UIEE: Secure and Efficient User-space Isolated Execution Environment for Embedded TEE Systems

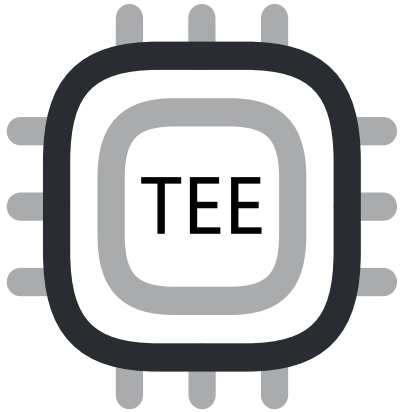
Huaiyu Yan†, Zhen Ling†, Xuandong Chen†, Xinhui Shao†§, Yier Jin‡, Haobo Li†
Ming Yang†, Ping Jiang†, Junzhou Luo†¶
Presenter: Guangchi Liu†

†*Southeast University*, ‡*University of Science and Technology of China*
§*City University of Hong Kong*, ¶*Fuyao University of Science and Technology*

Trusted Execution Environment

Touch ID

Face ID

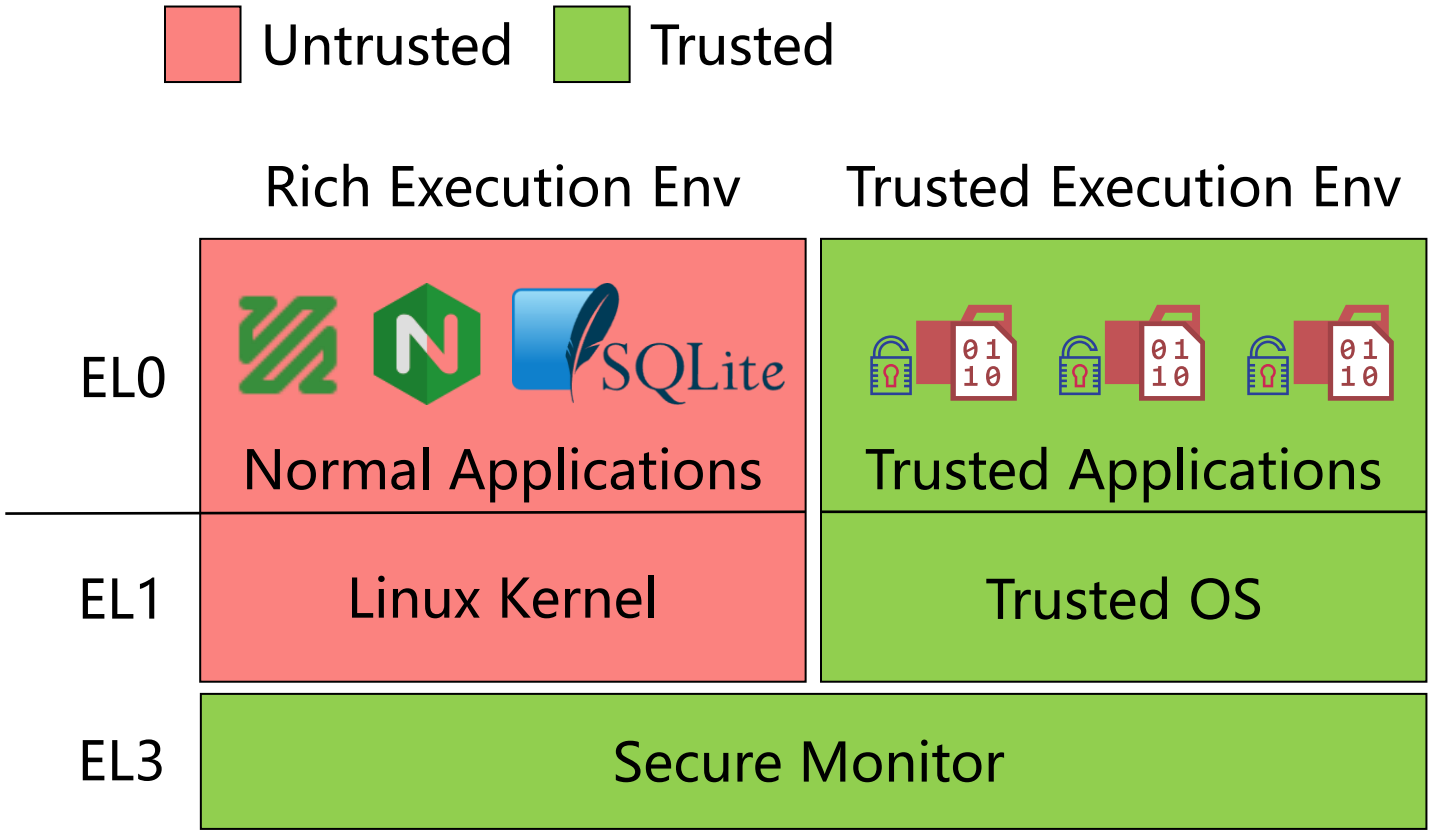


DRM Video Stream

Mobile Payment

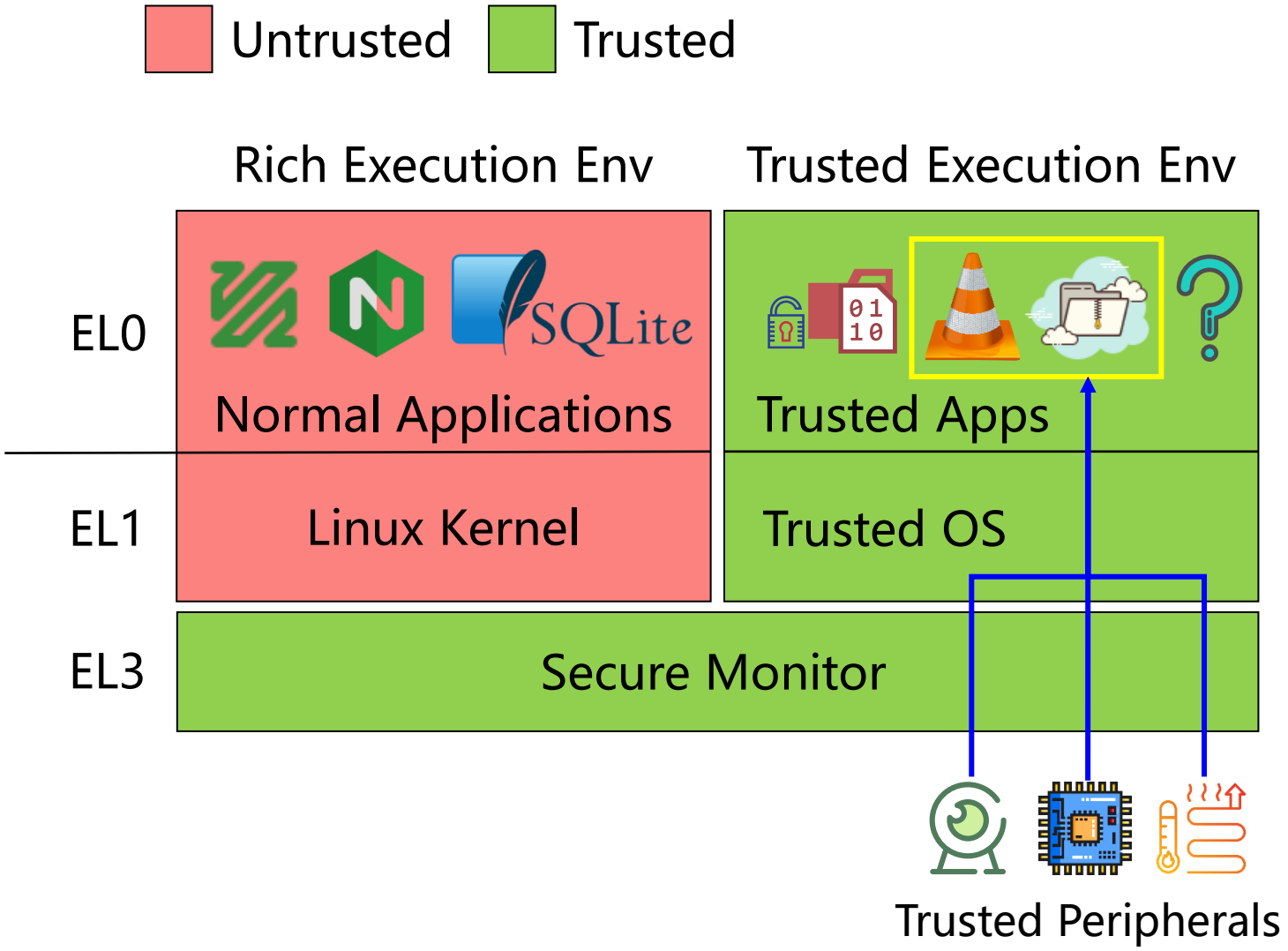
- Secure execution environment for sensitive operations
- Compact to reduce attack surfaces
  - Limited secure operations, such as cryptographical operations

ARM TrustZone



- ARM TrustZone divides system resources into two domains: Rich Execution Environment (**REE**) & Trusted Execution Environment (**TEE**)
- REE for normal applications with **rich** functionalities
- TEE for trusted applications with security-oriented while **limited** capabilities

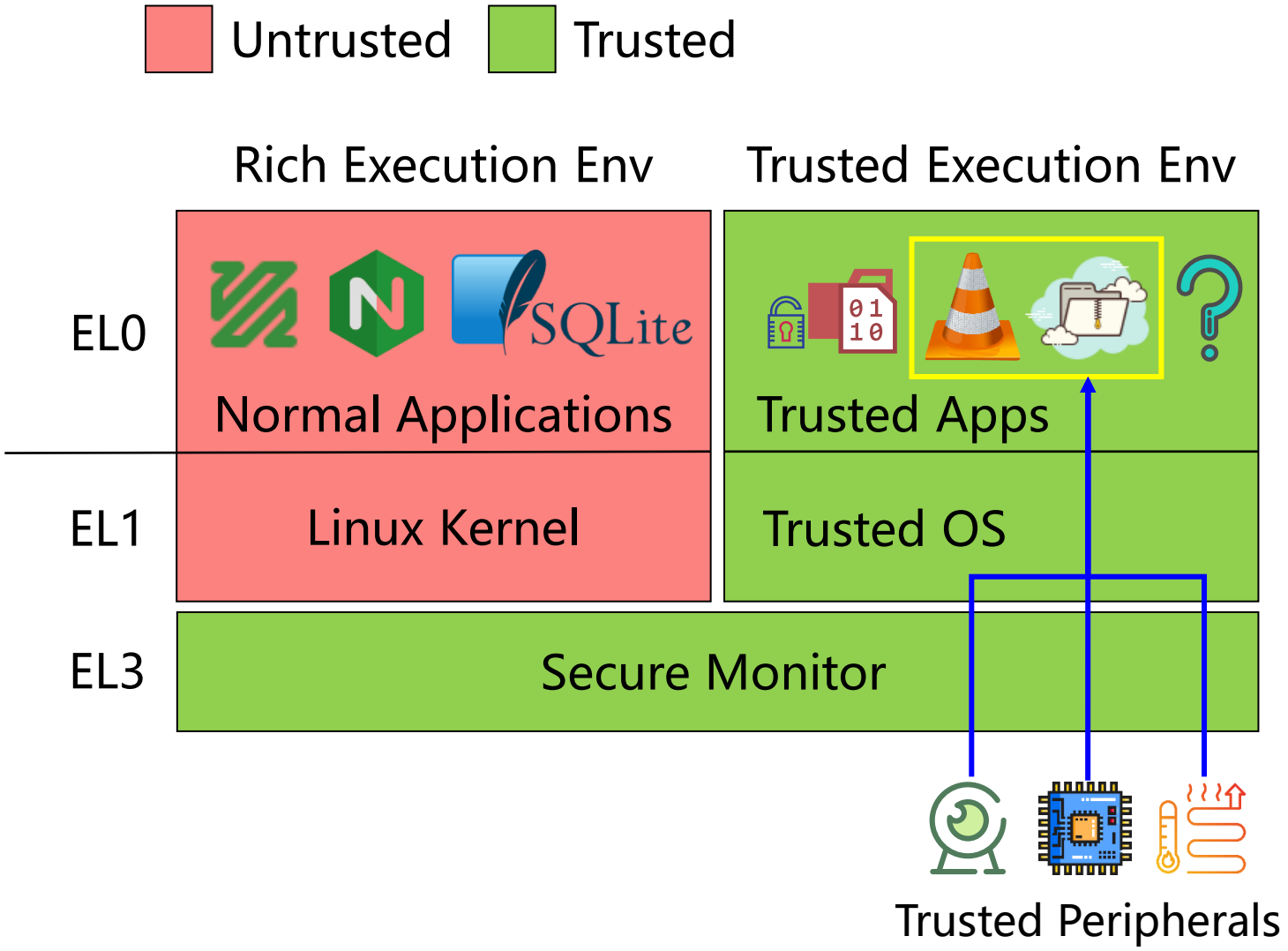
Motivations



Goals: Trusted applications (TA) with **rich** capabilities

- Versatile data processing
- Secure input & output
- Secure machine learning & model inferencing
- Video & image codec

Motivations



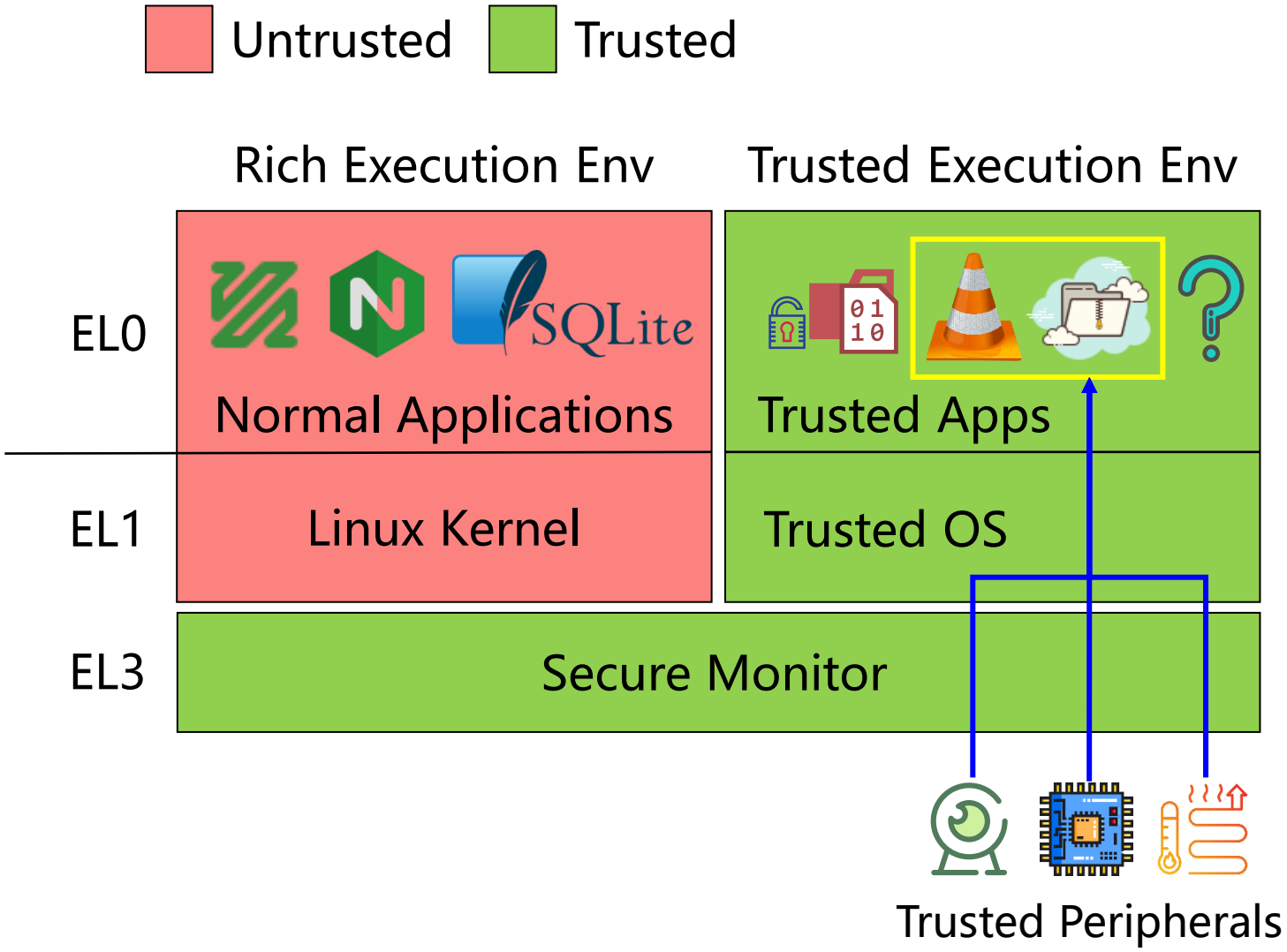
Goals: Trusted applications (TA) with **rich** capabilities

- Versatile data processing
- Secure input & output
- Secure machine learning & model inferencing
- Video & image codec

Limitations: **Limited** software runtime support inside TEE

- Limited **API** support
- Limited **system call** support

Motivations



Goals: Trusted applications (TA) with **rich** capabilities

- Versatile data processing
- Secure input & output
- Secure machine learning & model inferencing
- Video & image codec

Contradict

Limitations: **Limited** software runtime support inside TEE

- Limited **API** support
- Limited **system call** support

Our Approach – UIEE

Our Goal

Towards versatile applications with **rich capabilities** inside TEE

User-space Isolated Execution Environment

- Create an **isolated execution environment** (IEE) separated from both TEE and REE for applications
- Build a C language runtime based on a **library operating system** (LibOS) so as to run existing Linux applications without modification

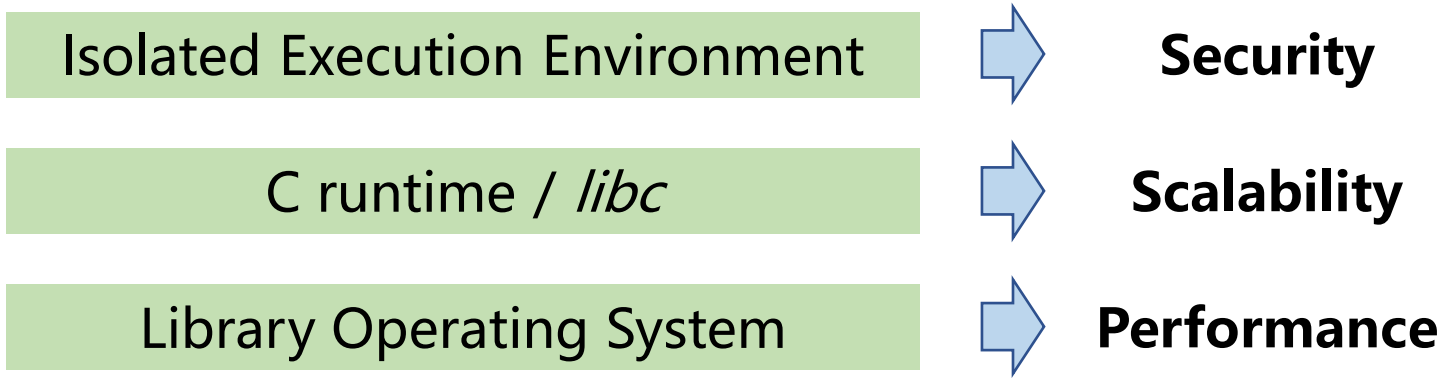
Our Approach – UIEE

Our Goal

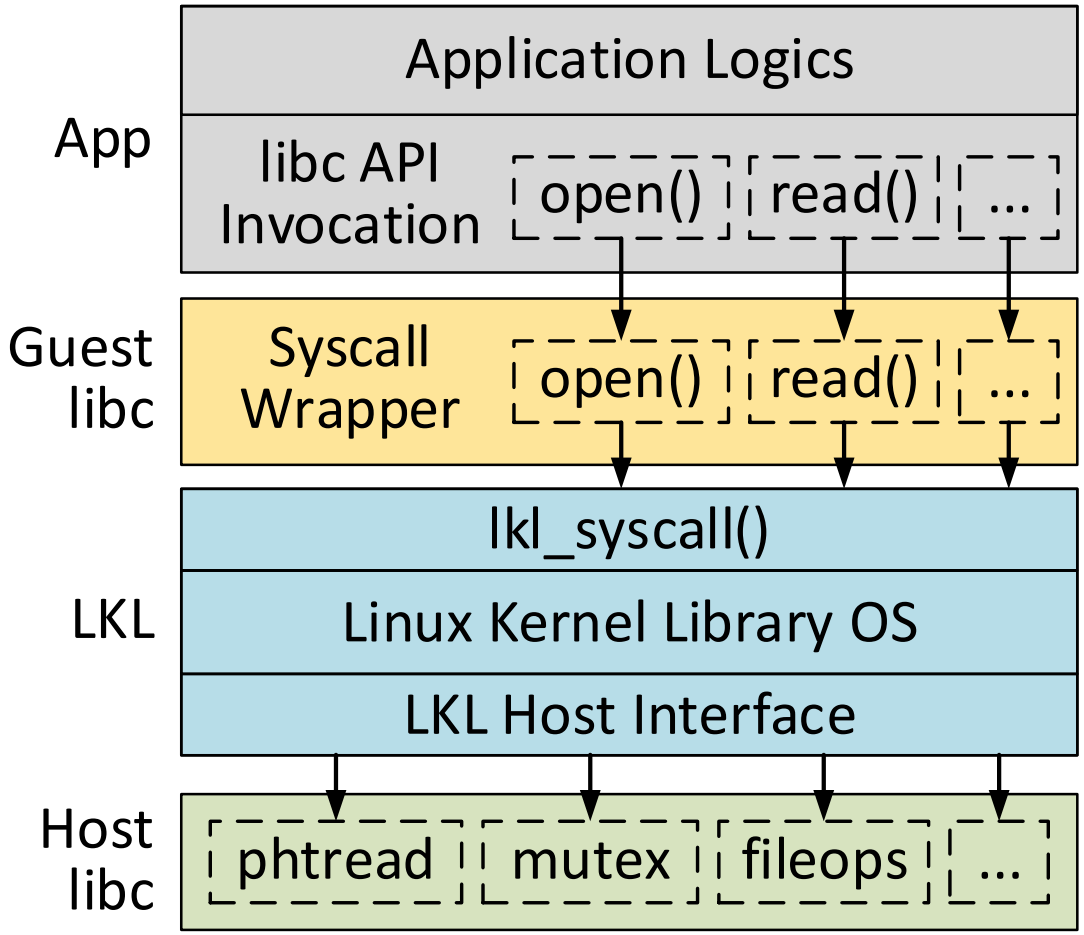
Towards versatile applications with **rich capabilities** inside TEE

User-space Isolated Execution Environment

- Create an **isolated execution environment** (IEE) separated from both TEE and REE for applications
- Build a C language runtime based on a **library operating system** (LibOS) so as to run existing Linux applications without modification



Linux Kernel Library (LKL) Library OS



- **App** : unmodified user-space applications
- **Guest libc** : a customized C library providing *libc* APIs to applications
- **LKL** : a library OS based on the Linux kernel providing system call services to the guest *libc* through function calls instead of traps
- **Host *libc*** : a standard C library providing dependency functions to implement the LKL host interfaces

Challenges

Challenge 1: Thread Creation Issues

LKL is a multi-threading library OS and requires ***pthread*-compatible APIs** to create LKL kernel threads, which is not supported inside TEE

Challenge 2: Thread Context Switching Issues

An LKL thread context is managed by both user-space host libc and Linux kernel. LKL **cannot conduct thread context switching inside UIEE** since the Linux kernel thread context is not accessible from UIEE

Challenges & Solutions

Challenge 1: Thread Creation Issues

LKL is a multi-threading library OS and requires ***pthread*-compatible APIs** to create LKL kernel threads, which is not supported inside TEE

Two-stage Bootstrapping

- First **initialize LKL inside REE** using host libc and then go into UIEE for application execution

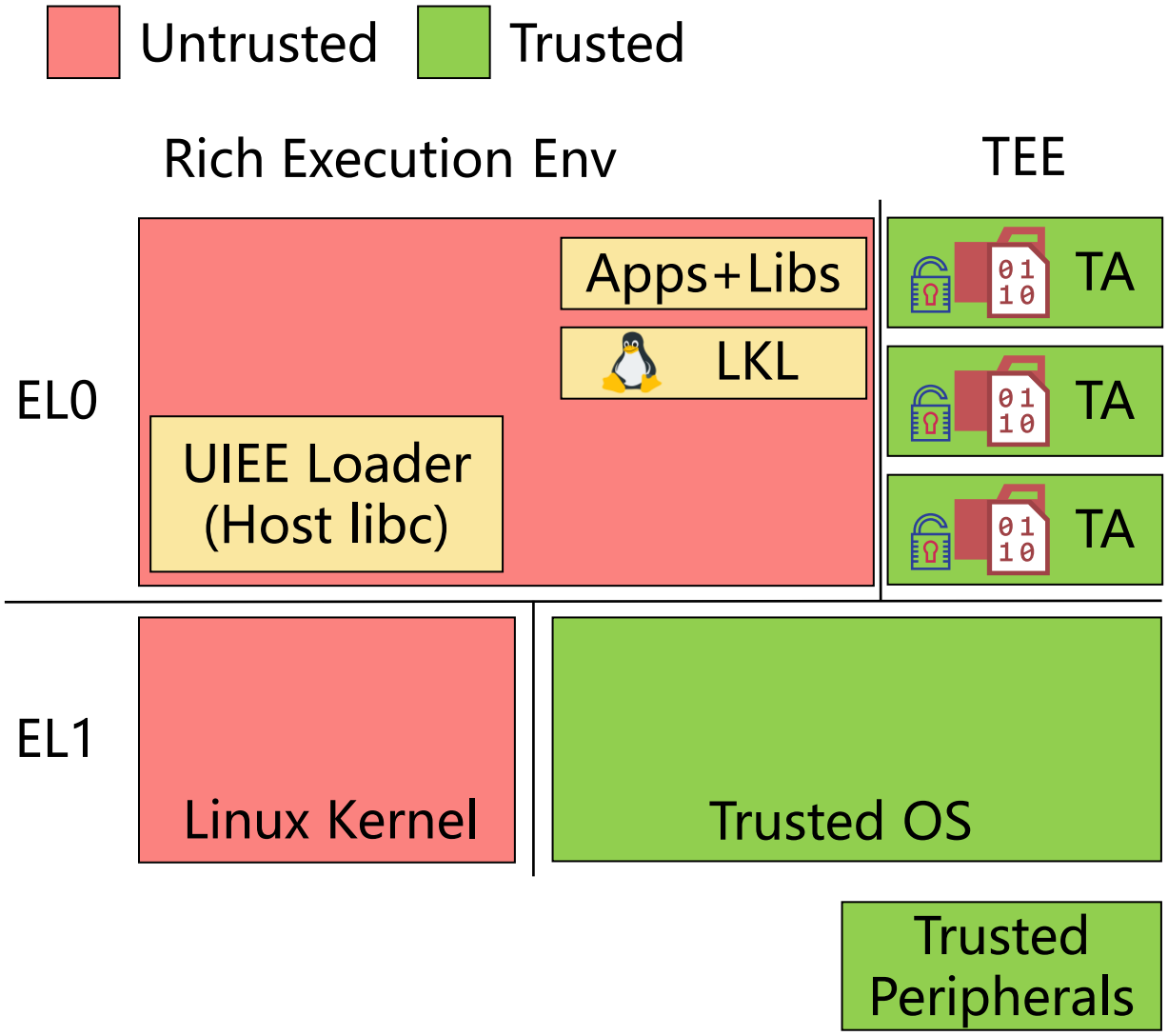
Challenge 2: Thread Context Switching Issues

An LKL thread context is managed by both user-space host libc and Linux kernel. LKL **cannot conduct thread context switching inside UIEE** since the Linux kernel thread context is not accessible from UIEE

On-demand Thread Migration

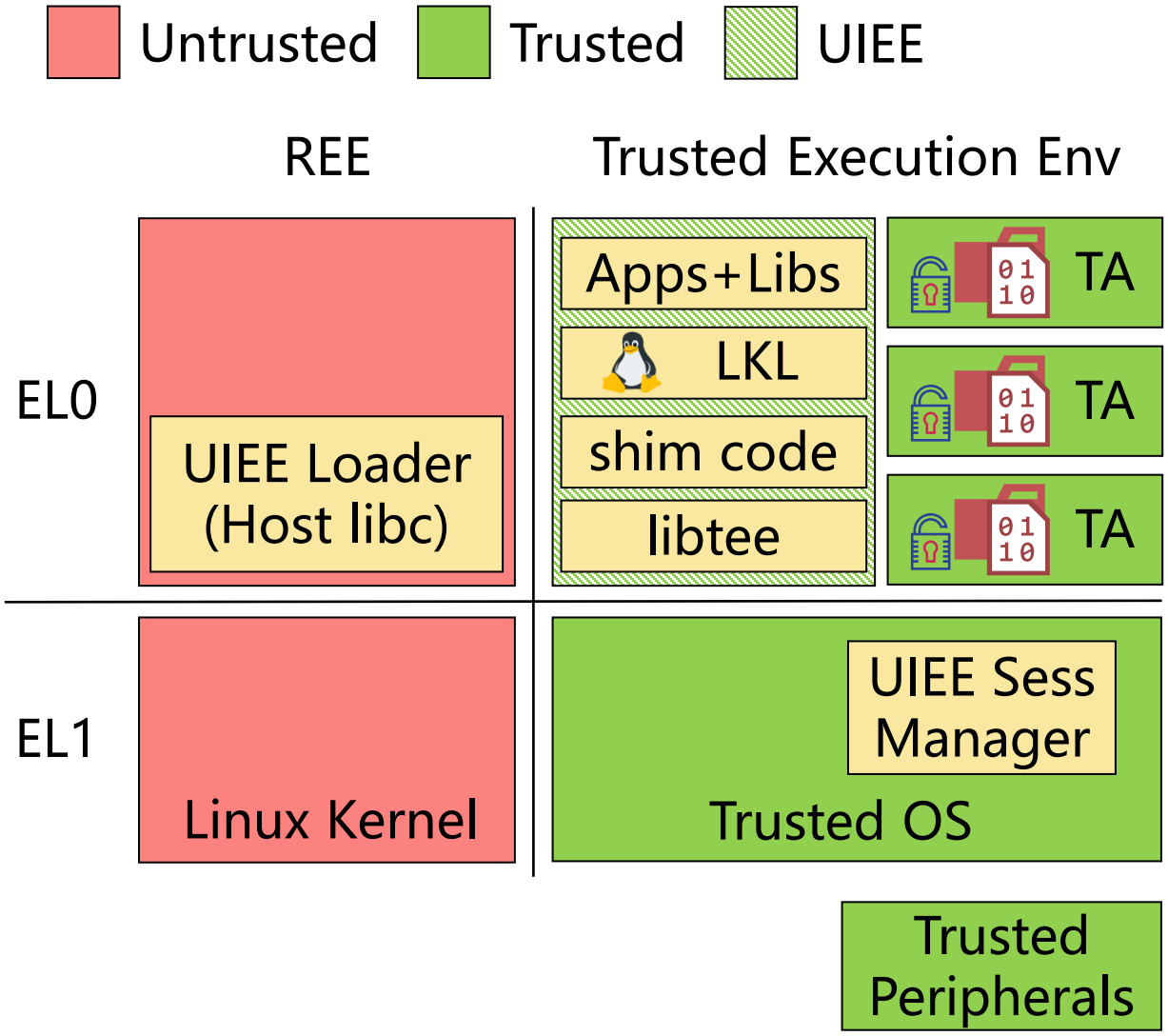
- **Reconstruct** LKL thread context inside trusted OS only when an LKL thread is scheduled

First-stage Bootstrapping



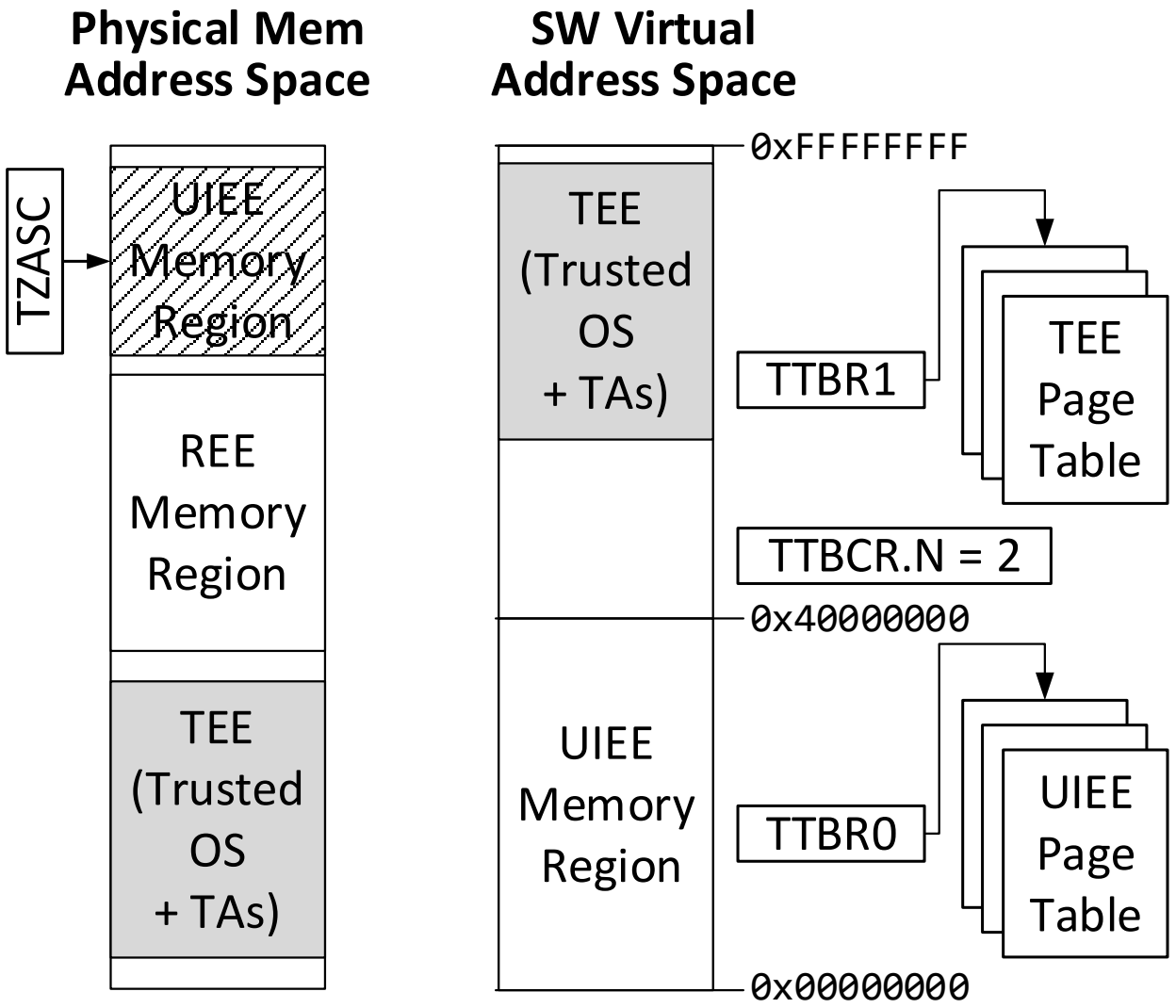
- After secure boot, UIEE loader **allocates UIEE memory region** using the Linux kernel continuous memory allocator (CMA)
- UIEE loader **loads images** of app, libraries and LKL, **relocates images** based on their ELF headers, and **initializes stacks and heaps**
- LKL **creates LKL threads and gets initialized** using host interfaces based on host *libc*

Second-stage Bootstrapping



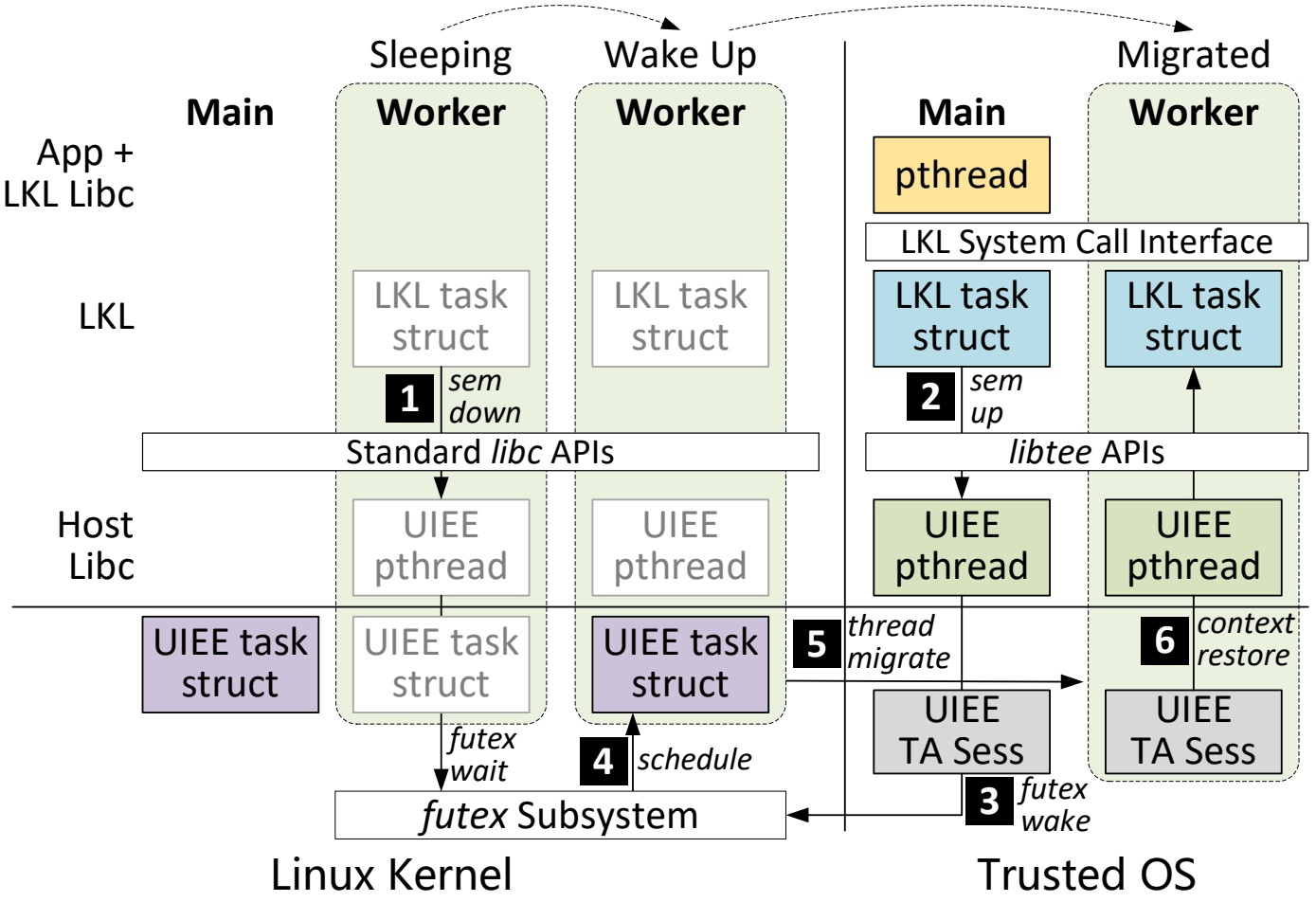
- UIEE session manager **configures UIEE memory region as secure memory**, isolating UIEE from REE
- A new TEE-side library ***libtee*** is loaded to implement LKL host interfaces inside UIEE
- UIEE session manager **creates thread contexts** for UIEE inside trusted OS and **manages UIEE life cycle**

Memory Isolation



- UIEE memory region is configured as secure using TrustZone address space controller (**TZASC**)
- UIEE memory region occupies **the first 1 GB** of the whole virtual address space (32-bit)
- A new page table is **reconstructed** inside the trusted OS based on the original page table inside the REE Linux kernel used during the first bootstrapping phase

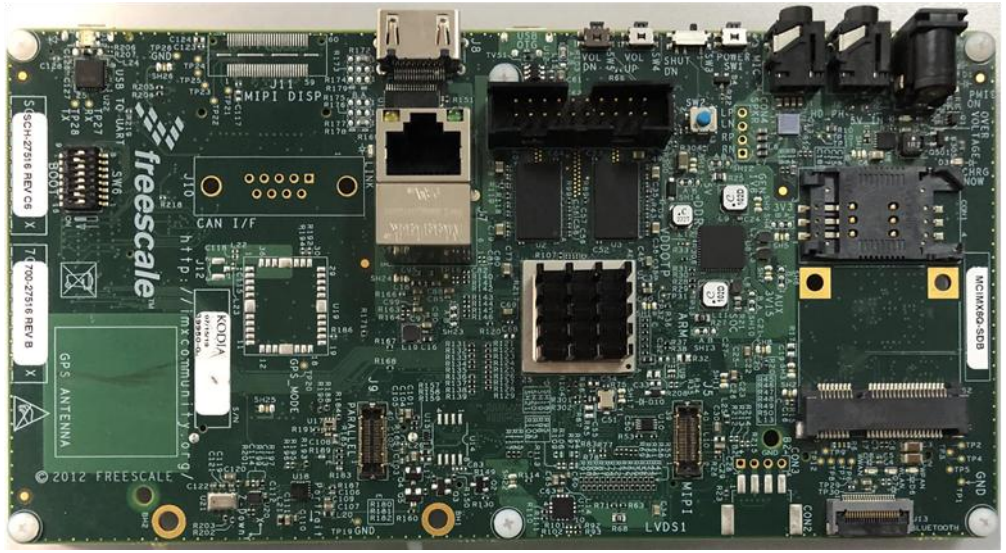
On-demand Thread Migration



During UIEE execution, an LKL kernel thread context is migrated from REE into TEE in an **on-demand** way

- Main thread wakes up an LKL kernel thread through **semaphore**
- UIEE redirects corresponding **fast mutex system call** back to REE Linux kernel
- Instead of going into REE user space, the LKL kernel thread **resumes executions inside UIEE**, where UIEE session manger reconstructs its thread context inside trusted OS

UIEE Prototype



IMX6Q SABRESD

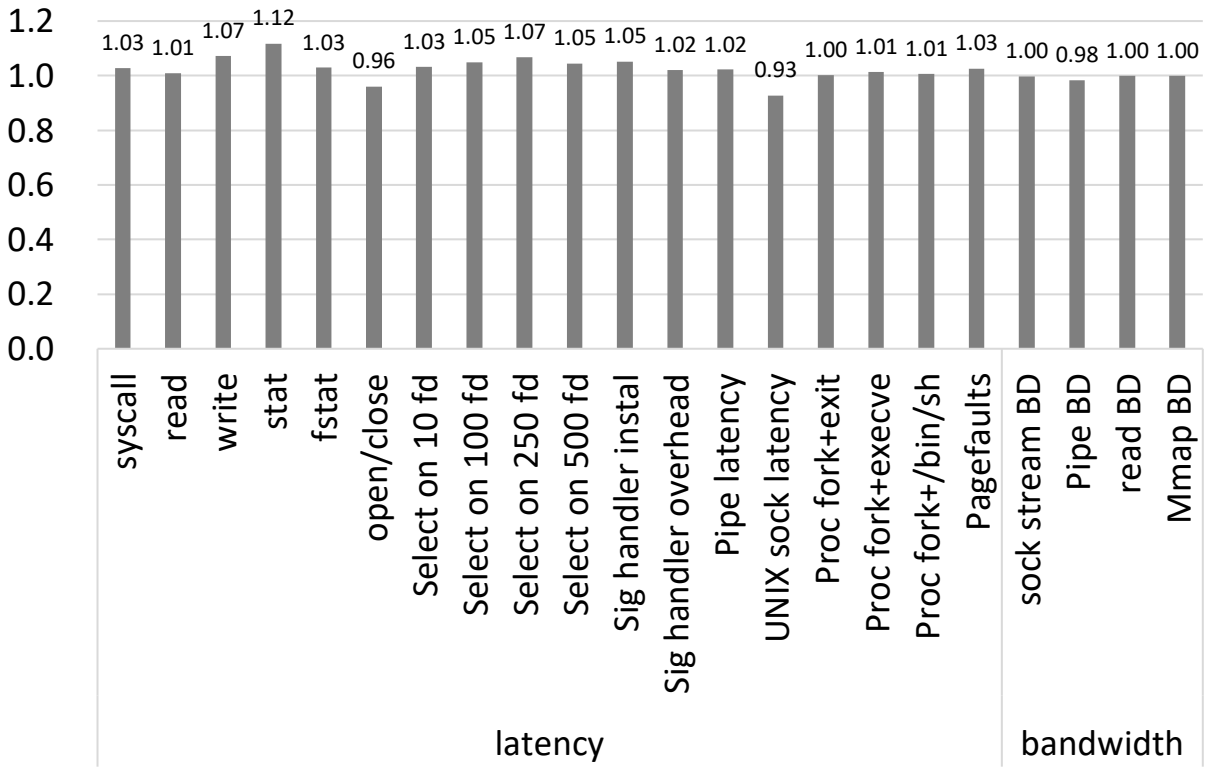
- Qual-core **Cortex-A9**
- 1GB RAM
- Rich Device Support

Line of Code (LoC) Statistics of UIEE Components

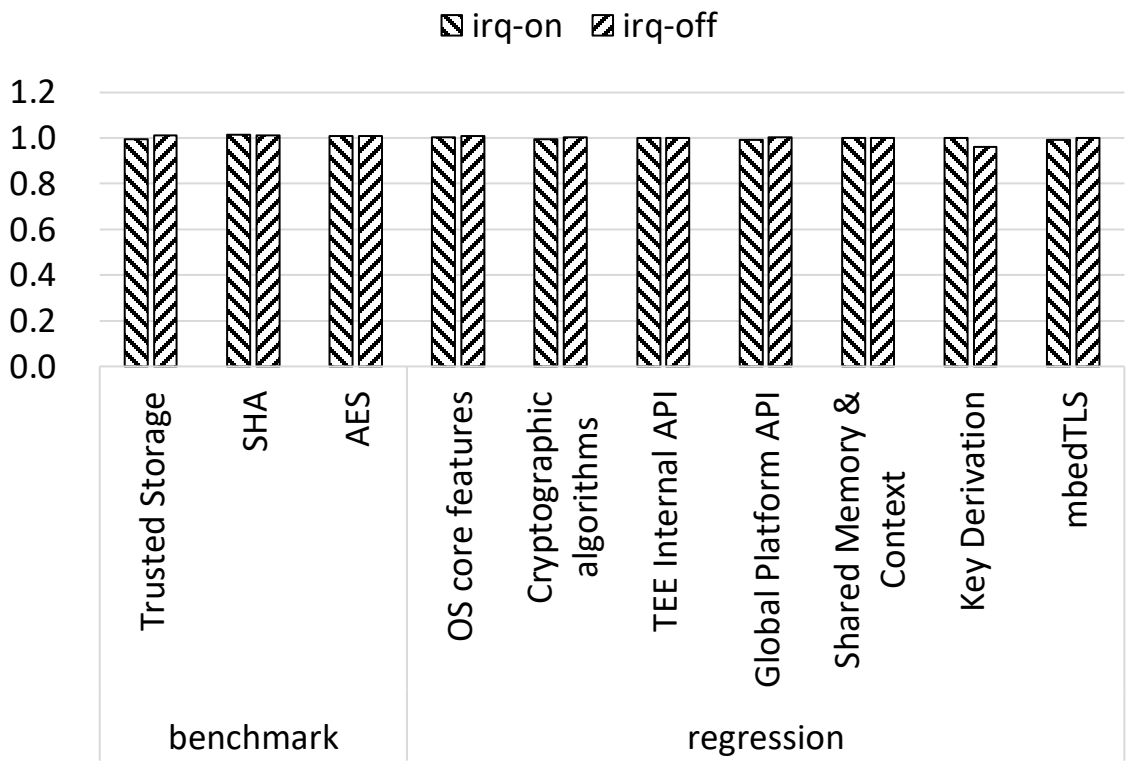
Component	Version	Added LoC
REE OS	Linux 6.6.0	961
Trusted OS	OPTEE OS 4.3.0	1997
UIEE Driver	Customized LKM	532
UIEE Loader	musl libc 1.2.4	2633
LKL	Linux 4.14	679
LKL *libc*	musl libc 1.2.4	43
libtee	Customized TA	1232
TEE Deamon	OPTEE tee-supplicant	340
Total LoC		**8417**

- UIEE only introduces little TCB increase
  - LoC increase : **0.46%**
  - Trusted OS image size increase : **3.37%**

Performance Evaluation – System Benchmark



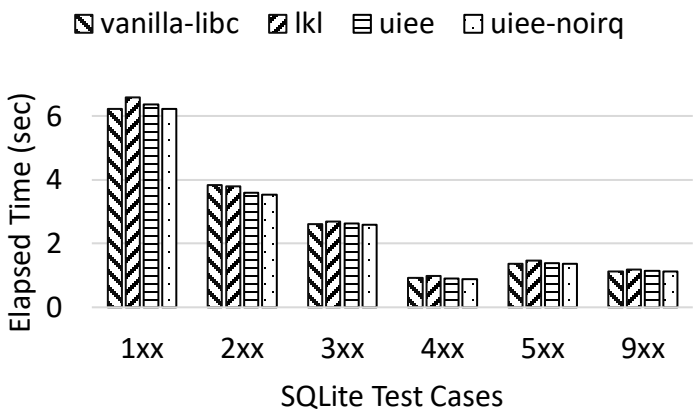
lmbench



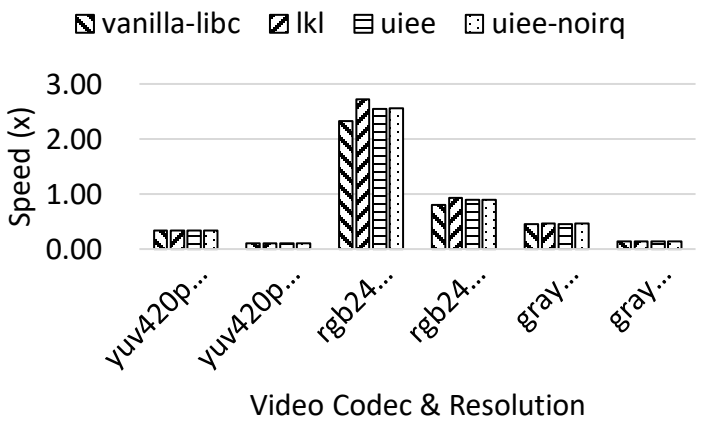
OPTEE xtest

- UIEE preserves all OPTEE functionalities and introduces **little** performance overhead to existing trusted services

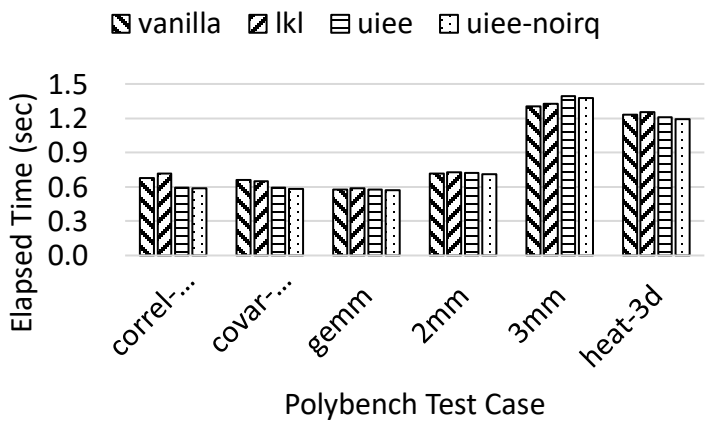
Performance Evaluation – Application Case Studies



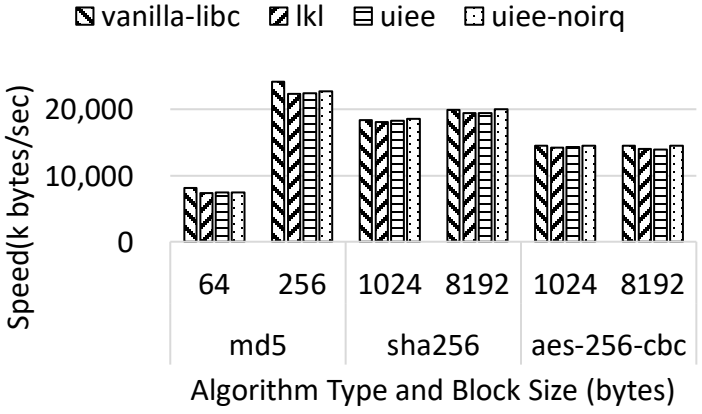
SQLite speedtest1



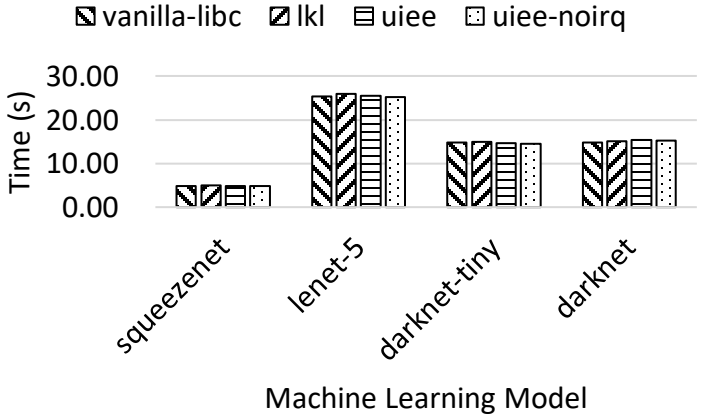
FFmpeg Video Encoding



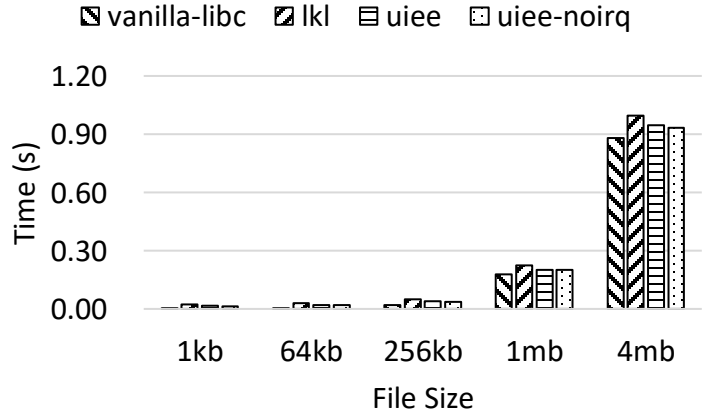
Polybench



OpenSSL



Machine Learning

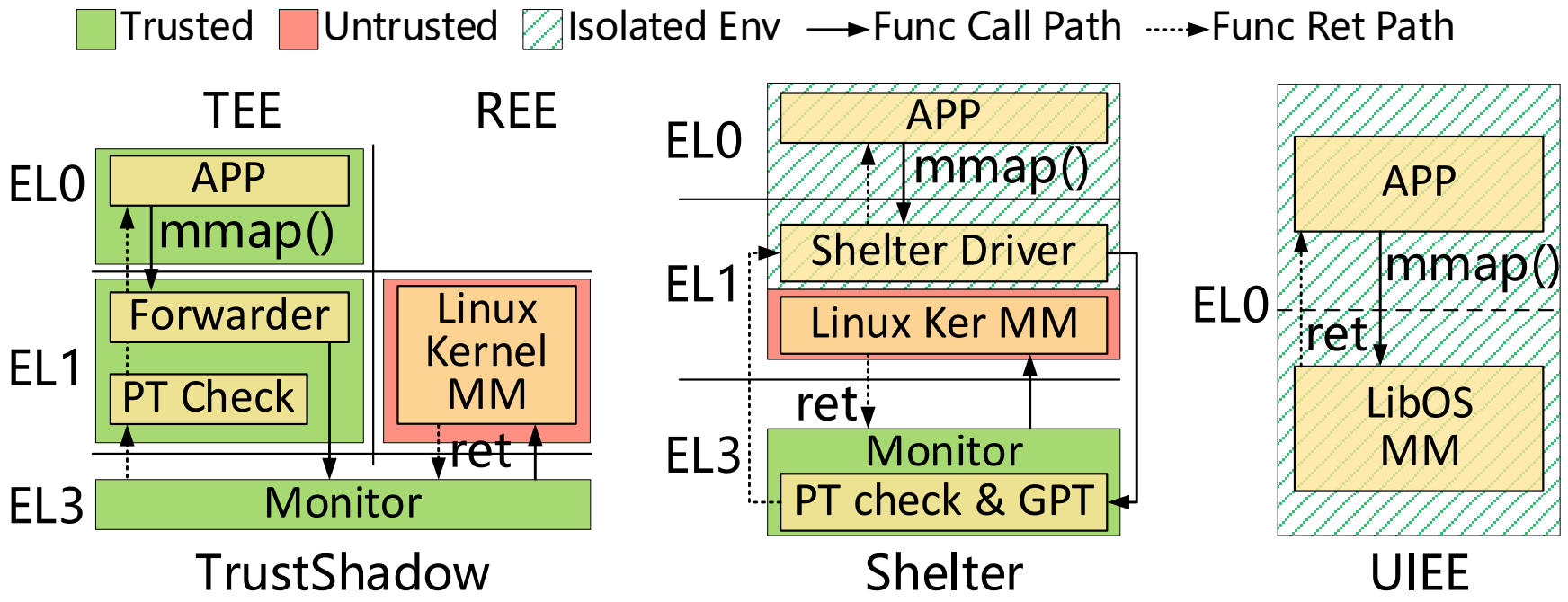


bzip2

- UIEE introduces **little** performance overhead to applications

Performance Evaluation – SOTA Comparison

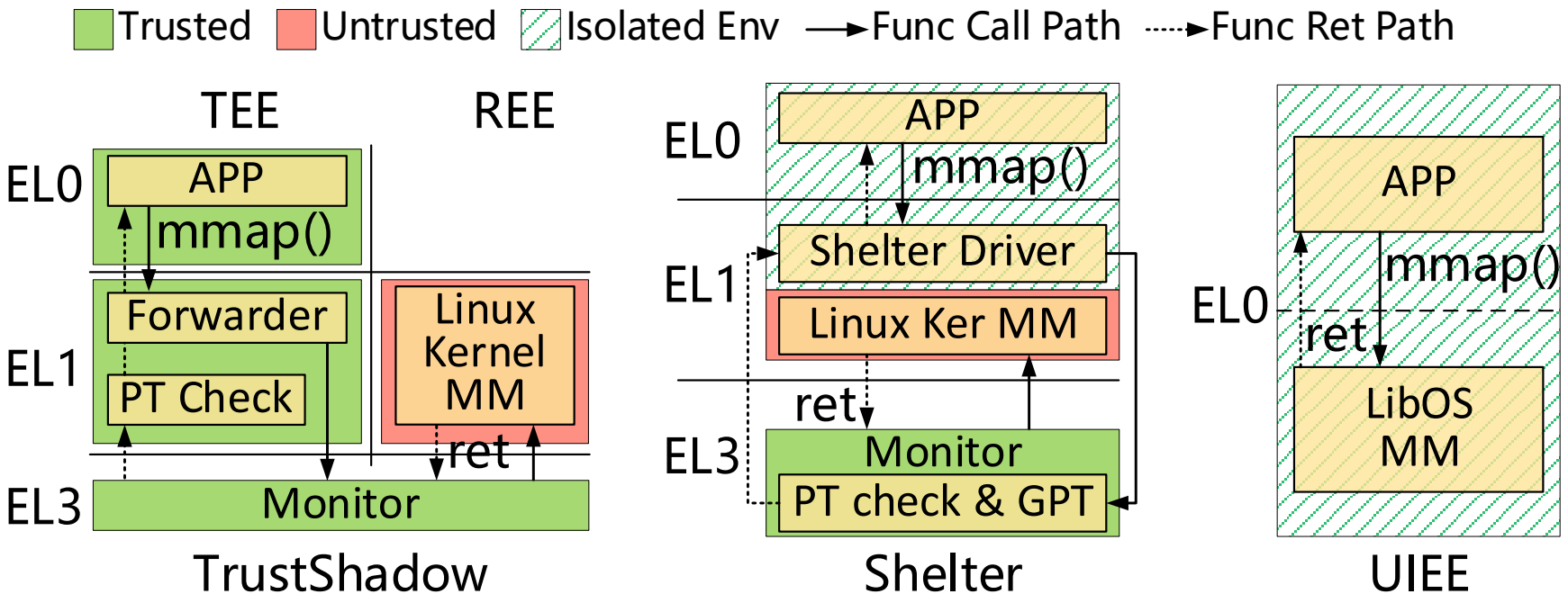
UIEE vs TrustShadow[1] & Shelter[2]



1. Guan L, Liu P, Xing X, Ge X, Zhang S, Yu M, Jaeger T. **Trustshadow: Secure Execution of Unmodified Applications with ARM TrustZone**. In Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys) 2017, pp. 488-501.
2. Zhang Y, Hu Y, Ning Z, Zhang F, Luo X, Huang H, Yan S, He Z. **SHELTER: Extending Arm CCA with Isolation in User Space**. In Proceedings of the 32nd USENIX Security Symposium (Security) 2023, pp. 6257-6274.

Performance Evaluation – SOTA Comparison

Theoretical Analysis



- For a single system call, both **TrustShadow** and **Shelter** involve **2 EL0-EL1** and **4 EL1-El3** exception level switches
- UIEE only involves **a user-space function call** and is thus expected to exhibit better performance

Performance Evaluation – SOTA Comparison Cont.

SQLite Application Benchmark

Case	syscall statistics			uiee	forward		forward & PT check	
	mmap #	munmap #	#/s	time (s)	time (s)	overhead	time (s)	overhead
1xx	1062	985	322.1	6.356	6.956	**9.43%**	7.041	**10.77%**
2xx	828	827	460.7	3.592	4.355	**21.24%**	4.448	**23.84%**
3xx	100	100	76.0	2.633	2.676	1.63%	2.689	2.11%
4xx	44	41	93.8	0.906	0.961	**5.99%**	0.968	**6.80%**
5xx	145	140	205.3	1.388	1.448	4.27%	1.460	**5.14%**
9xx	6	4	8.7	1.146	1.124	-1.92%	1.126	-1.77%

- We conduct performance comparison under 3 conditions:
  a) Run SQLite in UIEE and handle `m(un)map` system calls in UIEE
  b) Run SQLite in REE and forward `m(un)map` system calls to TEE to emulate world switches
  c) Run SQLite in REE and forward `m(un)map` system calls to TEE with page table (PT) checking
- For SQLite application benchmark, UIEE **outperforms** both TrustShadow & Shelter

Conclusion

- A **TrustZone-oriented Library OS** to provide standard C runtime inside TEE with minimal TCB increase
- A novel **two-stage bootstrapping** method and **on-demand thread migration** method to enable LKL thread creation and LKL thread context switching inside UIEE
- A UIEE prototype on IMX6Q SABRESD development board and **8** real-world application case studies with **low runtime overhead**

東南大學
SOUTHEAST UNIVERSITY


中国科学技术大学
University of Science and Technology of China


香港城市大學
City University of Hong Kong

福耀科技大學
FUYAO UNIVERSITY OF SCIENCE AND TECHNOLOGY

Q&A

Thanks!