Measurement & Attestation

1. System Boot

a.

The BIOS/UEFI sets up the hardware so the operating system can function, e.g., cpu, IO, memory.

b.

Data can be encrypted using a sealed key. The key is only going to be unsealed iff the code is measured to be trusted, hence, an attacker controlled code will be noticed and the key will not be unsealed.

c.

No, assume an attacker running concurrently to a benign application using a sealed key. Unless the benign process is isolated in a trusted execution environment and no side channel vulnerabilites are present, the unsealed key might end up in memory where the malicious process might be able to retrieve it.

d.

Yes, a BIOS is still needed for the reasons given in 1a). What late launch enables is to attest a state without needing to reboot and including all the previous state.

2. Remote attestation

a.

Binary attestation is more sensitive to changes, such the amount of allowed hashes potentially explodes easily. On the other hand, a third party is necessary to authenticate the properties and this may pose a pricacy concern.

b.

Yes, it can be run, however, since the tpm needs to be powered on simultaneously with the cpu, the state will not be trusted anymore.

c.

No, this system cannot attest itself as the digest will depend on the bios. Having a valid or invalid bios will change the digest.

d.

While this proves that the code running is the code we want to run, the result might still be invalid, e.g., due to implementation mistakes.

3.

a.

H(H(0 || TPM2_CC_PolicyOR ||

<

H(H(0 ‖ PolicyPCR ‖ 1 ‖ mOS)

‖ PolicyPCR ‖ 2 ‖ mA_1),

H(H(0 ‖ PolicyPCR ‖ 1 ‖ mOS)

‖ PolicyPCR ‖ 2 ‖ mA_2)

>)

‖ PolicyCommandCode ‖ RSA_Decrypt),

b.

v11 ← TPM2_PolicyPCR(1, mOS)

v12 ← TPM2_PolicyPCR(2, mA_1)

v22 ← TPM2_PolicyOR(<

(H(H(0 ‖ PolicyPCR ‖ 1 ‖ mOS)

PolicyPCR ‖ 2 ‖ mA_1),

(H(H(0 ‖ PolicyPCR ‖ 1 ‖ mOS)

PolicyPCR ‖ 2 ‖ mA_2)

>)

v23 ← TPM2_PolicyCommandCode(RSA_Decrypt)

TPM2_RSA_Decrypt(k_1, c)

c.

Difficult to modify. If an application changes, the bounded digest needs to be changed such that it can still access the key. But then all applications need to incorporate that change. Using PolicyAuthorize enables the developer to simply sign modified applications with its private key and now the key is boud to that public key.

d.

H(H(0 ‖ TPM2_CC_PolicyAuthorize ‖ H(PK_D)) ‖ PolicyCommandCode ‖ RSA_Decrypt)

e.

v11 ← TPM2_PolicyPCR(1, mOS)

v12 ← TPM2_PolicyPCR(2, mA_2)

v22 ← TPM2_PolicyAuthorize(PK_D)

v23 ← TPM2_PolicyCommandCode(RSA_Decrypt)

TPM2_RSA_Decrypt(k_1, c)

f.

Using the methods taught in the lecture, it is not possible since there is no revocation mechanism. However, it is possible using the NV counter mechanism.