Overview

The objective of this document is to provide a quick primer on PoET and the Trusted Execution Environment it depends on, Intel SGX.

Intel SGX

SGX1: Instruction set for instantiating an enclave
SGX2: Instruction set providing additional capabilities

Full specification of SGX can be found at:
https://software.intel.com/sites/default/files/managed/48/88/329298-002.pdf

Enclave

Enclave: a protected area in an application's address space which provides confidentiality and integrity even in the presence of privileged malware.

"Enclave" can also be used to refer to a specific enclave that has been initialized with a specific set of code and data.

An enclave can:

- Prove its identity to a remote party (see Attestation)
- Request enclave-specific and platform-specific keys that can be used to protect keys and data that it wishes to store **outside the enclave**.

Creation

1. Code and data are loaded from non-enclave memory
2. Enclave Measurement is calculated and stored
3. Sealing Authority is verified and stored
4. Control is transferred to the enclave code

Access Control

- Memory within an enclave cannot be accessed when the enclave does not have control.
- Memory outside an enclave cannot be accessed while the enclave has control.
- Interactions between code and data within and outside enclaves is done through "Enclave Interface Functions": Enclave Calls (ECalls) and Out Calls (OCalls)
    - An enclave API must be defined prior to initializing a new enclave for passing arguments into an enclave.

    - A set of calls out of the enclave (OCalls) must be defined prior to initializing a new enclave.
    - A pre-processor generates “proxy” functions to handle interactions between untrusted application code and trusted enclave code.
    - @jjason : “So, you define the external interface (i.e., functions) to the enclave in an enclave definition language (think IDL). You run that through a pre-processor. It creates a "proxy" or "bridge" for each of the functions, resulting in two sets of files (.c/.h files for the untrusted side and .c/.h files for the trusted side) that you compile into the appropriate code. The code outside the enclave calls the untrusted stub functions, which will marshal arguments into and unmarshals return values out of the function. This is very similar to how applications interact with drivers (call `DeviceIoControl` on Windows, or `ioctl` on Linux/UNIX). In the enclave, there are the other side of the "proxy" or "bridge" which unmarshals arguments into and marshals return values out of the enclave.”

Enclave Measurement

- “While the enclave is being built, a secure log is recorded reflecting the contents of the enclave and how it was loaded. This secure log is [used to calculate] the enclave’s measurement.” [1]
- The enclave’s measurement is the is a SHA-256 digest of the internal creation log.
- This measurement is used as the identity of the enclave.
- A.k.a., Enclave Identity, MRENCLAVE

Sealing Authority

- “The Sealing Authority is an entity that signs the enclave prior to distribution, typically the enclave builder.” [1]
- Upon creation of an enclave, the enclave is presented with a signed certificate containing an Enclave Measurement and the public key of the Sealing Authority. The enclave verifies the signature for the certificate and then compares the “proposed” measurement in the certificate against the measurement it computed locally through creation. If they are the same, the enclave stores the public key as the Sealing Authority for the enclave.
- A.k.a., Sealing Identity, MRSIGNER

Execution

- Untrusted application code calls an untrusted proxy function defined by the enclave API
- Proxy function prepares input arguments for the enclave
- Proxy functions makes an ECall and transfers control to the trusted enclave code
- Trusted enclave code executes making OCalls as needed to interact with OS and other untrusted code
- Trusted enclave code prepares output data for untrusted application code

- Trusted enclave code transfers control back to untrusted application code

Attestation

> Attestation: the process of demonstrating that a piece of software has been properly instantiated on the platform

- Within SGX, attestation is, “the mechanism by which another party can gain confidence that the correct software is running within an enclave on an enable platform.” [1]

Enclave Reports

- Enclaves can generate a report which should be used for attestation.
- When creating a report, the measurement of the target enclave (the intended receiver of the report) is required.
- Reports contain:
    - The enclave’s identity (Sealing Authority and Enclave Measurement)
    - Additional enclave attributes
    - The trustworthiness of the hardware TCB?
    - Additional developer-defined information
    - A message authentication code (MAC)

Intra-Platform Attestation

Intra-Platform attestation refers to attestation between two enclaves on the same platform. That is, they are both running on the same Intel SGX enabled processor and host.

1. A report is created for a given target enclave on the same platform
    a. The MAC associated with the report can be verified using a symmetric key accessible from within the target enclave called the Report Key.
    b. Something is handled opaquely by the hardware to make this possible
2. The target enclave verifies the report
3. A secure channel is then constructed between the two enclaves using standard encryption methods.

Inter-Platform Attestation

Inter-Platform attestation refers to attestation between two enclaves running on different platforms, most likely connected by a network connection. This form of attestation is used by PoET and Intel provides an implementation of this service through the Intel Attestation Service (IAS). It is also known as “remote attestation”.

A special “Quoting Enclave” is provided with SGX and is used in Inter-process Attestation:

- The Quoting Enclave verifies reports submitted to it from other enclaves on the same platform using the Intra-process Attestation described above
- It then replaces the MAC with a signature created with a device specific private key stored in the device hardware.
- This new structure is called a “quote”.

The Quoting Enclave uses the Intel Enhanced Privacy ID (EPID) signature scheme to overcome the privacy concerns associated with using a small number of asymmetric keys for signing over the life of a platform.

- “EPID is a group signature scheme that allows a platform to sign objects without uniquely identifying the platform or linking different signatures.” [1]
- In “anonymous” mode, a signature verifier cannot associate a given signature with a given identity.
- In “pseudonymous” mode, a signature verifier can determine whether it has verified a platform previously.

The remote attestation procedure is roughly:

1. A local application enclave wants to interact with a remote enclave so it connects to the remote service through the local application.
2. The remote service issues a challenge to the local application.
3. The local application retrieves the local quoting enclave’s identity and passes it into the the local application enclave along with the challenge.
4. The local application enclave generates a “manifest” that includes:
    a. A response to the challenge
    b. An ephemeral public key
5. The local application enclave generates a hash digest of the manifest. An enclave report is generated which includes the hash digest and has the platform’s (local) Quoting Enclave as its target enclave.
6. The enclave report and manifest are sent to the local application which forwards the report to the Quoting Enclave.
7. The Quoting Enclave verifies the report, generates an enclave quote using its EPID key, and forwards the quote to the local application.
8. The local application forwards the enclave quote and manifest to the remote service.
9. The remote service verifies the quote and manifest
    a. An EPID public key certificate along with revocation information can be used to verify both locally
    b. Alternatively, an attestation verification service can be used (see IAS below).

IAS

As an alternative to keeping track of a set of verified and revoked enclave identities, Intel provides an attestation service (IAS) which can be used to verify enclave quotes and challenge manifests. Roughly speaking, IAS provides:

- An up-to-date list of revoked EPID signatures (which correspond to processors)
- Enclave quote verification

Inter-Platform Attestation and IAS are used by PoET when a new validator is signing up with a network to verify:

- The validator is running the correct PoET code in a trusted enclave.
- The enclave is not running on a processor that has been revoked.

Each instance of Sawtooth running PoET must have a unique Service Provider ID which is used to communicate with IAS and identifies the network. Why?

PoET 1.0

The following gives a rough, mid-level description of PoET 1.0 and how it operates. It assumes an understanding of trusted execution environment such as Intel SGX as well as some background knowledge of consensus models. (At a minimum, the reader should have a high-level understanding of the “Byzantine Generals Problem” [4].) Many of the specific details are omitted because they are documented in the comprehensive “PoET enclave specification.” [5]

The Proof-of-Elapsed-Time or PoET Consensus method offers a solution to the Byzantine Generals Problem that utilizes a “trusted execution environment” to improve on the efficiency of present solutions such as Proof-of-Work. The following presentation assumes the use of Intel SGX as the trusted execution environment, but there is no reason a platform that provides equivalent security guarantees could not be used instead.

At a high-level, PoET stochastically elects individual peers to execute requests at a given target rate. Individual peers sample an exponentially distributed random variable and wait for an amount of time dictated by the sample. The peer with the smallest sample wins the election. Cheating is prevented through the use of a trusted execution environment, identity verification and blacklisting based on asymmetric key cryptography, and an additional set of election policies.

Trusted Computing Base

PoET leverages a trusted execution environment (TEE) to ensure fair elections. PoET was first designed and implemented with Intel SGX and makes the following assumptions about the trusted execution environment it depends on:

1. Code and data cannot be accessed or modified once they have been loaded into the TEE, except through a strict interface defined prior to loading.
2. Attestation of the code and data loaded into the TEE is possible.
3. Individual “platforms” have a hardware encoded identity in the form of an asymmetric key pair.
4. An external attestation service (EAS) is available which:
    a. Maintains a blacklist of identities
    b. Generates verifiable attestation reports for a given identity and code
5. Random number generation is possible within the TEE.

Peer Signup

The following describes the PoET signup process assuming a generic TEE:

1. PoET is loaded into the TEE by the application
2. Data required to receive attestation is submitted to the EAS by the application
3. Verifiable attestation data is received from the EAS
4. The application broadcasts a join request to the network which contains the attestation data and additional signup data

The following describes the signup process using Intel SGX and the Intel Attestation Service:

1. A new PoET enclave is created locally by the application
2. Network signup data is created by the enclave and exported out of the enclave to the application including an SGX report and application manifest
3. An SGX quote is generated based on the SGX report
4. The manifest and quote are submitted to IAS for approval
5. IAS responds with an approval (or denial) which is appended to the sign-up data
6. The application broadcasts a join request which contains the signup data
7. Individual peers on the network review the signup data and decide whether to accept the peer

Leader Election

Individual peers perform the following to participate in leader election:

1. Calculate the “local mean” of the network
    a. The local mean is computed so that the time between elections is approximately the target rate for the network given its size
    b. The target rate is a configurable parameter and should be “selected to minimize the probability of a collision.” [5]
2. Create a new “wait timer” and “wait timer signature” from within the TEE
    a. Sample an exponential distribution whose mean is the local mean

    b. The wait timer wraps the sample
3. Wait for the time specified in the wait timer
4. Create a new “wait certificate” and “wait certificate signature” based on the wait timer and wait timer signature from within the TEE
    a. The wait timer signature proves the wait timer is valid
    b. If the time elapsed since creating the timer is less than the sample, this fails
    c. If the time elapsed since creating the timer is greater than some expiration time, this fails
5. Broadcast the wait certificate and wait certificate signature along with the peer’s identity
6. Whichever peer sampled the smallest value from the distribution wins

Election Policies

In addition to the election process, the following additional policies are enforced to mitigate security risks. Each of the policies contains a configurable parameter and is named after that parameter.

K-policy

A peer must repeat the signup process after it has one *K* elections.

The signup process involves checking the identity of the peer against a blacklist maintained by the attestation service. Enforcing periodic attestation allows the network to identify peers that have been blacklisted by the attestation service since they first signed up.

c-policy

A peer may not win an election until at least *c* blocks have been committed since it signed up.

This policy prevents peers from manipulating elections by first generating multiple identities with corresponding wait timers, choosing a wait timer with a high probability of winning, and then signing up with whichever identity corresponded to the low wait timer.

z-policy

A peer may not win an election more often than is statistically probable.

As a final check that a peer is not manipulating elections, a statistic is calculated to determine the probability that a correct peer could win as often as the given peer has won. If the probability is below some threshold, then the peer ???

References

1. “Innovative Technology for CPU Based Attestation and Sealing.” Anati, Gueron, Johnson, Scarlata. Intel Corporation
2. “Intel® Software Guard Extensions Programming Reference.” 329298-002US. October 2014
3. “Intel® Software Guard Extensions: EPID Provisioning and Attestation Services.” Johnson, Scarlata, Rozas, Brickell, Mckeen. Intel Corporation.
4. “The Byzantine Generals Problem.” Lamport, Shostak, Pease. SRI International.
5. “PoET enclave specification.” Miele. April 2017.