
(b)



a) whole application



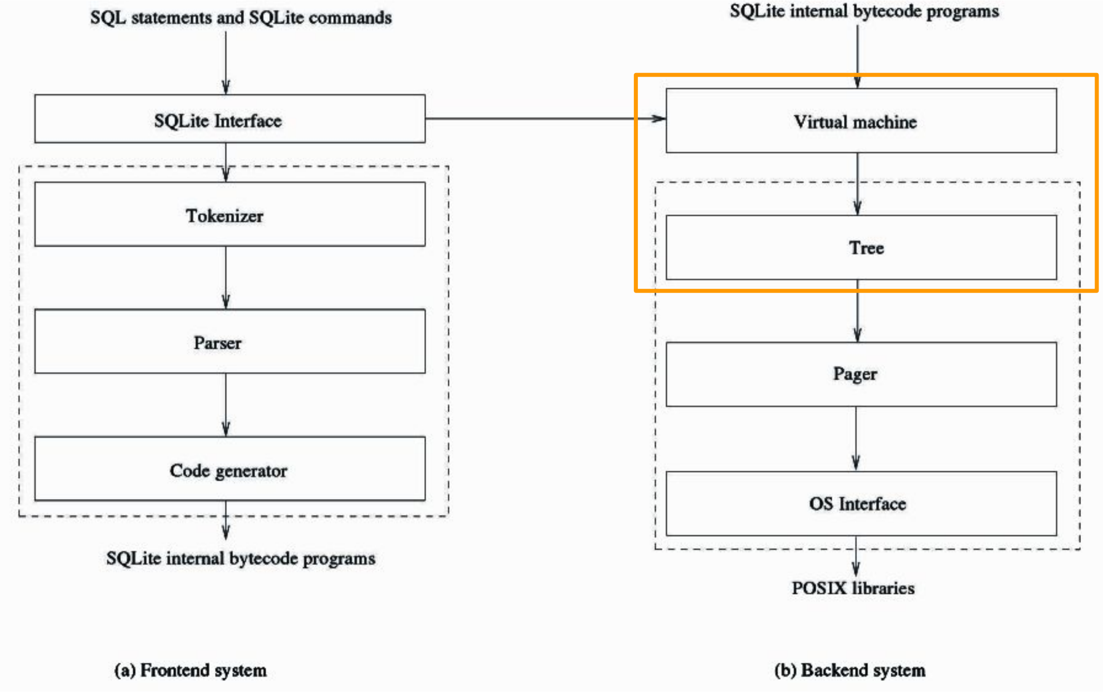
Case Study: KissDB File



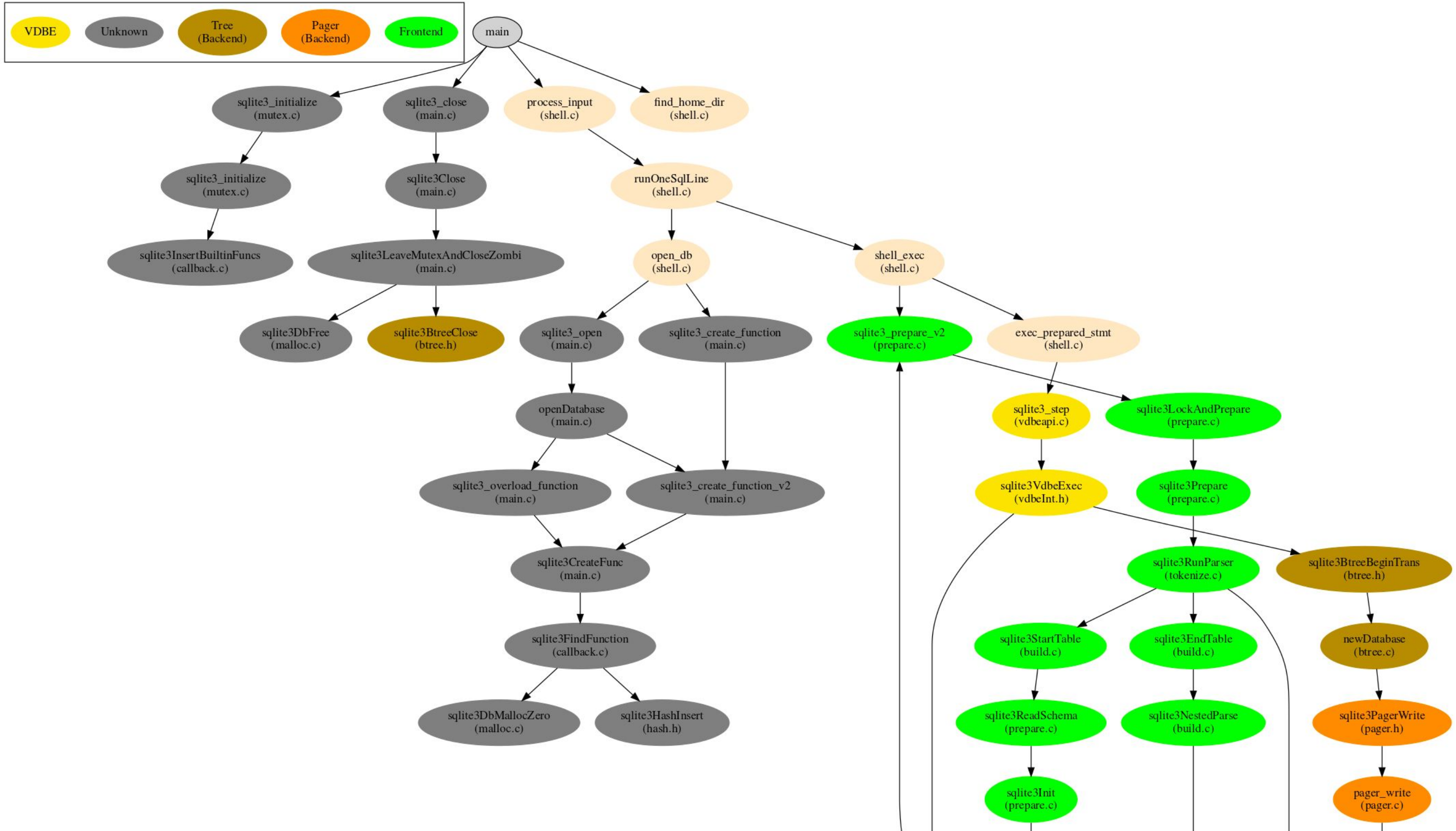
Sibsankar Haldar. „SQLite Database System Design and Implementation“

SQLite

- 113K SLOC
- hardening approaches
  - Virtual Machine
  - User Defined Functions



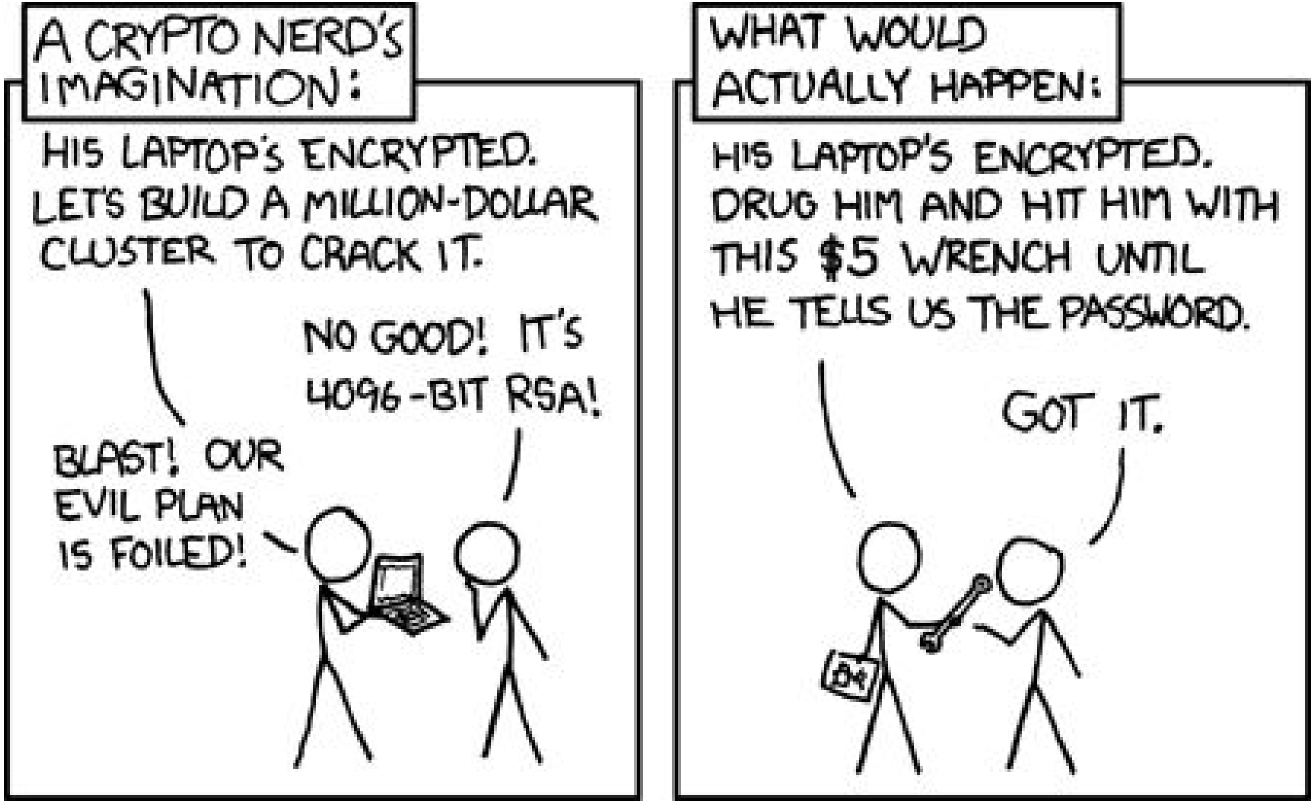
SQLite: Extracting the Virtual Machine


VDBE
Unknown
Tree (Backend)
Pager (Backend)
Frontend
main
sqlite3_initialize (mutex.c)
sqlite3_close (main.c)
process_input (shell.c)
find_home_dir (shell.c)
sqlite3_initialize (mutex.c)
sqlite3Close (main.c)
runOneSqlLine (shell.c)
sqlite3InsertBuiltinFuncs (callback.c)
sqlite3LeaveMutexAndCloseZombi (main.c)
open_db (shell.c)
shell_exec (shell.c)
sqlite3DbFree (malloc.c)
sqlite3BtreeClose (btree.h)
sqlite3_open (main.c)
sqlite3_create_function (main.c)
sqlite3_prepare_v2 (prepare.c)
exec_prepared_stmt (shell.c)
openDatabase (main.c)
sqlite3_step (vdbeapi.c)
sqlite3LockAndPrepare (prepare.c)
sqlite3_overload_function (main.c)
sqlite3_create_function_v2 (main.c)
sqlite3VdbeExec (vdbeInt.h)
sqlite3Prepare (prepare.c)
sqlite3CreateFunc (main.c)
sqlite3RunParser (tokenize.c)
sqlite3BtreeBeginTrans (btree.h)
sqlite3FindFunction (callback.c)
sqlite3StartTable (build.c)
sqlite3EndTable (build.c)
newDatabase (btree.c)
sqlite3DbMallocZero (malloc.c)
sqlite3HashInsert (hash.h)
sqlite3ReadSchema (prepare.c)
sqlite3NestedParse (build.c)
sqlite3PagerWrite (pager.h)
sqlite3Init (prepare.c)
pager_write (pager.c)

Conclusion

- timeline
    - implementation completed
- if there was more time
    - implement SQLite approach
    - benchmarks
    - SGX + partially homomorphic encryption
    - attestation

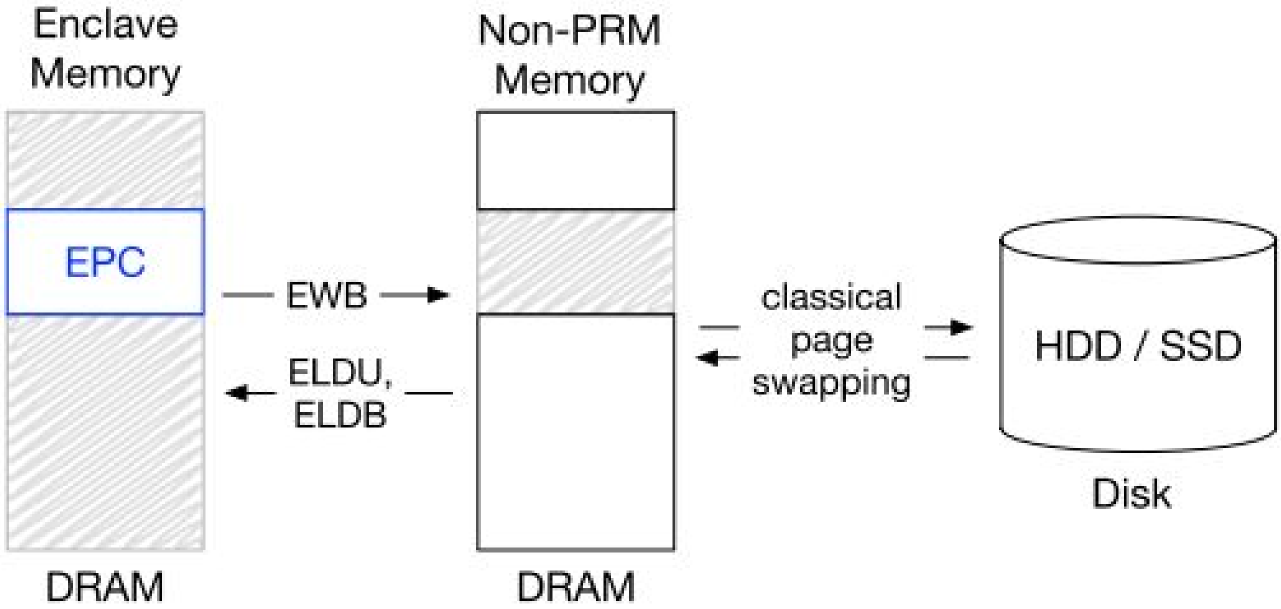
https://xkcd.com/538/


A CRYPTO NERD'S IMAGINATION:
HIS LAPTOP'S ENCRYPTED. LET'S BUILD A MILLION-DOLLAR CLUSTER TO CRACK IT.
NO GOOD! IT'S 4096-BIT RSA!
BLAST! OUR EVIL PLAN IS FOILED!
WHAT WOULD ACTUALLY HAPPEN:
HIS LAPTOP'S ENCRYPTED. DRUG HIM AND HIT HIM WITH THIS $5 WRENCH UNTIL HE TELLS US THE PASSWORD.
GOT IT.
$4

Backup

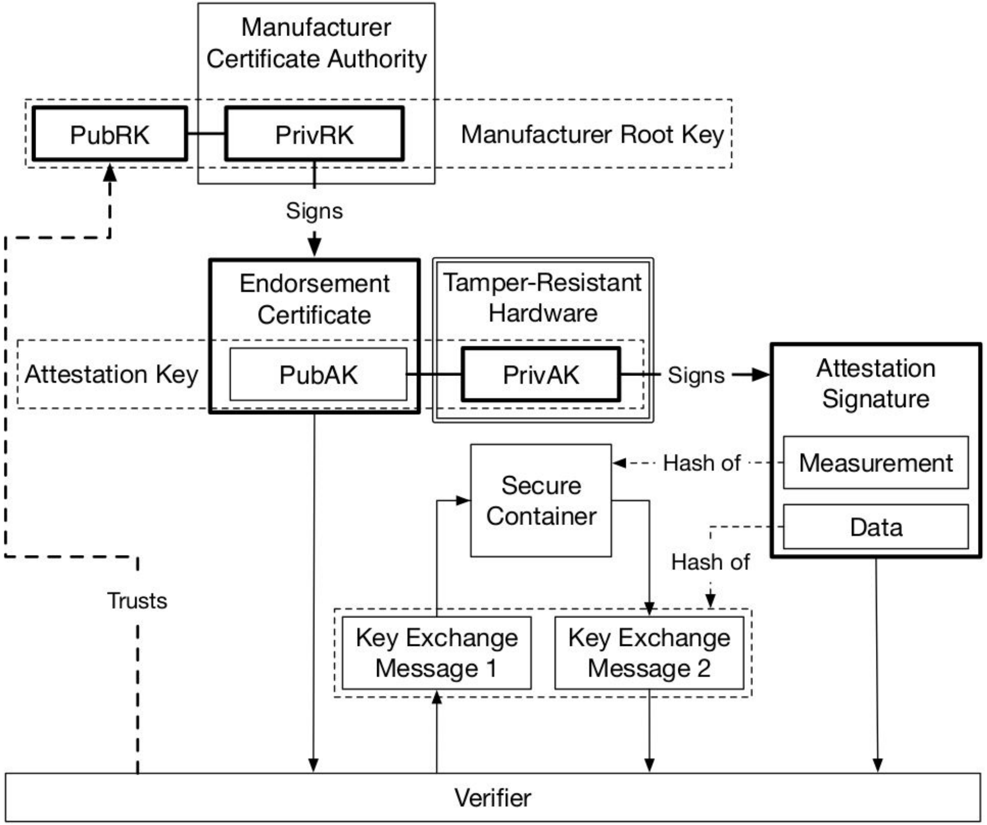
Victor Costan and Srinivas Devadas. „Intel SGX Explained“.

Intel SGX Two Stage EPC Paging



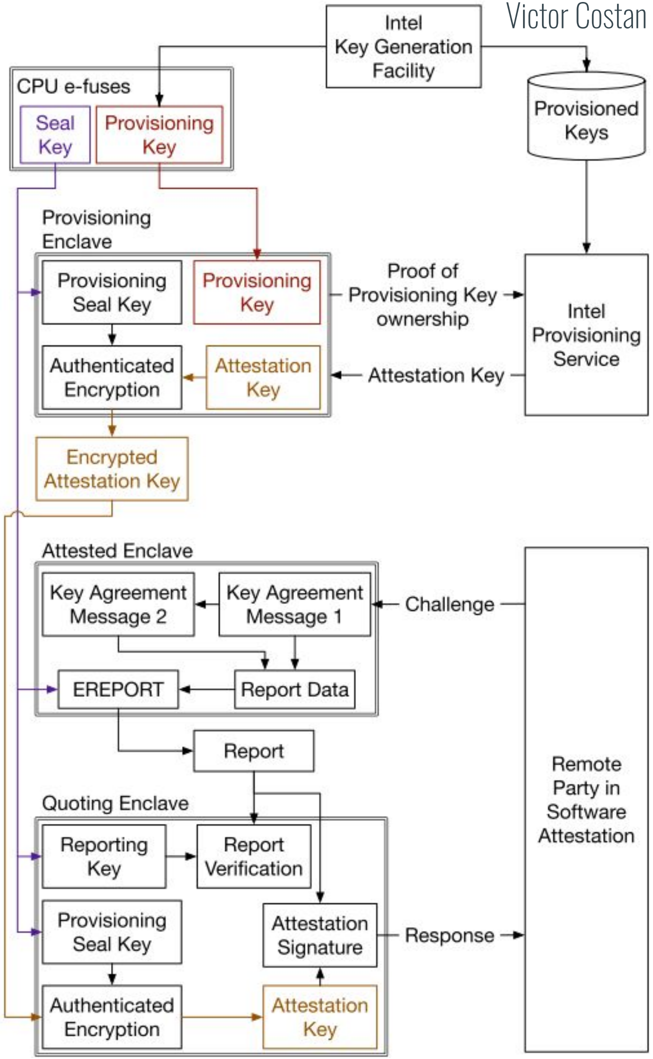
Victor Costan and Srinivas Devadas. „Intel SGX Explained“.

Software Attestation Chain of Trust

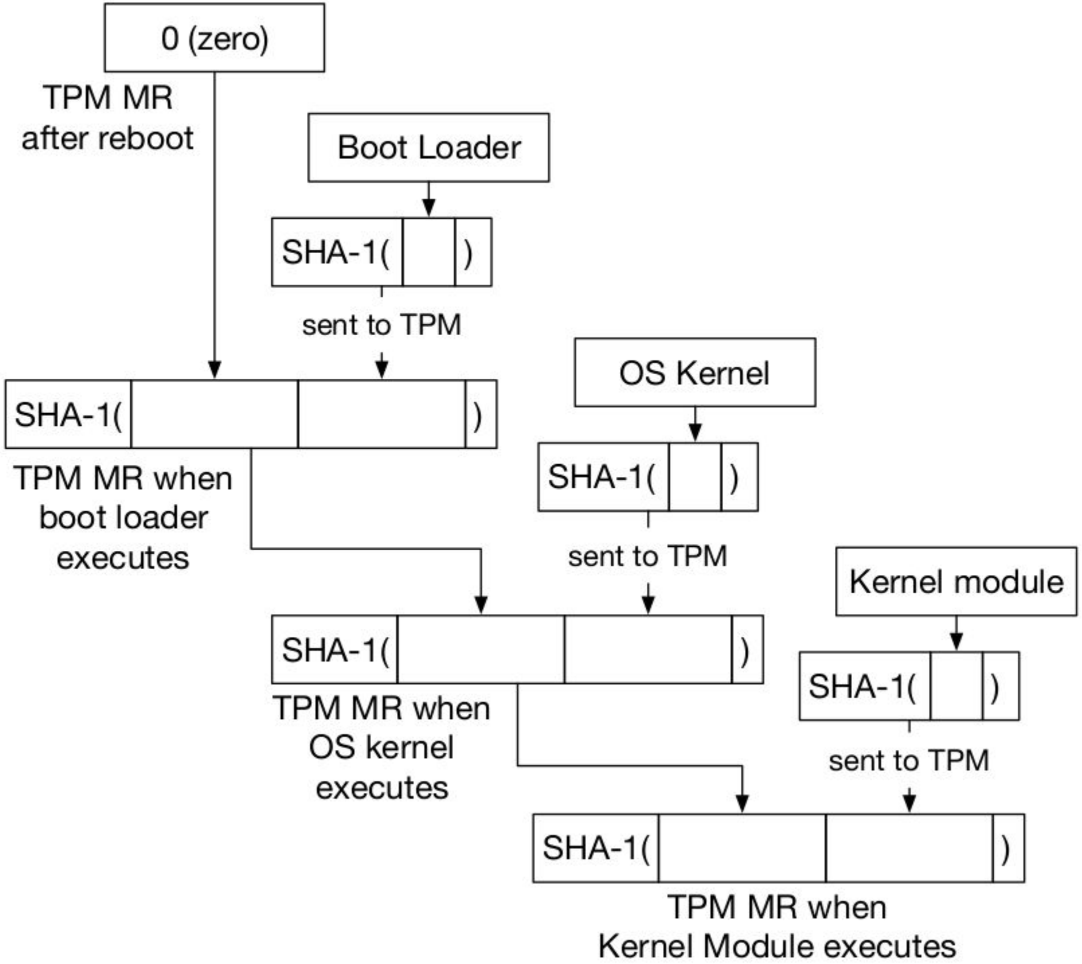


Victor Costan and Srinivas Devadas. „Intel SGX Explained“.

SGX Remote Attestation

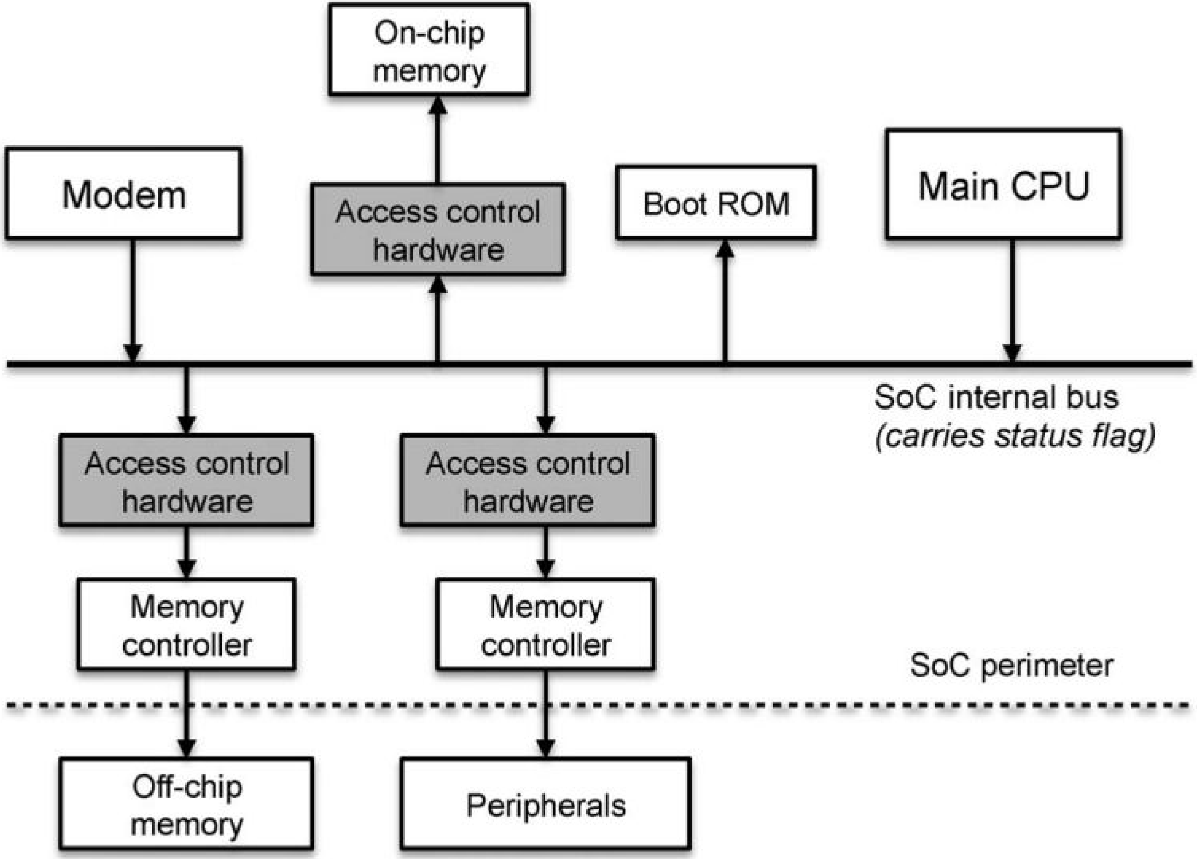


TPM SRTM


0 (zero)
TPM MR after reboot
Boot Loader
SHA-1( )
sent to TPM
SHA-1( )
TPM MR when boot loader executes
OS Kernel
SHA-1( )
sent to TPM
SHA-1( )
TPM MR when OS kernel executes
Kernel module
SHA-1( )
sent to TPM
SHA-1( )
TPM MR when Kernel Module executes

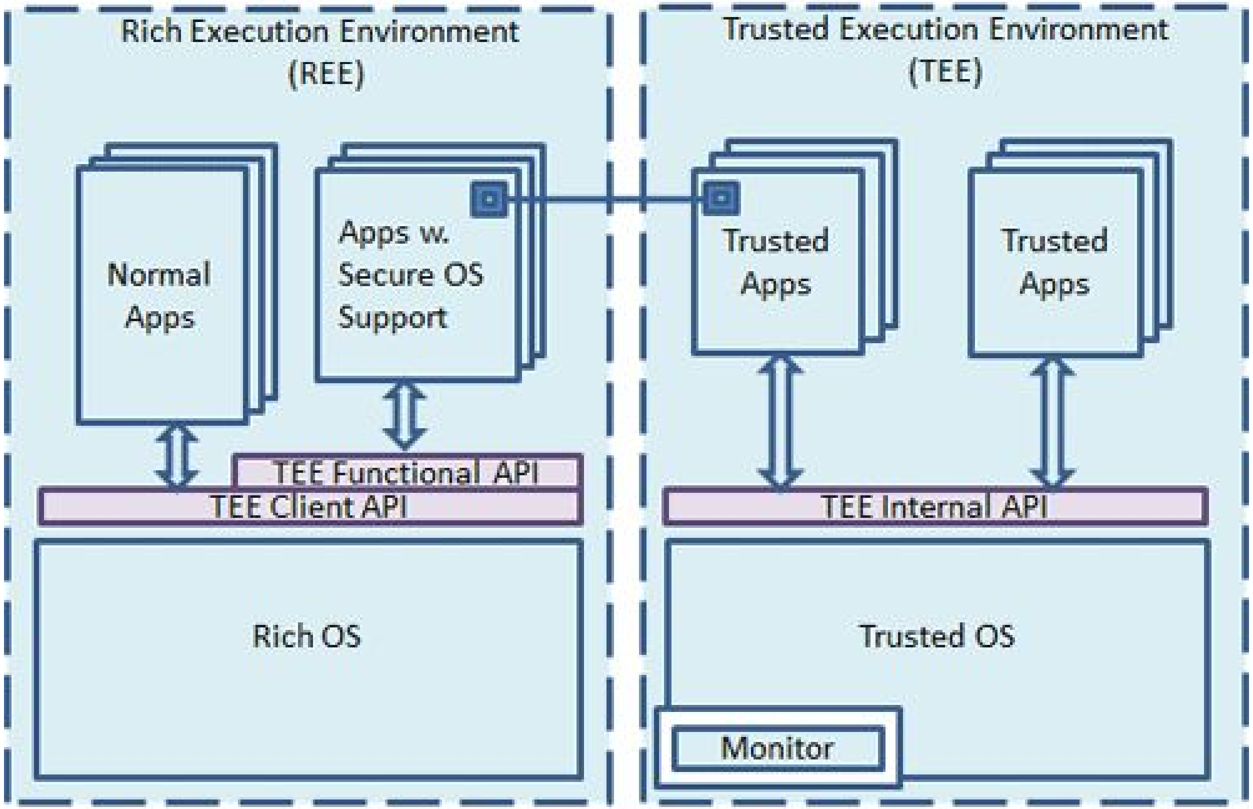
J.-E. Ekberg, K Kostiainen et al. „Trusted Execution Environments on Mobile Devices“

ARM TrustZone Access Control



ARM. Development of TEE and Secure Monitor Code.

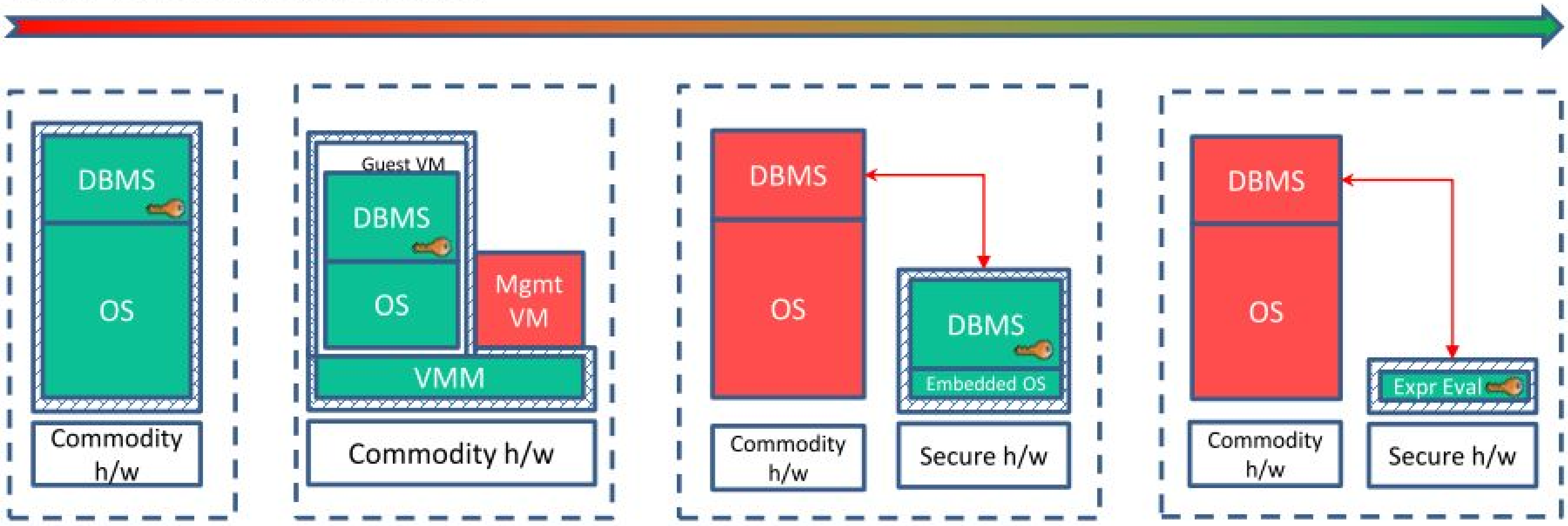
ARM TrustZone Worlds



Secure Databases

Larger Trusted Computing Base (TCB)

Smaller TCB



OS+

[AWSGC]

VMM+

CloudVisor [ZCC+11]

Drawbridge [PBH+ 11]

DBMS

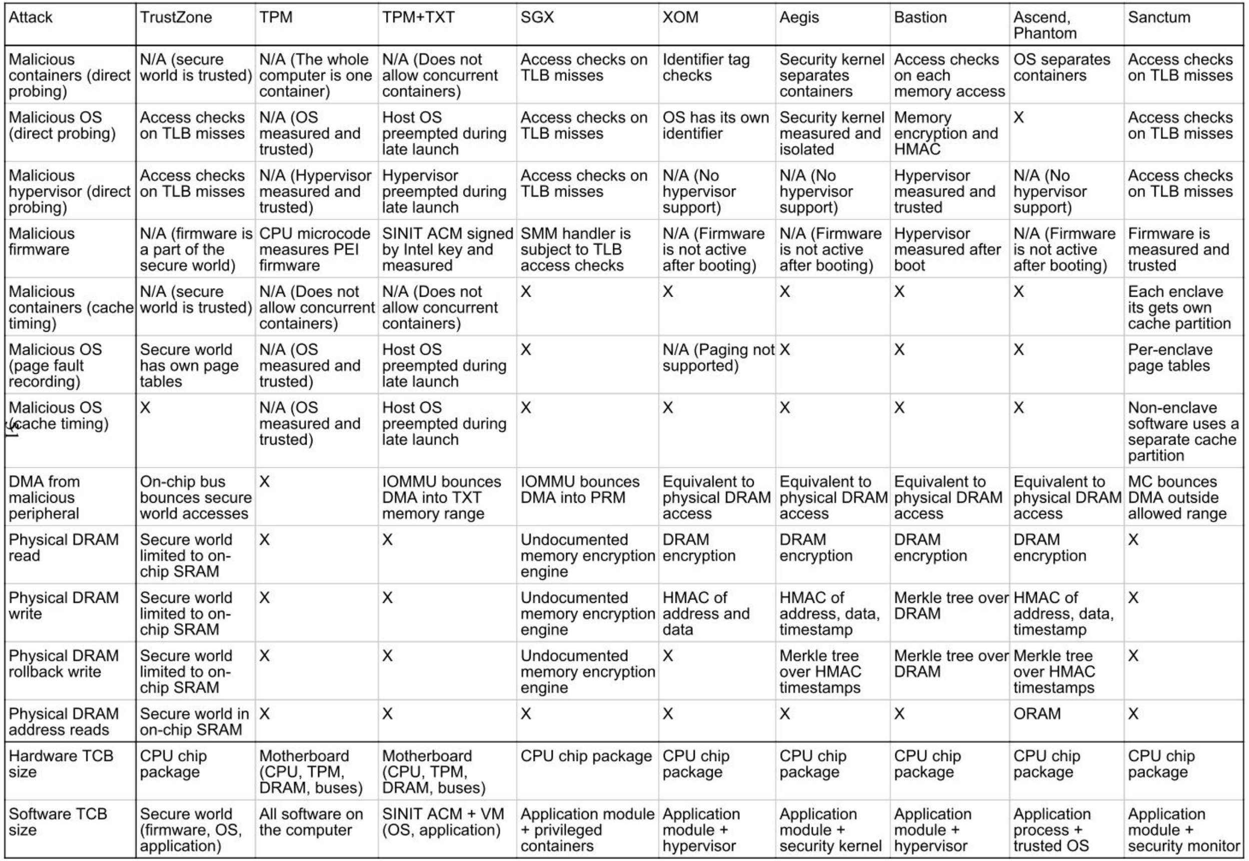
TrustedDB [BS11]

<< DBMS

Cipherbase [ABE+12]

Victor Costan and Srinivas Devadas. „Intel SGX Explained“.

Security Features Overview



Attack	TrustZone	TPM	TPM+TXT	SGX	XOM	Aegis	Bastion	Ascend, Phantom	Sanctum
Malicious containers (direct probing)	N/A (secure world is trusted)	N/A (The whole computer is one container)	N/A (Does not allow concurrent containers)	Access checks on TLB misses	Identifier tag checks	Security kernel separates containers	Access checks on each memory access	OS separates containers	Access checks on TLB misses
Malicious OS (direct probing)	Access checks on TLB misses	N/A (OS measured and trusted)	Host OS preempted during late launch	Access checks on TLB misses	OS has its own identifier	Security kernel measured and isolated	Memory encryption and HMAC	X	Access checks on TLB misses
Malicious hypervisor (direct probing)	Access checks on TLB misses	N/A (Hypervisor measured and trusted)	Hypervisor preempted during late launch	Access checks on TLB misses	N/A (No hypervisor support)	N/A (No hypervisor support)	Hypervisor measured and trusted	N/A (No hypervisor support)	Access checks on TLB misses
Malicious firmware	N/A (firmware is a part of the secure world)	CPU microcode measures PEI firmware	SINIT ACM signed by Intel key and measured	SMM handler is subject to TLB access checks	N/A (Firmware is not active after booting)	N/A (Firmware is not active after booting)	Hypervisor measured after boot	N/A (Firmware is not active after booting)	Firmware is measured and trusted
Malicious containers (cache timing)	N/A (secure world is trusted)	N/A (Does not allow concurrent containers)	N/A (Does not allow concurrent containers)	X	X	X	X	X	Each enclave its gets own cache partition
Malicious OS (page fault recording)	Secure world has own page tables	N/A (OS measured and trusted)	Host OS preempted during late launch	X	N/A (Paging not supported)	X	X	X	Per-enclave page tables
Malicious OS (cache timing)	X	N/A (OS measured and trusted)	Host OS preempted during late launch	X	X	X	X	X	Non-enclave software uses a separate cache partition
DMA from malicious peripheral	On-chip bus bounces secure world accesses	X	IOMMU bounces DMA into TXT memory range	IOMMU bounces DMA into PRM	Equivalent to physical DRAM access	Equivalent to physical DRAM access	Equivalent to physical DRAM access	Equivalent to physical DRAM access	MC bounces DMA outside allowed range
Physical DRAM read	Secure world limited to on-chip SRAM	X	X	Undocumented memory encryption engine	DRAM encryption	DRAM encryption	DRAM encryption	DRAM encryption	X
Physical DRAM write	Secure world limited to on-chip SRAM	X	X	Undocumented memory encryption engine	HMAC of address and data	HMAC of address, data, timestamp	Merkle tree over DRAM	HMAC of address, data, timestamp	X
Physical DRAM rollback write	Secure world limited to on-chip SRAM	X	X	Undocumented memory encryption engine	X	Merkle tree over HMAC timestamps	Merkle tree over DRAM	Merkle tree over HMAC timestamps	X
Physical DRAM address reads	Secure world in on-chip SRAM	X	X	X	X	X	X	ORAM	X
Hardware TCB size	CPU chip package	Motherboard (CPU, TPM, DRAM, buses)	Motherboard (CPU, TPM, DRAM, buses)	CPU chip package	CPU chip package	CPU chip package	CPU chip package	CPU chip package	CPU chip package
Software TCB size	Secure world (firmware, OS, application)	All software on the computer	SINIT ACM + VM (OS, application)	Application module + privileged containers	Application module + hypervisor	Application module + security kernel	Application module + hypervisor	Application process + trusted OS	Application module + security monitor