Assignment 1: Trusted Computing

mq1n22

---

part1

1

```
1. Crtpto: TPM 1.2 specification only allows the use of RSA and SHA-1 hash
algorithms, while TPM 2.0 enables greater encryption like SHA-2, SHA-3,
and others
2. Storage: TPM 1.2 anly support SHA-1 bank, but TPM 2.0 support more
banks and support for SHA2,3. TPM 1.2's area used to store acess is small
while the area of TPM 2.0 is larger.
3. Ownership/Policy: the platform owner of TPM1.2 is single but TPM2.0 is
multiple and can change on platformAuth. The way to change SRK and RTR is
different.
4. Integration: TPM 2.0 is available as a discrete (dTPM) silicon
component in a single semiconductor package, an integrated component
incorporated in one or more semiconductor packages - alongside other logic
units in the same package(s), and as a firmware (fTPM) component.
```

the main difference is TPM2.0 support more advanced cryptographic functions

---

2

```
1. The Storage Hierarchy is used for less sensitive operations, such as
storing keys and data used by the platform software and applications. It
can be easily accessed and used by the software without requiring strong
privacy protection.
The Endorsement Hierarchy is used for highly sensitive operations that
require strong privacy protection, such as digital signatures or
encryption.
In summary, while the Storage Hierarchy is intended for non-privacy-
sensitiveoperations the Endorsement Hierarchy is designed to providestrong
privacy protection for highly sensitive data and operations.
```

3

~~~~
```
# Create a primary key under the owner hierarchy
~~~~

~~~
tpm2_createprimary -C e -c primary.ctx

# Create a child key under the primary key
tpm2_create -G rsa -u rsa.pub -r rsa.priv -a
'fixedtpm|fixedparent|sensitivedataorigin|
userwithauth|restricted|decrypt' -C primary.ctx

# Load the child key into the TPM
tpm2_load -C primary.ctx -u rsa.pub -r rsa.priv -c rsa.ctx
```
In this way, I can generate a subkey in TPM2.0 that is specially used for
encryption but not for signature, and only TPM2.0 can access and use it.
~~~

4

```
TPM 2.0 supports virtualization extensions that allow multiple virtual
machines to share a physical TPM. This is done by using a virtual TPM
(vTPM), which is a software emulation of a TPM that provides each virtual
machine with its own key hierarchy and storage space. The vTPM can use the
physical TPM's features to protect its own keys and data, but it keeps
them separate from other vTPMs and the physical TPM.
Yes, each user can own their respective hierarchies.
```

5

1. PCR0: The firmware code that runs before exiting boot services.

   PCR1: The platform configuration options, such as firmware settings and boot order.

   PCR2: The code of the option ROMs and UEFI drivers.

   PCR3: The configuration data of the option ROMs and UEFI drivers.

   PCR4: The code of the OS loader and OS kernel.

   PCR5: IPL Configuration.

   PCR6: State transition.

   PCR7: The configuration data of OS startup applications and drivers.

   These measurements can help verify that the system has booted with trusted software components and has not been tampered with by malicious actors.

2. > tpm2_quote -c key.ctx -l sha256:0,1,2,3,4,5,6,7 -q "this is some additional data" -n 0xdeadbeef -m quote.msg

The "-c" option specifies the handle of the key that will be used to sign the quote, which in this case is "key.ctx".

The "-l" option specifies the hash algorithm that will be used to compute the PCR digest values that will be included in the quote. In this case, the SHA-256 algorithm is being used to compute the digest values of PCR registers 0 through 7 (inclusive).

The "-q" option specifies any additional data that will be included in the quote. In this case, the string "this is some additional data" is being included.

The "-n" option specifies a nonce value that will be included in the command. Nonces are used to ensure freshness and prevent replay attacks.

3.
    1. magic: the indication that this structure was created by a TPM always TPM2_GENERATED_VALUE
    2. type: The type of the attestation structure. It is TPM2_ST_ATTEST_QUOTE for the type that has the PCR information and being discussed at length here in.
    3. qualifiedSigner: Qualified Name of the signing key. The term qualified name is the digest of all the Names of all the ancestor keys back to the Primary Seed at the root of the hierarchy.
    4. extraData: External information supplied by caller. The NONCE generated by the “Service-Provider“ is added here in this field.
    5. Clock: The time in milliseconds during which the TPM has been powered. This value is reset to zero when the Storage Primary Seed is changed TPM2_Clear.
    6. resetCount: The number of occurrences of TPM Reset since the last TPM2_Clear.
    7. restartCount: The number of times that TPM2_Shutdown or _TPM_Hash_Start have occurred since the last TPM Reset or TPM2_Clear.
    8. Safe: Indicates that no value of Clock greater than the current value of Clock has been previously reported by the TPM. Set to YES on TPM2_Clear.
    9. firmwareVersion: TPM vendor-specific value identifying the version number of the firmware.
    10. pcrSelect: The information on algID, PCR selected, and the digest.
    11. count: The number of selection structures. A value of zero is allowed. This indicates the count of selected PCR banks (SHA1, SHA256, etc.)
    12. pcrSelections: This is a list of PCR selection structures. hash: The hash algorithm associated with the selection bank.
    13. sizeofSelect: The size in octets of the pcrSelect array. This indicates number of bytes required to represent all the PCRs in a bank. Every PCR is represented as a bit.
    14. pcrSelect: The bit map of selected PCR (the least significant byte first)
    15. pcrDigest: The digest of selected PCR using hash algorithm of the signing key.

    By examining these data, I can ensure that the quote is authentic, consistent, and accurate. I can also compare the PCR values with a reference baseline or a policy to determine if the system meets certain security requirements or expectations.

4. The nonce is necessary to ensure freshness and prevent replay attacks. Use a combination of a timestamp and a random number as the nonce, such as 64 bits of timestamp and 64 bits of random number, so that you can check if the timestamp is within five days.

---

part2 Secure Boot

1

```
Secure Boot is a security feature that helps ensure that a device boots
using only software that is trusted by the original equipment
manufacturer. It prevents malicious software from loading during the boot
process and compromising the system integrity. Any operating system or
hardware driver that wants to be loaded on this motherboard must pass the
authentication of these public keys.Secure Boot is necessary because it
provides a foundation for platform trust and protects the device from
unauthorized modifications. Without Secure Boot, an attacker could install
malware such as rootkits or bootkits that can bypass the operating system
security and gain full control of the device.
```

2

```
  Threats: Rootkits, bootkits, ransomware, backdoors, Trojan.
  Adversaries: Hackers, cybercriminals, insiders.
  Attacks: spoofing, Man-in-the-middle (MITM), replay attack.
```

3

```
Verified boot and measured boot are two different approaches to ensure the
integrity and security of the boot process. Measured boot stores the
measurements for later verification and Verified boot verifies that each
measured component is appropriate per theplatform boot policy.Verified
boot is better, because in Verified boot verification occurs immediately
after measurement which means immediate notification of bad entity and
stop bad entity from running. Verified also means that if verification
fails, boot fails.
```

4

```
  a. AES, RSA, Elliptic Curve Cryptography, Secure Hash Algorithm.

  b. Yes, the engineer requires a root of trust for a secure boot process.
The purpose of a root of trust is to establish a chain of trust that
ensures that only trusted and authorized components are used in the
system,A component that must alwaysbehave in the expected manner,because
its misbehaviour cannotbe detected.
  A root of trust is necessary for a secure boot process because it
provides a secure foundation for the system's security. It is the first
```

```
element in a trust chain.

  c. non-volatile storage. Root of trust persists even when the power is
turned off. A root of trust needs to have area to store sensitive
information such as cryptographic keys, digital certificates, and
measurements of the boot process. The storage should be tampered-
resistant.

  d. Platform Key (PK), Key Exchange Key (KEK), Signature Database (db),
Forbidden Signature Database (dbx).

  e.
     1. Using an external TPM that is connected to the CPU via a secure
interface.
     2. Using a software-based root-of-trust
     3. Using a hardware security module (HSM) that provides cryptographic
services and key management

  f.
     1. Press the power button, the firmware(BIOS or UEFI) start.
     2. Power-On Self-Test (POST)
     3. The UEFI firmware checks if Secure Boot is enabled
     4. If Secure Boot is enabled, the UEFI firmware verifies the digital
signature of the bootloader1 using the keys stored in the Platform Key
(PK) and Key Exchange Key (KEK) variables2
     5. If the signature is valid, the UEFI firmware loads and executes
the bootloader1, which is responsible for loading the bootloader2
     6. The bootloader1 then checks if Trusted Boot is enabled, which is a
feature that verifies every component of the Windows startup process using
the keys stored in the key database (db) and forbidden signatures database
(dbx) variables.
     7. The bootloader1 verifies the digital signature of the bootloader2
using the keys in db and dbx
     8. If the signature is valid, the bootloader1 loads and executes the
bootloader2
     9. The bootloader2 then hands over execution to the Windows kernel,
which completes the boot processs
```

---

part3

1

```
Intel ME
```

2

```
UEFI is usually stored on a NOR-based EEPROM that is located on the
mainboard or it may be loaded from a hard drive or network share at boot.
```

3

```
UEFI Secure Boot specifies four key databases, which are used in a chain
of trust. The databases are stored in the UEFI variable store. In user
mode, the secure variables can only be updated with a valid signature from
the corresponding key
```

4

```
The OS cannot access the UEFI keys location directly, because they are
stored in UEFI runtime variables, which are special memory locations that
can only be accessed by the firmware and the operating system through UEFI
runtime services.
```

5

1. The UEFI specification addresses revocation by using using UEFI Revocation List Files. These files are used to update the Secure Boot Forbidden Signature Database (dbx). The dbx contains a list of signatures of unauthorized UEFI applications and drivers that cannot be loaded during the boot process.
2. To guarantee that a specific cryptographic key is never used in the secure boot process, the signature of the key should be added to the dbx variable. This will prevent any UEFI component that has been signed by that key from being loaded during the boot process. However, we should make sure the UEFI firmware is not compromised,

6

```
I will delete the Platform Key (PK). Because the PK is the root of trust
for UEFI Secure Boot and is used to verify the signature of the Key
Exchange Key (KEK).If the PK is deleted, the firmware will not be able to
verify or update the KEK, which means that the Signature Database (db) and
the Forbidden Signature Database (dbx) cannot be verified or updated
either. This would allow me to load any UEFI application or driver that
has a valid signature, regardless of whether it is authorized or not.
```