TEE² – Combining Trusted Hardware to Enhance the Security of TEEs

Master-Thesis of Fritz Alder
In cooperation with Aalto University, Finland

Supervisor: Prof. Katzenbeisser
Supervisor at Aalto University: Dr. Andrew Paverd and Prof. Asokan
Motivation

- Trusted Execution Environments (TEEs) provide isolated execution of security sensitive pieces of code that can be attested by remote parties.
  - TEEs have the potential to ensure security in cloud computing environments
- Real-world TEEs and their implementations are prone to attacks and bugs
  - Trust in real-world TEEs is difficult to achieve, possibly slowing down adoption
- Can we combine multiple TEEs to achieve security even if all but one TEE is compromised?
Trusted Execution Environment (TEE)

- Separated from Rich Execution Environment (REE)
- Executes Trusted Applications (TAs)
- Provides:
  - Code integrity
  - Isolated execution
  - Sealed data
  - TEE attestation

- Code integrity verified by TEE (e.g. with certificates)
- No direct access to TAs from REE
- Access only through predefined call-gates
Trusted Execution Environment (TEE)

- Separated from Rich Execution Environment (REE)
- Executes Trusted Applications (TAs)
- Provides:
  - Code integrity
  - Isolated execution
  - Sealed data
  - TEE attestation

Physical host

TEE

REE

No access from REE

TA1

No access from other TAs

TA2

Sealed storage

Remote attestation
System model – *Ideal TEE*

- Similar to Honest but Curious cloud provider model:
  - User communicates with TEE
  - Adversary has full control of host
  - Adversary has no interest in DoS

- Adversary goal: Undermine any of the four TEE properties
  - Code integrity
  - Isolated execution
  - Sealed data
  - TEE attestation
Real-world TEE adversaries

Weak attacker

- Compromises TEE confidentiality
  - Can read run-time secrets and sealed data
  - But: Can not fake attestations or impact TEE integrity

Strong attacker

- Compromises TEE confidentiality and integrity
- Has access to architectural secrets or can influence TEE integrity
  - Can fully impersonate the TEE
**Combined TEE – Design**

- User communicates with two unique TEEs
- Adversary has full control over both untrusted hosts
- Adversary can choose to compromise any TEE
  - User stays unaware of choice
- *Combined TEE* remains secure as long as at least one TEE is uncompromised
Random Number Generation

Goal:
1. Generate a random string.
2. ...that is unknown by an attacker
3. ...and can be attested by remote parties as being actually randomly generated
Random Number Generation – weak adversary

Goal:

- Generate a random string.
- ...that is unknown by an attacker
- ...and can be attested by remote parties as being actually randomly generated
Random Number Generation – weak adversary

**Goal:**

1. Generate a random string.
2. ...that is unknown by an attacker
3. ...and can be attested by remote parties as being actually randomly generated
Random Number Generation—strong adversary

Goal:
1. Generate a random string.
2. ...that is unknown by an attacker
3. ...and can be attested by remote parties as being actually randomly generated
Random Number Generation–strong adversary

Goal:

1. Generate a random string.
2. ...that is unknown by an attacker
3. ...and can be attested by remote parties as being actually randomly generated
Random Number Generation—strong adversary

Goal:

1. Generate a random string.
2. ...that is unknown by an attacker
3. ...and can be attested by remote parties as being actually randomly generated

combine

responses + attestations

Check commitments
Random Number Generation–strong adversary

1. User requests commitment from TEE
2. TEE binds commitment to a chain
3. TEE reveals commitment to user
4. User combines commitments with TEE

Diagram:

- User
- TEE
- Request commitment
- Bind to commitment chain
- Reveal
- Combine
Random Number Generation–strong adversary

Request commitment

Bind to commitment chain

Reveal

Combine
Protocol Design

- Combined TEE protocols differ from Ideal TEE protocols
  - No TEE can have knowledge of or control over any part of a secret
  - Instead, protocols need to protect against compromised TEEs
- Defined a range of utility, one-party, and two-party protocols
  - Key Exchange
  - Messaging
  - Random Number Generation
  - ElGamal operations
  - Signing
  - Store-and-forward
  - Oblivious Transfer
ElGamal operations – key generation

Goal:
1. Operate on private keys held by the TEEs...
2. ...that are attestable
3. ...and can not be learned during decryption
4. ...but can be used for confidential messages to the user
ElGamal operations – decryption

Goal:
1. Operate on private keys held by the TEEs...
2. ...that are attestable
3. ...and can not be learned during decryption
4. ...but can be used for confidential messages to the user

Ciphertext C → private key → request decryption

private key → return decryption shares

combine shares
ElGamal operations – decryption

Ciphertext C

private key

request decryption

private key

return decryption shares

combine shares

Goal:
1. Operate on private keys held by the TEEs...
2. ...that are attestable
3. ...and can not be learned during decryption
4. ...but can be used for confidential messages to the user
ElGamal operations – decryption

Goal:
1. Operate on private keys held by the TEEs...
2. ...that are attestable
3. ...and can not be learned during decryption
4. ...but can be used for confidential messages to the user

Ciphertext $C$

private key

request decryption

private key

return decryption shares

combine shares

S

$y_1 = x_1 \cdot G ; y_2 = x_2 \cdot G ; Y = y_1 + y_2$

$C_1 = k \cdot G ; C_2 = M + k \cdot x_1 \cdot G + k \cdot x_2 \cdot G$

$C = (C_1, C_2) ; d_1 = -x_1 \cdot C_1 ; d_2 = -x_2 \cdot C_1$

$M = C_2 + d_1 + d_2$
Two-party protocols

- Both parties can try to cheat and can compromise N-1 TEEs
- ...but do not collaborate
- Protocols require active participation from both users
  - More than a simple attestation verification
Policy based store-and-forward

Goal:
1. Secretly share data with user B
2. Only B can reveal the secret
3. B can not reveal the secret if a policy is not matched
Oblivious transfer – ideal version

A has a list L of n items

Goals:
1. B can select up to m items
2. B should not learn more than m items
3. A should not learn B‘s choices (except the value of m)
4. No third party should learn any items or choices
Oblivious transfer – *Combined TEE*

A has a list $L$ of $n$ items

**Goals:**
1. B can select up to $m$ items
2. B should not learn more than $m$ items
3. A should not learn B’s choices (except the value of $m$)
4. No third party should learn any items or choices
Oblivious transfer – Combined TEE

A has a list \( L \) of \( n \) items

**Goals:**

1. B can select up to \( m \) items
2. B should not learn more than \( m \) items
3. A should not learn B’s choices (except the value of \( m \))
4. No third party should learn any items or choices
Oblivious transfer – *Combined TEE*

A has a list L of $n$ items

Goals:

1. B can select up to $m$ items
2. B should not learn more than $m$ items
3. A should not learn B’s choices (except the value of $m$)
4. No third party should learn any items or choices
Oblivious transfer – Combined TEE

A has a list L of n items

Goals:
1. B can select up to m items
2. B should not learn more than m items
3. A should not learn B’s choices (except the value of m)
4. No third party should learn any items or choices

A → TEE: establish n keys → TEE: send encrypted L → B
1. Involve B
2. Shuffle keys

request m keys ← B

check policy

encrypt L with keys

reveal m keys

check policy
Oblivious transfer – *Combined TEE*

A has a list L of *n* items

### Goals:

1. B can select up to *m* items
2. B should not learn more than *m* items
3. A should not learn B’s choices (except the value of *m*)
4. No third party should learn any items or choices
Verification & Implementation

- Formally verified subset of protocols with the Tamarin Prover
  - Key exchange & RNG with weak adversary
  - Comparing security properties of *Combined TEE* protocols to security properties of *Ideal TEE* protocols

- Implemented subset of protocols with Intel SGX
  - Key exchange, both RNG protocols, and signature generation
  - Based on C++ (SGX) and Python (Client)
    - Passing JSON strings to exchange requests and responses
Evaluation

- Key exchange
- Randomness strong adversary
- Randomness weak adversary

Seconds vs. # of TEEs
Future work

• Explore different system models
  • Small and Big TEE instead of system of equals
    • ARM Trustzone + Intel SGX
    • Key management enclave + disk encryption enclave
  • System of TEEs for safety and reliability
    • E.g. TEEs in a car: Parts may fail frequently
• Port complex applications
  • Only cryptographic building blocks so far
  • No real-world use case
Summary

• TEEs have great potential
• 2 types of real-world adversaries exist: Weak and strong attackers
• Combined TEE alleviates impact of compromises
  • Range of protocols for arbitrary many cooperating TEEs: RNG, PK-Encryption, Signing, Store and forward, Oblivious Transfer
• Subset of protocols formally verified with Tamarin prover
• Implementation based on Intel SGX and Python
  • Shows a reasonable performance overhead for cryptographic building blocks
• Future work: Port complex applications, explore different TEE² scenarios (big-small TEE combination, multiple TEEs for safety and reliability, etc)
Oblivious transfer – Offline phase

A

Random t1, t2
Random A1,..,An

Random Q1,..,Qn

Random R1,..,Rn

TEE

TEE

B

Random permutation

[t1 ⊕ A1, t1 ⊕ A2, ...]

[t2 ⊕ A1, t2 ⊕ A2, ...]

[t1 ⊕ A1 ⊕ Q1, ...]

[t1 ⊕ A1 ⊕ R1, ...]

Permutated list

Combine:
[t1 ⊕ t2 ⊕ Q1 ⊕ R1, ...]
Permute

Eliminate t1 ⊕ t2 → Permuted
[Q1 ⊕ R1, ...]

But: B should not see keys in plain!
→ Hide with homomorphic encryption
\textbf{$\mathcal{C}_n^\alpha$: Offline phase of oblivious $m$ of $n$ transfer between two users with Combined TEE.}

User $A$
Secrets $S = \{s_1, ..., s_n\}$
with $s_i \in \mathcal{D}_i$
GM pair $p_{k_A}, s_{k_A}$

\begin{align*}
\alpha_1, \alpha_2 & \overset{\$}{\xleftarrow{} D_f} \\
A &= \left[ A_i \overset{\$}{\xleftarrow{} D_f} \right] \\
& \text{for } 1 \leq i \leq n
\end{align*}

\begin{align*}
Q &= \left[ Q_i \overset{\$}{\xleftarrow{} D_f} \right] \\
& \text{for } 1 \leq i \leq n
\end{align*}

\begin{align*}
R &= \left[ R_i \overset{\$}{\xleftarrow{} D_f} \right] \\
& \text{for } 1 \leq i \leq n
\end{align*}

\begin{align*}
\mathcal{M}_{\mathcal{I} \rightarrow \mathcal{T}}(\text{Enc}_A(\alpha_1 \oplus A), m)
\end{align*}

\begin{align*}
\mathcal{M}_{\mathcal{I} \rightarrow \mathcal{T}}(\text{Enc}_A(\alpha_2 \oplus A), m)
\end{align*}

\begin{align*}
\mathcal{M}_{\mathcal{T} \rightarrow \mathcal{U}}(\text{Enc}_A(\alpha_1 \oplus A \oplus Q))
\end{align*}

\begin{align*}
\mathcal{M}_{\mathcal{T} \rightarrow \mathcal{U}}(\text{Enc}_A(\alpha_2 \oplus A \oplus R))
\end{align*}

\begin{align*}
\mathcal{T}_1 = \text{Enc}_A(\alpha_1 \oplus A \oplus Q) \quad \mathcal{T}_2 = \text{Enc}_A(\alpha_2 \oplus A \oplus R)
\end{align*}

\begin{align*}
\mathcal{Z} = \text{Dec}_A(\mathcal{T}_1 \oplus \mathcal{T}_2)
\end{align*}

\begin{align*}
(\mathcal{Z}(\mathcal{Z}))
\end{align*}

\begin{align*}
\text{Decrypt } Z
\end{align*}

\begin{align*}
\text{Calculate: } K = [Z_i \oplus \alpha_1 \oplus \alpha_2]
\end{align*}

\begin{align*}
\text{Each } K_j \text{ is now: } K_j = Q_j \oplus R_j \text{ with } j = \sigma(i)
\end{align*}

\textbf{$\mathcal{C}_n^\beta$: Online phase of oblivious $m$ of $n$ transfer between two users with Combined TEE.}

User $A$
Secrets $S = \{s_1, ..., s_n\}$
with $K_i = Q_i \oplus R_i$

\begin{align*}
\mathcal{T}_1, \mathcal{T}_2
\end{align*}

\begin{align*}
Q = \left[ Q_{1,1}, ..., Q_{a,1} \right]
\end{align*}

\begin{align*}
R = \left[ R_{1,1}, ..., R_{n,1} \right]
\end{align*}

\begin{align*}
\mathcal{M}_{\mathcal{I} \rightarrow \mathcal{T}}(J)
\end{align*}

\begin{align*}
\mathcal{M}_{\mathcal{I} \rightarrow \mathcal{T}}(J)
\end{align*}

\begin{align*}
\mathcal{M}_{\mathcal{T} \rightarrow \mathcal{U}}(Q_M)
\end{align*}

\begin{align*}
\mathcal{M}_{\mathcal{T} \rightarrow \mathcal{U}}(R_M)
\end{align*}

\begin{align*}
\text{Only once: If } |J| \leq m: \text{ Return } Q_M \text{ with } Q_M = \left[ Q_{j,1}, ..., Q_{j,n} \right] \text{ Delete } Q
\end{align*}

\begin{align*}
\text{Only once: If } |J| \leq m: \text{ Return } R_M \text{ with } R_M = \left[ R_{j,1}, ..., R_{j,n} \right] \text{ Delete } R
\end{align*}

\begin{align*}
\text{Retrieve } S_M = \{S_{M_1}, ..., S_{M_n}\} \text{ with } S_{M_i} = S_i \oplus Q_{j,i} \oplus R_{j,i}
\end{align*}
Signing

Goal:
1. Operate on private signing keys held by the TEEs...
2. ...that are attestable and linkable to these TEEs

Both signatures + attestations require all signatures to verify