TEE API의 정형 명세 및 모델 검증

유근열

POSTECH

커지는 보안의 중요성

무단 도용 방지
(저작권 보호)

Disney+

NETFLIX

자산 탈취 방지
(결제 정보 보호)

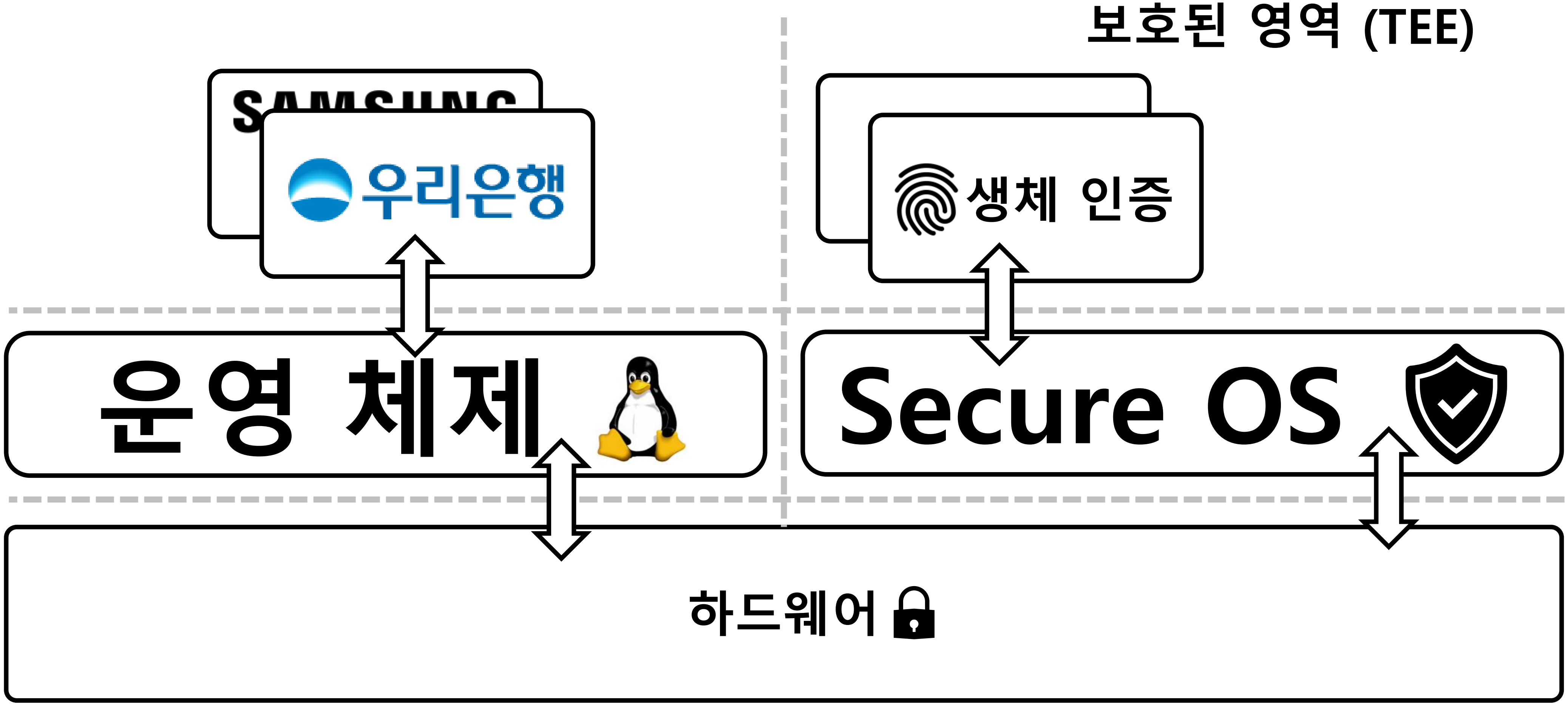
SAMSUNG pay

우리은행

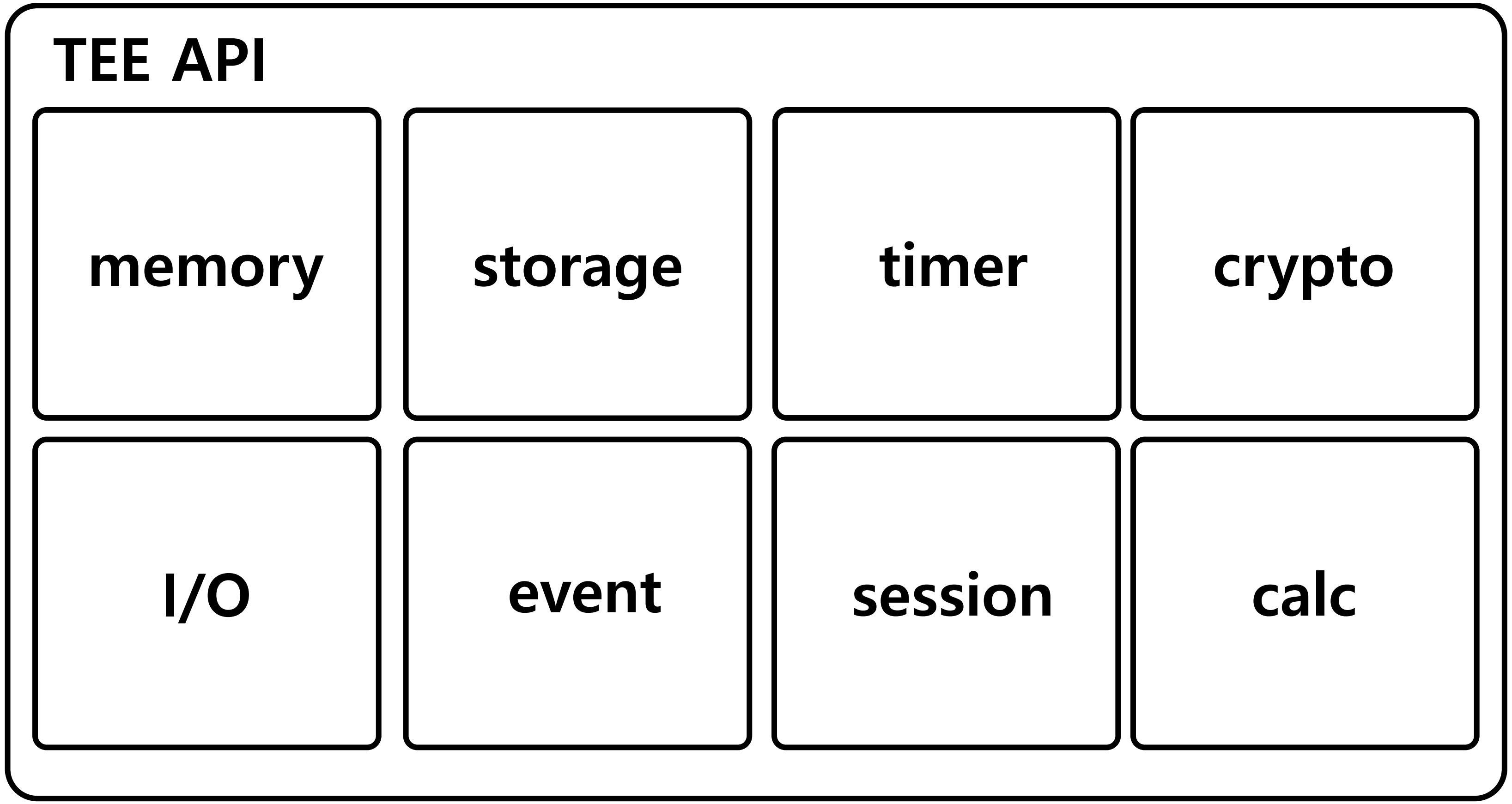
→ 기존보다 더 강력한 보안 요구

Secure OS

연산을 보호된 환경에서 처리하는 운영체제



Secure OS 표준 문서



TEE API

memory	storage	timer	crypto
I/O	event	session	calc

GLOBALPLATFORM®
THE STANDARD FOR SECURE DIGITAL SERVICES AND DEVICES

GlobalPlatform Technology
TEE Internal Core API Specification
Version 1.1.2.50 (Target v1.2)

Public Review
June 2018
Document Reference: GPD_SPE_010

연구 동기

- 표준 명세 자체에 설계 결함이 있다면?

- 표준 명세를 따른 구현이 명세 요구 사항을 만족하는가?

→ 검증의 필요성

어려운 점

- 문서를 읽고 정형 명세하는 것 자체에 많은 노력과 시간이 필요
    - 자연어로 적힌 문서 → 모호한 표현 해석 필요

- 동시성을 어떻게 고려할 것인가?
    - Code-based testing & static analysis로는 하기 어려움

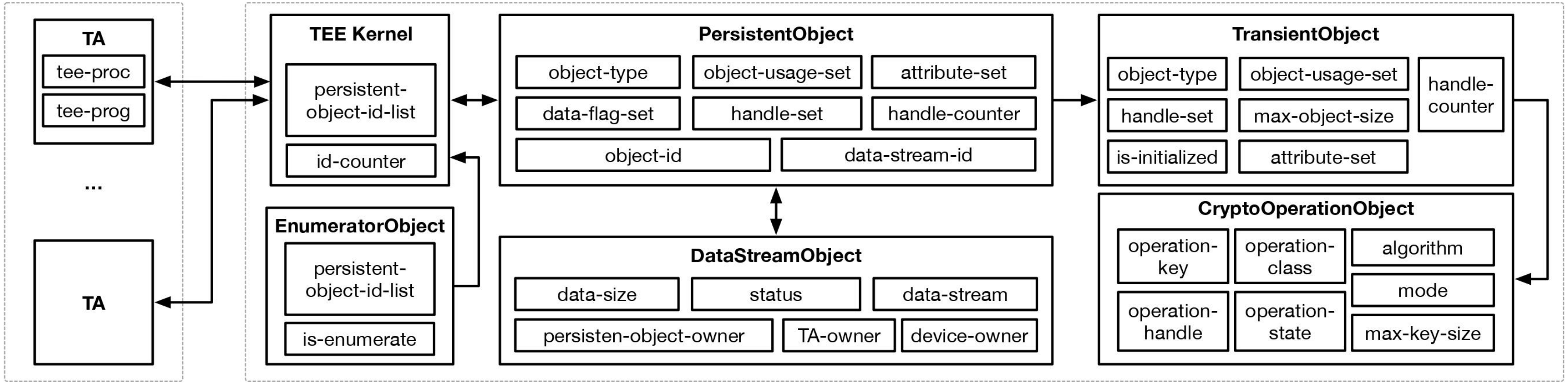
- 구현 되어있는 Real-world 프로그램을 어떻게 검증할 것인가?
    - Real-world 프로그램은 C/C++로 작성되어 있음

연구 전략

- 문서를 읽고 정형 명세하는 것 자체에 많은 노력과 시간이 필요
  - → 열심히 하자!
- 동시성을 어떻게 고려할 것인가?
  - → Maude로 명세
- 구현 되어있는 Real-world 프로그램을 어떻게 검증할 것인가?
  - → C-like language 지원

TEE 정형 명세

- 주요 컴포넌트 정의 및 명세
  - 어플리케이션, 커널, TEE 리소스



TA
tee-proc
tee-prog
...
TA

TEE Kernel
persistent-object-id-list
id-counter

EnumeratorObject
persistent-object-id-list
is-enumerate

PersistentObject
object-type
object-usage-set
attribute-set
data-flag-set
handle-set
handle-counter
object-id
data-stream-id

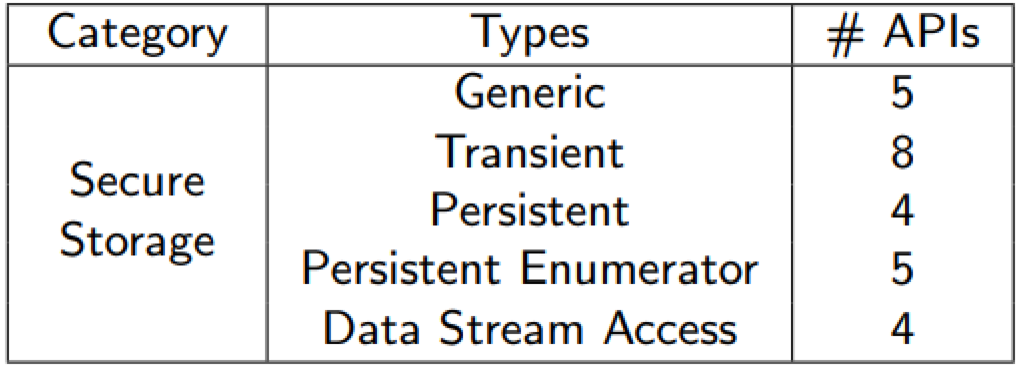
DataStreamObject
data-size
status
data-stream
persisten-object-owner
TA-owner
device-owner

TransientObject
object-type
object-usage-set
handle-counter
handle-set
max-object-size
is-initialized
attribute-set

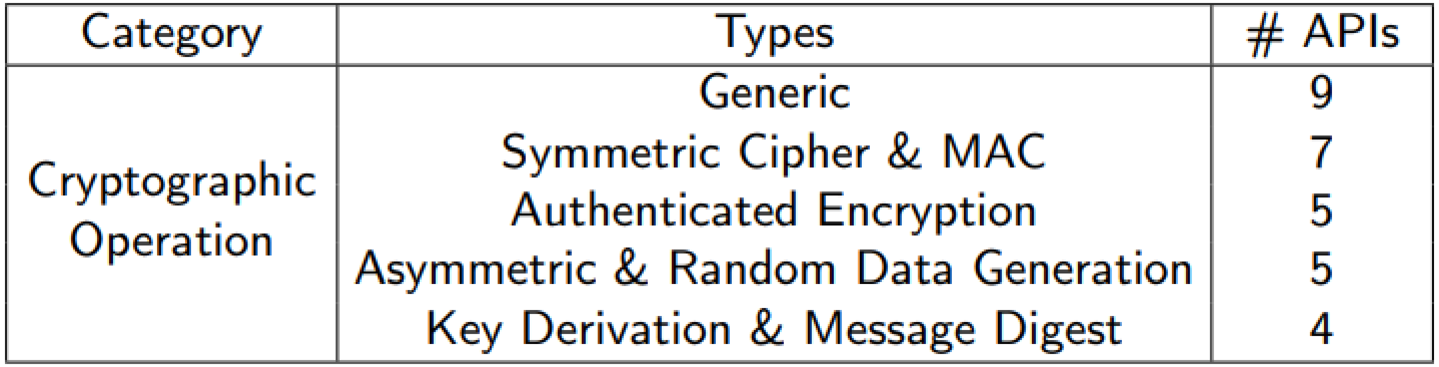
CryptoOperationObject
operation-key
operation-class
algorithm
mode
operation-handle
operation-state
max-key-size

TEE 정형 명세

- TEE 표준 API 모델링



Category	Types	# APIs
Secure Storage	Generic	5
	Transient	8
	Persistent	4
	Persistent Enumerator	5
	Data Stream Access	4



Category	Types	# APIs
Cryptographic Operation	Generic	9
	Symmetric Cipher & MAC	7
	Authenticated Encryption	5
	Asymmetric & Random Data Generation	5
	Key Derivation & Message Digest	4

TEE 정형 명세

- Real-world 프로그램 검증 위한 언어 실행 지원
    - C-like 프로그램 syntax

```
struct Person { var age } ;
Person john ; john.age = 0 ;
while (john.age < 10) { john.age += 1 }
```

- C-like 프로그램 semantics
    - Memory model (x) multi-threading (x)

TEE 모델 검증 예시

- TEE API spec 검사
  - Reachable state analysis, LTL model checking

```
search run(teeApi) ⇒∗ RESULT such that checkSpec(RESULT) .
red modelCheck(teeApi, [] invariant(teeApi)) .
```

- TEE 프로그램 검증
  - E.g., process2는 항상 process1 보다 나중에 끝나야 함

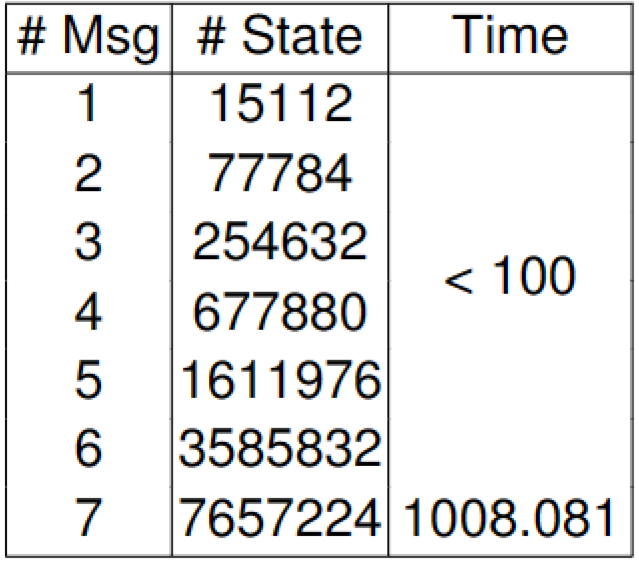
```
red modelCheck(init(proc1) init(proc2), [] exitAfter(proc1, proc2)) .
```

- 상태 공간 축소 기법
  - Invisible transition reduction, partial order reduction

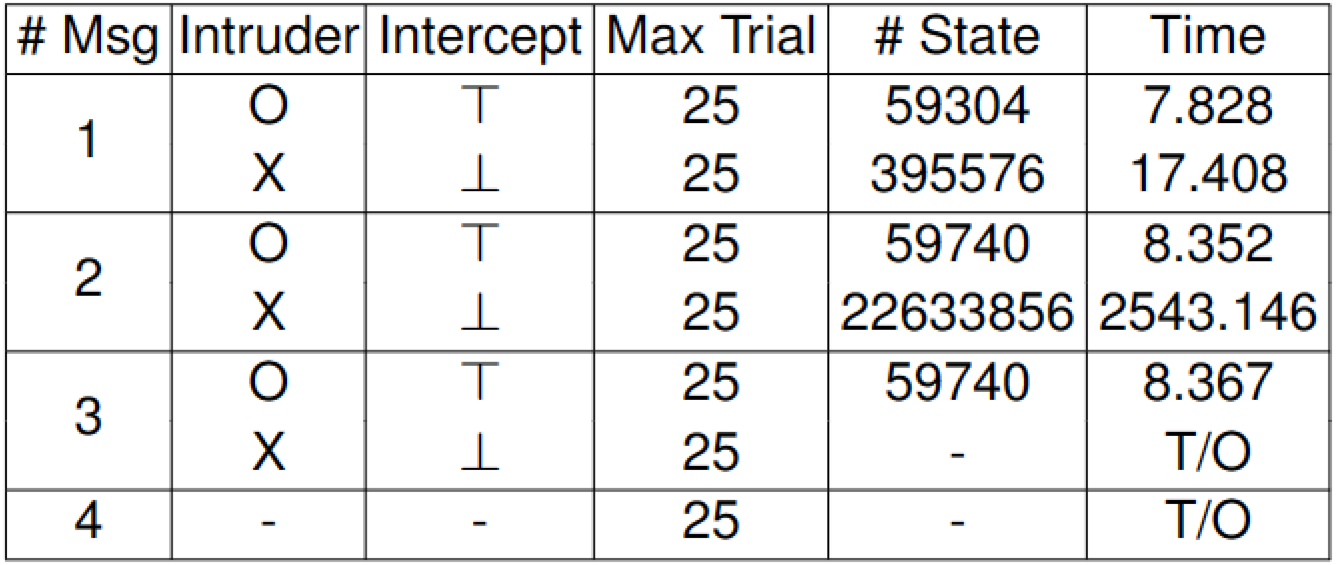
Case study: MQT-TZ

- IoT message protocol을 사용한 real-world 프로그램
  - TEE를 사용하여 message 탈취 방지

- TEE 여부에 따른 message 탈취 가능성 검증



# Msg	# State	Time
1	15112	< 100
2	77784	
3	254632	
4	677880	
5	1611976	
6	3585832	
7	7657224	1008.081



# Msg	Intruder	Intercept	Max Trial	# State	Time
1	O	⊤	25	59304	7.828
	X	⊥	25	395576	17.408
2	O	⊤	25	59740	8.352
	X	⊥	25	22633856	2543.146
3	O	⊤	25	59740	8.367
	X	⊥	25	-	T/O
4	-	-	25	-	T/O

Formal Specification and Verification of TEE

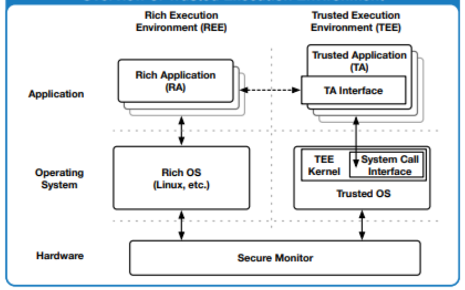
Geunyeol Yu

Software Verification Lab., Pohang University of Science and Engineering, South Korea

Background

- A trusted execution environment (TEE) is an isolated code execution environment to provide high-level of trust.
- Global Platform defines standard APIs and architectures for TEE and device vendors provide their own TEE implementations.
- Maude is a language and tool for formal specification and analysis of distributed systems.

Overview of Trusted Execution Environment



Motivation

- What if there is a design flaw in the standard APIs?
- Does a TEE implementation follow the standard?
- Is a TEE application safe and bug-free?

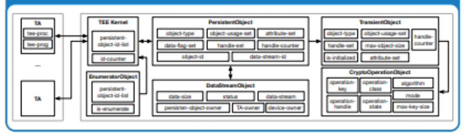
Challenge

- Specifying the standard APIs is itself challenging.
  - e.g., free all the resources (?) after TEE_FreeTransientObject.
- How to verify concurrent behaviours?
  - code-based testing (x), code-based static analysis (x).
- How to verify real-world TEE applications only with their code?

Formal Specification of TEE in Maude

- Specify models for trusted & rich applications.
- Specify abstracted objects for REE & TEE kernels.
- Specify objects representing TEE resources.
  - handle objects, secure storage objects, cryptographic objects.
- Define overall relations b/w them.

Overall Specification



- Specify the standard TEE APIs.

Category	Types	# API	Category	Types	# API
Secure Storage	Generic	5	Crypto Operation	Generic	9
	Transient	8		Symmetric Cipher & MAC	7
	Persistent	4		Authenticated Encryption	5
	Persistent Enumerator	5		Asymmetric & Message Digest	7
	Data Stream Access	4		Key Deriv. & Rand. Generation	4

Programming Language Semantics for TEE

- C-like language syntax.
  - e.g., structure, if-else, loop, function call.

```
struct Person { var age } ;
Person john ; john.age = 0 ;
while (john.age < 10) { john.age += 1 }
```

- C-like language semantics.
  - memory model (x), multi-threading (x), executing a program (o)

Formal Verification of TEE using Maude

- Verify the TEE formal specification using Maude.
  - reachable state analysis, LTL model checking

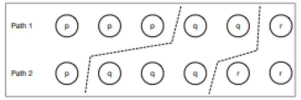
```
search teeApi =>* STATE such that checkSpec(STATE) .
red modelCheck(teeApi, [] invariant(teeApi)) .
```

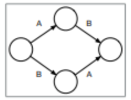
- Verify real-world TEE applications using Maude.
  - using C-like language semantics
  - high-level behaviour analysis (o), code-level analysis (x)
  - e.g., program2 always exits after program1

```
red modelCheck(init(p1 p2), [] exitAfter(p1, p2)) .
```

State Space Reduction

- Invisible transition reduction
  - reduce a transition b/w equivalent states.
  - i.e., remove stuttering equivalent paths.

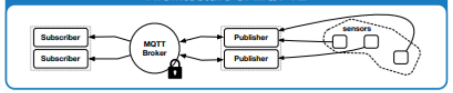




- Partial order reduction
  - explore only necessary paths w.r.t independent relations.
  - i.e., do not consider all interleavings.

Case study: MQT-TZ IoT Application

Architecture of MQT-TZ



- MQTT is a standard messaging protocol for the IoTs.
- MQT-TZ protects MQTT broker using TEE.
  - preventing message interception, modification
- Analyzing MQT-TZ using formal specification.
  - Model an intruder that tries to intercept messages.
  - Simulate the intruder can intercept a message w/o TEE.
  - Verify the intruder fails to intercept any message w/ TEE.

# Msg	# State	Time
1	15112	< 100
2	77784	
3	254632	
4	677880	
5	1611976	
6	3585832	
7	7657224	1008.081

# Msg	Intruder	Intercept	Max Trial	# State	Time
1	O	⊤	25	59304	7.828
	X	⊥	25	395576	17.408
2	O	⊤	25	59740	8.352
	X	⊥	25	22633856	2543.146
3	O	⊤	25	59740	8.367
	X	⊥	25	-	T/O
4	-	-	25	-	T/O

Contact: rgyen@postech.ac.kr