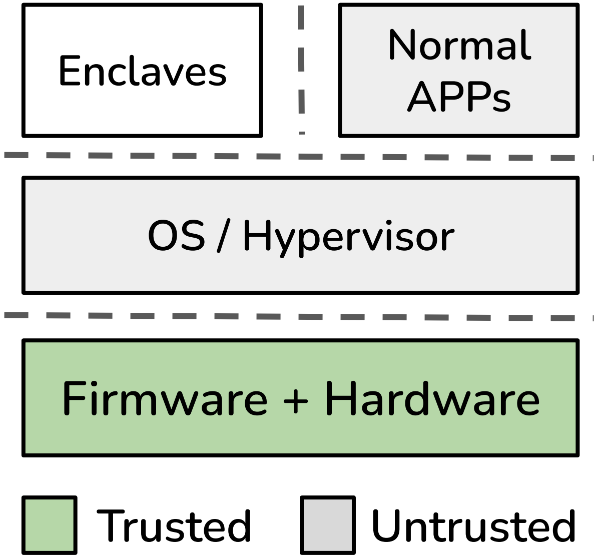
A Formal Approach to Multi-Layered Privileges for Enclaves

Ganxiang Yang, Chenyang Liu, Zhen Huang, Guoxing Chen, Hongfei Fu, Yuanyuan Zhang, Haojin Zhu

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

Enclave / TEE (Trusted Execution Environment)

- **TEE protects enclaves from untrusted (privileged) software by**
  - Spatial Isolation
  - Execution Isolation
- **TEE trusts secure hardware and firmware supports including**
  - Secure CPU
  - Trusted On-chip Modules



Enclave / TEE (Trusted Execution Environment)

Intel
SGX, TDX

AMD SEV

ARM
TrustZone, CCA

Sanctum, Keystone,
Penglai

TEEs are widely used in various remote computation scenarios

- Secure Machine Learning
- Secure Service
- Secure Storage

Motivation: Restriction of Current Enclaves

Restriction of Deploying Enclaves: **Usability**

- ➢ Lack of common features (e.g. Memory Sharing, Introspection, etc.)
- ➢ Incompatible with cloud/VM scenarios (e.g. cold-boot, migration, etc.)
- ➢ …

Motivation: Restriction of Current Enclaves

Restriction of Deploying Enclaves: **Usability**

- Lack of common features (e.g. Memory Sharing, Introspection, etc.)
- Incompatible with cloud/VM scenarios (e.g. cold-boot, migration, etc.)
- ...

Root Cause: **Spatial Isolation** and **Execution Isolation** of enclaves

Motivation: Restriction of Current Enclaves

Restriction of Deploying Enclaves: **Usability**

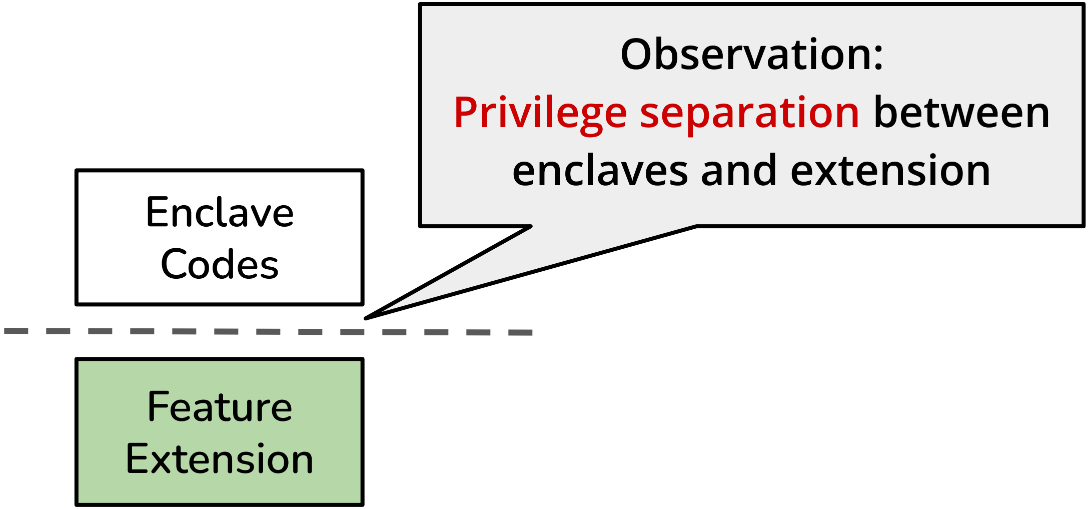
- ➢ Lack of common features (e.g. Memory Sharing, Introspection, etc.)
- ➢ Incompatible with cloud/VM scenarios (e.g. cold-boot, migration, etc.)
- ➢ …

Root Cause: **Spatial Isolation** and **Execution Isolation** of enclaves

Feature Extensions: always equipped with **"Privileges"**, including **Spatial Control** and **Execution Control**

Previous Works

Providing TEE extensions based on **Privilege Separation**



Observation:
Privilege separation between enclaves and extension

Previous Works

Enclave Codes

Feature Extension

(CCS ’22) Cerberus: A Formal Approach to Secure and Efficient Enclave Memory Sharing

(Security ’22) Elasticlave: An Efficient Memory Model for Enclaves

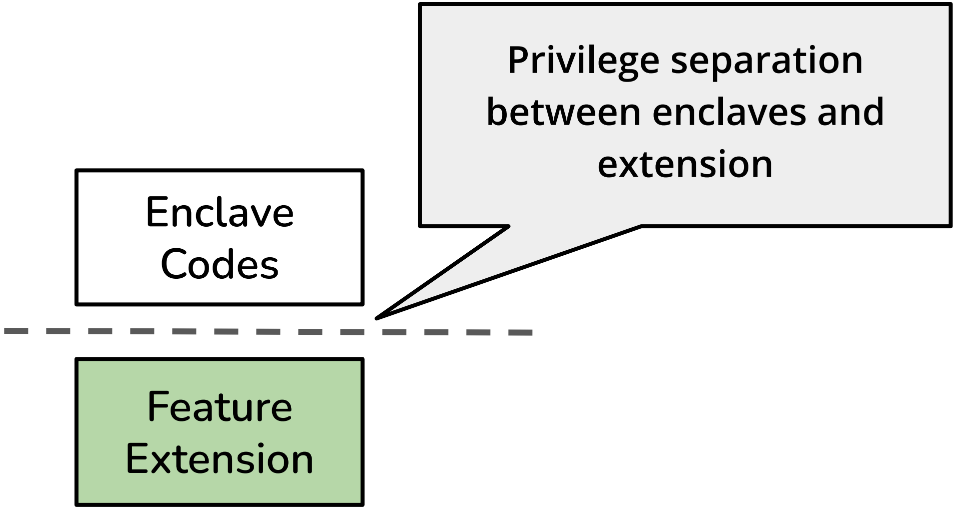
(ISCA ’20) Nested Enclave: Supporting Fine-grained Hierarchical Isolation with SGX

(Security ’23) Reusable Enclaves for Confidential Computing

…

Previous Works

🤔 **Where to put the extensions?**

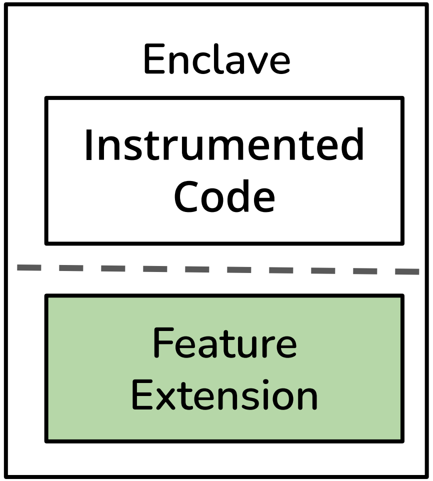


Privilege separation between enclaves and extension

Previous Works

🤔 **Where to put the extensions?**

1. Inside the enclave **(Intra-Enclave Compartmentalization*)**

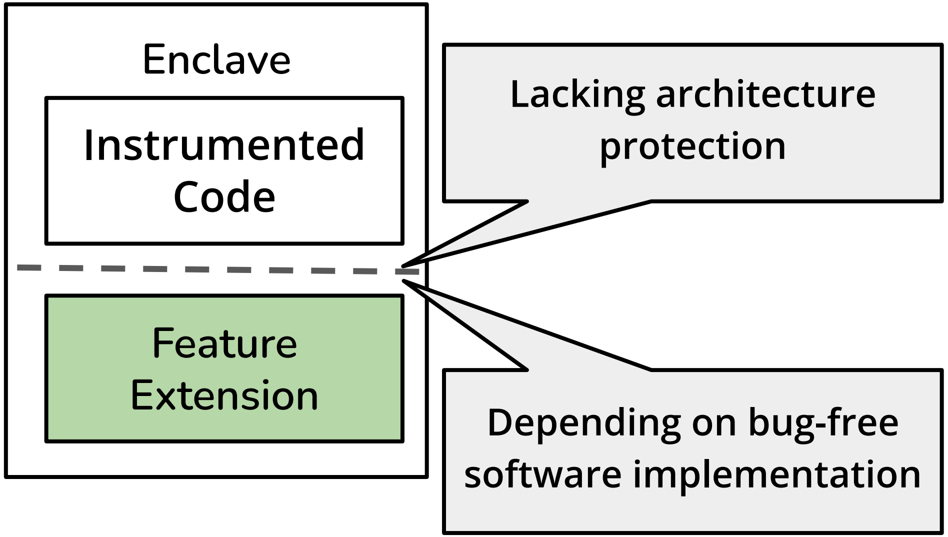


** Reusable Enclave (Security '23), SGX-Migration (DSN '17), etc.*

Previous Works

🤔 **Where to put the extensions?**

1. Inside the enclave **(Intra-Enclave Compartmentalization)**



Enclave

Instrumented Code

Feature Extension

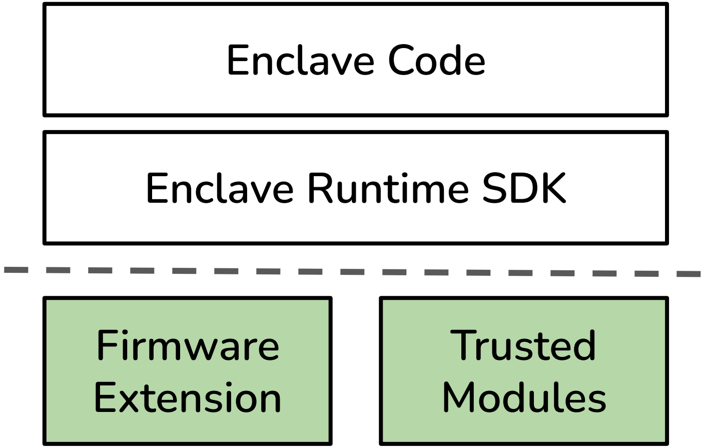
Lacking architecture protection

Depending on bug-free software implementation

Previous Works

🤔 **Where to put the extensions?**

2. Architecture-level Design



Enclave Code

Enclave Runtime SDK

Firmware Extension

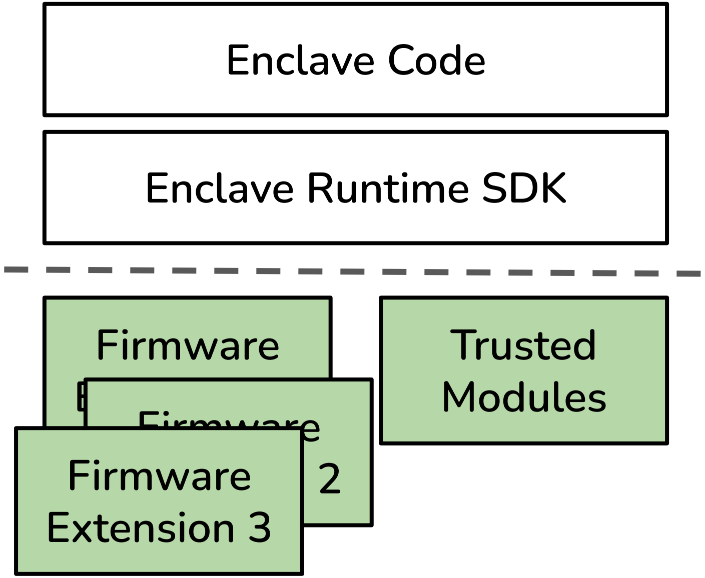
Trusted Modules

** Cerberus (CCS ’22), SMILE (S&P ’22), etc.*

Previous Works

🤔 **Where to put the extensions?**

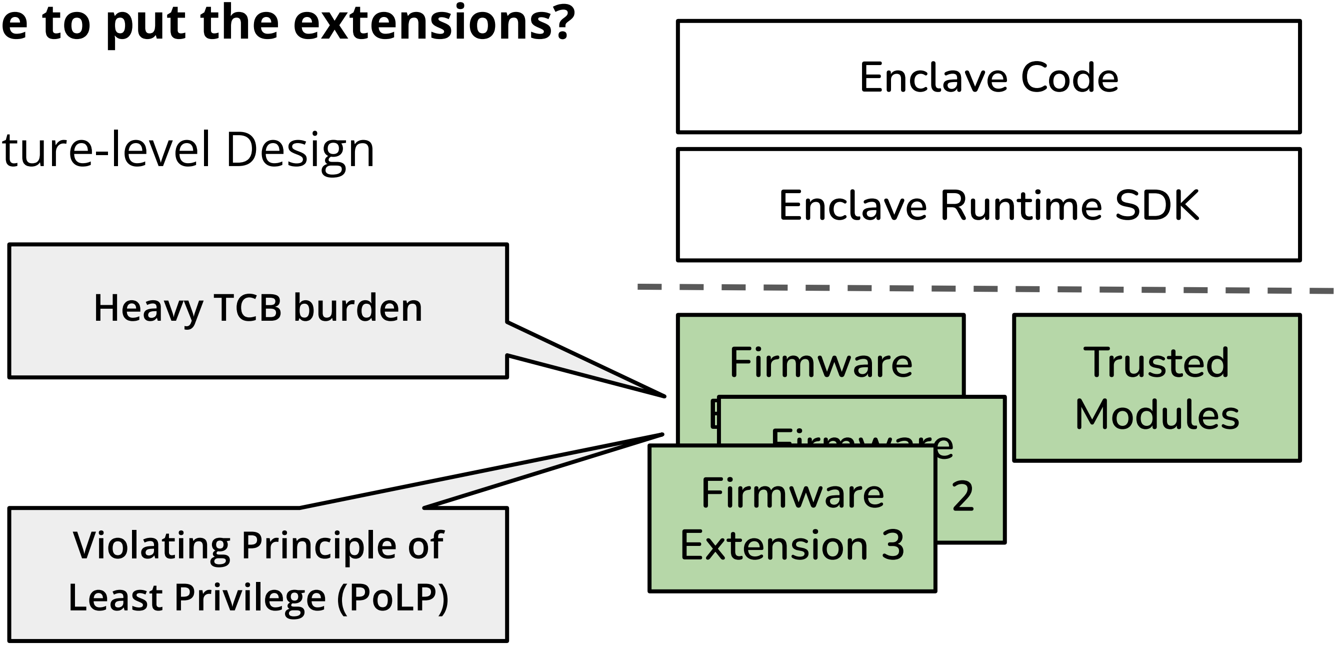
2. Architecture-level Design



Previous Works

🤔 **Where to put the extensions?**

2. Architecture-level Design



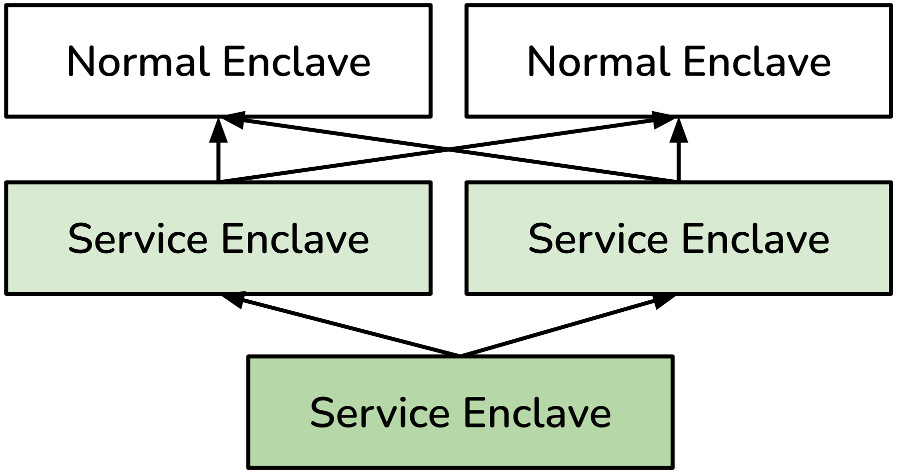
Heavy TCB burden

Violating Principle of Least Privilege (PoLP)

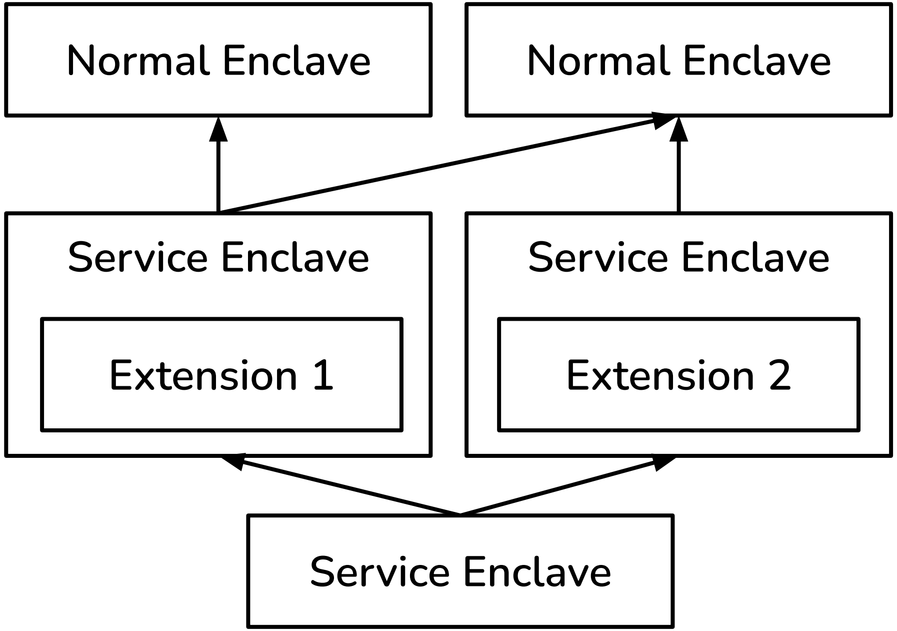
Previous Works

🤔 **Where to put the extensions?**

3. Inter-enclave Privileges?



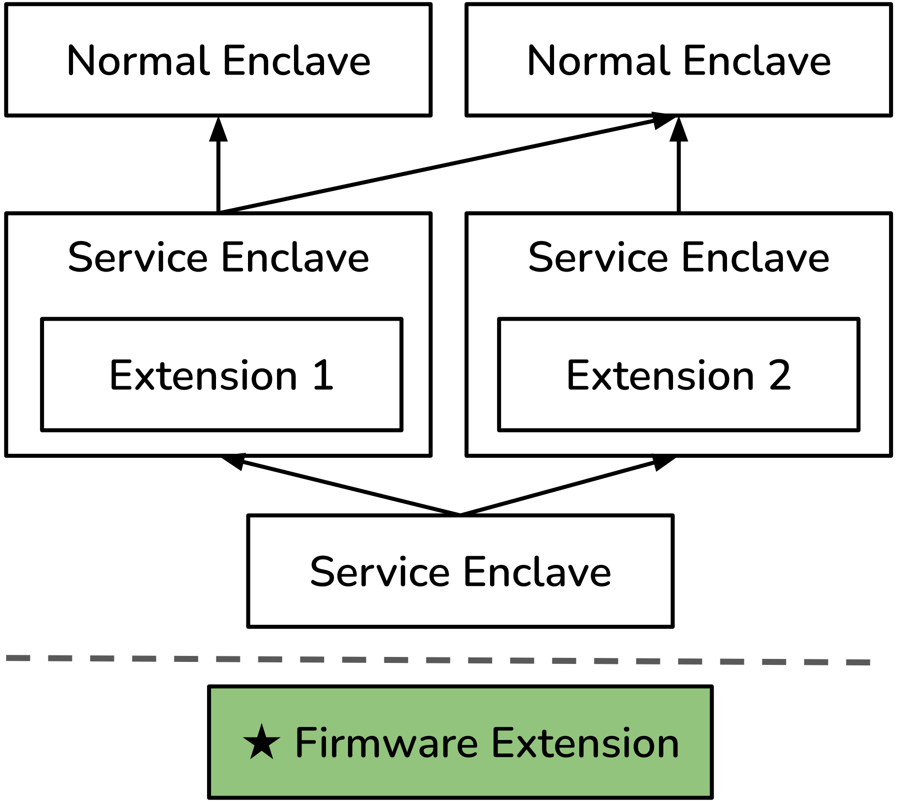
Idea: Inter-Enclave Privilege Separation



Idea: Inter-Enclave Privilege Separation

Advantages:

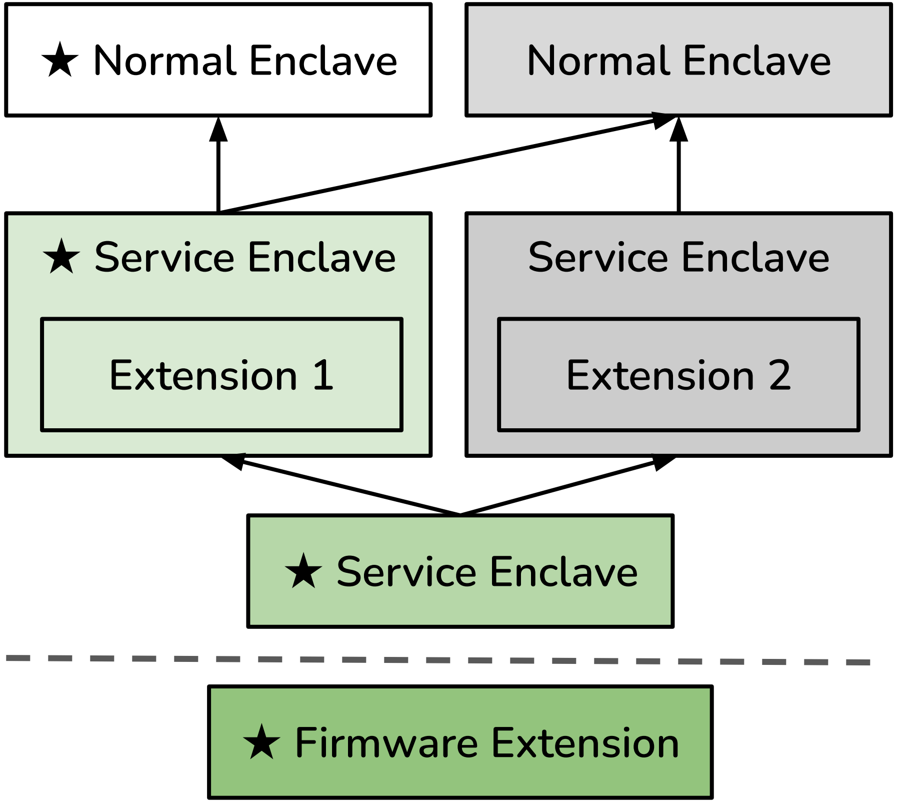
★ **Architecture-based security** guarantees



Idea: Inter-Enclave Privilege Separation

Advantages:

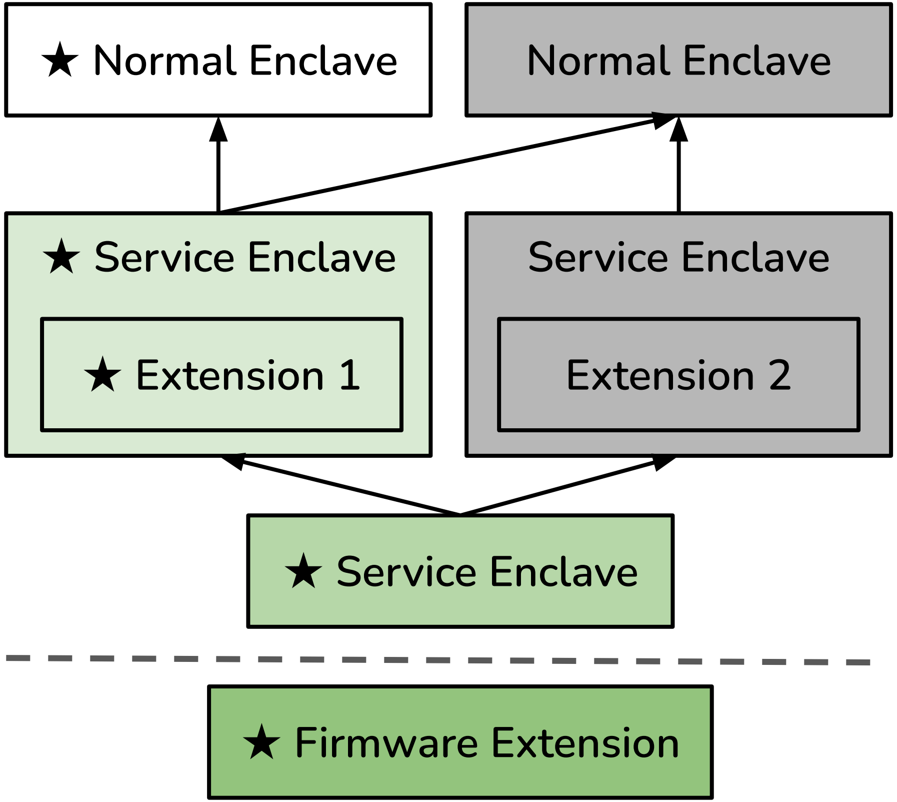
- ★ **Architecture-based security** guarantees
- ★ Use only necessary extensions, **minimize TCB**



Idea: Inter-Enclave Privilege Separation

Advantages:

- ★ **Architecture-based security** guarantees
- ★ Use only necessary extensions, **minimize TCB**
- ★ **Customizable extensions** in userspace, easy to program



Idea: Inter-Enclave Privilege Separation

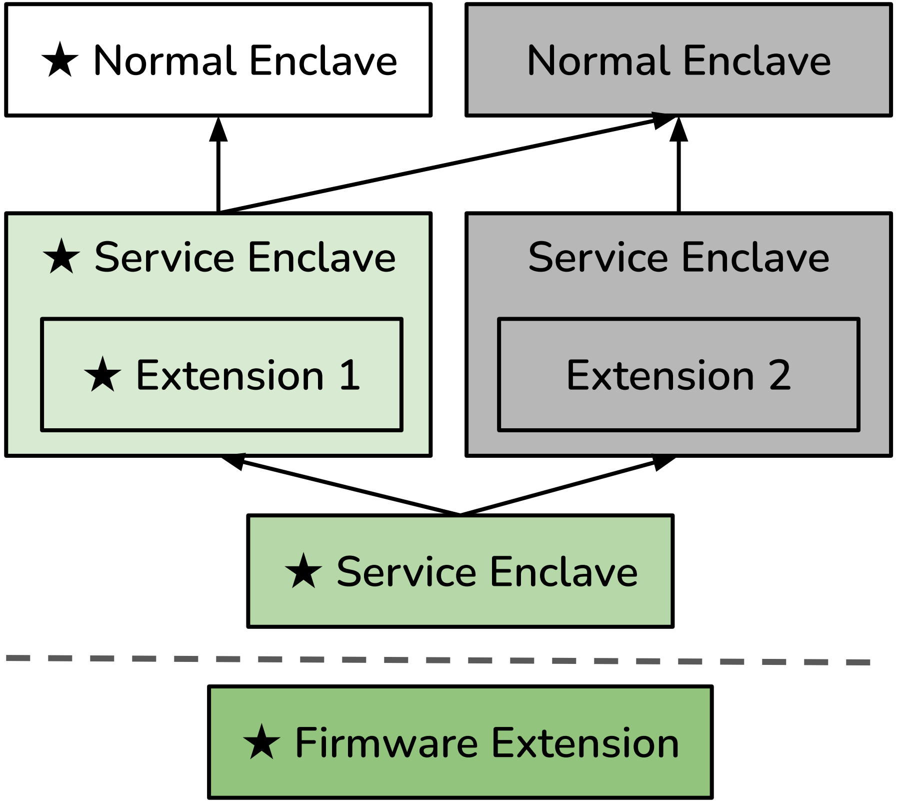
Previous Work:

Nested Enclave (*ISCA '20*)

CapStone (*Security '23*)

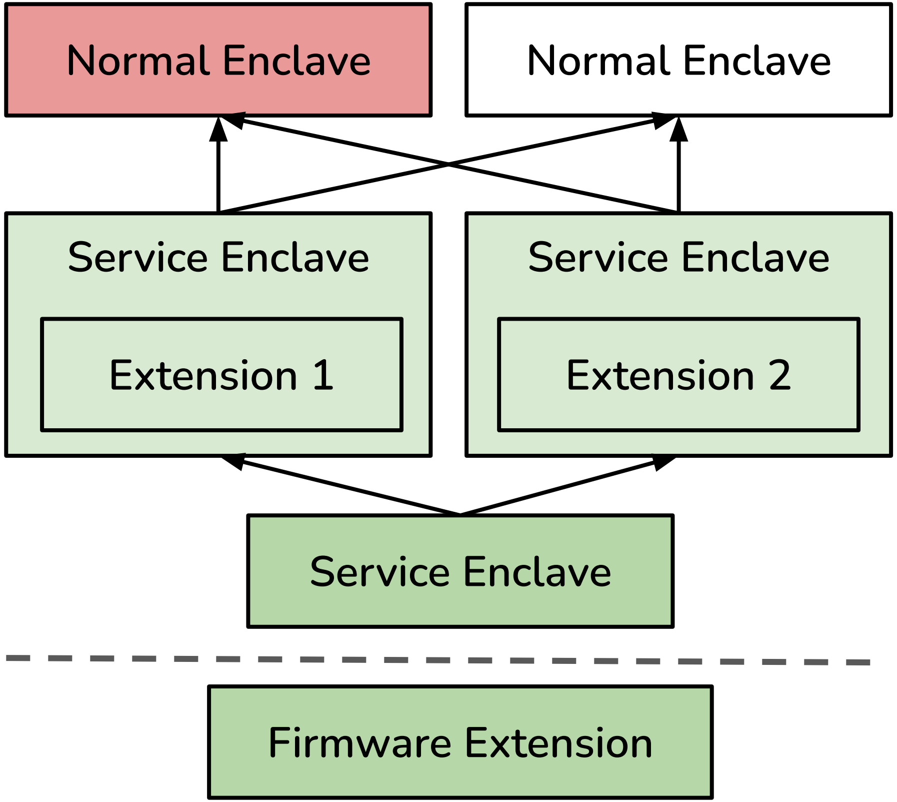
! No formal security guarantees

! Single-layer separation



Idea: Inter-Enclave Privilege Separation

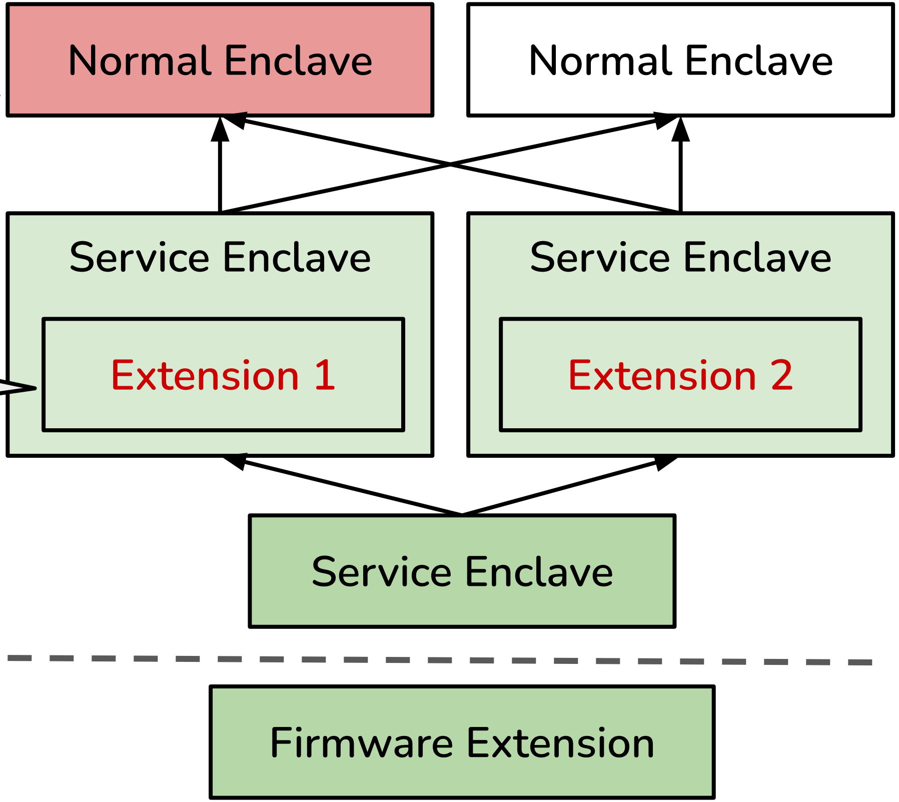
Challenge 1: Security
of Parent Enclaves (PE) when
Child Enclave (CE) compromised



Idea: Inter-Enclave Privilege Separation

Challenge 1: Security
of Parent Enclaves (PE) when
Child Enclave (CE) compromised

Challenge 2: Scalability
of Feature Extensions.
Proving security for any extension
/ any layer number



A Formal Approach to Multi-Layered Privileges for Enclaves

Threat Model

- Malicious OS
- Side-channel attacks and DoS attacks are out of scope
- Concerning about the security of a Parent Enclave when any these enclaves are compromised:

  (1) its own Children Enclaves;

  (2) any other legacy enclaves;

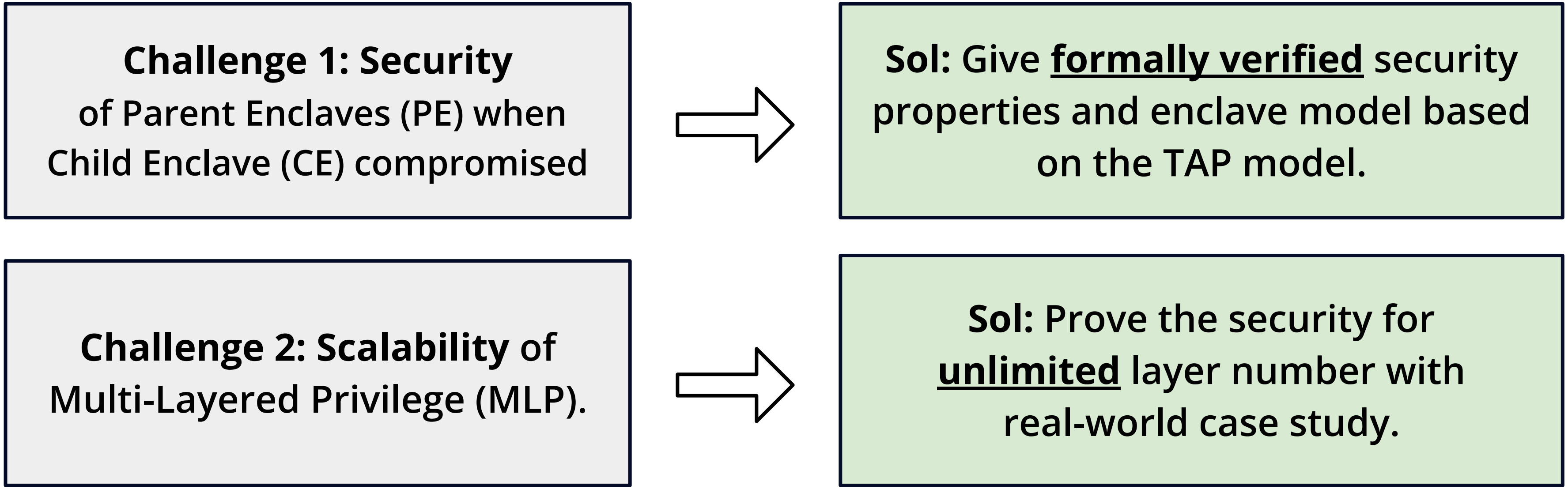
  (3) other non-ancestor Children Enclaves

Our Design

Challenge 1: Security of Parent Enclaves (PE) when Child Enclave (CE) compromised

Challenge 2: Scalability of Multi-Layered Privilege (MLP).

Our Design



Challenge 1: Security of Parent Enclaves (PE) when Child Enclave (CE) compromised

Sol: Give **formally verified** security properties and enclave model based on the TAP model.

Challenge 2: Scalability of Multi-Layered Privilege (MLP).

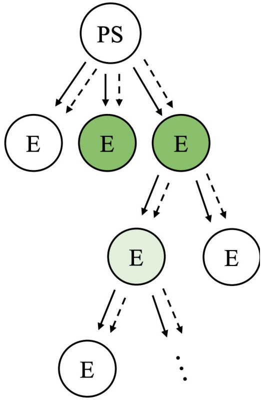
Sol: Prove the security for **unlimited** layer number with real-world case study.

Our Design

Challenge 1: Security of Parent Enclaves (PE) when Child Enclave (CE) compromised

Solution

1. Define 7 privilege instructions from Parent Enclave (PE) to its Children Enclave (CE)
2. Build an abstract enclave platform model supporting Multi-Layered Privileges



(c)
Multi-Layered
Privileges (✓)

Operation
LAUNCH
ENTER
PAUSE
RESUME
EXIT
DESTROY
INSPECT†

Our Design

Challenge 1: Security of Parent Enclaves (PE) when Child Enclave (CE) compromised

Solution

3. Define the Secure Remote Computation (SRE) property for Multi-Layered Privileges (MLP)

4. Use Z3 prover and inductions to verify security

e.g. Formalizing the Integrity

$$\begin{tikzcd}
\pi_1^{\langle 0\rangle} \arrow[r, "eop_0"] \arrow[d, phantom, "\approx_I \approx_E"] & \cdots & \pi_1^{\langle i\rangle} \arrow[r, "\mathcal{A}_1"] \arrow[d, phantom, "\approx_I"] & \pi_1^{\langle i+1\rangle} \arrow[r, "eop_1"] \arrow[d, phantom, "\approx_I \;\; \approx_E, \approx_O"] & \pi_1^{\langle i+2\rangle} \arrow[d, phantom, "\approx_I \;\; \approx_E, \approx_O"] & \cdots & \pi_1^{\langle j\rangle} \arrow[r, "\mathcal{A}_1"] \arrow[d, phantom, "\approx_I \;\; \approx_E, \approx_O"] & \pi_1^{\langle j+1\rangle} \arrow[r, "eop_2"] \arrow[d, phantom, "\approx_I"] & \cdots \\
\pi_2^{\langle 0\rangle} \arrow[r, "eop_0"] & \cdots & \pi_2^{\langle i\rangle} \arrow[r, "\mathcal{A}_2"] & \pi_2^{\langle i+1\rangle} \arrow[r, "eop_1"] & \pi_2^{\langle i+2\rangle} & \cdots & \pi_2^{\langle j\rangle} \arrow[r, "\mathcal{A}_2"] & \pi_2^{\langle j+1\rangle} \arrow[r, "eop_2"] & \cdots
\end{tikzcd}$$

$$\begin{aligned}
&\forall \pi_1, \pi_2 \in TRACE(TS). \\
&\Big(E_e(\pi_1^{\langle 0\rangle}) = E_e(\pi_2^{\langle 0\rangle}) && \wedge \\
&\forall i \in \mathbb{N}.\ \pi_1^{\langle i\rangle}.curr = e \iff \pi_2^{\langle i\rangle}.curr = e && \wedge \\
&\forall i \in \mathbb{N}.\ \pi_1^{\langle i\rangle}.curr = e \Rightarrow I_e(\pi_1^{\langle i\rangle}) = I_e(\pi_2^{\langle i\rangle})\Big) && \Longrightarrow \\
&\Big(\forall i \in \mathbb{N}.\ E_e(\pi_1^{\langle i\rangle}) = E_e(\pi_2^{\langle i\rangle}) \wedge O_e(\pi_1^{\langle i\rangle}) = O_e(\pi_2^{\langle i\rangle})\Big)
\end{aligned} \tag{4}$$

Our Design

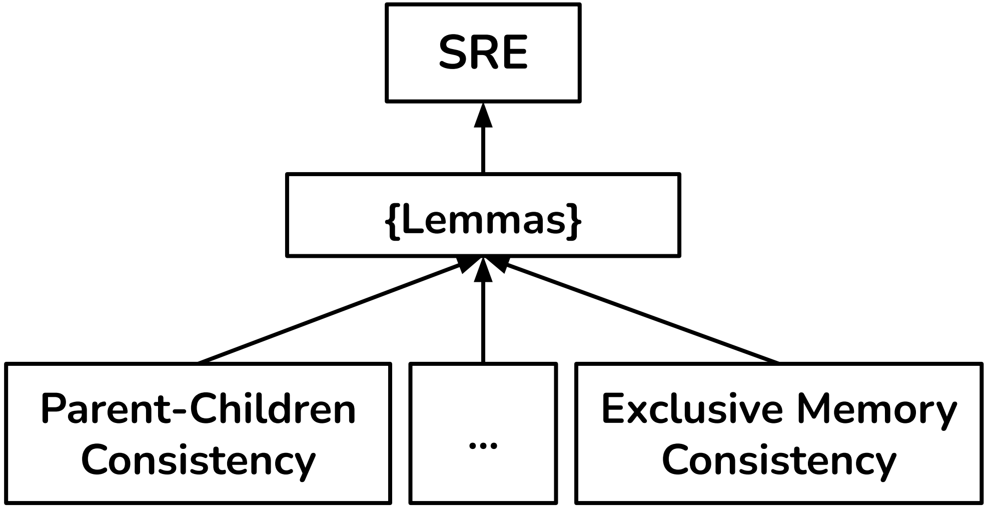
Challenge 1: Security of Parent Enclaves (PE) when Child Enclave (CE) compromised

Solution

3. Define the Secure Remote Computation (SRE) property for Multi-Layered Privileges (MLP)

4. Use Z3 prover and inductions to verify security

Proof Tree:



Our Design

Challenge 2: Scalability of Multi-Layered Privilege (MLP).

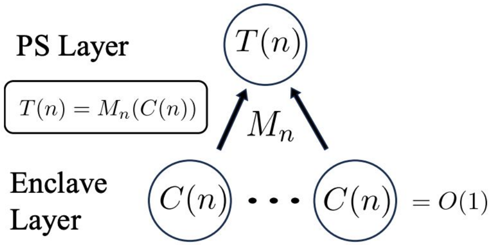
→ Introducing inter-enclave privileges

Our Design

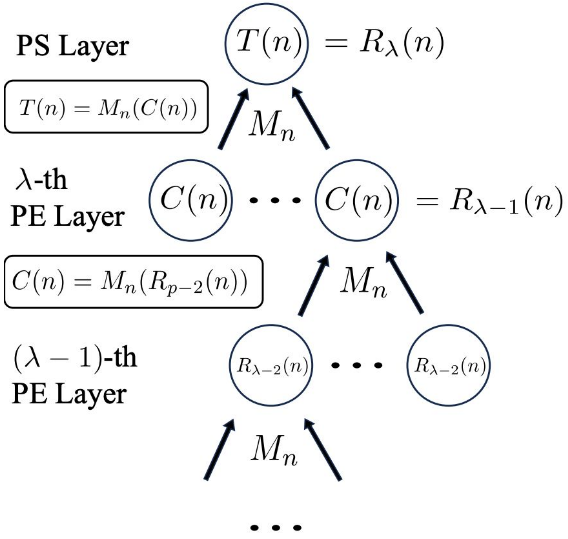
Challenge 2: Scalability of Multi-Layered Privilege (MLP).

→ **Introducing inter-enclave privileges**

→ **Introducing new execution-flow**



(b) Legacy PS-Enclave Model



(d) Multi-Layered Privilege Model

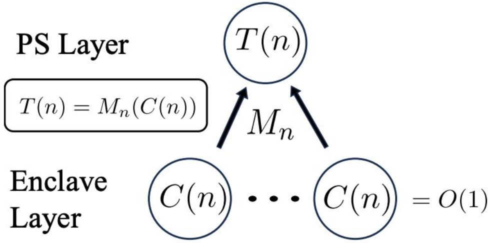
Our Design

Challenge 2: Scalability of Multi-Layered Privilege (MLP).

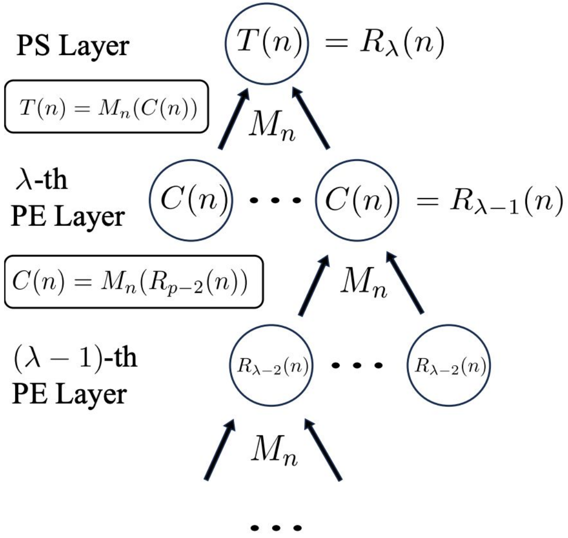
→ Introducing inter-enclave privileges

→ Introducing new execution-flow

→ **Verification state explodes!**



(b) Legacy PS-Enclave Model



(d) Multi-Layered Privilege Model

Our Design

Challenge 2: Scalability of Multi-Layered Privilege (MLP).

→ Introducing inter-enclave privileges

→ Introducing new execution-flow

→ **Verification state explodes!**

Poly(n) ⇒ 2EXP(n)

Model Complexity Explosion

Legacy TEE Platform: **Poly(n)**

MLP TEE Platform: **2EXP(n)***

*Complexity analysis refers our paper appendix

Our Design

Challenge 2: Scalability of Multi-Layered Privilege (MLP).

Solution

1. Z3 optimizations

   Skolemization

   Relevancy Propagation

2. Parameterizing layer depth λ

   Proof by Induction

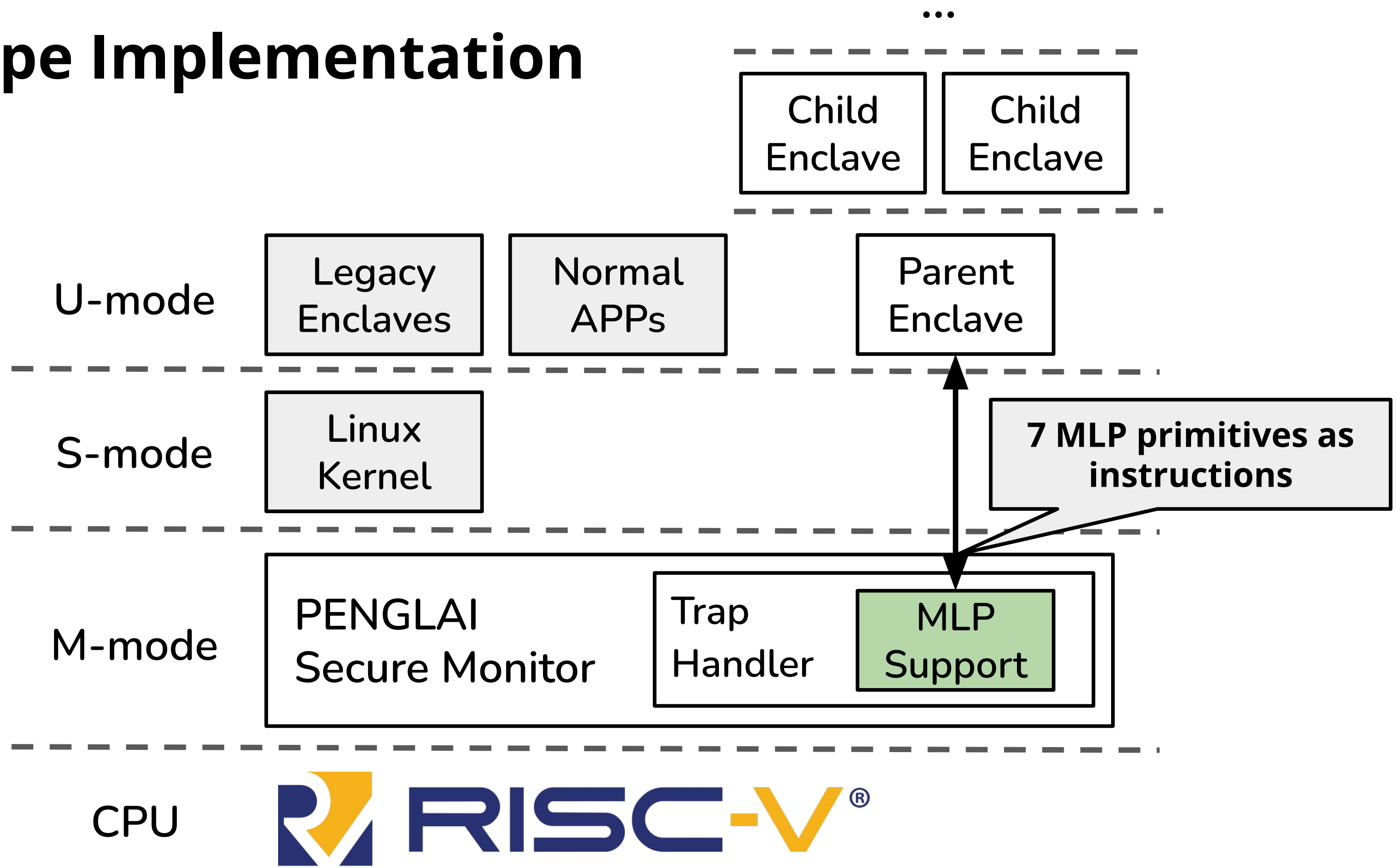
Model Complexity Explosion

(solved)

Legacy TEE Platform: **verified!**

MLP TEE Platform: **verified!**

Prototype Implementation



Evaluation: Implementation

Implementation Efforts

- **Formal Model**

  ~ 800 LoC of Formal Model

  ~ 5,000 LoC for Security Proof

- **TEE Platform**

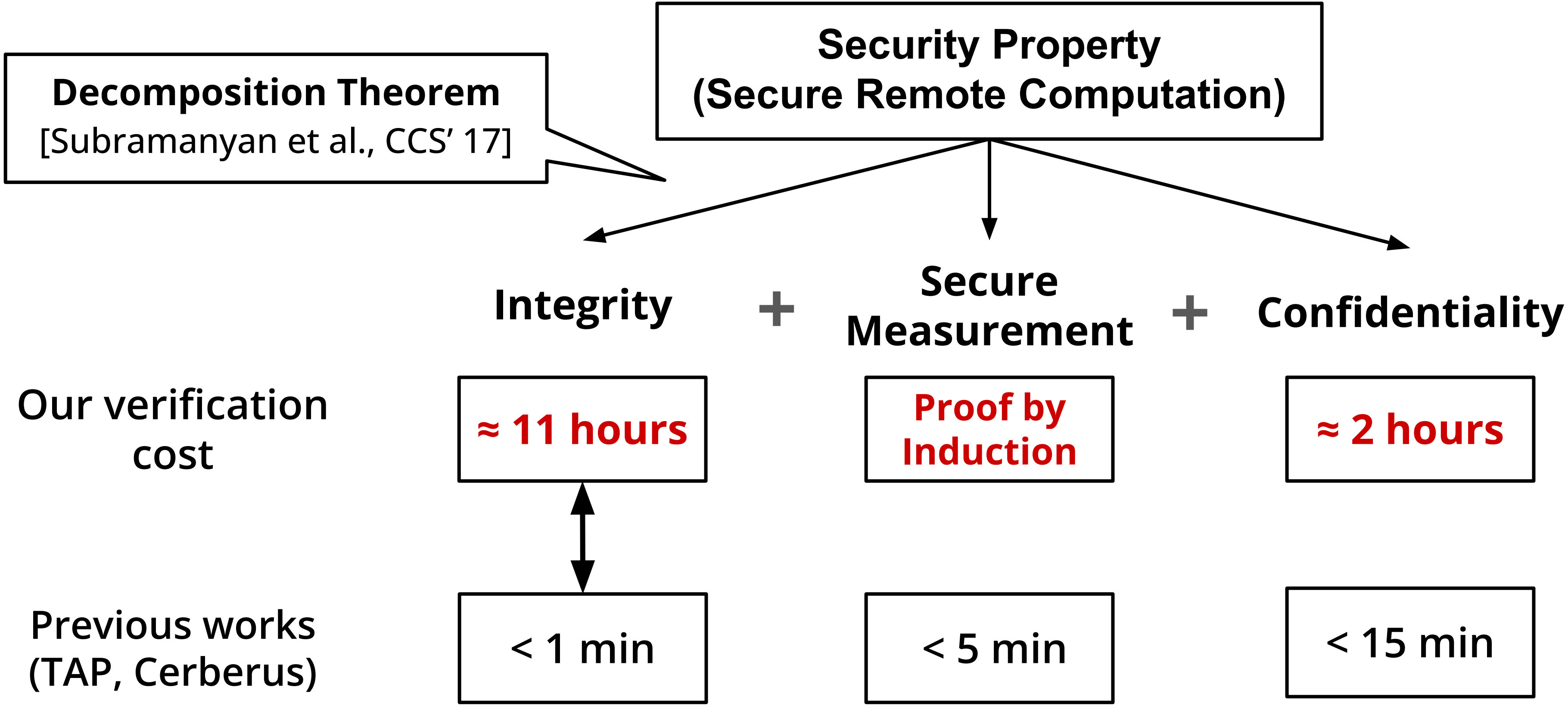
  ~ 5,000 LoC (3,300 LoC in TCB)

Environment

2 Intel Xeon Gold 5318Y CPUs, each 48 cores, 512 GB Memory

Z3 4.8.7, Boogie 2.16.0

Evaluation: Verification Costs



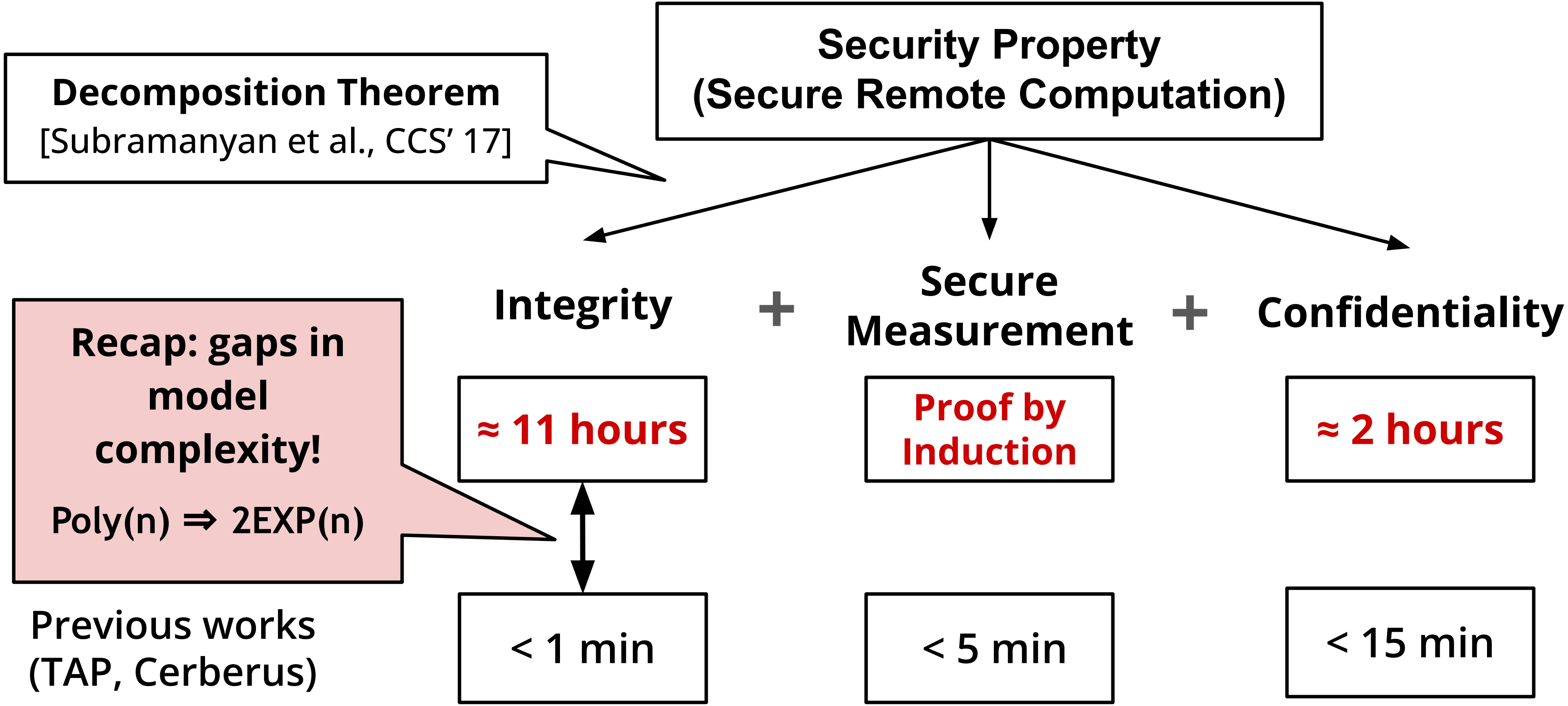
Decomposition Theorem
[Subramanyan et al., CCS’ 17]

Security Property (Secure Remote Computation)

Integrity + **Secure Measurement** + **Confidentiality**

	Integrity	Secure Measurement	Confidentiality
Our verification cost	**≈ 11 hours**	**Proof by Induction**	**≈ 2 hours**
Previous works (TAP, Cerberus)	< 1 min	< 5 min	< 15 min

Evaluation: Verification Costs



Security Property
(Secure Remote Computation)

Decomposition Theorem
[Subramanyan et al., CCS’ 17]

Integrity + Secure Measurement + Confidentiality

≈ 11 hours

Proof by Induction

≈ 2 hours

Recap: gaps in model complexity!

Poly(n) ⇒ 2EXP(n)

Previous works
(TAP, Cerberus)

< 1 min

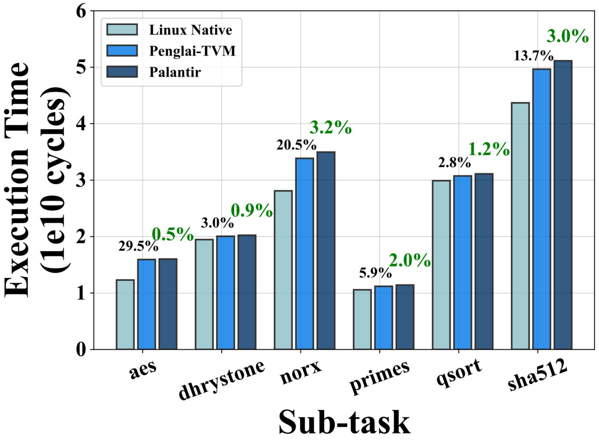
< 5 min

< 15 min

Evaluation: Implementation

Q1: Burden of PE-CE context switches?

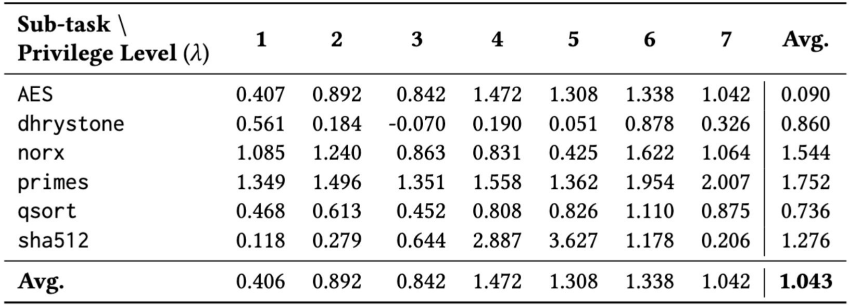
Overhead: **< 5%**



Q2: Burden of Multi-Layering?

Insight: Context switches among different layers are independent.

Overhead: **Should be a constant! (< 3%)**



Sub-task \ Privilege Level (λ)	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**Avg.**
AES	0.407	0.892	0.842	1.472	1.308	1.338	1.042	0.090
dhrystone	0.561	0.184	-0.070	0.190	0.051	0.878	0.326	0.860
norx	1.085	1.240	0.863	0.831	0.425	1.622	1.064	1.544
primes	1.349	1.496	1.351	1.558	1.362	1.954	2.007	1.752
qsort	0.468	0.613	0.452	0.808	0.826	1.110	0.875	0.736
sha512	0.118	0.279	0.644	2.887	3.627	1.178	0.206	1.276
Avg.	0.406	0.892	0.842	1.472	1.308	1.338	1.042	**1.043**

Q3: Memory Overhead for each extension?

Overhead: **Reduce O(n) to O(1) by a sharable PE.**

Evaluation: Usability

- Hierarchical Deterministic Wallet (~ 200 LoC in PE, ~27,000 LoC as runtime lib)
- Reusable Enclaves (~ 500 LoC in PE)
- Inter-Enclave Memory Sharing (~500 LoC in PE)
- Runtime Attestation (~ 100 LoC in PE)
- Enclave Introspection…

All above can be integrated into PEs!

Q & A

Artifact Available: https://github.com/arxgy/Palantir (Implementation)

https://github.com/arxgy/TAP-lambda (Formal Model)

Thanks!