Trustworthy Computing

CS642:
Computer Security

TCG: Background

TCG consortium. Founded in 1999. Lots of companies.

Goals:

- **Hardware protected (encrypted) storage**:
  - Only "authorized" software can decrypt data
  - e.g.: protecting key for decrypting file system
    ⇒ only "authorized" software can boot
- **Attestation**: Prove to remote server what software started on my machine.

TCG: changes to the PC

Extra hardware: Trusted Platform Module (**TPM**) chip (33Mhz)

- Available on many laptops

Software changes:

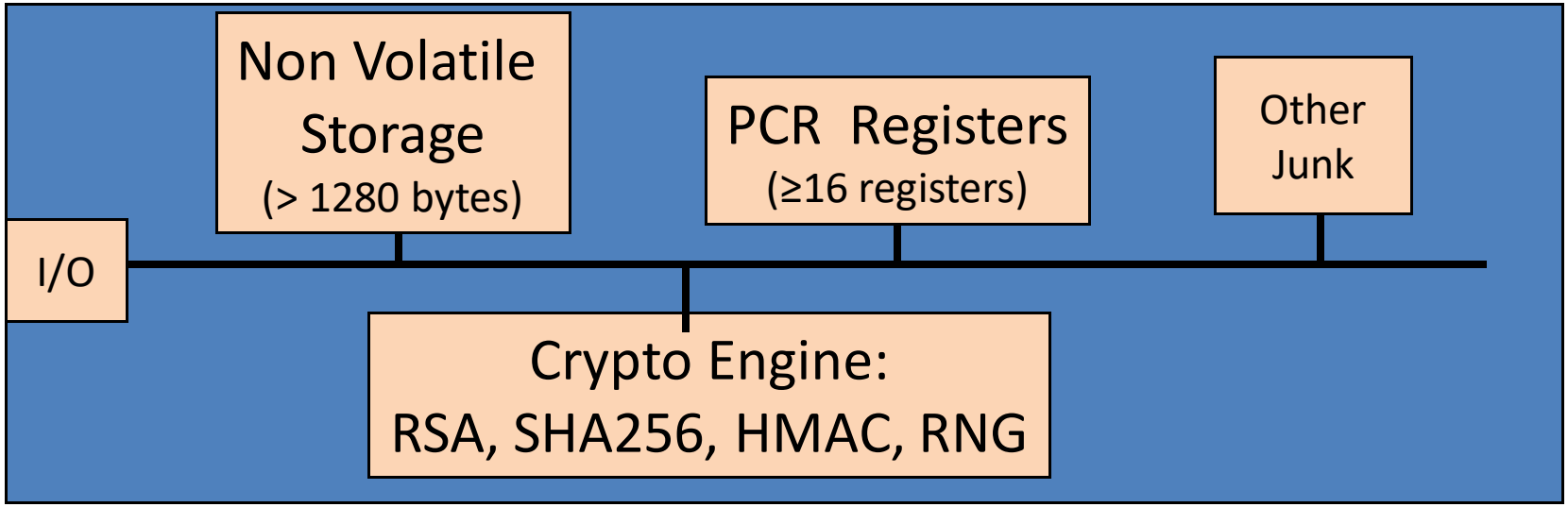
Hardware layer: BIOS, EFI (UEFI)

Software: OS and apps

Trusted Computing

What is the TPM?

Components on TPM chip



RSA: 1024, 2048 bit modulus

SHA256: Outputs 32 byte digest

Non-volatile storage

1. Endorsement Key (EK) (2048-bit RSA)
   - Created at manufacturing time. Cannot be changed.
   - Used for "attestation" (described later)

2. Storage Root Key (SRK) (2048-bit RSA)
   - Used for encrypted storage. Created after running
     TPM_TakeOwnership(OwnerPassword, …)
   - Can be cleared later with TPM_ForceClear from BIOS

3. OwnerPassword (160 bits) and persistent flags

Private: EK, SRK, and OwnerPwd never leave the TPM

PCR: the heart of the matter

PCR: Platform Configuration Registers

- Many PCR registers on chip (at least 16)
- Contents: 32-byte SHA256 digest (+junk)

Updating PCR #n :

- TPM_Extend(n,D): PCR[n] ← SHA256 (PCR[n] || D)
- TPM_PcrRead(n): returns value(PCR(n))

PCRs initialized to default value (e.g. 0) at boot time

Using PCRs: the TCG boot process (SRTM)

On power-up: TPM receives a TPM_Init signal from LPC bus.

BIOS **boot block** executes:

- Calls TPM_Startup (ST_CLEAR) to initialize PCRs to 0
  [can only be called once after TPM_Init]
- Calls PCR_Extend(n, <BIOS code>)
- Then loads and runs BIOS post boot code

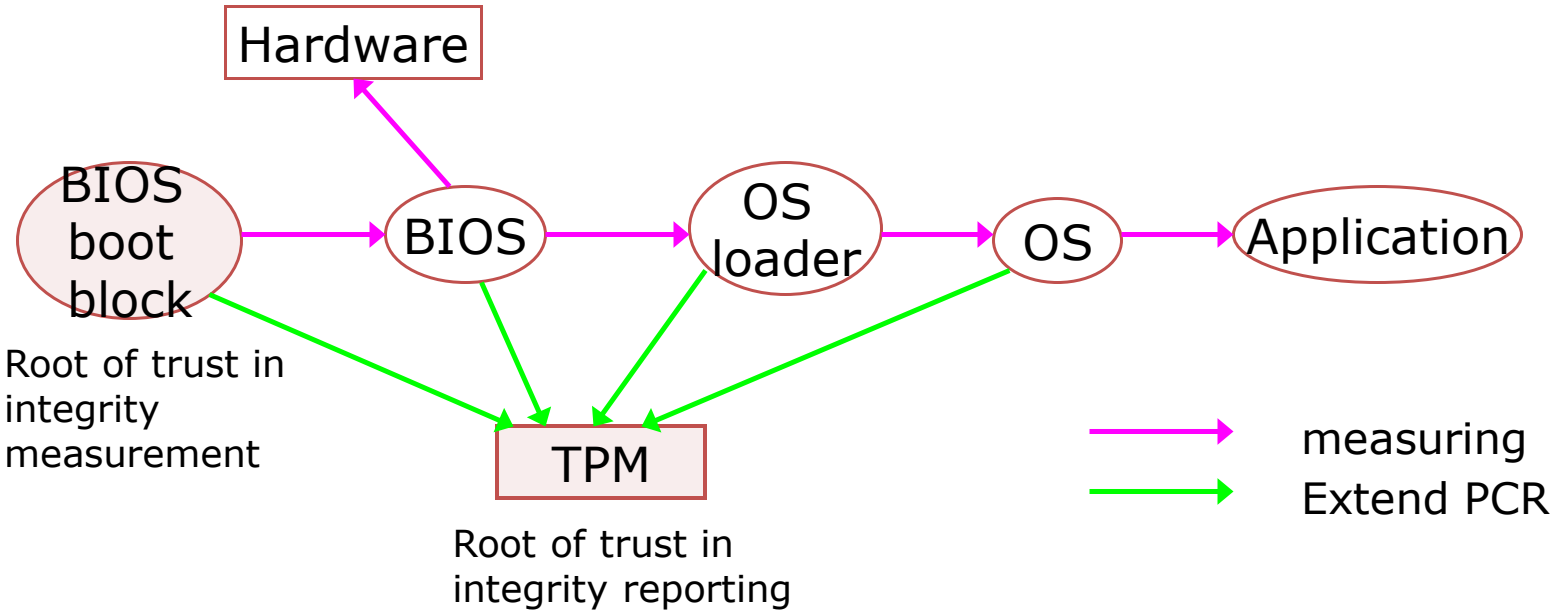
BIOS executes: Calls PCR_Extend(n, <MBR code>)

- Then runs MBR (master boot record), e.g. GRUB.

MBR executes: Calls PCR_Extend(n, <OS loader code, config>)

- Then runs OS loader … and so on

In a diagram



After boot, PCRs contain hash chains of booted software

Collision resistance of SHA256 ensures commitment

The main point

After boot completes, PCR registers measure the entire
software stack that booted on the machine:

- BIOS and hardware configuration
- Boot loader and its configuration
- Operating system
- Running apps

What would go wrong if TPM_Startup (ST_CLEAR) could be called at any time after boot?

Malicious OS could reset PCRs post-boot and then set them to a valid OS hash. PCRs would then look as if valid OS loaded.

TPM Counters

- TPM must support at least four hardware counters
  - Increment rate: every 5 seconds for 7 years.
- Applications:
  - Provide time stamps on blobs.
  - Supports “music will pay for 30 days” policy.

Trusted Computing

Attestation

Attestation: what it does

Goal: prove to remote party what software loaded on my machine

Good applications:

- Bank allows money transfer only if customer's machine runs "up-to-date" OS patches
- Enterprise allows laptop to connect to its network only if laptop runs "authorized" software
- Gamers can join network only if their game client is unmodified

DRM: MusicStore sells content for authorized players only.

Attestation: how it works

Recall: EK private key on TPM.

- Cert for EK public-key issued by TPM vendor.

Step 1: Create Attestation Identity Key (AIK)

- Details not important here
- AIK Private key known only to TPM
- AIK public cert issued only if EK cert is valid

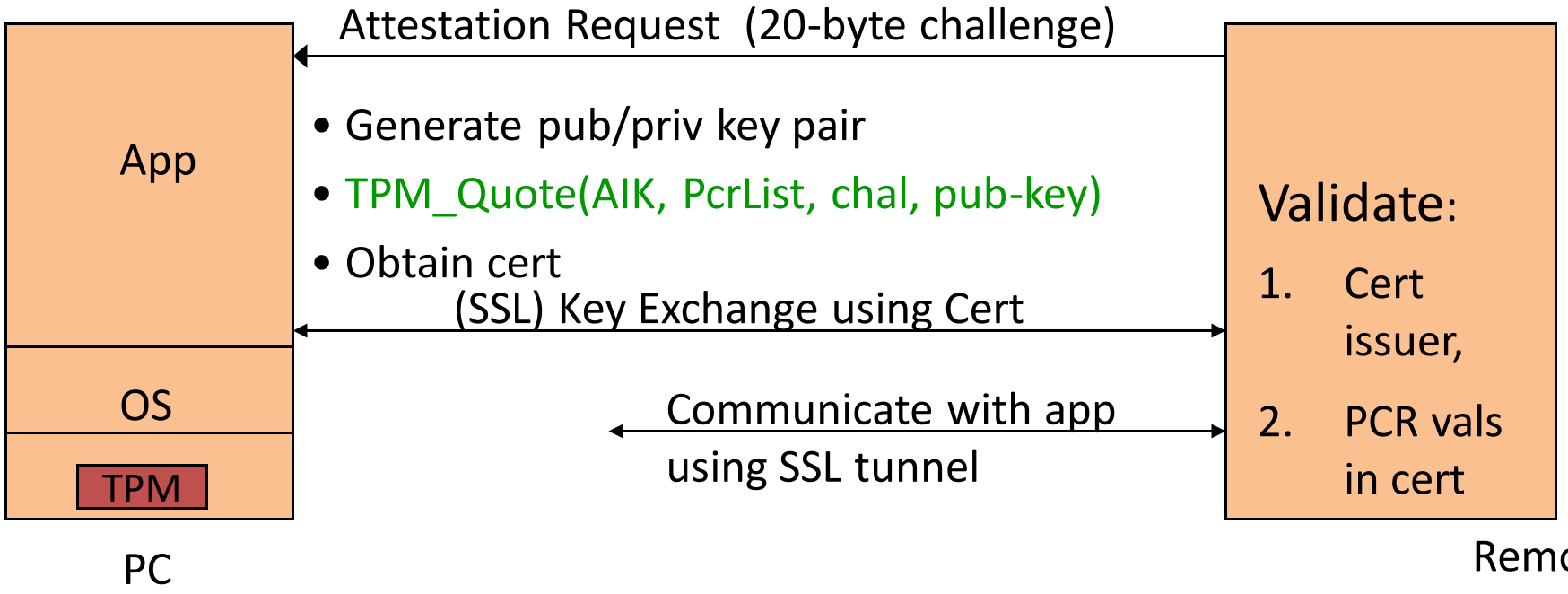
Attestation: how it works

Step 2: sign PCR values (after boot) with TPM_Quote.
Arguments:

- keyhandle: which AIK key to sign with
- KeyAuth: Password for using key `keyhandle'
- PCR List: Which PCRs to sign.
- Challenge: 20-byte challenge from remote server
  - Prevents replay of old signatures.
- Userdata: additional data to include in sig.

Returns signed data and signature.

Attestation: how it works



- Attestation typically includes key-exchange
- App must be isolated from rest of system

What would go wrong if communication between app. and server were done in the clear?

User can reboot machine after attestation and run arbitrary software pretending to be app.

Trusted Computing

Attestation: challenges

1. Attesting to Current State

- Attestation only attests to what code was loaded.
- Does not say whether running code has been compromised.
  - Problem: what if Quake vulnerability exploited after attestation took place?
- Can we attest to the current state of a running system?
  - … or is there a better way?

2. Encrypted viruses

Suppose malicious music file exploits bug in video player.

- Video file is encrypted.
- TCG prevents anyone from getting video file in the clear.
- Can anti-virus companies study virus without seeing its code in the clear?
- How would you solve this?

3. TPM Compromise

Suppose one TPM Endorsement Private Key is exposed

- Destroys all attestation infrastructure:
  - Embed private EK in TPM emulator.
  - Now, can attest to anything without running it.

⇒ Certificate Revocation is critical for TCG Attestation.

Intel SGX

SGX: Goals

Extension to Intel processors that support:

- **Enclaves**: running code and memory isolated from the rest of system
- **Attestation**: prove to local/remote system what code is running in enclave
- **Minimum TCB**: only processor is trusted
  nothing else: DRAM and peripherals are untrusted
  ⇒ all writes to memory must be encrypted

Applications

Server side:

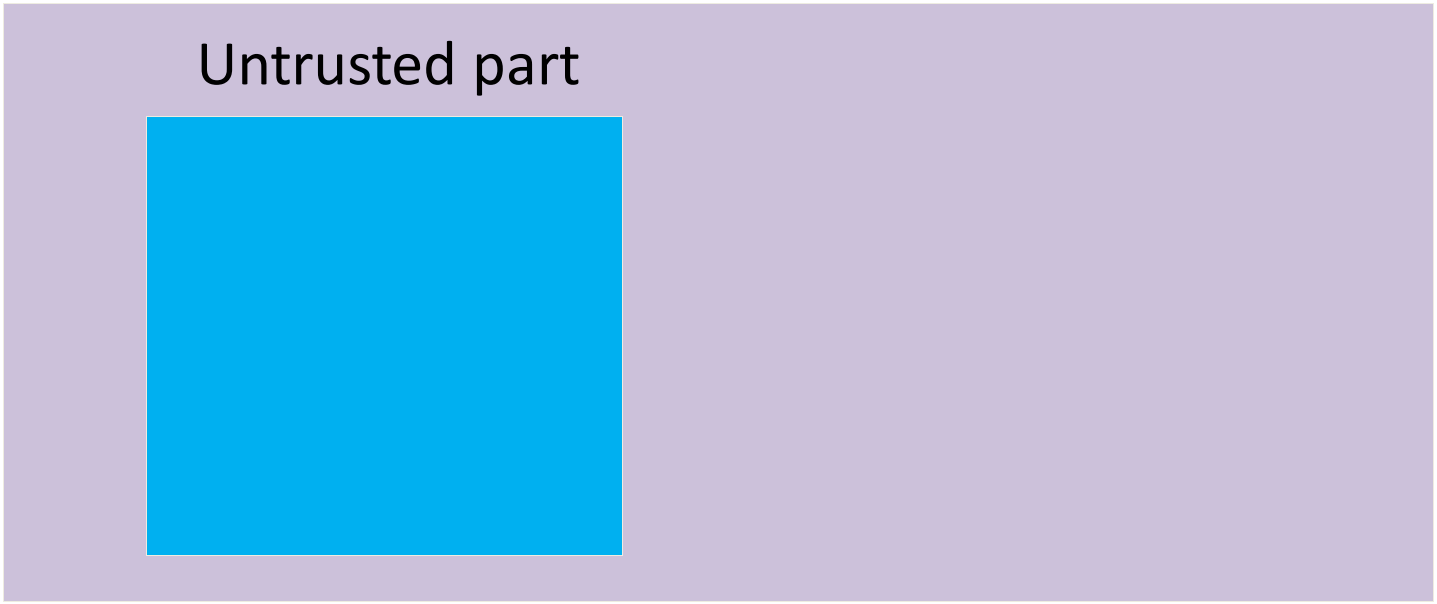
- Storing a Web server HTTPS secret key
  secret key only opened inside of an enclave
  malware cannot get the key
- Running a private job in the cloud: job runs in enclave
  Cloud admin cannot get code or data of job

Client side:

- Hide anti-virus (AV) signatures:
  AV signatures are only opened inside an enclave
  not exposed to adversary in the clear

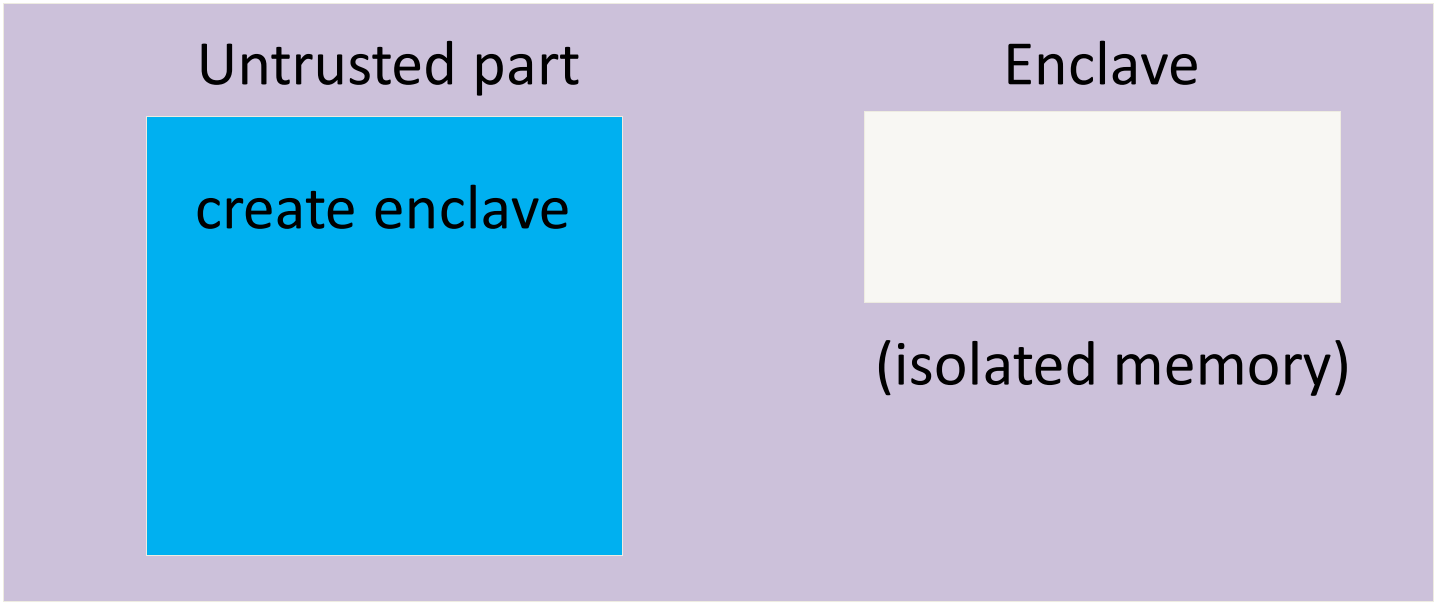
How does it work?

An application defines part of itself as an enclave



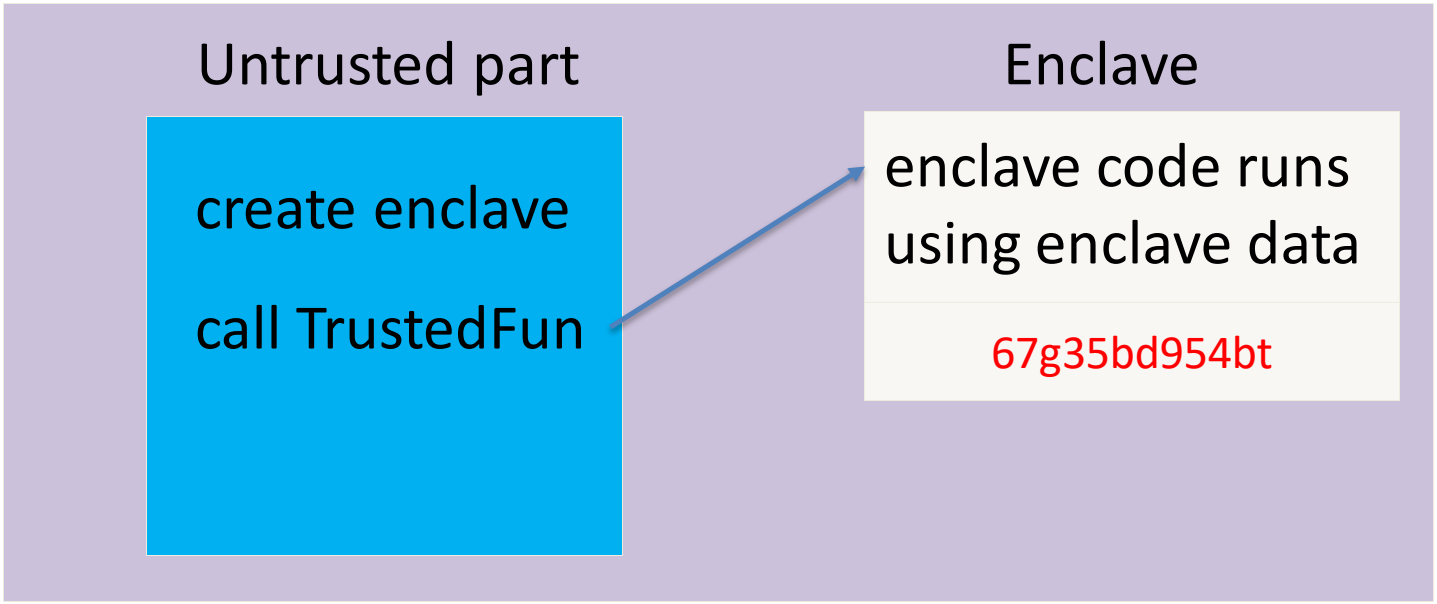
How does it work?

An application defines part of itself as an enclave



How does it work?

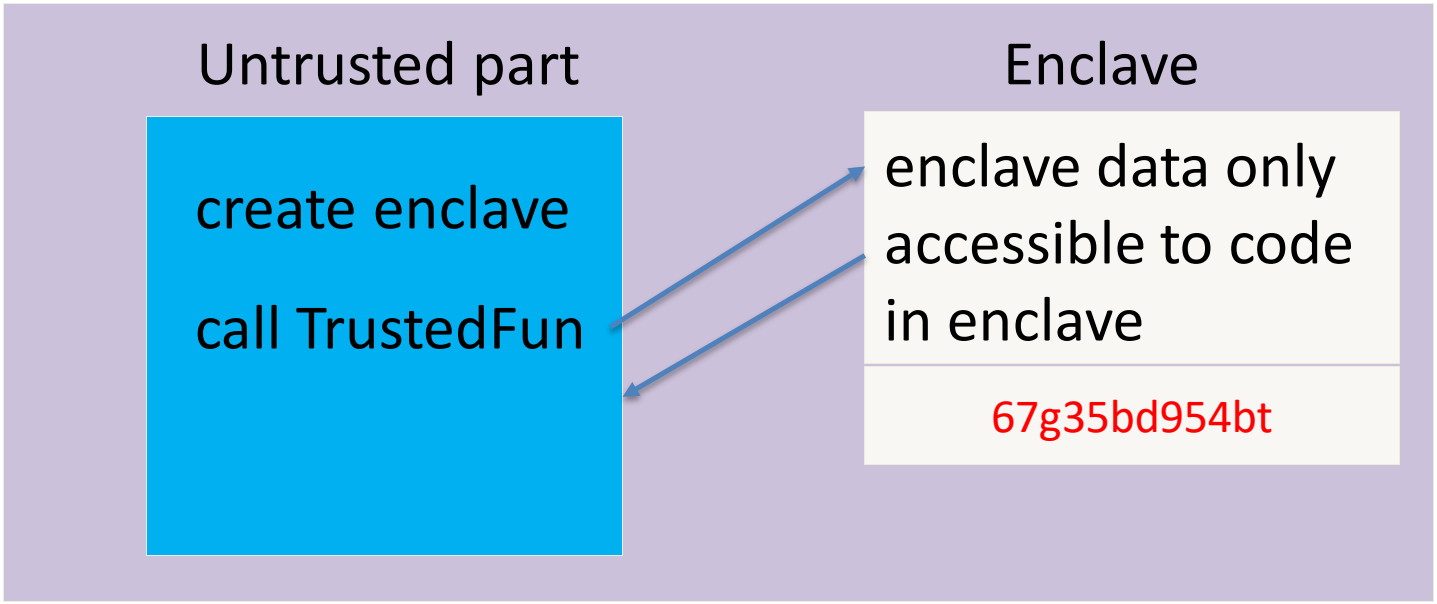
An application defines part of itself as an enclave



Application

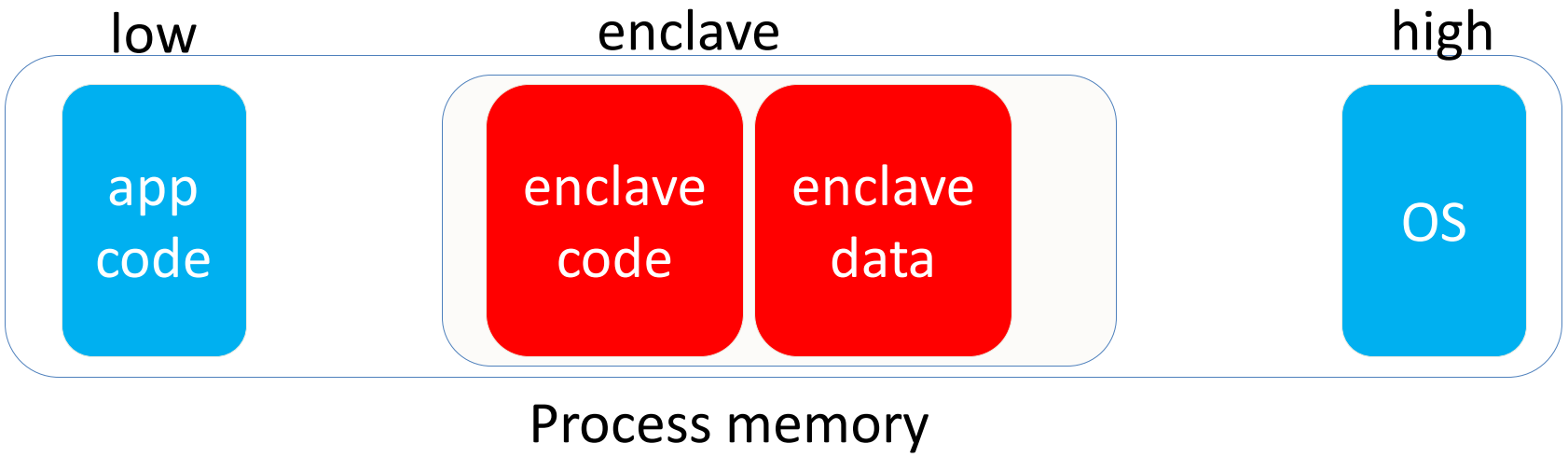
How does it work?

An application defines part of itself as an enclave



How does it work?

Part of process memory holds the enclave:



Creating an enclave: new instructions

- **ECREATE**: establish memory address for enclave
- **EADD**: copies memory pages into enclave
- **EEXTEND**: computes hash of enclave contents (256 bytes at a time)
- **EINIT**: verifies that hashed content is properly signed
  if so, initializes enclave (signature = RSA-3072)

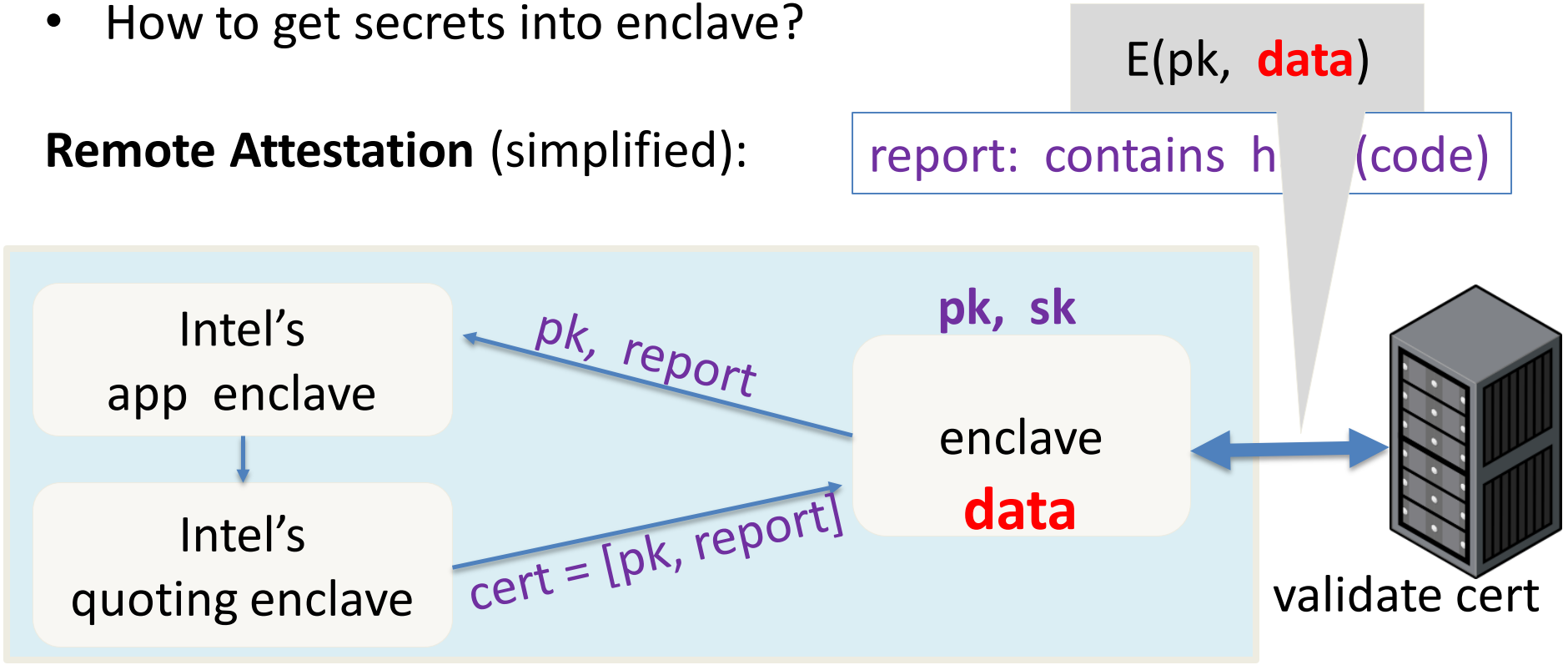
- **EENTER**: call a function inside enclave
- **EEXIT**: return from enclave

Provisioning enclave with secrets: attestation

The problem: enclave is in the clear prior to activation (EINIT)

- How to get secrets into enclave?

Remote Attestation (simplified):



Summary

SGX: a powerful architecture for managing secret data

- Enables processing of data that cannot be read by anyone, except for code running in enclave
- Minimal TCB: nothing trusted except for x86 processor
- Not suitable for legacy applications

THE END