SGX attestation process

Research seminar in Cryptograpy

Author: Hiie Vill
Supervisor: Pille Pullonen

Introduction

Software Guard Extensions (SGX) is a technology, the main function of which is to establish special protected software containers, also known as enclaves. These enclaves can be used for provisioning sensitive parts of a software executable in order to protect them from malicious entities. In order to verify remotely that an application is running securely within an enclave, a remote attestation must be performed. This report gives an overview of the attestation process, the background related to it and the necessary requirements for remote attestation [1][2].

1. Software Guard Extensions

Software Guard Extensions (SGX) is a technology introduced by Intel in 2015, which provides hardware assisted security for the application layer. In itself, SGX is a set of processor extensions for establishing a protected execution environment, referred to as an enclave, and the software related to it. More specifically, enclaves are protected areas of execution in memory, providing a trusted execution environment, even on a compromised platform. In order to protect an application from malicious entities, the software's code, data and stack are stored within the enclave and protected by hardware enforced access control policies, making the application inaccessible to any malware on the platform. This means that SGX enables applications to defend themselves, protecting any sensitive data used by the application (for example credentials and cryptographic keys), while retaining its integrity and confidentiality [1][2].

2. Preliminaries

In this section, some terminology will be briefly explained that is used in the remote attestation process in order to familiarize the reader with the background information.

2.1 Hardware instructions

In order to carry out remote attestation, two hardware instructions are called during the process. The two instructions are EREPORT and EGETKEY, which are both provided by the Intel SGX Architecture. EREPORT provides cryptographical structures, also known as REPORTs, special signed structures that bind a key to the hardware (the enclave). EGETKEY enables the enclave to access the REPORT key used in attestation. This REPORT key can then be useb for verifying REPORTs [1].

2.2 Sigma protocol

It should be noted that remote attestation uses a slightly altered Sigma protocol for assisting Diffie-Hellmann key exchange between the client (the application) and the service provider (the challenger). Sigma protocol is a proof that consists of commitment, challenge and response. As a result of this exchange between the client and the service provider, a shared key between the enclave and the challenger is produced that can be used for encypting secrets that are to be provisioned in the enclave. Once inside the enclave, these secrets could then be decrypted by the application [1].

2.3 Diffie-Hellmann key exchange

Diffie-Hellmann key exchange (DHKE) used during the attestation process is a method for exchanging keys over a public channel without leaking the actual key to other listeners [5].

2.4 EPID

Another term relevant to the attestation process is EPID – Intel Enhanced Privacy ID. It is an extension to an existing Direct Anonymous Attestation scheme with some additions, for example the use of SigRL, described in paragraph 2.5. EPID enables signing objects without leaving a trace that can be uniquely backtracked to the signer, making the signing process anonymous. This is done by dividing signers to groups (also known as EPID groups), based on their processor type. This way they create signatures with their own secret keys, but the signatures can be verified only with the public key of the group they belong to, making it possible to check that the signer belongs to the right group, but impossible to uniquely identify the signer [3][6].

2.5 SigRL

Signature Revocation List, also known as SigRL, is a list of untrustworthy signatures, signed by the revocation authority. Whenever a signature is determined to be untrustworthy by the revocation authority, the corresponding key is revoked and the signature is placed in the revocation list. SigRL is also one of the differences between Direct Anonymous Attestation scheme and EPID – it is present in the latter, but not in the former [3].

2.6 TCB

Intel Trusted Computing Base (TCB) is an entity responsible for protecting the secrets provisioned to the enclave (both the processor's hardware and the software running inside the enclave) [3].

2.7 Quoting enclave

The quoting enclave is a special enclave on every SQX processor and is tasked entirely with handling the remote attestation. It receives REPORTs from other enclaves, verifies them and signs them with an asymmetric key before returning the result, also known as a QUOTE, to the application [1].

3. Remote attestation

Remote attestation, an advanced feature of Intel SGX, is the process of proving that an enclave has been established in a secure hardware environment. This means that a remote party can verify that the right application is running inside an enclave on an Intel SGX enabled platform. Remote attestation provides verification for three things: the application's identity, its intactness (that it has not been tampered with), and that it is running securely within an enclave on an Intel SGX enabled platform. Attestation is necessary in order to make remote access secure, since very often the enclave's contents may have to be accessed remotely, not from the same platform [1].

For remote attestation, both symmetric and asymmetric key systems are used. The symmetric key system is used in intra-platform enclave attestation with only the quoting enclave and the EREPORT instruction (explained in paragraph 2.1) having access to the authentication key. Asymmetric key system is used for creating an attestation that can be verified from other platforms (inter-platform attestation). The attestation key itself is asymmetric. [1]

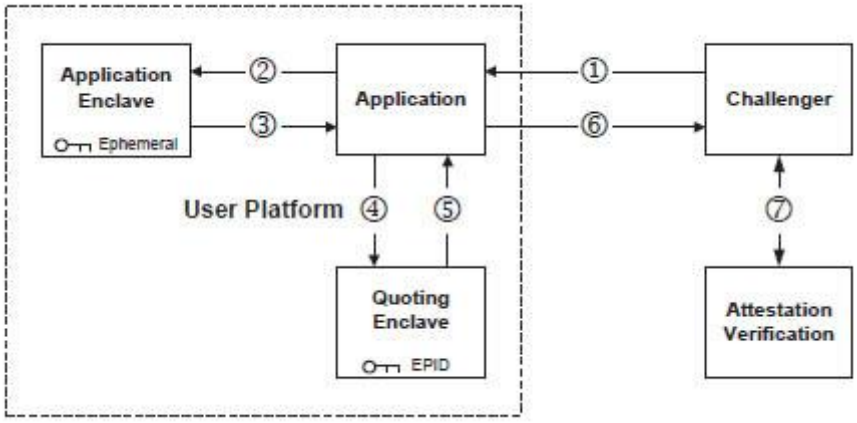

Image 1. Remote attestation example [1]

While *Image 1* only depicts seven major flows of communication for simplicity, in reality, messages are exchanged between the application and the challenger nearly at every stage, as shown in *Image 2.*

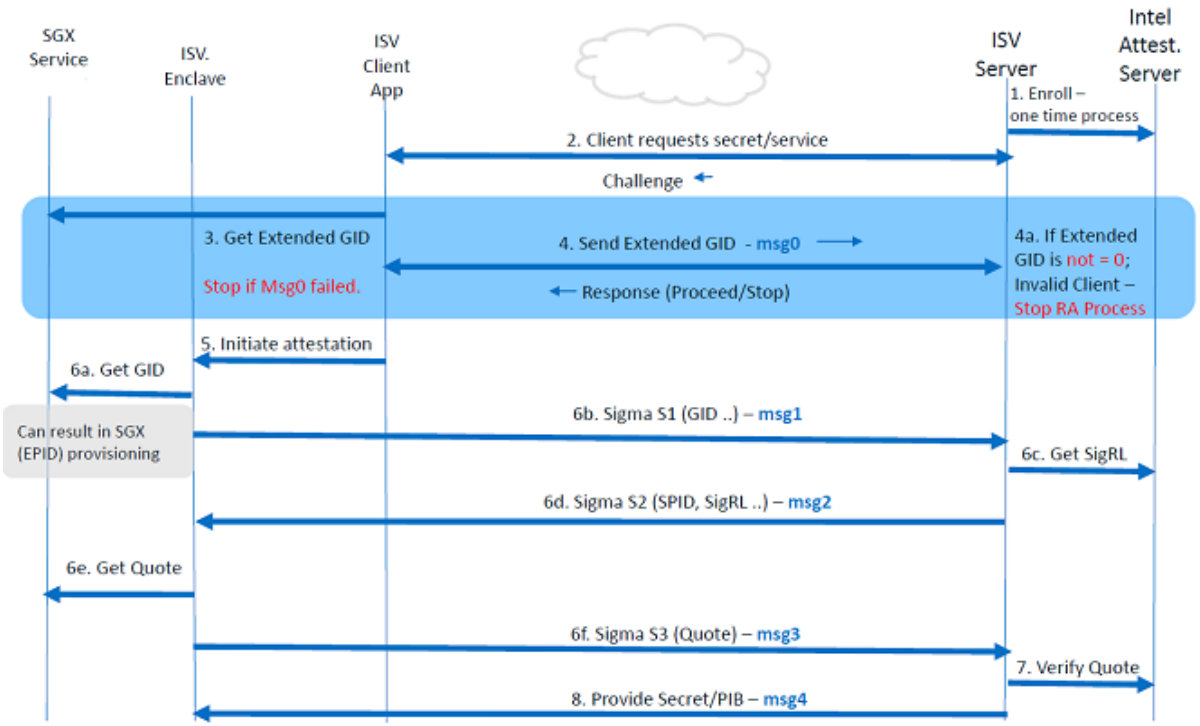

Image 2. Full remote attestation flow [4]

The attestation process consists of seven stages, encompassing several actors, namely the service provider (referred to as a challenger) on one platform, and the application, its enclave and its quoting enclave on another platform. A separate entity in the attestation process is Intel Attestation Service (IAS), which carries out the verification of the enclave [1][3].

In short, the seven stages of remote attestation comprise of making a remote attestation request (stage 1), performing a local attestation (stages 2-3), converting the local attestation to a remote attestation (stages 4-5), returning the remote attestation to the challenger (stage 6) and verifying the remote attestation (stage 7) [3].

The seven stages of the attestation flow are described in more detail in section 4.

Image 2 depicts the communication between the entities participating in the remote attestation process. The five special messages (msg0…msg4), built in the SGX software, which are also described in paragraph 4, are brought out here as well.

The seven stages cannot be mapped to the messages uniquely due to overlaps, but in a broad view, stage 1 of *Image 1* corresponds to number 2 in *Image 2*, stages 2-3 (roughly) to 5…6.d, stage 4 (roughly) to 6.e, stage 5 to 6e, stage 6 to 6f and stage 7 to 7 [1][4].

4. The stages of attestation

4.1 Stage 1

The attestation process begins with the application establishing a communication with the challenger – an entity that wishes to validate whether the application is running securely within

the enclaves. The challenger must be located remotely, meaning that it is not on the same platform as the application itself.

Firstly, after a communication is established by the application between the platform and the service providing system (the latter representing the challenger), the application asks the challenger to provision secrets. The challenger then replies by sending an attestation request to the application. The attestation request sent out is in form of a nonce (a random number generated for this one occasion) [1][4].

4.2 Stage 2

Every application is linked to its quoting enclave by storing its Enclave Identity (a cryptographic hash of the enclave's log that restricts access to the enclave's contents for unauthorized parties). In this stage, the application sends the challenge and the enclave identity to the application enclave.

In greater detail, this stage begins with the application performing ECALL (a call to the enclave) to initialize the remote attestation by executing *sgx_ra_init()*. This function takes the challenger's public key as an argument and creates a context for the Diffie-Hellmann key exchange (DHKE), which will take place later during the remote attestation. Optionally, it is also possible to use the Platform Services Enclave (another special enclave provided by Intel) during the attestation. The advantages of PSE include a secure calculation of the duration the secret shall be valid, and replay protection during nonce generation.

Next, the enclave returns a response to the application, and with it, the DHKE context. If the response from *sgx_ra_init()* was positive, the application calls *sgx_get_extended_epid_group_id()*, a method for getting the Extended Group ID (also known as Extended GID) of the Intel Enhanced Privacy ID (referred to as EPID), an anonymous signature scheme that used for identification. The application then sends a message (msg0) back to the challenger, containing the Extended GID. The format of that message is not restricted, but dependant on the challenger. Once the challenger receives the Extended GID, it has the option to verify whether it is legal and whether to commence the attestation or to halt the process. For example, in the case of Intel Attestation Service, only the Extended GID value 0 is considered legal [1][4].

4.3 Stage 3

In the application enclave, an ephemeral public key is generated - one that only lasts for a short period of time. This key is to be used by the challenger for provisioning secrets to the enclave. The enclave also generates a response to the challenger. The response and the ephemeral public key are tied together into a manifest and a hash digest is created of the combined key and response manifest, which is then sent back to the application in the form of a REPORT by calling EREPORT (a built-in hardware instruction that links the combo to the enclave).

Upon getting a green light from the challenger to proceed with the attestation, the application calls the method *sgx_ra_get_msg1()*, which takes as parameters the application's public key for the DHKE, the DHKE context found previously, and a pointer to the stub function *sgx_ra_get_ga()*, which computes the application-side DHKE secret. The application then sends a message (msg1) consisting of the aforementioned data to the challenger.

Having received the previous message from the application, containing information about DHKE, the challenger checks the values contained in the request and generates its own DHKE parameter. After that, the challenger queries the IAS for a Signature Revocation List (SigRL). The challenger's public key and SigRL data are combined together and sent to the application [1][4].

4.4 Stage 4

In the fourth stage, the application simply receives the REPORT, which included information about the key and its originating enclave, and forwards it to the quoting enclave to be signed [1].

4.5 Stage 5

The quoting enclave receives the REPORT and calls the hardware instruction EGETKEY to get its REPORT key. Using this REPORT key, the quoting enclave verifies the REPORT. After that, a QUOTE structure is created and signed by the quoting enclave, using its EPID key, making it only verifiable by IAS. QUOTE in itself is merely a signed REPORT. This QUOTE will be used to attest this particular application's enclave. Finally, the quoting enclave sends the signed QUOTE to the application [1][4].

4.6 Stage 6

In this stage, the application receives the QUOTE produced and signed by the quoting enclave, and forwards it back to the challenger.

In detail, the application calls the function *sgx_ra_get_msg3_trusted()*. This function's tasks are verifying the challenger's signature, checking the SigRL provided by the challenger, and generating a reply (msg3) to the challenger. The reply message will include the QUOTE that was produced by the quoting enclave. It will also include information about aforementioned PSE that may be used on the platform. This stage ends with the application sending msg3 to the challenger. [1][4]

4.7 Stage 7

In the final stage, the challenger uses the EPID public key certificate to validate the signature over the QUOTE.

Message-wise, the challenger receives that message msg3 sent by the application in stage 6, and proceeds to check any parameters contained in the message, such as the DHKE parameters and the application enclave's identity (embedded in the QUOTE). After a successful check, the challenger forwards the QUOTE and the signature to the IAS to be properly verified.

Once the IAS has verified the QUOTE and the challenger has received the verification results from IAS, the challenger generates a reply message (msg4) to the application. This message contains the attestation result and optionally (only if both the enclave and PSE are trusted by IAS) the secret that is to be provisioned within the now trusted enclave and can be encrypted by the shared key obtained during the DHKE phase earlier. [1][4]

5. Attestation requirements

In order to perform remote attestation, several requirements must be met. First of all, Intel Trusted Computing Base (TCB) must be represented by a special platform attestation key, which has to be provided. Secondly, In case of an upgrade is made to the platform, the attestation key must be replaced. Thirdly, unsafe parts must have no access to additional attestation keys. And lastly, the attestation process must guarantee user privacy – that it would be impossible to uniquely identify who created the signature. Fulfilling those four requirements will now be described in more detail, although the full descriptions can be read from [3].

5.1 The creation of attestation keys

Creating an attestation key representing the hardware and software TCB of Intel SGX consists of two stages – one stands for updating the processor hardware, the other stands for linking the software components of TBC and the hardware TBC key [3].

5.2 Replacing attestation keys

After a TCB upgrade, a new attestation key is requested. The platform is only allowed to join a new EPID group once it has been determined that none of the keys issued to that platform earlier have been revoked. If none have been revoked, the platform is assigned to a new EPID group [3].

5.3 Preventing compromised entities from accessing additional attestation keys

Compromised platforms (processors) are prevented from gaining access to additional attestation keys by applying the current revocation list before issuing new keys. If an attestation key had been assigned to that platform before, but has been revoked, then the process fails [3].

5.4 User privacy guarantee

User privacy is guaranteed by using Intel Enhanced Privacy ID, also known as EPID. With EPID, it is possible to sign objects (such as QUOTE) without identifying the platform or linking different signatures. No signer has their own public key, instead signers are divided into groups and the group's public key is used for verifying signatures. This prevents from uniquely identifying the real signer, only the group it belongs to [3].

Sources

[1] I. Anati, S. Gueron, S. Johnson, V. Scarlata, “Innovatiove Technology for CPU Based Attestation and Sealing“, https://software.intel.com/en-us/articles/innovative-technology-for-cpu-based-attestation-and-sealing (visited 25.05.2017)

[2] https://software.intel.com/en-us/sgx (visited 25.05.2017)

[3] S. Johnson, V. Scarlata, C. Rozas, E. Brickell, F. Mckeen, “Intel® Software Guard Extensions: EPID Provisioning and Attestation Services“, https://software.intel.com/sites/default/files/managed/ac/40/2016%20WW10%20sgx%20provisioning%20and%20attesatation%20final.pdf (visited 25.05.2017)

[4] https://software.intel.com/en-us/articles/intel-software-guard-extensions-remote-attestation-end-to-end-example (visited 25.05.2017)

[5] W. Diffie, M. Hellmann, “New Directions in Cryptography“, https://ee.stanford.edu/~hellman/publications/24.pdf (visited 25.05.2017)

[6] E. Brickell, J. Camenisch, L. Chen, “Direct Anonymous Attestation“, http://eprint.iacr.org/2004/205.pdf (visited 25.05.2017)