Sealing data/Secure storage

Peng Xu

June 4, 2019
Writing a sealing application for Intel SGX

1. What is Digital rights management (DRM)?
   ▶ Definition: a set of access control technologies for restricting the use of proprietary hardware and copyrighted works
   ▶ TEE-based DRM mechanism

2. Sealed Data in Intel SGX
   ▶ Motivation
     ▶ When an enclave is instantiated, the hardware provides protections (confidentiality and integrity) to its data
     ▶ When the enclave process exits, the enclave will be destroyed and any data that is secured within the enclave will be lost
     ▶ The enclave must make special arrangements to store the data outside the enclave in order to be re-used later
Writing a sealing application for Intel SGX

1. What is Digital rights management (DRM)?
   ▶ Definition: a set of access control technologies for restricting the use of proprietary hardware and copyrighted works
   ▶ TEE-based DRM mechanism

2. Sealed Data in Intel SGX
   ▶ Motivation
     ▶ When an enclave is instantiated, the hardware provides protections (confidentiality and integrity) to its data
     ▶ When the enclave process exits, the enclave will be destroyed and any data that is secured within the enclave will be lost
     ▶ The enclave must make special arrangements to store the data outside the enclave in order to be re-used later
   ▶ Two policies for Seal Keys
     ▶ MRENCLAVE: Sealing to the Enclave Identity
     ▶ MRSIGNER: Sealing to the Sealing Authority Identity
SGX-assisted DRM application

1. DRM-APP
   - App.cpp
   - ReplayProtectedDRM.cpp/.h
   - TimeBasedDRM.cpp/.h

2. DRM-Enclave
   - Enclave.cpp
   - Enclave.edl
   - Other files, like *.pem, config files
Sealing and unsealing operations

1. Several key APIs
   - `sgx_calc_sealed_data_size(...)`
   - `sgx_seal_data(...)` and `sgx_seal_data_ex(...)`
   - `sgx_unseal_data(...)` and `sgx_unseal_data_ex(...)"`
Sealing and unsealing operations

1. Several key APIs
   ▶ sgx_calc_sealed_data_size(…)
   ▶ sgx_seal_data(…) and sgx_seal_data_ex(…)
   ▶ sgx_unseal_data(…) and sgx_unseal_data_ex(…)

2. Sgx_seal_data() function: sealing the plaintext to ciphertext. The ciphertext can be delivered outside of enclave.
   ▶ Keys: MRENCLAVE and MRSIGNER
   ▶ Parameters:
     ▶ additional_MACtext_length - length of the plaintext data stream in bytes. The additional data is optional and thus the length can be zero if no data is provided
     ▶ p_additional_MACtext - pointer to the plaintext data stream to be GCM protected
     ▶ text2encrypt_length - length of the data stream to encrypt in bytes
     ▶ p_text2encrypt - pointer to data stream to encrypt
     ▶ sealed_data_size - Size of the sealed data buffer passed in
     ▶ p_sealed_data - pointer to the sealed data structure containing protected data
Sealing and unsealing operations

1. Sgx_unseal_data() function: Unseal the sealed data structure passed in and populate the MAC text and decrypted text buffers with the appropriate data from the sealed data structure.
   - Keys: MRENCLAVE and MRSIGNER
   - Parameters:
     - p_sealed_data - pointer to the sealed data structure containing protected data
     - p_additional_MACtext - pointer to the plaintext data stream which was GCM protected
     - p_additional_MACtext_length - pointer to length of the plaintext data stream in bytes
     - p_decrypted_text - pointer to decrypted data stream
     - p_decrypted_text_length - pointer to length of the decrypted data stream to encrypt in bytes
Similar with SGX: writing an application with optee

1. Host (Client Application)
   - host.c/.h
   - Makefile

2. TA (Trusted Application)
   - math.c/.h
   - Makefile

3. Both Host and TA sides are written in C
1. APIs
   - TEE_Malloc()
   - TEE_MemMove()
   - TEE_CreatePersistentObject()
   - TEE_WriteObjectData()
   - TEE_OpenPersistentObject()
   - TEE_ReadObjectData()
   - TEE_CloseAndDeletePersistentObject()
Similar with SGX: writing an application with optee

1. APIs
   - TEE_Malloc()
   - TEE_MemMove()
   - TEE_CreatePersistentObject()
   - TEE_WriteObjectData()
   - TEE_OpenPersistentObject()
   - TEE_ReadObjectData()
   - TEE_CloseAndDeletePersistentObject()

2. Keys
   - Secure Storage Key (SSK)
   - Trusted Application Storage Key (TSK)
   - File Encryption Key (FEK)
Key Generation

1. **HUK**: *Hardware Unique Key*
2. **SSK**: $HMAC_{SHA256}(HUK, ChipID || \text{\textbackslash staticstring})$
3. **TSK**: $HMAC_{SHA256}(SSK, TA_U UID)$
4. **PRNG**: *pseudo random number generator*
5. **FEK**: $f(P RNG)$
Meta Data Encryption

Figure: Meta Data Flow
Figure: Block Data Encryption
Questions?