CIS 4360
Secure Computer Systems

SGX

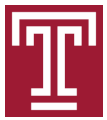
Professor Qiang Zeng

Spring 2017

Some slides are stolen from Intel docs

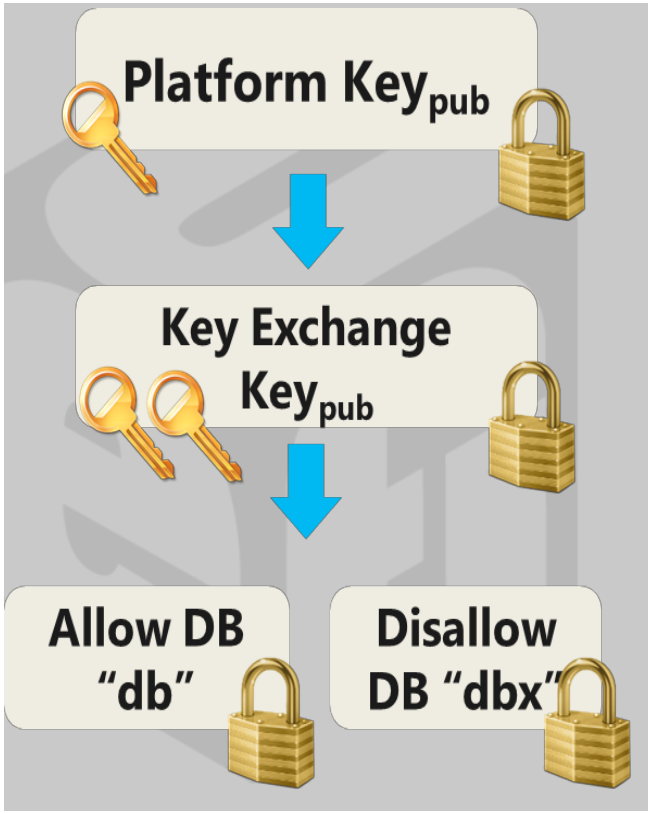
Previous Class

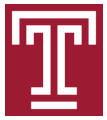
- UEFI Secure Boot
- Windows's Trusted Boot
- Intel's Trusted Boot



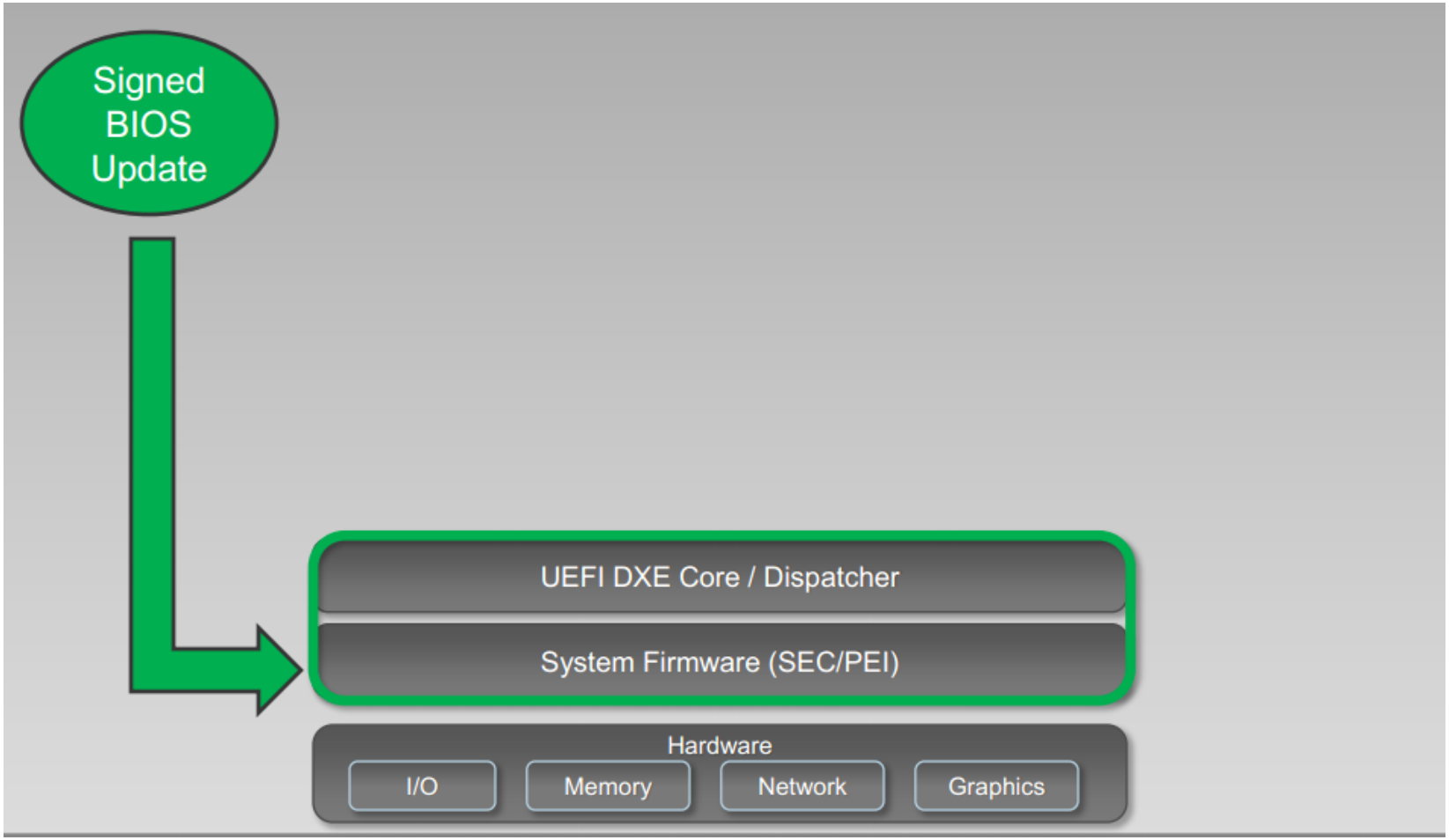
Keys in UEFI Secure Boot

- Platform Key (PK)
  - Allows modification of KEK
  - Initialized by the h/w vendor
- Key Exchange Key (KEK)
  - Allows modification of db and dbx
  - Can be multiple
- Authorized Database (db)
  - Legal CAs, signatures and hashes
- Forbidden Database (dbx)
  - Illegal CAs, signatures and hashes





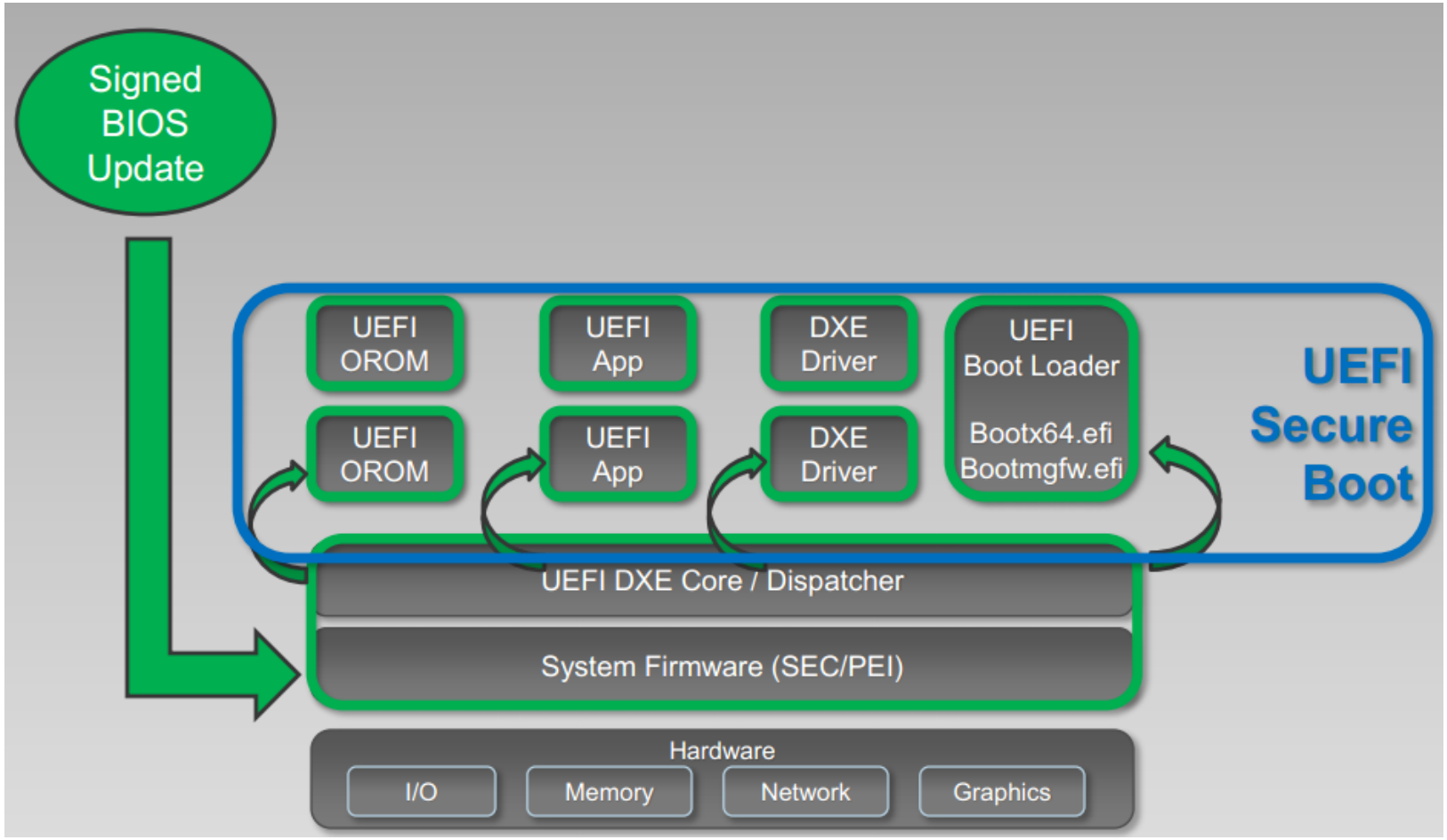
Firmware Signing



- Flash-based UEFI components are verified only during the update process when the whole BIOS image has its signature verified

http://c7zero.info/stuff/Windows8SecureBoot_Bulygin-Furtak-Bazhniuk_BHUSA2013.pdf

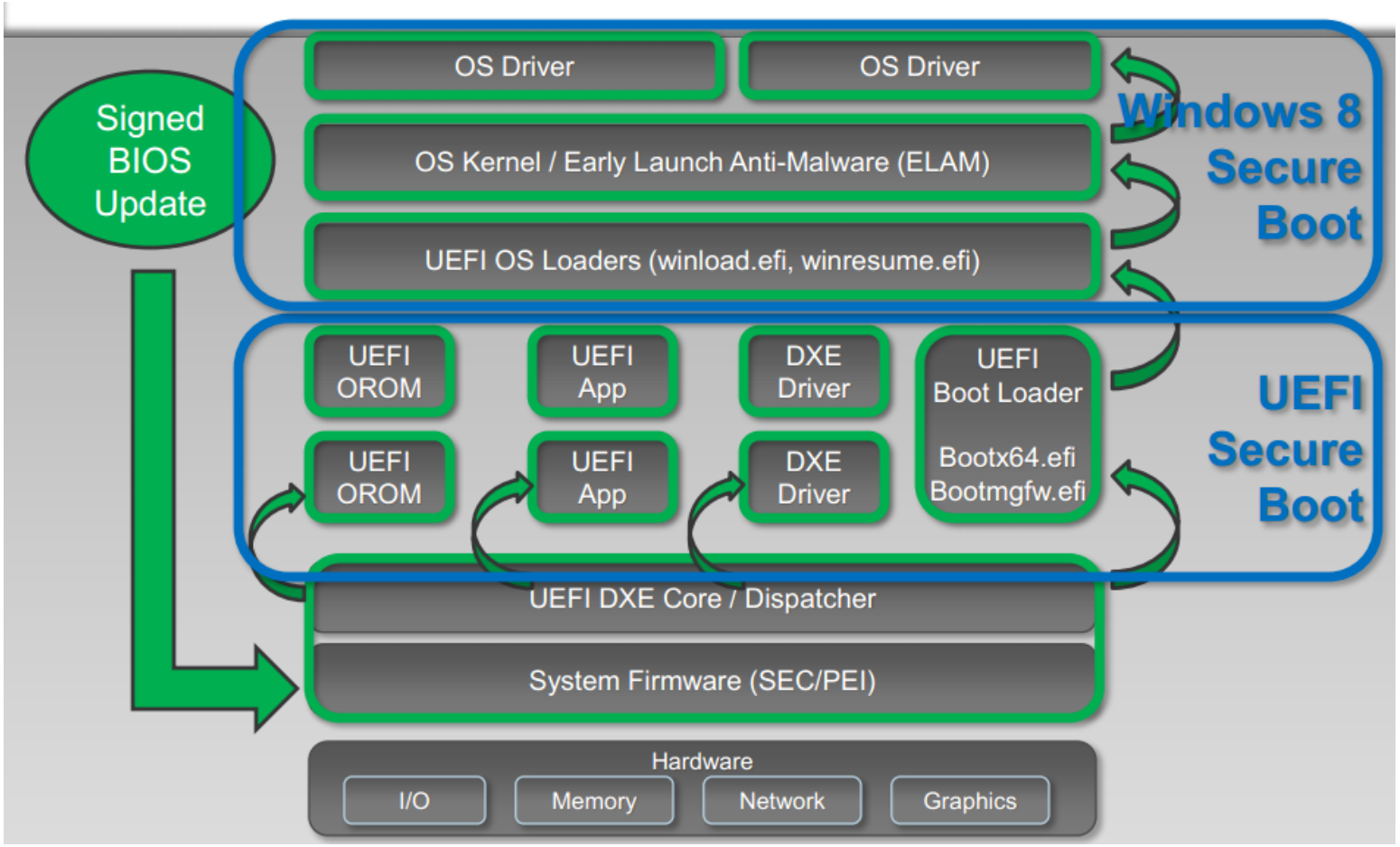
UEFI Secure Boot



- DXE verifies non-embedded XROMs, DXE drivers, UEFI applications and boot loader(s)
- This is the UEFI Secure Boot process

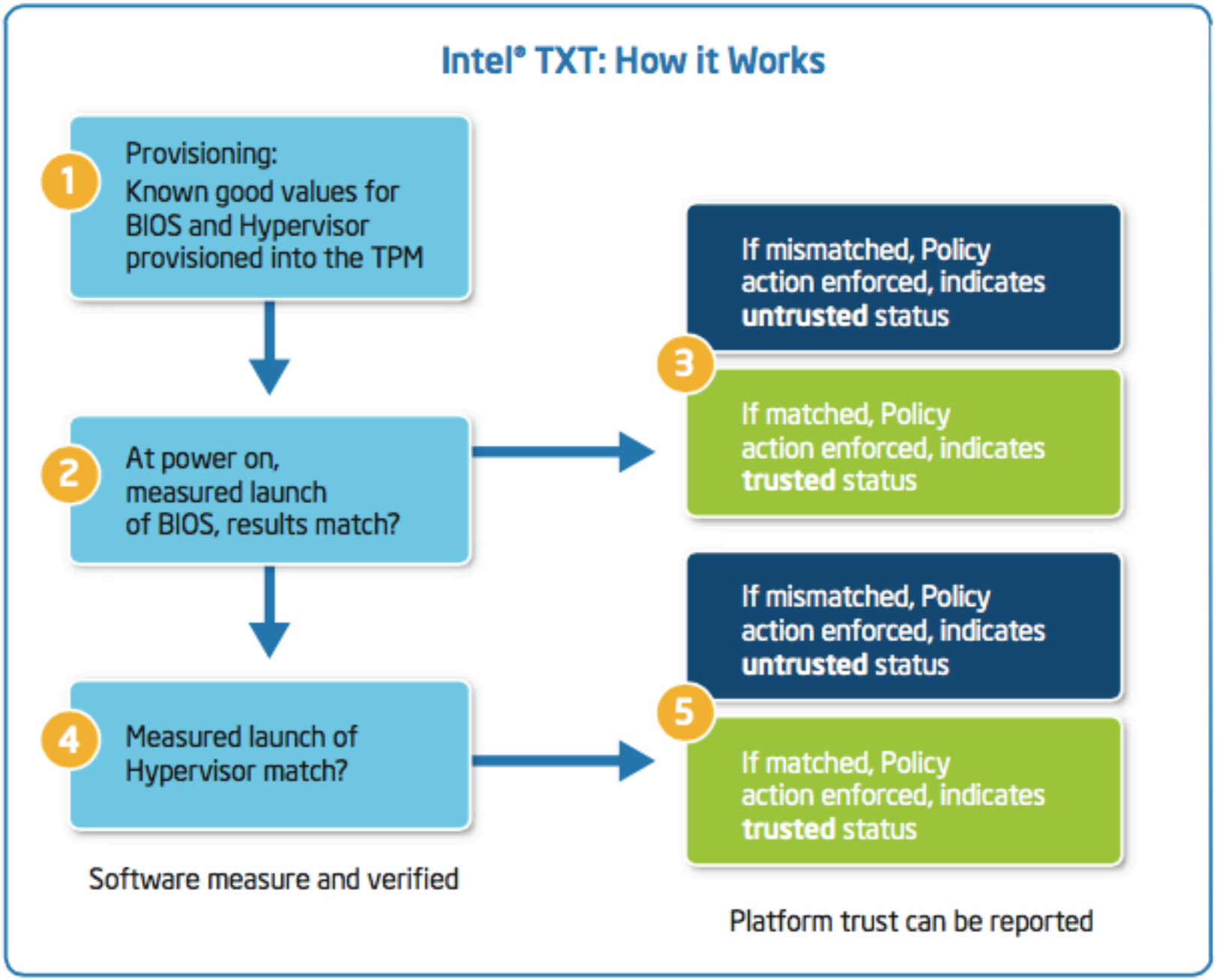
http://c7zero.info/stuff/Windows8SecureBoot_Bulygin-Furtak-Bazhniuk_BHUSA2013.pdf

Windows 8 Secure Boot



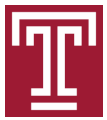
- Microsoft Windows 8 adds to the UEFI secure boot process
- Establishes a chain of verification
- UEFI Boot Loader -> OS Loader -> OS Kernel -> OS Drivers

http://c7zero.info/stuff/Windows8SecureBoot_Bulygin-Furtak-Bazhniuk_BHUSA2013.pdf


Intel® TXT: How it Works
1
Provisioning:
Known good values for BIOS and Hypervisor provisioned into the TPM
2
At power on, measured launch of BIOS, results match?
3
If mismatched, Policy action enforced, indicates untrusted status
If matched, Policy action enforced, indicates trusted status
4
Measured launch of Hypervisor match?
5
If mismatched, Policy action enforced, indicates untrusted status
If matched, Policy action enforced, indicates trusted status
Software measure and verified
Platform trust can be reported

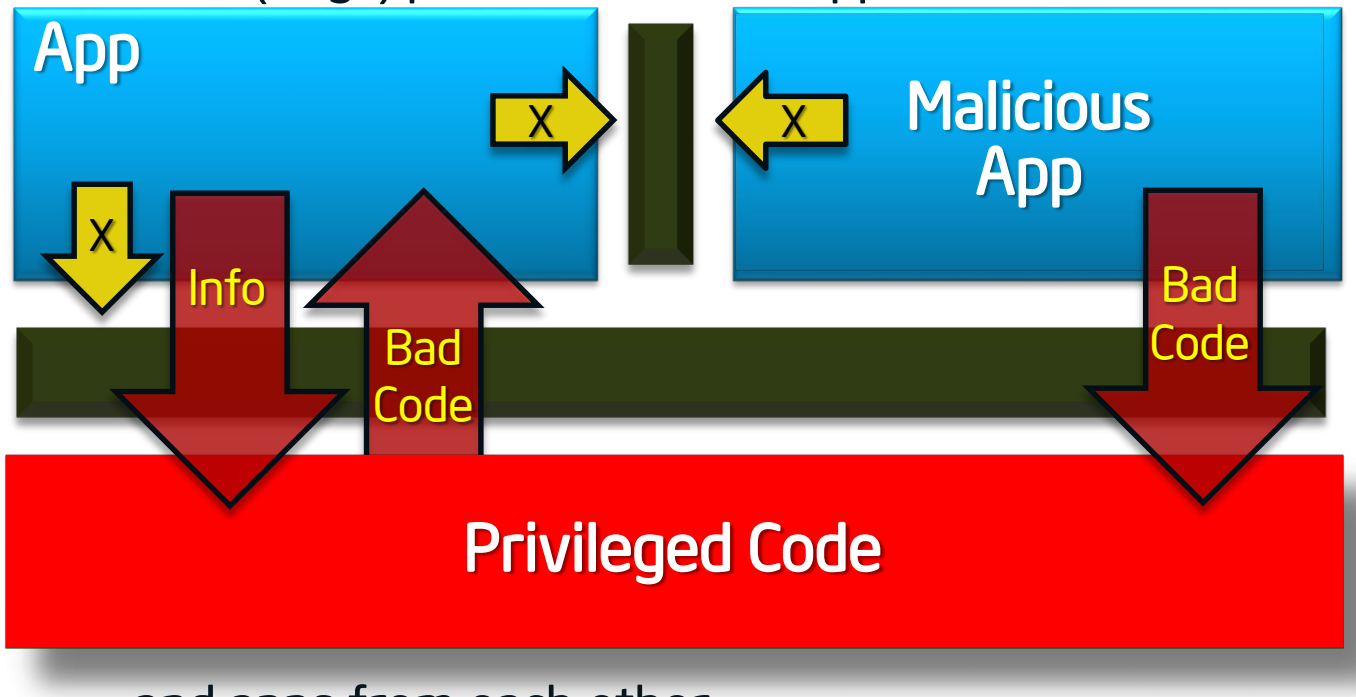
Outline

- What is SGX?
- How does SGX work?
- How to use it?



Problems of Current Computation

Protected Mode (rings) protects OS from apps …



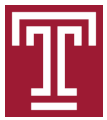
… and apps from each other …

… UNTIL a malicious app exploits a flaw to gain full privileges and then tampers with the OS or other apps

Apps not protected from privileged code attacks

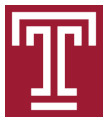
Intel SGX

- SGX: Software Guard Extensions
- Introduced in 2015 with 6th generation Intel CPUs
- It provides a trusted execution environment and guarantees confidentiality and integrity, even when the underlying components, e.g., the BIOS, VMM, and OS, are compromised
  - The trusted execution environment is called “enclave”
- A direct advantage: a customer who purchased cloud service from Amazon can have peace of mind that its data assets cannot be stolen by Amazon



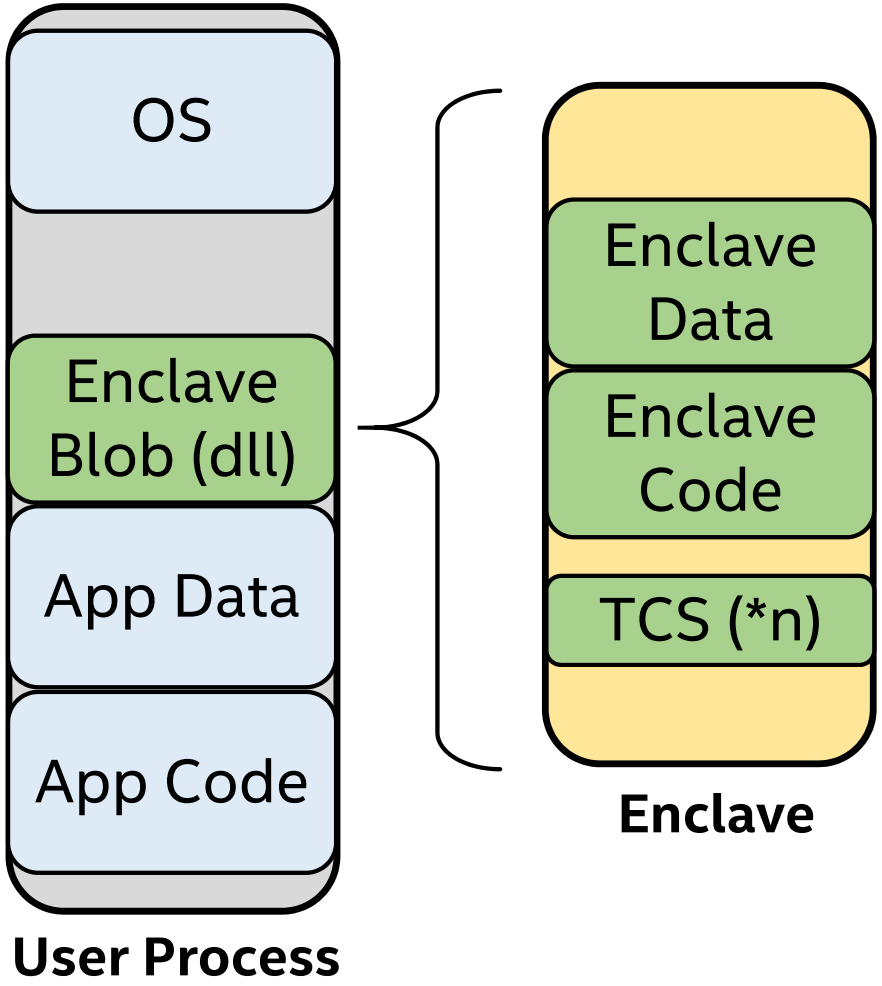
What kind of attacks are prevented

- Even when your BIOS/SMM/Operation System/ Virtual Machine Monitor (SMM) are all compromised, the attacker cannot steal secret protected by SGX
- Even when you launch DMA attack and Cold Boot attack against RAM, the attacker cannot steal secret protected by SGX



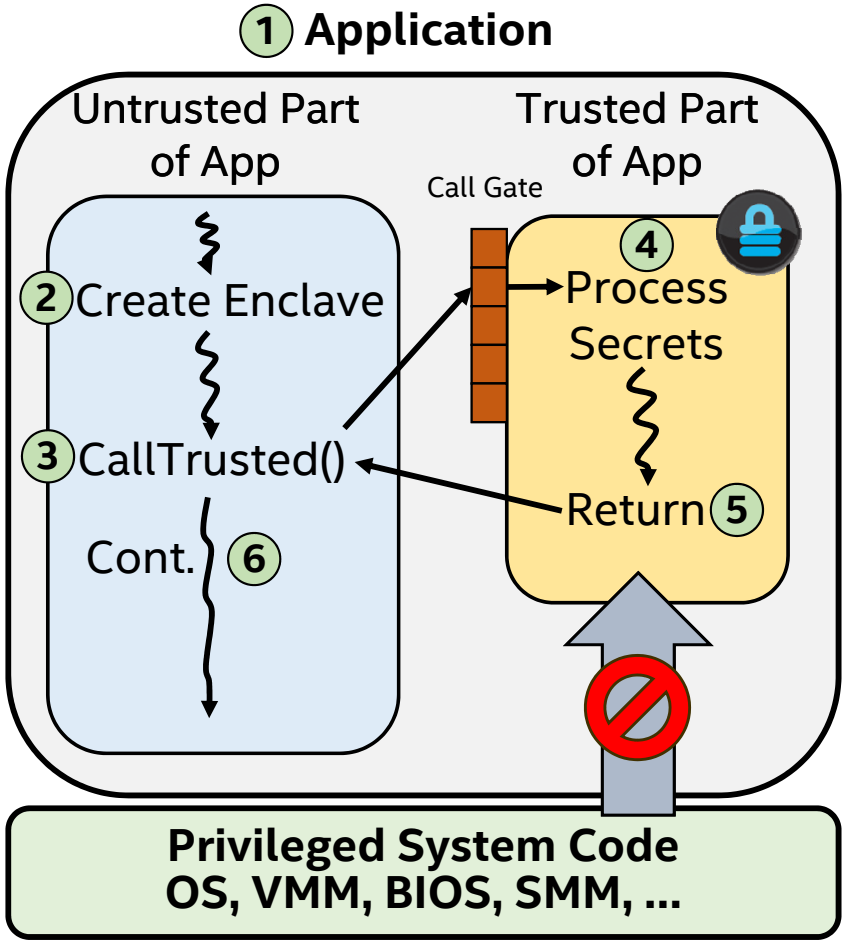
Process View

Protected execution environment embedded in a process

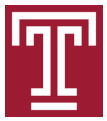


- With its own code and data
- Providing Confidentiality & Integrity
- Controlled entry points
- Multi-thread support
- Full access to app memory and processor performance

EXECUTION FLOW

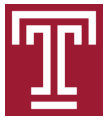


1. App built with trusted and untrusted parts
2. App runs & creates the enclave which is placed in trusted memory
3. Trusted function is called, execution transitioned to the enclave
4. Enclave sees **all process data** in clear; external access to enclave data is denied
5. Trusted function returns; enclave data remains in trusted memory
6. Application continues normal execution



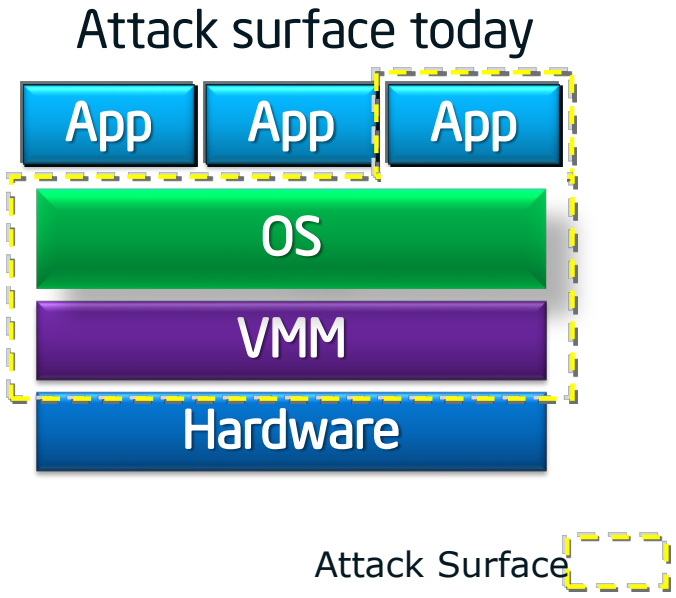
What is SGX?

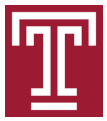
- Intel SGX is essentially a new mode of execution on the CPU, a new memory protection semantic, plus a couple of new instructions to manage this all
- You create an enclave by filling its pages with desired code, then you lock it down, measure the code there, and if everything's fine, you send in your secret data securely and start the execution inside the enclave
- Since now on, no entity, including the kernel (ring 0) or hypervisor (ring “-1”), or SMM (ring “-2”) or AMT (ring “-3”), has no right to read nor write the memory pages belonging to the enclave



Attack Surface for non-SGX computation

- Reduced attack surface with SGX



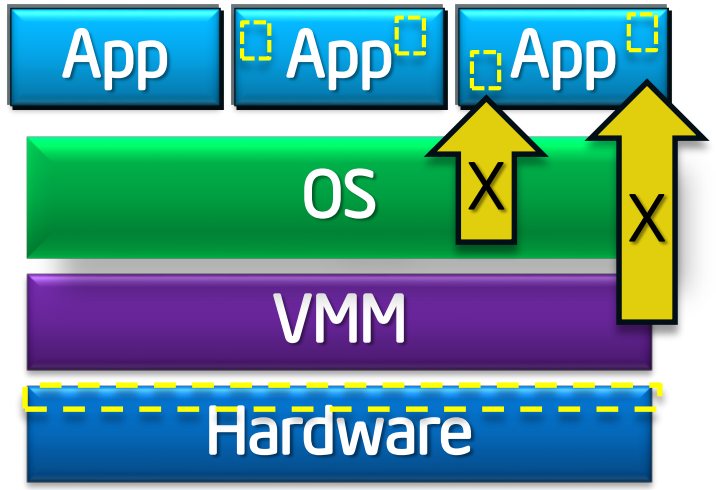


Reduced attack surface with SGX

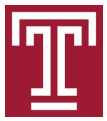
Application gains ability to defend its own secrets

- Smallest attack surface (App + processor)
- Malware that subverts OS/VMM, BIOS, Drivers etc. cannot steal app secrets

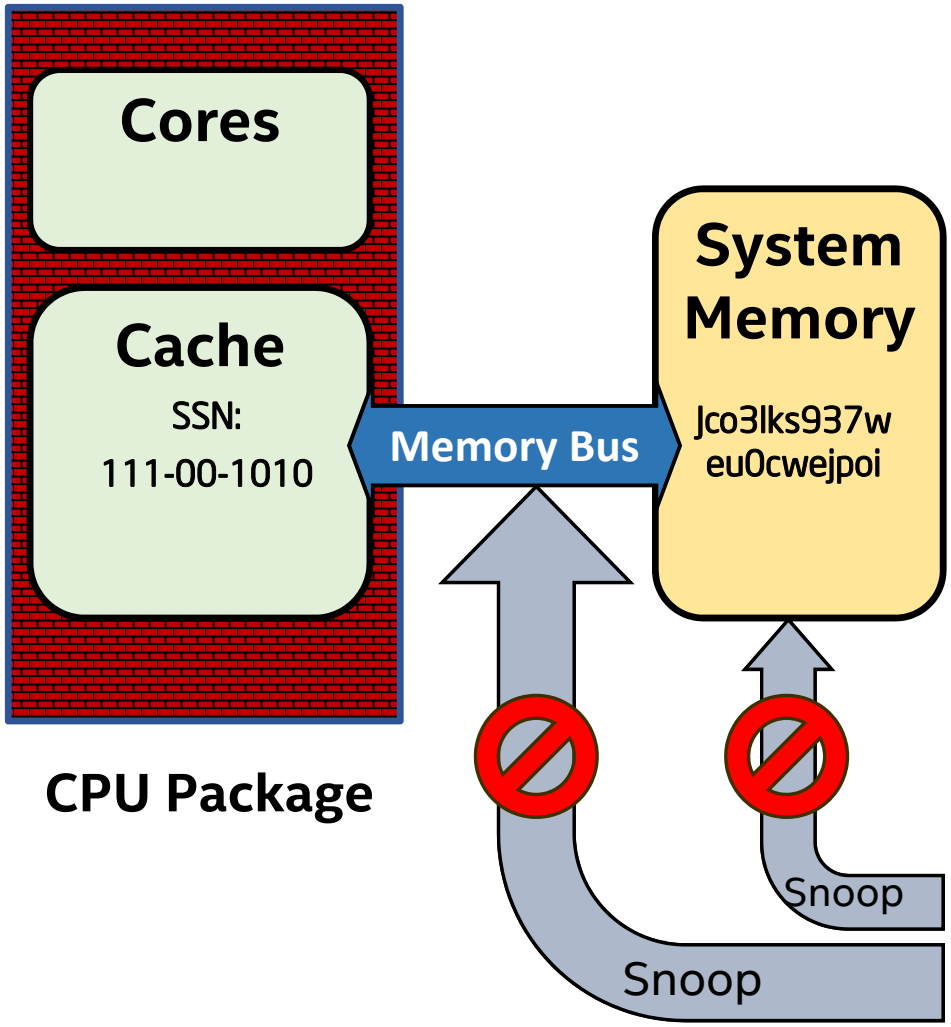
Attack surface with Enclaves



Attack Surface



SECURITY PERIMETER



- Security perimeter is the CPU package boundary
- Data and code unencrypted inside CPU package
- Data and code outside CPU package is **encrypted and integrity checked**
- External memory reads and bus snoops see only encrypted data

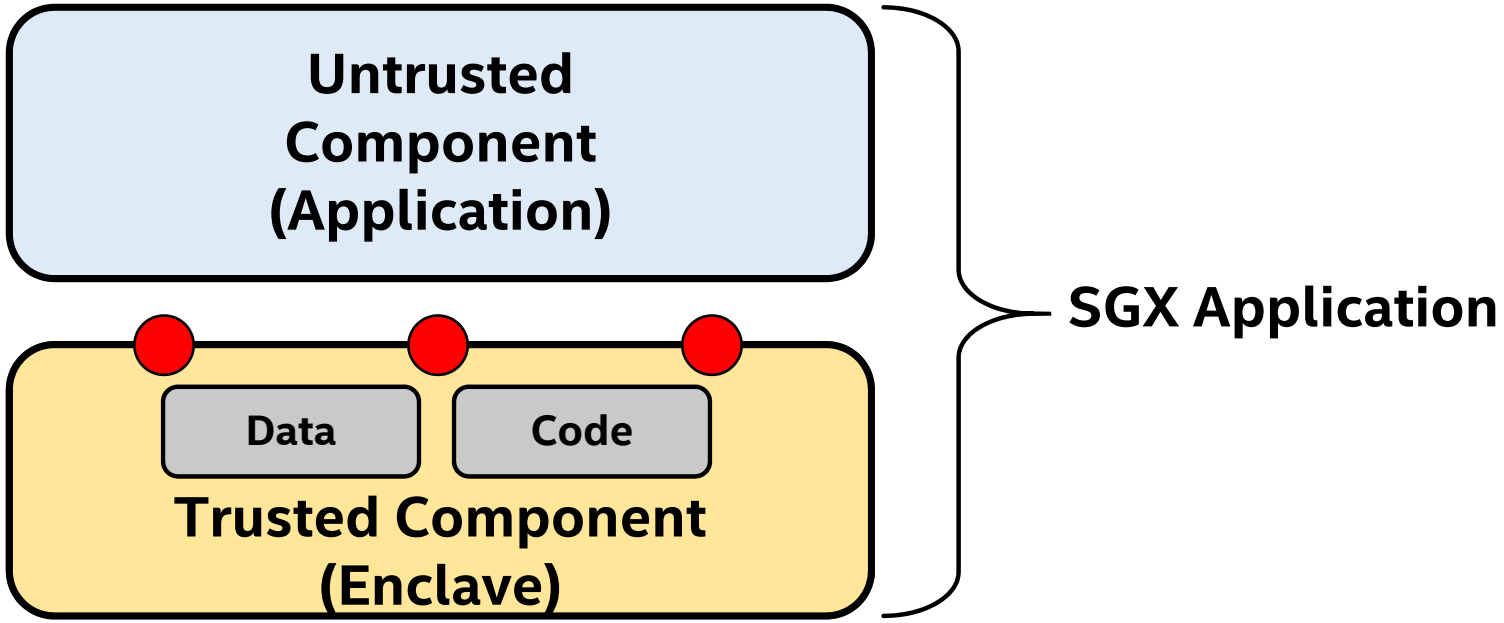
PARTITIONING

Identify sensitive application data (secrets) and the operations that work on/with that data

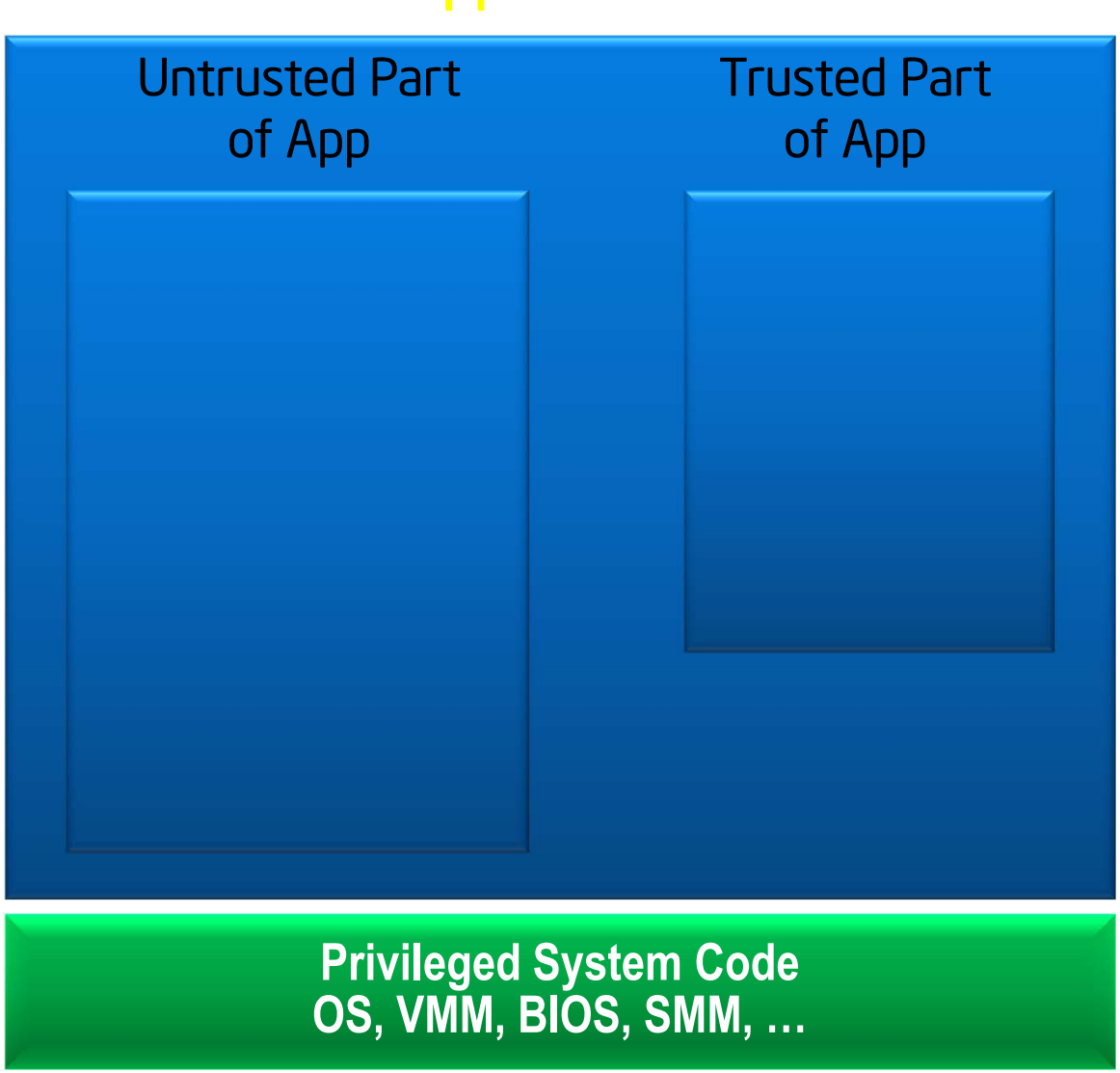
- Ex. Key material, proprietary algorithms, biometric data, CSR generation, etc.

Partition this **functionality** to an enclave

- Do not hard code secrets into the enclave

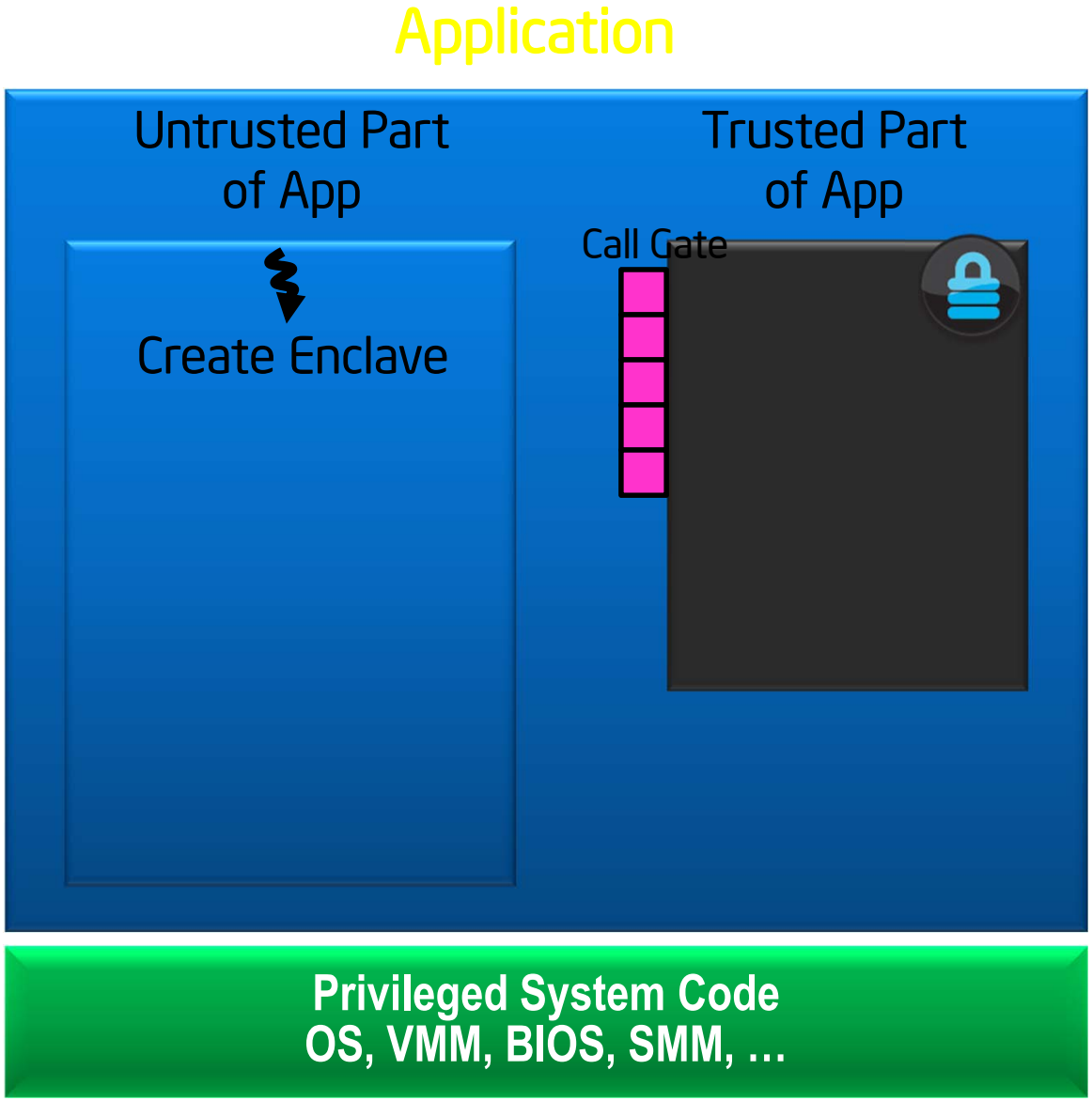


How SE Works: Protection vs. Software Attack



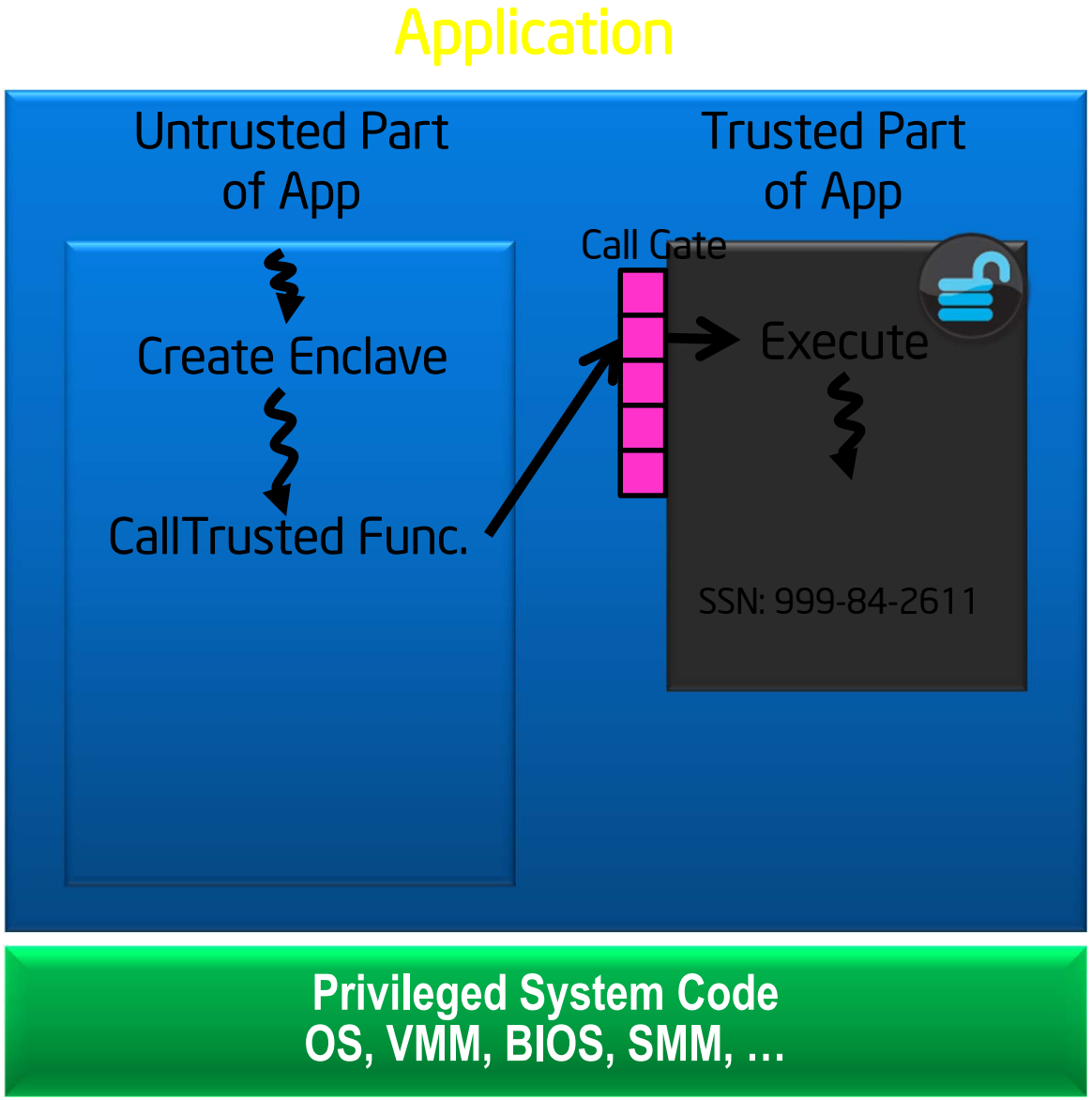
1. App is built with trusted and untrusted parts

How SE Works: Protection vs. Software Attack



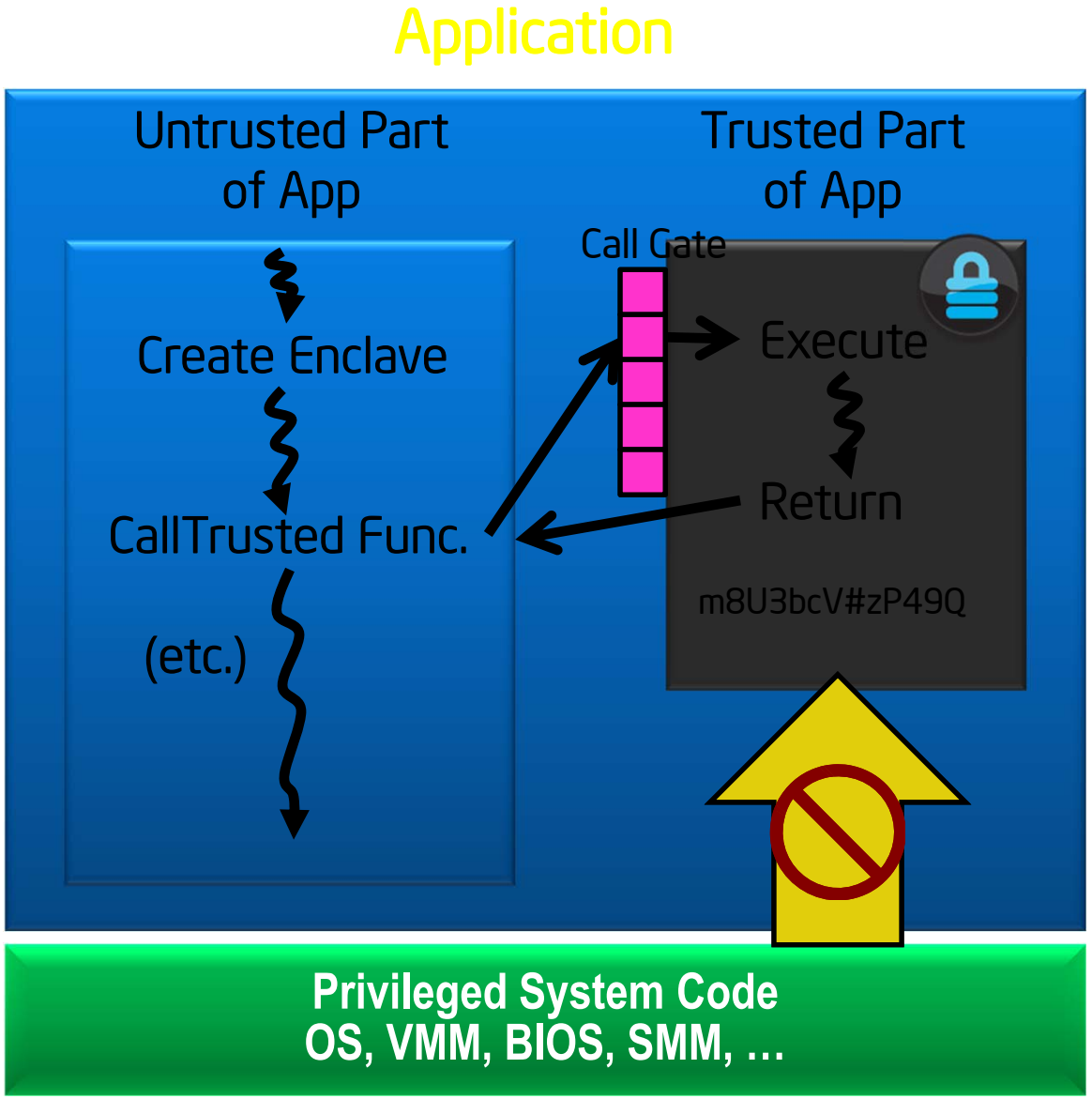
1. App is built with trusted and untrusted parts
2. App runs & creates enclave which is placed in trusted memory

How SE Works: Protection vs. Software Attack



1. App is built with trusted and untrusted parts
2. App runs & creates enclave which is placed in trusted memory
3. Trusted function is called; code running inside enclave sees data in clear; external access to data is denied

How SE Works: Protection vs. Software Attack



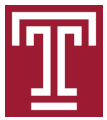
1. App is built with trusted and untrusted parts
2. App runs & creates enclave which is placed in trusted memory
3. Trusted function is called; code running inside enclave sees data in clear; external access to data is denied
4. Function returns; enclave data remains in trusted memory

PROVISIONING ENCLAVE SECRETS

Well designed enclaves never contain hard coded secrets, instead, they are provisioned or created **after** the enclave is loaded

- Enclave binaries are un-encrypted and inspectable
- Should verify that the 'right app/enclave is executing on the right platform' using attestation

Persist provisioned Secrets across execution runs using HW provided Seal Key



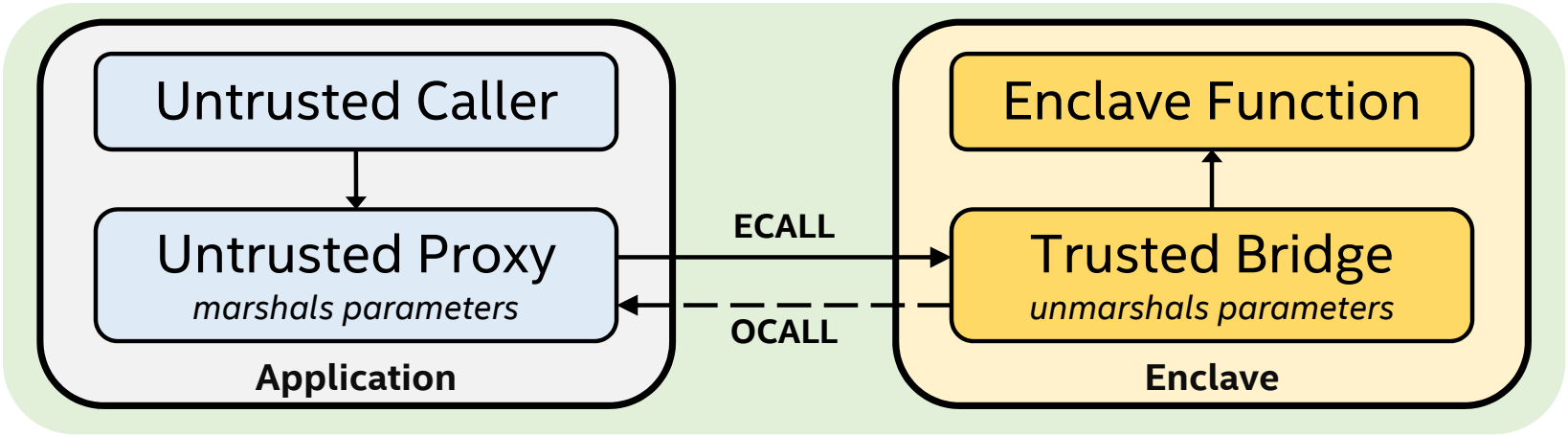
ENCLAVE INTERFACE DEFINITION

Careful definition of the enclave interface is critical

- The enclave's interface is it's attack surface; it should be minimal and avoid data leakage

Enclave Definition Language (EDL) is used to define an enclave's Trusted and Untrusted interface functions

- Tools process the EDL to create proxy / bridge code to call into (ECALL) and return from (OCALL) an enclave



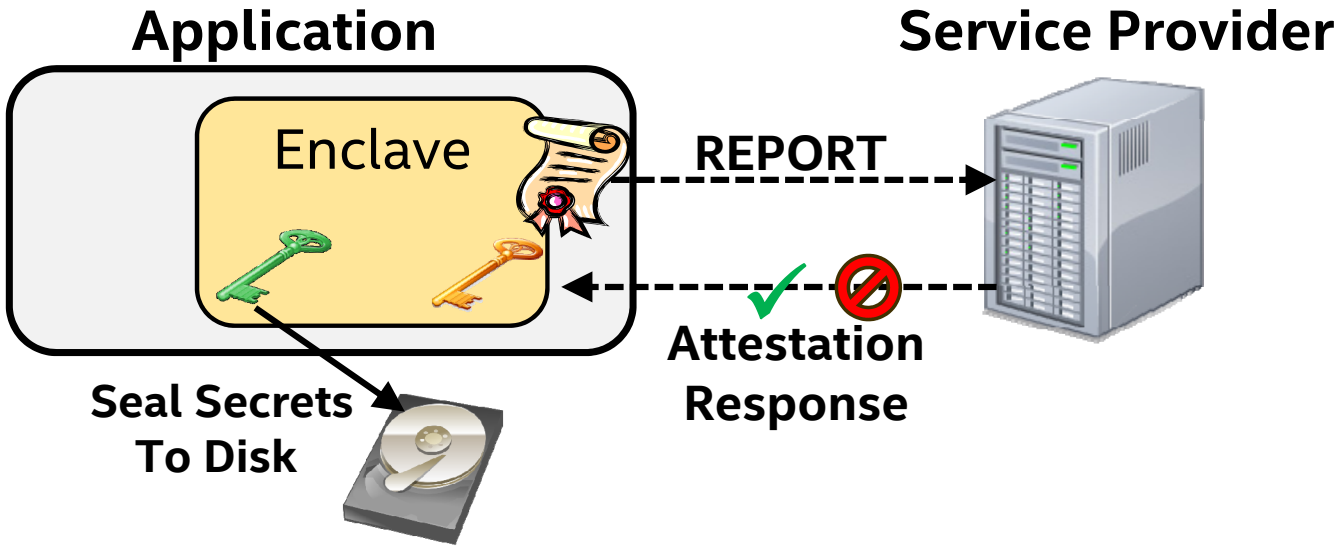
ATTESTATION & SEALING

Support for enclave attestation to a 3rd party

- Can attest; Enclave SW, CPU Security level, Sealing Identity, and more.

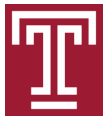
Data can be sealed against an enclave using a hardware derived Seal Key

- The Seal Key is unique to the CPU and the specific enclave environment



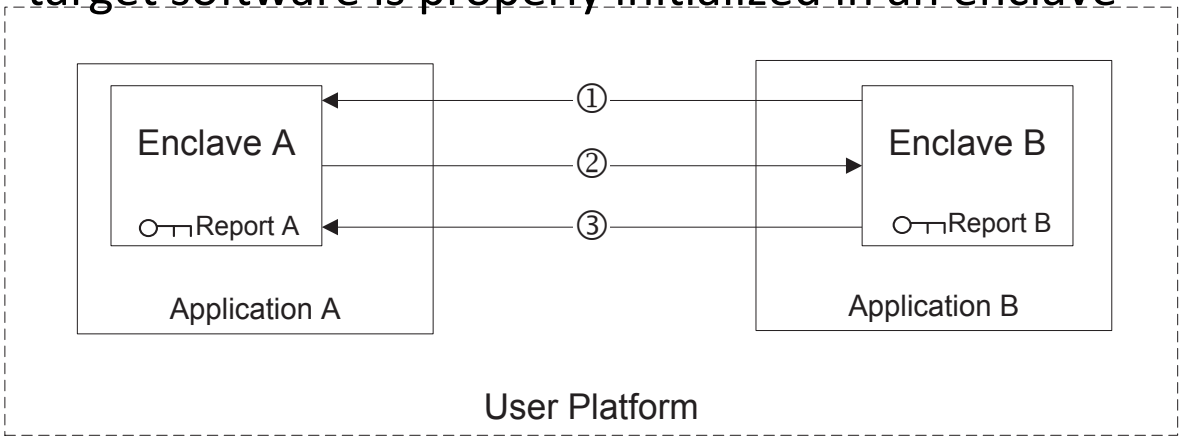
Enclave Certificate: SIGSTRUCT

- The enclave builder (also called “Sealing Authority”) signs a certificate for each enclave. The certificate contains
  - The expected value of the Enclave Identity, which is a hash of contents and properties of the enclave pages
  - The public key of the Sealing Authority
  - The product ID
- For each enclave, there two registers
  - MRENCLAVE: it records the measurement of the contents and properties of the pages of *an initialized enclave*
  - MRSIGNER: after CPU checks the MRENCLAVE value against the expected one in SIGSTRUCT, it stores a hash of the public key of the Sealing Authority in MRSIGNER
- The MRENCLAVE value represents the enclave identity, while MRSIGNER represents the enclave builder identity



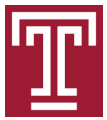
Local Attestation

In SGX, Attestation is the process of proving that some target software is properly initialized in an enclave

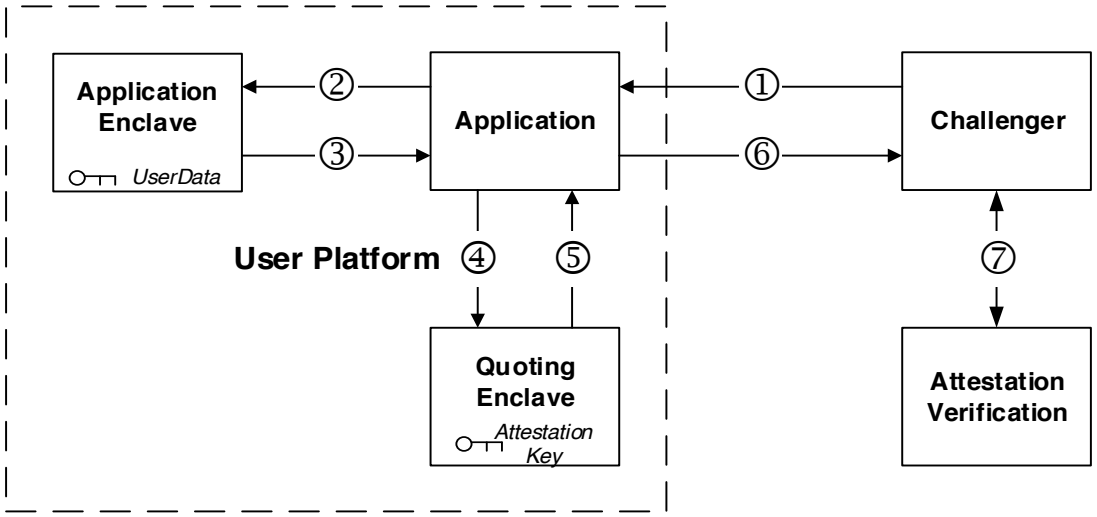


Enclave B wants to verifies Enclave A in the same machine

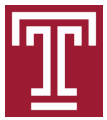
1. So B sends $MRENCLAVE_B$ to A
2. A asks the h/w to create a report about its identities with a MAC tag (the tag is created using a key only known by B). B then verifies the report using the Report Key (each enclave has its unique Report Key)
3. Optionally, B asks the h/w to create a report and then sends it to A



Remote Attestation Flow

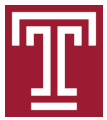


- When an app receives an attestation request (containing a nonce) from an off-platform challenger (1), the app requests its enclave produce a report (2), which is then sent to the Quoting Enclave (by Intel) (3 and 4); it verifies the report then signs it with a private key (Enhanced Privacy ID Key) (5). The challenger then asks Intel's attestation service to verify the signature.
- The challenger should not send secret to the enclave until its identity is what the challenger expects and the signature has been verified



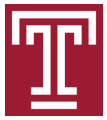
Sealing

- After an enclave receives secrets from the remote challenger, it may want to store the secret in disk, so that it can be retrieved for later uses
- However, the secret should not be revealed unless the same h/w and s/w environments are present
- Thus, the secret should be encrypted using a Seal Key that is derived from
  - The Root Provisioning Key that reflects the h/w
  - MRENCLAVE value or MRSIGNER value
  - Intel SGX libs
- The effect is that nobody can steal the secret



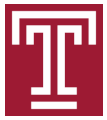
Caveats

- Direct system calls (e.g., I/O) are prohibited inside the enclave, as the kernel is not trusted by the enclave
- Workaround: You can invoke the external untrusted app code or APIs to perform IO, but this is risky
- https://software.intel.com/en-us/forums/intel-software-guard-extensions-intel-sgx/topic/539803
- http://stackoverflow.com/questions/28114746/what-does-it-implies-to-disable-syscall-in-intel-sgx



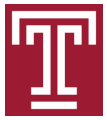
References

- Poster: A Primer on Intel SGX
  - https://software.intel.com/sites/default/files/managed/3e/b9/SF15_ISGC003_81_SGX_DL_100_small.pdf
- Thoughts on Intel's upcoming Software Guard Extensions (Part 2). 2013, Rutkowska.
  - http://theinvisiblethings.blogspot.com/2013/09/thoughts-on-intels-upcoming-software.html
- Intel SGX slides
  - https://software.intel.com/sites/default/files/332680-002.pdf (2015)
  - https://web.stanford.edu/class/ee380/Abstracts/150415-slides.pdf (2015; confusing)



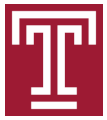
References

- Intel SGX Explained. Costan and Devadas, 2016
  - https://eprint.iacr.org/2016/086.pdf
- Intel SGX Notes
  - https://intelsgx.blogspot.com/2016/05/intel-sgx-vs-txt.html
- Intel *SGX Programming Reference*
  - https://software.intel.com/sites/default/files/managed/48/88/329298-002.pdf
- Intel SGX Developer Guide
  - https://download.01.org/intel-sgx/linux-1.5/docs/Intel_SGX_Developer_Guide.pdf



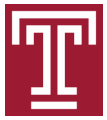
References

- SCONE: Secure Linux Containers with Intel SGX (OSDI 16)
  - It emphasizes the system call limitation
- Innovative Technology for CPU Based Attestation and Sealing
  - https://software.intel.com/en-us/articles/innovative-technology-for-cpu-based-attestation-and-sealing
- Intel SGX Details Discussion
  - https://security.stackexchange.com/questions/136993/intel-sgx-details



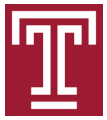
References

- "Intel Software Guard Extensions: EPID Provisioning and Attestation Services", by Johnson, Scarlata, Rozas, Brickell, and Mckeen.
  - https://software.intel.com/en-us/blogs/2016/03/09/intel-sgx-epid-provisioning-and-attestation-services
- SGX Secure Enclaves in Practice: Security and Crypto Review (blackhat, 2016)
  - https://www.blackhat.com/docs/us-16/materials/us-16-Aumasson-SGX-Secure-Enclaves-In-Practice-Security-And-Crypto-Review-wp.pdf



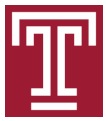
References

- SGX Instructions
  - https://software.intel.com/en-us/blogs/2016/06/10/overview-of-intel-software-guard-extensions-instructions-and-data-structures
  - The ISCA slides contain clear interpretation of instructions
- Comparison of Trusted Computing by Murdoch
  - http://sec.cs.ucl.ac.uk/users/smurdoch/talks/rhul14tee.pdf (Introduction to Trusted Execution Environments (TEE) – IY5606)
- Example SGX applications
  - https://software.intel.com/en-us/articles/using-innovative-instructions-to-create-trustworthy-software-solutions

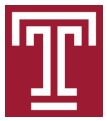


References

- Beautiful slides by Dr. Sadeghi
  - https://www.trust.cased.de/fileadmin/user_upload/Group_TRUST/LectureSlides/ESS-SS2014/Exercise_02_SS2014.pdf



Backup slides



SGX and I/O

- system calls for read/write operation is same as interrupt which is not supported within Enclave. (IO operations are not supported by SGX)

- But SGX includes instructions to let you temporarily exit the enclave in order to call untrusted code so that you can perform I/O, make general system calls, etc. and then return to the enclave. This imposes a performance penalty, and you are executing untrusted code during this time (which is not protected by SGX), so this should only be done when necessary and the amount of time spent outside the enclave should be minimized.
- https://software.intel.com/en-us/forums/intel-software-guard-extensions-intel-sgx/topic/532658