Remote attestation

Peng Xu

May 14, 2019
1. What is attestation?

- Attestation is a mechanism for software to prove its identity
- Attestation is the process of demonstrating that a piece of software has been established on a platform

2. Why we need attestation?

- To prove to a remote party that your operating system and application software are intact and trustworthy

3. How can we implement attestation by Intel SGX?

- Local (Intra-platform) attestation
  - a mechanism for creating a basic assertion between enclaves running on the same platform
- Remote (Inter-platform) attestation
  - a mechanism that provides the foundation for attestation between an enclave and a remote third party
1. What is attestation?
   ▶ Attestation is a mechanism for software to prove its identity
   ▶ Attestation is the process of demonstrating that a piece of software has been established on a platform

2. Why we need attestation?
   ▶ To prove to a remote party that your operating system and application software are intact and trustworthy
1. What is attestation?
   ▶ Attestation is a mechanism for software to prove its identity
   ▶ Attestation is the process of demonstrating that a piece of software has been established on a platform

2. Why we need attestation?
   ▶ To prove to a remote party that your operating system and application software are intact and trustworthy

3. How can we implement attestation by Intel SGX?
   ▶ Local (Intra-platform) attestation
     ▶ a mechanism for creating a basic assertion between enclaves running on the same platform
   ▶ Remote (Inter-platform) attestation
     ▶ a mechanism that provides the foundation for attestation between an enclave and a remote third party
SGX Application - Remote Attestation

Figure: remote_attestation

https://software.intel.com/en-us/node/702987
1. A client’s enclave can attest to a remote entity that it is trusted.
2. establishes an authenticated communication channel with that entity.
3. the client’s enclave proves the following:
   ▶ Its identity
   ▶ That it has not been tampered with
   ▶ That it is running on a genuine platform with Intel SGX enabled
4. the remote server can safely provision secrets to the enclave.
Basic Concepts

1. Cryptography
   ▶ Private key/symmetric cryptography

**Figure:** Symmetric key encryption
Basic Concepts

1. Cryptography
   - Public key/asymmetric cryptography

Figure: Asymmetric_key_encryption
1. Signature

- Public key/asymmetric cryptography

**Figure:** Private_key_signing
Basic Concepts

1. Client-Server protocol - Sigma protocol
   - Commitment, challenge and response

Sigma Protocol (Basic)

Output from DH-value $g^{xy}$:
- encryption key $K_e$
- mac key $K_m$

Figure: Sigma Protocol
SGX Application - Remote Attestation

1. Create enclave and call enclave_init_ra(b_pse)
2. sgx_create_pse_session
3. sgx_ra_init(g_sp_pub_key, b_pse)
4. sgx_close_pse_session
5. context
6/7. msg0
8. sgx_ra_get_msg1(context, enclave_id, sgx_ra_get_ia)
9. msg1
10. ra_network_send_receive
11. msg2
12. sgx_ra_proc_msg2(context, enclave_id, sgx_ra_proc_msg2_trusted_t, sgx_ra_get_msg3_trusted_t, msg2)
13. msg3
14. ra_network_send_receive
15. result
1. Attestation application structure
   ▶ ISV_Enclave(secure_world)
   ▶ ISV_App(Non-secure_world)
   ▶ Service Provider
   ▶ Makefile
   ▶ Include
SGX Application - Attestation

1. Attestation application structure
   - ISV Enclave(secure_world)
   - ISV App(Non-secure_world)
   - Service Provider
   - Makefile
   - Include

2. Basic concepts
   - Public key/Private key
   - Signature
   - Service provider
   - etc.
Logical functionalities - Preparing msg2

1. Checks the values in the request
2. Generates its own DHKE parameter,
3. Sends a query to the IAS to retrieve the Signature Revocation List (SigRL) for the Intel EPID GID sent by the client.
Logical functionalities - Processing msg2

1. Generate a random EC key using the P-256 curve. This key will become Gb.

2. Derive the key derivation key (KDK) from Ga and Gb:
   ▶ Compute the shared secret using the client’s public session key, Ga, and the service provider’s private session key (obtained from Step 1), Gb. The result of this operation will be the x coordinate of Gab, denoted as Gabx.
   ▶ Convert Gabx to little-endian byte order by reversing its bytes. Perform an AES-128 CMAC on the little-endian form of Gabx using a block of 0x00 bytes for the key.
   ▶ The result of 2.3 is the KDK.

3. Derive the SMK from the KDK by performing an AES-128 CMAC on the byte sequence:
   ▶ \( \text{sgx\_ra\_msg2\_t* p\_msg2\_body} = (\text{sgx\_ra\_msg2\_t*})(\text{uint8\_t* p\_msg2\_full} + \text{sizeof(ra\_samp\_response\_header\_t)}) \)
   ▶ PRINT_ATTESTATION_SERVICE_RESPONSE()
Logical functionalities - Processing msg2

1. Verifies the service provider signature.
2. Checks the SigRL.
3. Returns msg3, which contains the quote used to attest that particular enclave.

- `sgx_ra_proc_msg2(context, enclave_id, sgx_ra_proc_msg2_trusted, sgx_ra_get_msg3_trusted, p_msg2_body, p_msg2_full_size, &p_msg3, &msg3_size);
- `sgx_status_t sgx_ra_proc_msg2_trusted();`
- `sgx_status_t sgx_ra_get_msg3_trusted();`
import *;

tenclave {
    from "sgx_tae_service.edl" import *
    trusted {
        public sgx_status_t sgx_ra_get_msg3_trusted();
    }
};
Question?

Questions?