Information Flow Control for Distributed Trusted Execution Environments

Anitha Gollamudi
Stephen Chong

Owen Arden

HARVARD UNIVERSITY

UNIVERSITY OF CALIFORNIA SANTA CRUZ
Trusted Execution Environment (TEE)

- Protected memory region for code and data
- Offers isolated execution and remote attestation
- Only host can communicate with TEE
- Hardware feature
  - e.g. Intel SGX, ARM TrustZone
- Good fit for offering security in distributed settings
TEEs ⇒ Security Guarantees
TEEs $\not\Rightarrow$ Security Guarantees

if secret
send cipher on public
else ()
TEEs $\not\implies$ Security Guarantees

Ah, secret is 1

if secret
send cipher on public
else

TEEs ⇔ Security Guarantees

Information Flow Control (IFC) techniques!
IFC for Distributed TEEs: Challenges
1. Choose *right* abstractions for crypto and TEEs

- Focus on application-level security
- *Reflect* the capabilities and limitations of TEEs
  - e.g. TEE can communicate only with the host
- Implementable!
IFC for Distributed TEEs: Challenges

1. Choose *right* abstractions for crypto and TEEs
   - Focus on application-level security
   - *Reflect* the capabilities and limitations of TEEs
     - e.g. TEE can communicate only with the host
   - Implementable!

2. Enforce security
Contributions
Contributions

1. Distributed Flow-limited Authorization calculus for TEEs (DFLATE)
   - Supports distributed TEEs
   - Design mapping to real system
Contributions

1. Distributed Flow-limited Authorization calculus for TEEs (DFLATE)
   - Supports distributed TEEs
   - Design mapping to real system

2. A permissive security type system
   - Enforces security (noninterference) for confidentiality and integrity
DFLATE

- Simply typed lambda calculus extended with
  - Communication primitives (send/receive/spawn)
  - Abstractions for crypto and TEE
  - Security types
Address Challenge #1

1. Choose *right* abstractions for crypto and TEEs
   - Abstractions *should reflect* the capabilities and limitations of TEEs
     - e.g. TEE can only communicate with host
   - Focus on application-level security
     - Implementable!

2. Enforce security
Communication

Alice \((a)\)

send *blue* on \(ch_{ab}\)

Bob \((b)\)

recv \(ch_{ab}\) as \(x\) in

...
**Communication**

Alice (a)

**send** blue on \( ch_{ab} \)

Bob (b)

**recv** \( ch_{ab} \) as \( x \) in ...

[Diagram showing Alice sending a message labeled "blue" on channel \( ch_{ab} \) to Bob, who receives it as \( x \).]
Communication

Alice (a)

send blue on $ch_{ab}$

Bob (b)

recv $ch_{ab}$ as x in ...

...
Securing Communication

if secret
  send blue on public
else ()

Public channel
Securing Communication

if secret
    send blue on public
else ()
Securing Communication

if secret
    send blue on public
else ()

Public channel
Securing Communication

Security labels on channels prevent leaks due to communication
Communication through trusted/untrusted nodes
Communication through trusted/untrusted nodes

• Bob can not learn/modify the content of the message
• Bob may learn the existence of the message
Bob can not learn/modify the content of the message
Bob may learn the existence of the message

Bob can learn the message received from Alice

Communication through trusted/untrusted nodes
• Bob can not learn/modify the content of the message
• Bob may learn the existence of the message

Bob can learn the message received from Alice

Communication through trusted/untrusted nodes
1. Bob cannot learn/modify the content of the message.
2. Bob may learn the existence of the message.

Bob can learn the message received from Alice.

Communication through trusted/untrusted nodes.
Carol **may** learn the contents of the blue message

Carol **must not** learn the contents of the blue message
2a

Carol may learn the contents of the blue message

2b

Carol must not learn the contents of the blue message
Bob cannot learn/modify the orange message

Support for communication with enclaves
DFLATE abstracts crypto mechanisms using *protected expressions*
Protected Expression

\[ ch_{ab} \]
Protected Expression

\( ch_{ab} \)

\((\eta_a \ 42)\)

Protected expression
Protected Expression

Protected expressions abstract encryption and signing
Protected Expression

\[ (\eta_a \ 42) \text{ has type } \ a \text{ says } \text{int} \]
Operating on Protected Expressions

\[ \text{send } (\eta_a \ 42) \text{ on } ch_{ab} \]

\[ \text{recv } ch_{ab} \text{ as } enc \text{ in } \]

\[ \text{bind } x = enc \text{ in } (f \ x) \]

Bind abstracts
decryption and signature verification
Operating on Protected Expressions

send \((\eta_a 42)\) on \(ch_{ab}\)

recv \(ch_{ab}\) as \(enc\) in

\[
\text{bind } x = \text{enc} \text{ in } (f\ x)
\]

Bind abstracts
decryption and signature verification
To successfully decrypt, Alice must authorize Bob.
Bob $\geq$ Alice

Principals delegate authority using acts-for ($\geq$)
 Principals delegate authority using acts-for (≽)
Delegation of Authority

\begin{align*}
\text{assume } b \geq a \text{ in} \\
\text{send } (\eta_a \ 42) \text{ on } ch_{ab}
\end{align*}

\begin{align*}
\text{recv } ch_{ab} \text{ as } enc \text{ in} \\
\text{bind } x = enc \text{ in} \\
(f \ x)
\end{align*}
Delegation of Authority

\[ \text{assume } b \geq a \text{ in } \]
\[ \text{send } (\eta_a 42) \text{ on } ch_{ab} \]

\[ \text{recv } ch_{ab} \text{ as } enc \text{ in } \]
\[ \text{bind } x = enc \text{ in } (f x) \]
Delegation of Authority

Assume abstracts key sharing among principals

\[
\text{assume } b \geq a \text{ in } \\
\text{send } (\eta_a 42) \text{ on } ch_{ab}
\]

\[
\text{recv } ch_{ab} \text{ as } enc \text{ in } \\
\text{bind } x = enc \text{ in } (f x)
\]
• Bob must not learn/modify the content of the message
• Bob may learn the existence of the message

Bob can learn the message received from Alice
Bob can learn the message received from Alice.

- Bob must not learn/modify the content of the message.
- Bob may learn the existence of the message.
Bob must not learn/modify the content of the message
Bob may learn the existence of the message

Bob can learn the message received from Alice

Types enable reasoning about Bob’s power
Secure Bind (2)

assume $b \geq a$ in
send $(\eta_a 42)$ on $ch_{ab}$

recv $ch_{ab}$ as $enc$ in
bind $x = enc$ in
$(f x)$
Secure Bind (2)

assume $b \geq a$ in 
send $(\eta_a 42)$ on $ch_{ab}$

recv $ch_{ab}$ as $enc$ in 
bind $x = enc$ in 
$(f x)$

Bob must not leak the decrypted value
The output type of bind must protect Alice
The output type of bind must protect Alice
Type system ensures that Bob uses the decrypted value securely

Carol may learn the contents of the blue message

Carol must not learn the contents of the blue message
assume $t \geq a$ in
send $(\eta_a, \text{blue})$ on $ch_{ab}$

given $t \geq b$
recv $ch_{ab}$ as $x$
send $x$ on $ch_{bt}$
recv $ch_{bt}$ as $x'$ in
...

$\mathsf{TEE}^{t}$
{
...  
  send $v$ on $ch_{bt}$
}

assume $t \geq c$ in
recv $ch_{bc}$ as $x$ in
...

Abstracting TEE
Abstracting TEE

assume $t \geq a$ in
send $(\eta_a, blue)$ on
recv $ch_{ab}$ as $x$

send $x$ on $ch_{bt}$
recv $ch_{bt}$ as $x'$ in

... 

$\text{TEE}^t \{ ...
\text{send } v \text{ on } ch_{bt} \}

\text{assume } t \geq c \text{ in}
recv $ch_{bc}$ as $x$ in

...
Abstracting TEE

\begin{align*}
\text{assume } t & \geq a \text{ in } \\
\text{send } (\eta_a \text{ blue}) \text{ on } \\
\text{recv } ch_{ab} \text{ as } x \\
\text{send } x \text{ on } ch_{bt} \\
\text{recv } ch_{bt} \text{ as } x' \text{ in } \\
\text{...} \\
\text{recv } ch_{bc} \text{ as } x' \text{ in } \\
\text{...} \\
\text{send } v \text{ on } ch_{bt} \\
\end{align*}
assume \( t \geq a \) in
send \((\eta_a \text{ blue})\) on \( ch_{ab} \)

recv \( ch_{ab} \) as \( x \)
send \( ch_{bt} \) \( x \) then
recv \( ch_{bt} \) as \( x' \) in
...

\( \text{TEE}^t \) 
\{ ...
send \( v \) on \( ch_{bt} \)
\}

assume \( t \geq c \) in
recv \( ch_{bc} \) as \( x \) in
...

Abstracting TEE

26
assume $t \geq a$ in
send $(\eta_d \text{ blue})$ on $ch_{ab}$

\[
\begin{align*}
\text{recv } ch_{ab} & \text{ as } x \\
\text{send } ch_{bt} & \text{ as } x \text{ then} \\
\text{recv } ch_{bt} & \text{ as } x' \text{ in} \\
\text{...} & \\
\text{TEE}^t & \{ \\
\text{...} & \\
\text{send } v & \text{ on } ch_{bt} \\
\} & \\
\text{...} & \\
\end{align*}
\]

assume $t \geq c$ in
recv $ch_{bc}$ as $x$ in

Abstracting TEE
Bob cannot learn/modify the orange message
Bob cannot learn/modify the orange message
Implementing DFLATE
Design

- DFLATE abstractions are implementable!
Design

• DFLATE abstractions are implementable!

• TEEs can be implemented by Intel SGX enclaves
  • SGX provides remote attestation
  • SGX enclave communicates through the host
Design

- DFLATE abstractions are implementable!

- TEEs can be implemented by Intel SGX enclaves
  - SGX provides remote attestation
  - SGX enclave communicates through the host

- Protected expressions can be implemented using public key encryption and digital signatures
  - However, this requires access to the corresponding signing/decryption keys

- Key distribution, especially for enclaves, is non-trivial
Key Distribution

- A global key master has key pairs for all principals
- Key master provisions the nodes and enclaves with necessary private keys
Key Distribution

To obtain keys, a node proves its identity to the key master
To obtain keys, a node proves its identity to the key master.
Key Distribution

To obtain keys, a node proves its identity to the key master.
Key Distribution

To obtain keys, a node proves its identity to the key master.
Key Distribution

To obtain keys, an enclave attests itself to the key master
To obtain keys, an enclave attests itself to the key master
Key Distribution

To obtain keys, an enclave attests itself to the key master
Address Challenge #2

1. Choose *right* abstractions for crypto and TEEs
   
   - Abstractions *should reflect* the capabilities and limitations of TEEs
     
     - e.g. TEE can only communicate with host
   
   - Focus on application-level security
     
     - Implementable!

2. Enforce security
Security

• Formal definition of security is noninterference (NI)

  • Confidentiality NI: private inputs can’t influence public outputs

  • Integrity NI: Low integrity inputs can’t influence high integrity outputs

• Type system enforces security
Example Revisited

Ah, secret is 1

Alice

Bob

it secret
send cipher on public
Example Revisited

Ah, secret is 1

Program is ill-typed
Confidentiality Theorem

Secret inputs from Alice

Public outputs
Confidentiality Theorem

Secret inputs from Alice

Public outputs
Confidentiality Theorem

Secret inputs from Alice

Public outputs
Integrity Theorem

Untrusted inputs from Bob

Trusted outputs to Carol
Integrity Theorem

Untrusted inputs from Bob

Trusted outputs to Carol
Compromised-node
Noninterference
Compromised-node
Noninterference

Trace for blue execution
Compromised-node Noninterference

Traces observed (by Bob) for executions with different secret inputs are equal
Compromised-node Noninterference
Compromised-node
Noninterference

Trace for blue execution
Compromised-node Noninterference

Traces observed can be different
Confidentiality vs Integrity Guarantees

• Asymmetry due to the ability to suppress messages

• Faithfully models the expressive power of the integrity attacker

  • Without undermining the guarantees of cryptography and TEEs
Conclusion

• DFLATE: A programming model for distributed TEEs
• Design for implementing the abstractions in DFLATE
• DFLATE enforces confidentiality and integrity
Backup Slides
Nested Protection

\( ch_{ab} \) 

\( (\eta_a \ 42) \)
Nested Protection

$\eta_a^{42}$

$\chi_{ab}$

43
Nested Protection

\[(\eta_b (\eta_a 42)) \text{ has type } b \text{ says } a \text{ says } \text{int}\]
send $c_{ab} (\eta_a \ m_{blue})$

recv $c_{ab}$ as $x$ in

send $c_{bc} \ x$
\textbf{send} \(ch_{ab} (\eta_a \ m_{blue})\)

\textbf{assume} \(b \geq a\) \textbf{in}

\textbf{recv} \(ch_{ab}\) \textbf{as} \(x\) \textbf{in}

\textbf{send} \(ch_{bc}\) \(x\)
send $ch_{ab}(\eta, m_{blue})$

\begin{align*}
\text{assume } b &\geq a \text{ in} \\
\text{recv } ch_{ab} &\text{ as } x \text{ in} \\
\text{send } ch_{bc} &\ x
\end{align*}

Malicious declassification
\textbf{Type system prevents malicious declassifications and endorsements}

\begin{align*}
\text{send } ch_{ab} (\eta_a m_{\text{blue}}) \\
\text{assume } b \geq a \text{ in} \\
\text{recv } ch_{ab} \text{ as } x \text{ in} \\
\text{send } ch_{bc} x
\end{align*}

Malicious declassification
send $ch_{ab}(\eta_a m_{\text{blue}})$

assume $b \geq a$ in

recv $ch_{ab}$ as $x$

send $ch_{bc} x$

Insufficient authority to add delegation