MAGE: Mutual Attestation for a Group of Enclaves without Trusted Third Parties

Guoxing Chen
Shanghai Jiao Tong University

Yinqian Zhang
Southern University of Science and Technology
Trusted execution environment (TEE)
Remote attestation

TEE Platform

Enclave code and data

Attester

Operating System

Confidentiality

Integrity

Verifier

Remote Attestation
Q1: Is the attester an enclave?
Q2: What is the attester enclave’s identity?

Enclave measurement: the cryptographic hash of the initial code and data of an enclave, as the identity of the enclave.
Q3: Is the identity trusted?

Enclave measurement

Report

Signature

=?

Trusted enclave’s measurement
Q3: Is the identity trusted?

Verifier enclave

Enclave measurement

= ?

Trusted enclave’s measurement

Verifier enclave
Mutual attestation with TTPs

- Trusting multi-enclave applications via **mutual attestation**

- TTPs increase the TCB and might incur extra costs for running PKIs
Mutual attestation without TTPs

Circular dependency
Measurement calculation

The measurement calculation (e.g., SHA-256) is deterministic and sequential.
Measurement calculation

**Key observation:** knowing the intermediate hash and information to perform subsequent measuring operations would be sufficient to derive the final output
MAGE

During enclave creation
Introduce an **common part**
at the end of each enclave

Enclave A

```
Enclave A
SHA_INIT
  intermediate hash
initialization
```

```
Enclave B
SHA_INIT
  intermediate hash
initialization
```

```
A: intermediate hash
B: intermediate hash
```

```
SHA_FINAL
  final hash
finalization
```

```
SHA_FINAL
  final hash
finalization
```

```
SHA_UPDATE
  intermediate hash
update
```

```
SHA_UPDATE
  intermediate hash
update
```

---

```
Enclave A
```

```
Enclave B
```

---

```
Enclave A
```

```
Enclave B
```

---

```
Enclave A
```

```
Enclave B
```
MAGE

During enclave creation

Enclave A

- SHA_UPDATE
- intermediate hash
- update

Enclave B

- SHA_UPDATE
- intermediate hash
- update

A:
- intermediate hash

B:
- intermediate hash

---

Final Hash

Finalization
MAGE

During enclave run time

Derive Enclave A’s Measurement

Derive Enclave B’s Measurement

A:
- intermediate hash

B:
- intermediate hash

SHA_UPDATE
- update

SHA_FINAL
- finalization

SHA_UPDATE
- update

SHA_FINAL
- finalization

SHA_UPDATE
- update

SHA_FINAL
- finalization

SHA_UPDATE
- update

SHA_FINAL
- finalization

SHA_UPDATE
- update

SHA_FINAL
- finalization
Implementation for Intel SGX

• MAGE library:
  • Reserve a read-only data section, named .sgx_mage
  • Provide APIs for deriving measurements from .sgx_mage
Implementation for Intel SGX

- Modified enclave loader:
  - Load .sgx_mage section after all other enclave code and data
Implementation for Intel SGX

- Modified signing tool:
  - Extract intermediate hashes from enclaves.
  - Insert intermediate hashes into .sgx_mage section
Performance

- **Memory overhead**
  - Linear with the number of trusted enclaves
  - 48 bytes to store auxiliary information (e.g., intermediate hashes, page metadata) for deriving one enclave measurement
  - One 4KB page could support 85 enclaves

- **Measurement derivation efficiency**
  - Linear with the size of `.sgx_mage` section
  - 21.7 microseconds to derive one measurement when `.sgx_mage` section consists of one page
Discussion

• Alternative designs
  • Extending MAGE with untrusted storage for better scalability.

• Extensions to other TEEs
  • Even between different types of TEEs.

• Supporting enclave updates/private code
Thank You!

GitHub repo  https://github.com/donnod/linux-sgx-mage

Guoxing Chen  guoxingchen@sjtu.edu.cn
Yinqian Zhang  vinqianz@acm.org