

安徽工业大学
ANHUI UNIVERSITY OF TECHNOLOGY

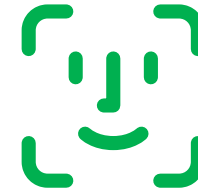
LDR: Secure and Efficient Linux Driver Runtime for Embedded TEE Systems

*Huaiyu Yan**Zhen Ling**Haobo Li**Lan Luo**Xinhui Shao**Kai Dong**Ping Jiang**Ming Yang**Junzhou Luo**Xinwen Fu*
(Presenter)

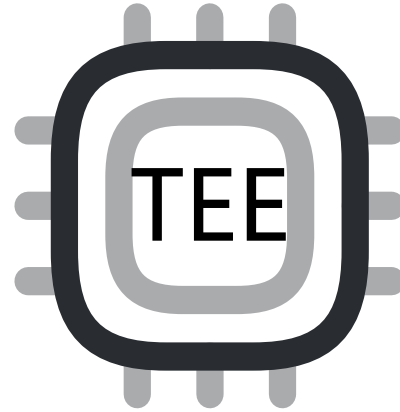
Trusted Execution Environment



Touch ID



Face ID



DRM Video
Stream

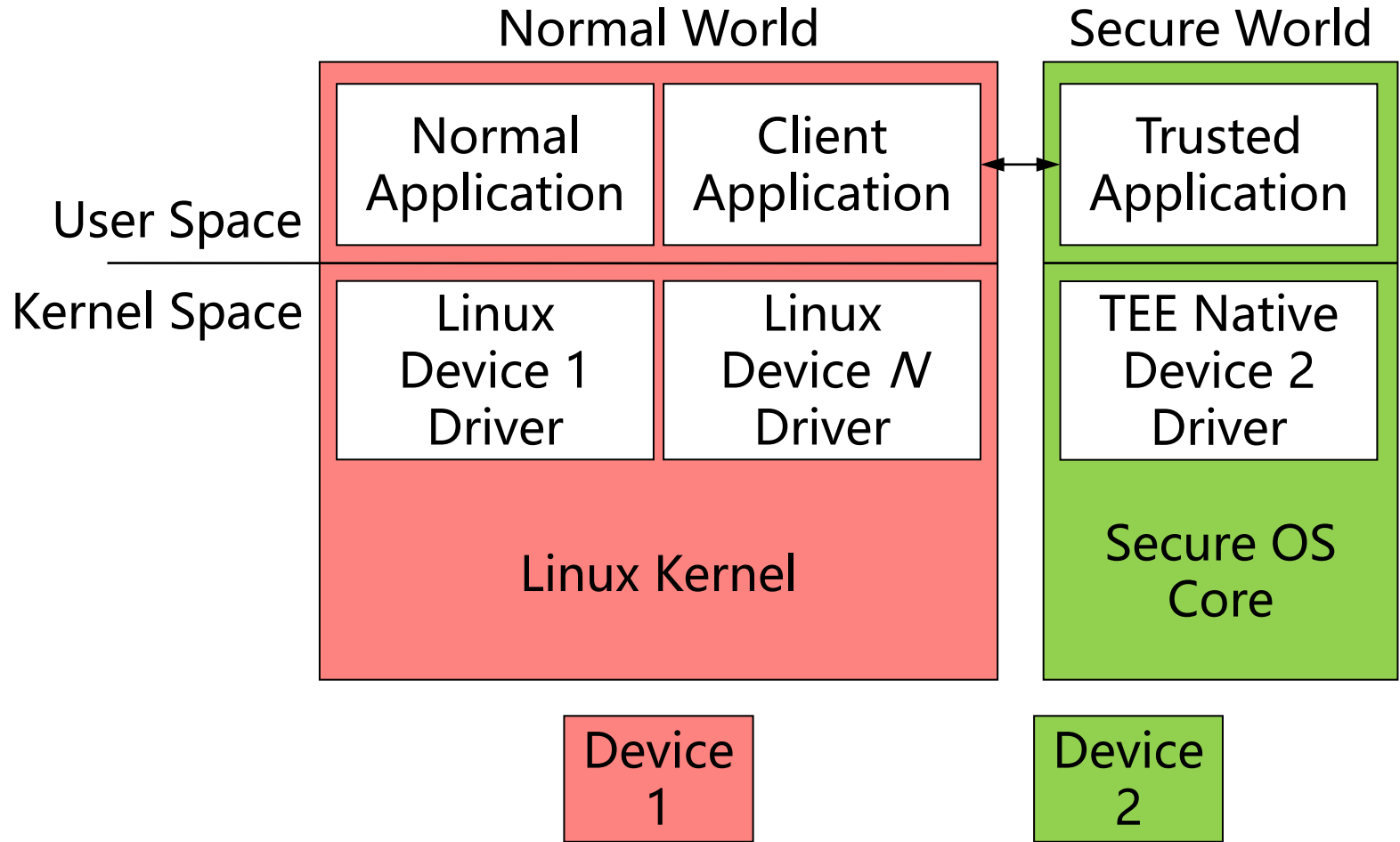


Mobile
Payment

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

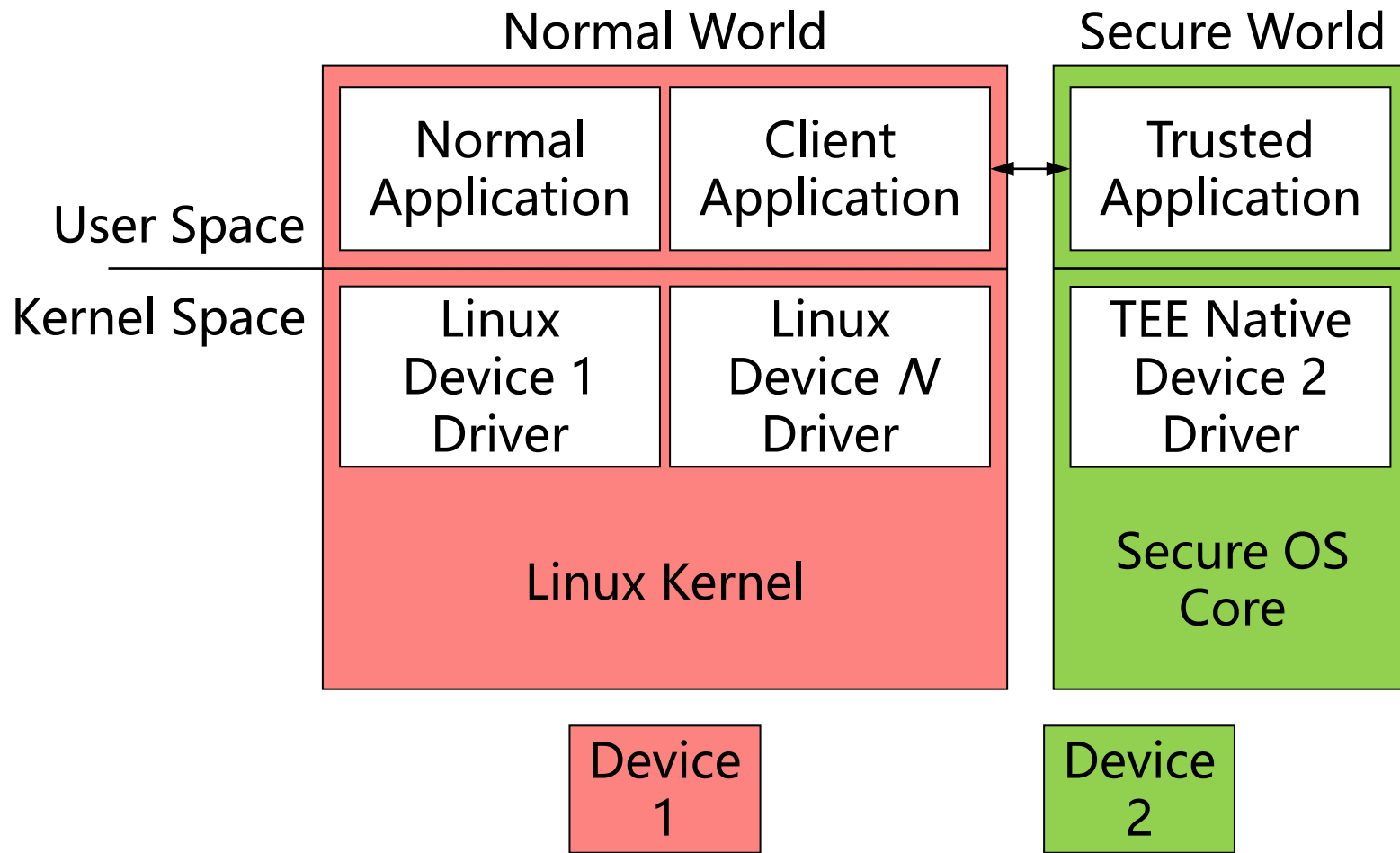
ARM TEE Architecture

Untrusted Trusted



Motivations

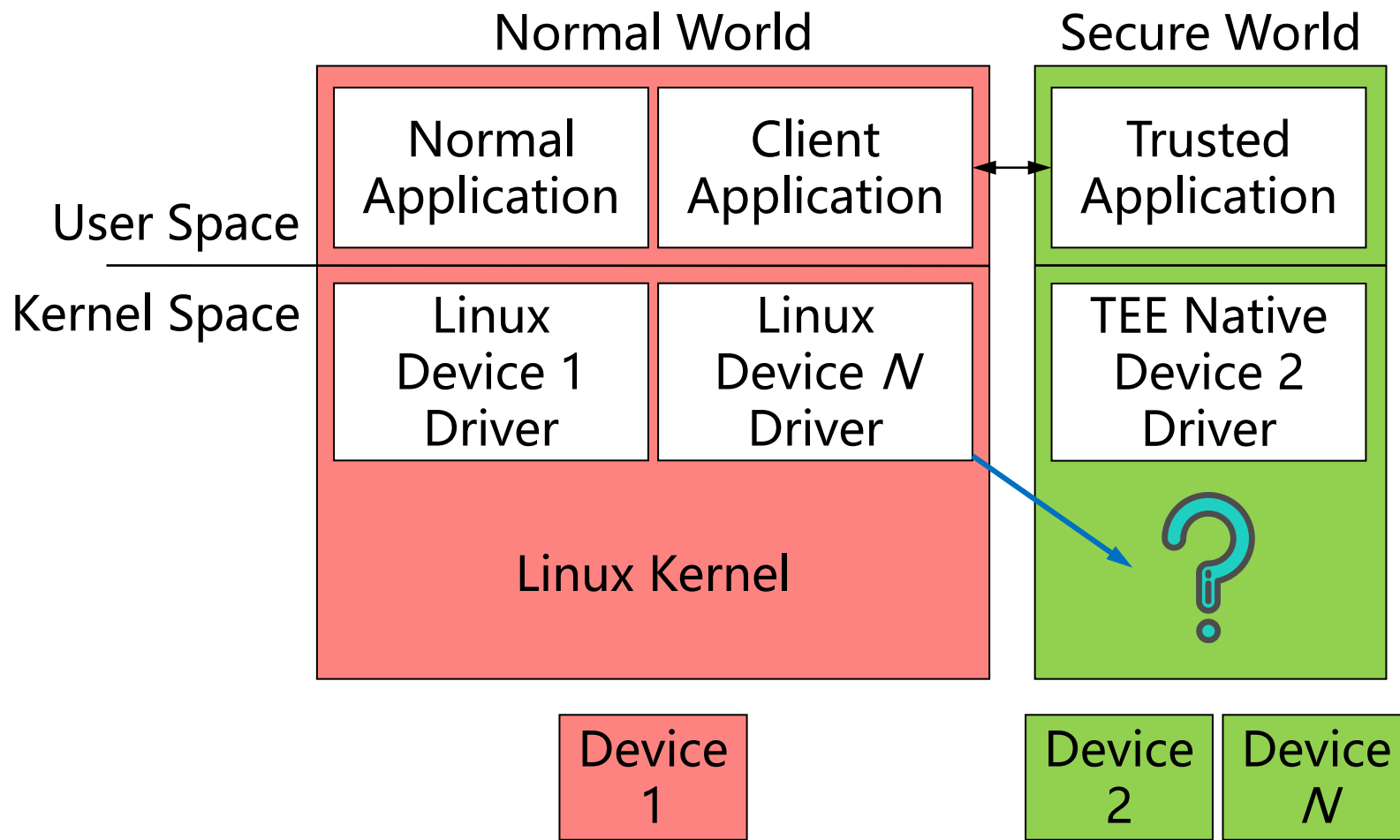
Untrusted Trusted



- TEE has poor device driver support
- Driver porting requires significant efforts

Motivations

Untrusted Trusted



- TEE has poor device driver support
- Driver porting requires significant efforts

Can we reuse existing Linux kernel drivers?

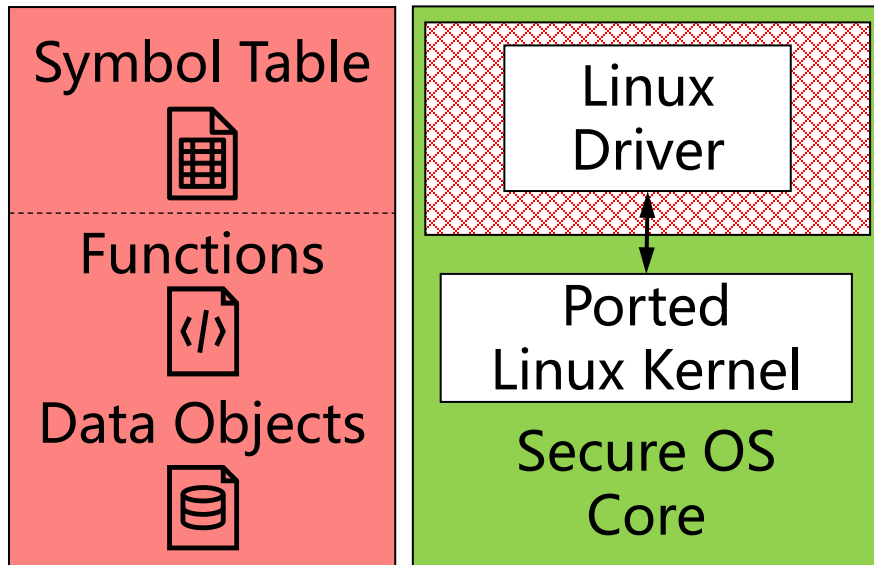
Intuitive Design Options

- Drivers is **sandboxed** for a minimal Trusted Computing Base (TCB)
- Driver **dependency functions** are provided inside secure world

Untrusted Trusted Sandbox

Normal World

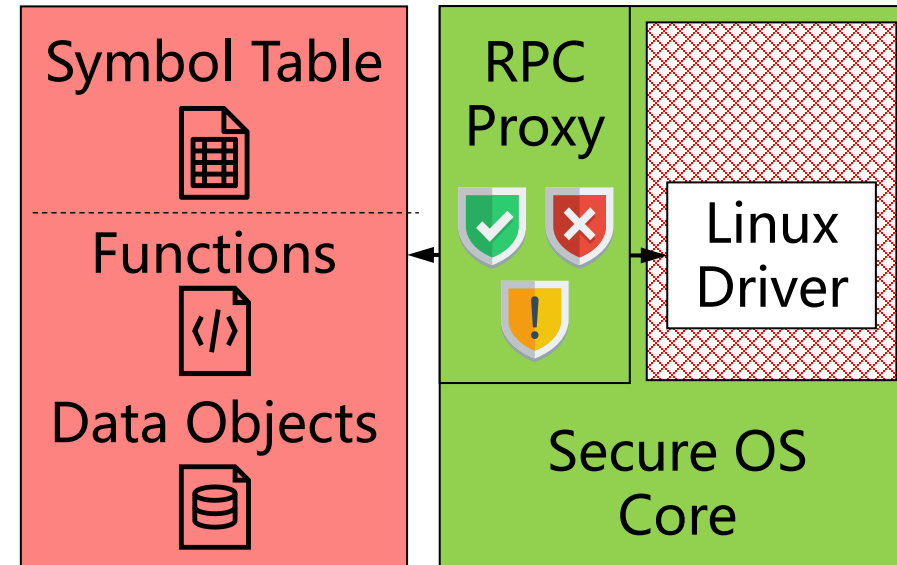
Secure World



Option 1: Linux Kernel Porting

Normal World

Secure World



Option 2: Function Redirection

Problems of Intuitive Design Options

Problems of option 1: Linux Kernel Porting

- Significant engineering efforts
- Massive TCB expansion

Problems of option 2: Function Redirection

- Performance overhead due to frequent world switching
- Security issues

Problems of Intuitive Design Options

Problems of option 1: Linux Kernel Porting

- Significant engineering efforts
- Massive TCB expansion

Problems of option 2: Function Redirection

- Performance overhead due to frequent world switching
- Security issues



Are all driver functions needed for I/O support inside TEE?

Problems of Intuitive Design Options

Problems of option 1: Linux Kernel Porting

- Significant engineering efforts
- Massive TCB expansion

Problems of option 2: Function Redirection

- Performance overhead due to frequent world switching
- Security issues



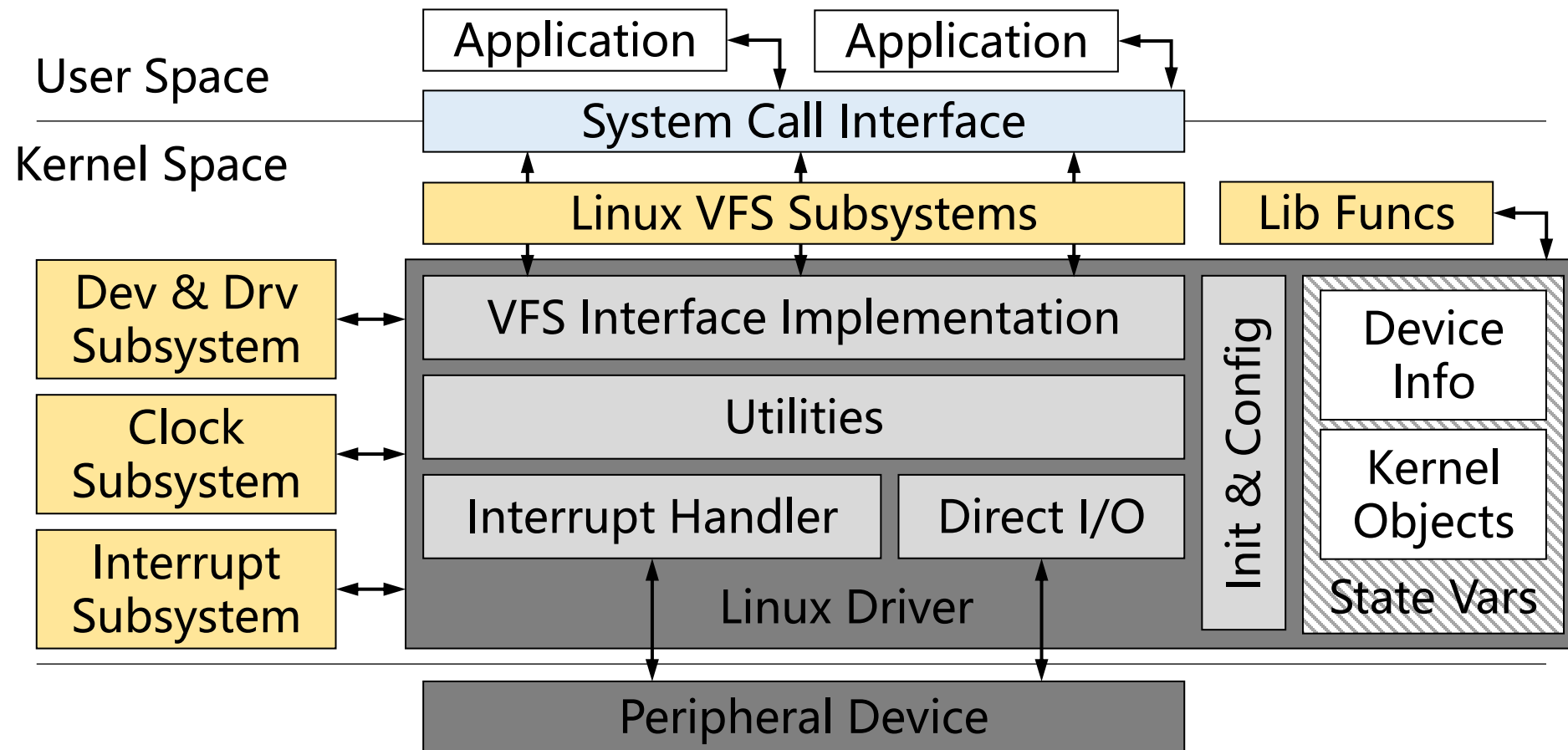
Are all driver functions needed for I/O support inside TEE?



Do dependency functions need to be provided inside secure OS or redirected to NW Linux kernel?

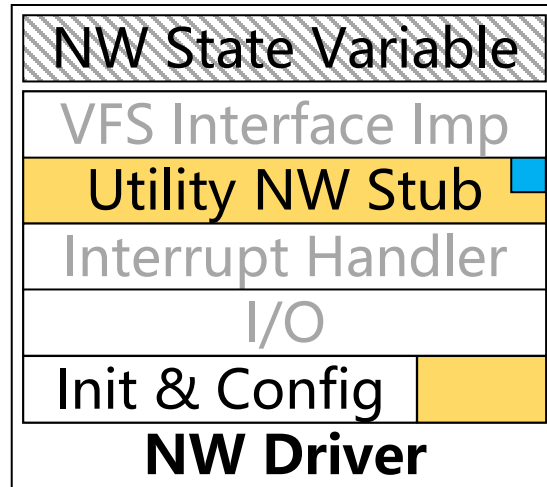
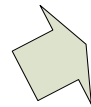
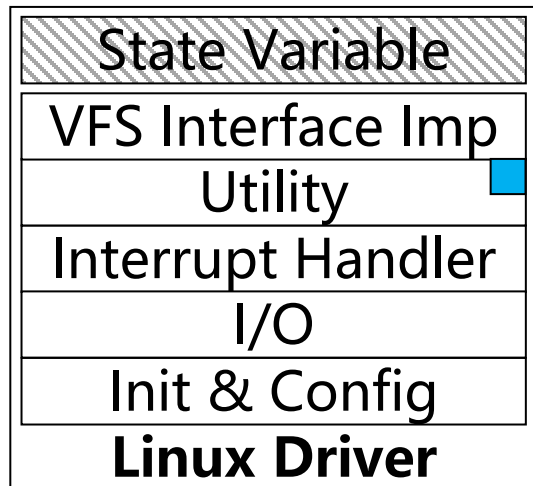
Linux Device Driver Model

- Internally, a Linux driver consists of **state variables** and **driver functions**
- Externally, a driver invokes **library functions** or **Linux kernel subsystem functions**



Our Approach — Twin Drivers

- Exported Function
- LDR Function

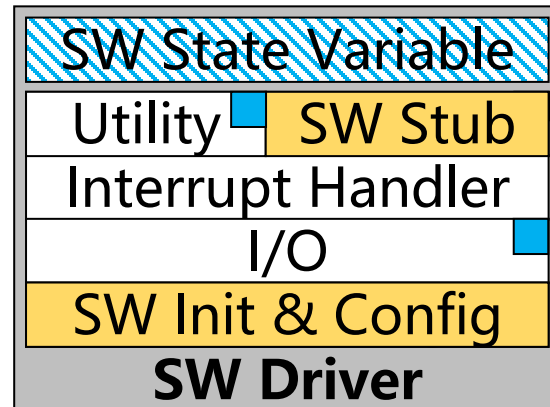
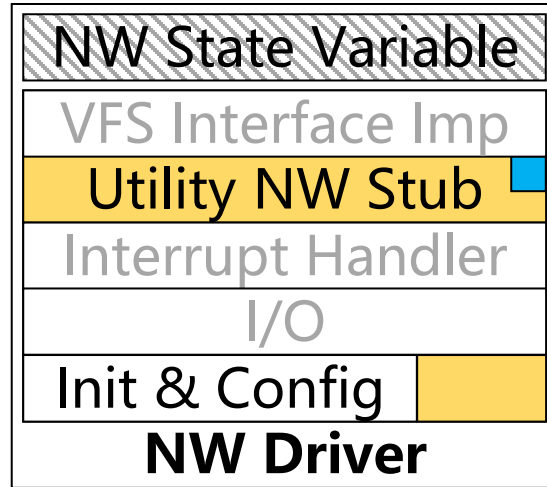
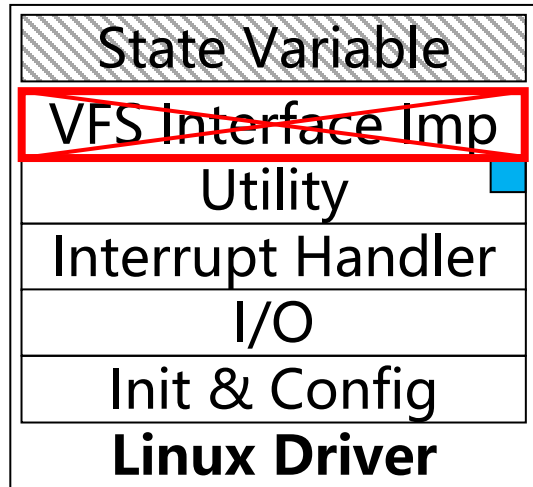


NW Driver

- Init & Conf function is augmented with **state variable transmission code** to transfer initialized NW state variables to SW driver
- Utility function is replaced by **utility NW stub** to retrieve secure peripheral data

Our Approach — Twin Drivers



- Exported Function
- LDR Function

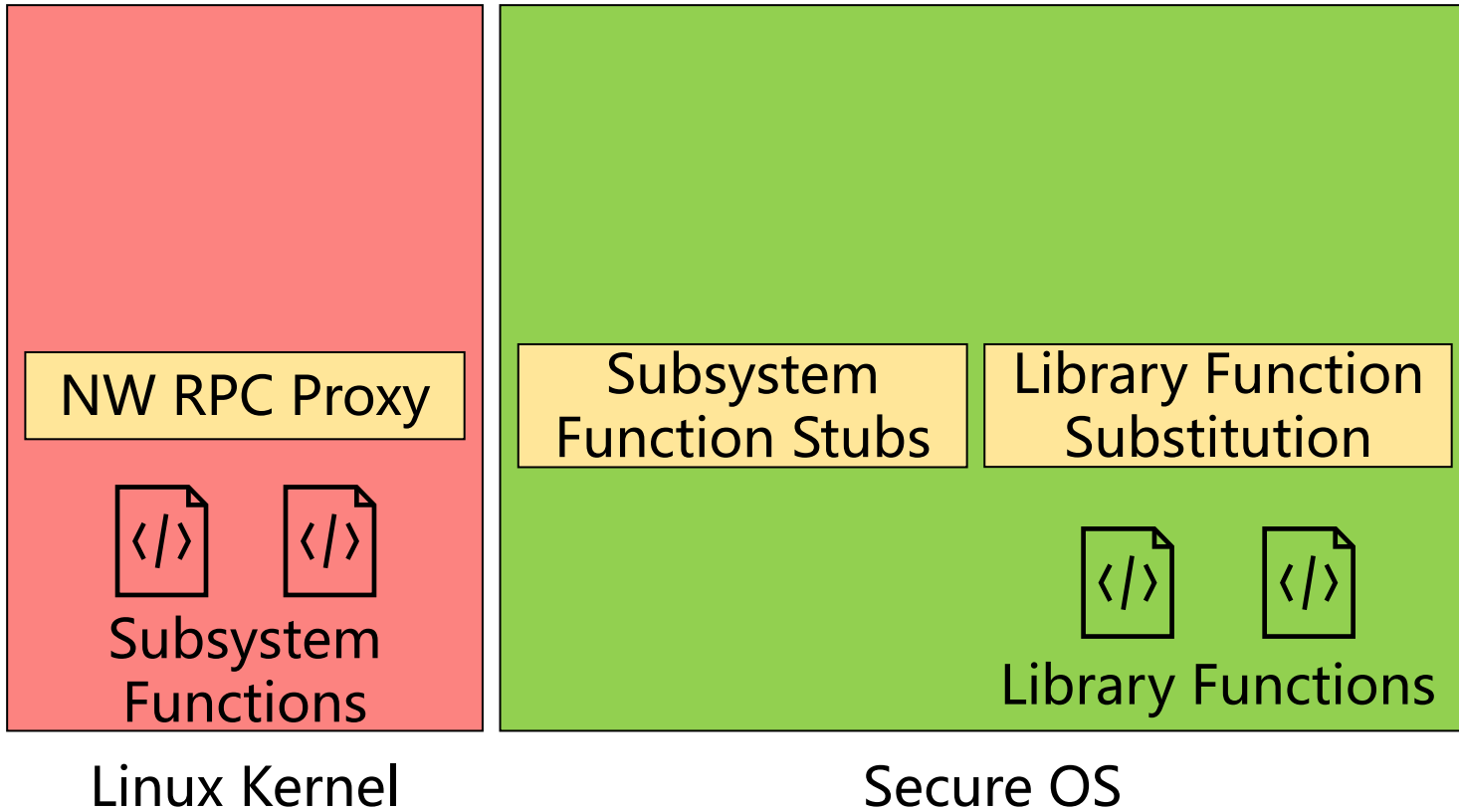


SW Driver

- VFS functions are **removed**
- New Init & Conf** function receives NW state variables from NW world for SW driver Init & Conf
- I/O functions are **exported** for trusted application use
- Utility SW stub** is added to transfer secure peripheral data to NW

Driver Dependency Function Support

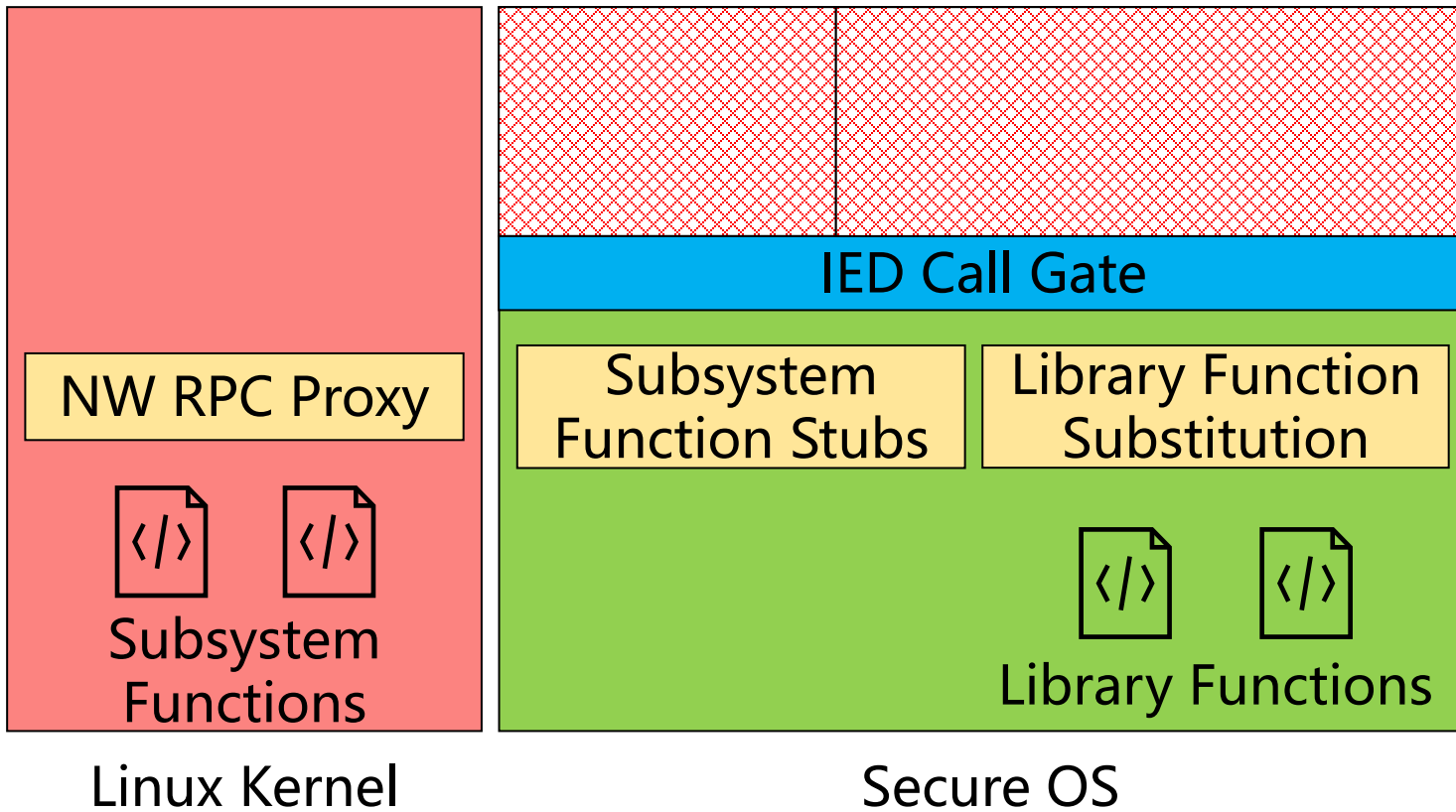
 Trusted  Untrusted



- **Reuse library functions** of secure OS
- **Construct Linux kernel subsystem function stubs** to redirect kernel subsystem function calls back to NW Linux kernel

Isolated Execution Domain (IED)

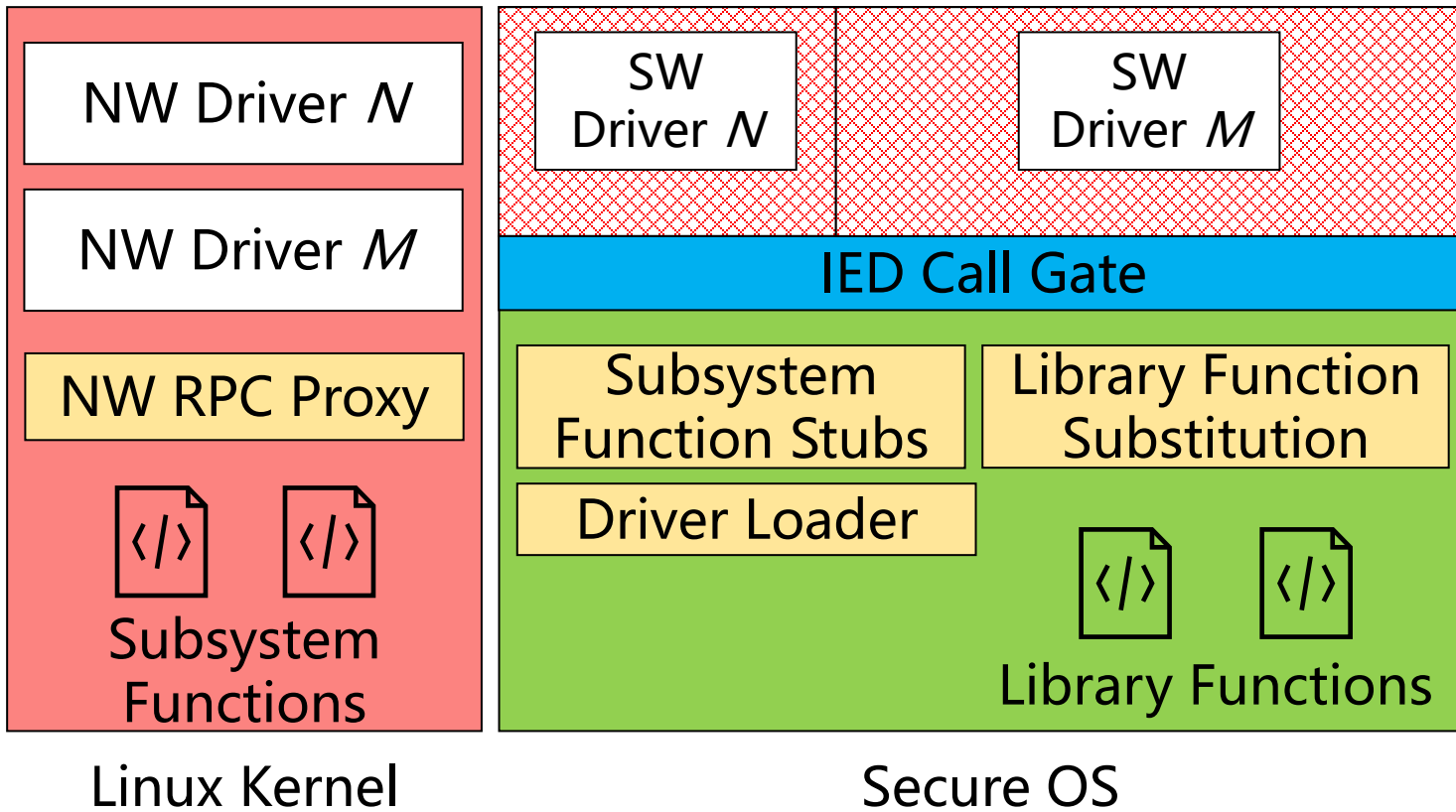
Trusted Untrusted Sandbox



- Build sandboxes by constructing **isolated execution domains (IED)**
- Design **IED call gate** to intercept dependency function invocations issued from SW drivers and conduct security checking

Driver Loading

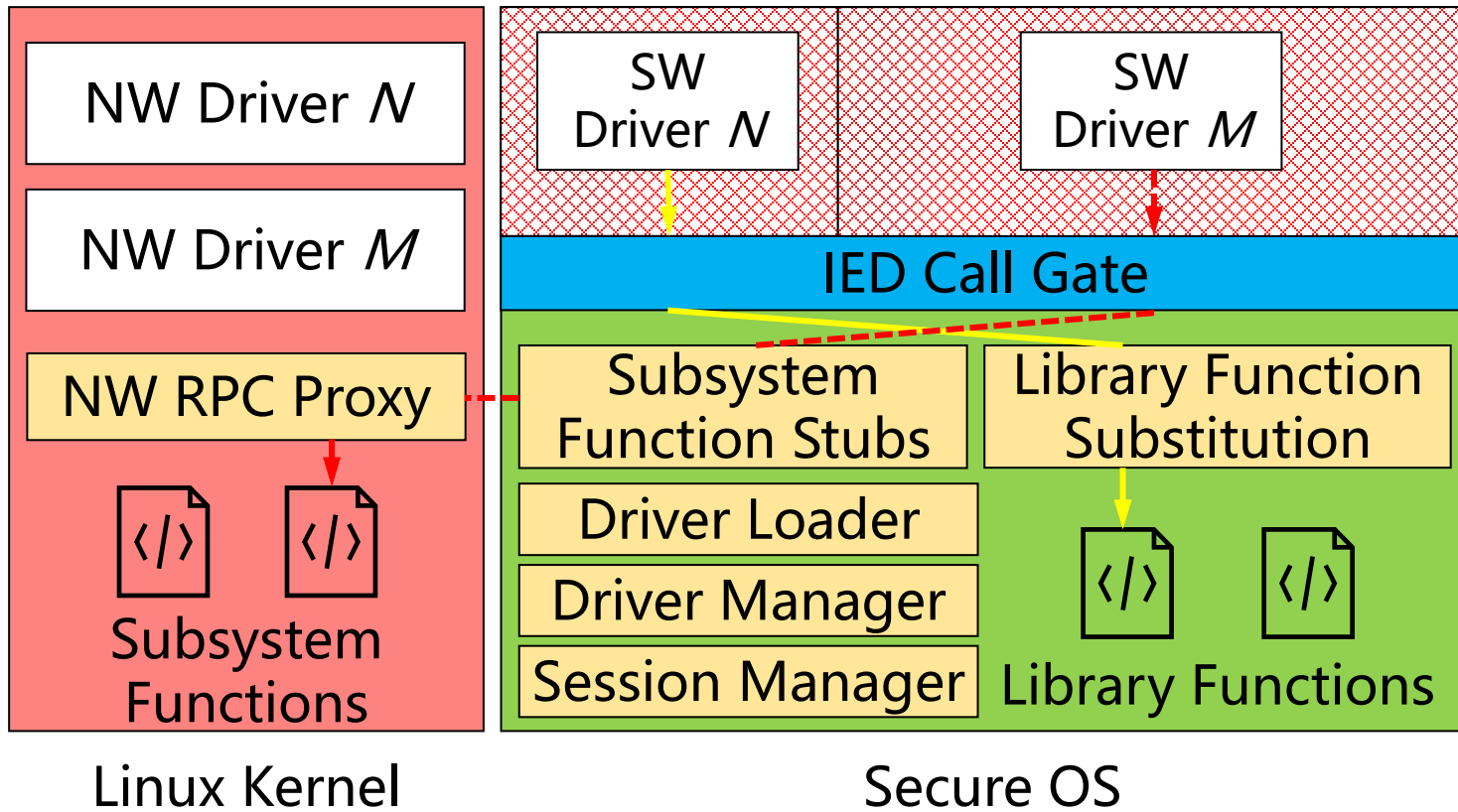
Trusted Untrusted Sandbox



- **Driver loader** loads SW driver into assigned sandbox
- Corresponding NW driver is loaded into Linux kernel

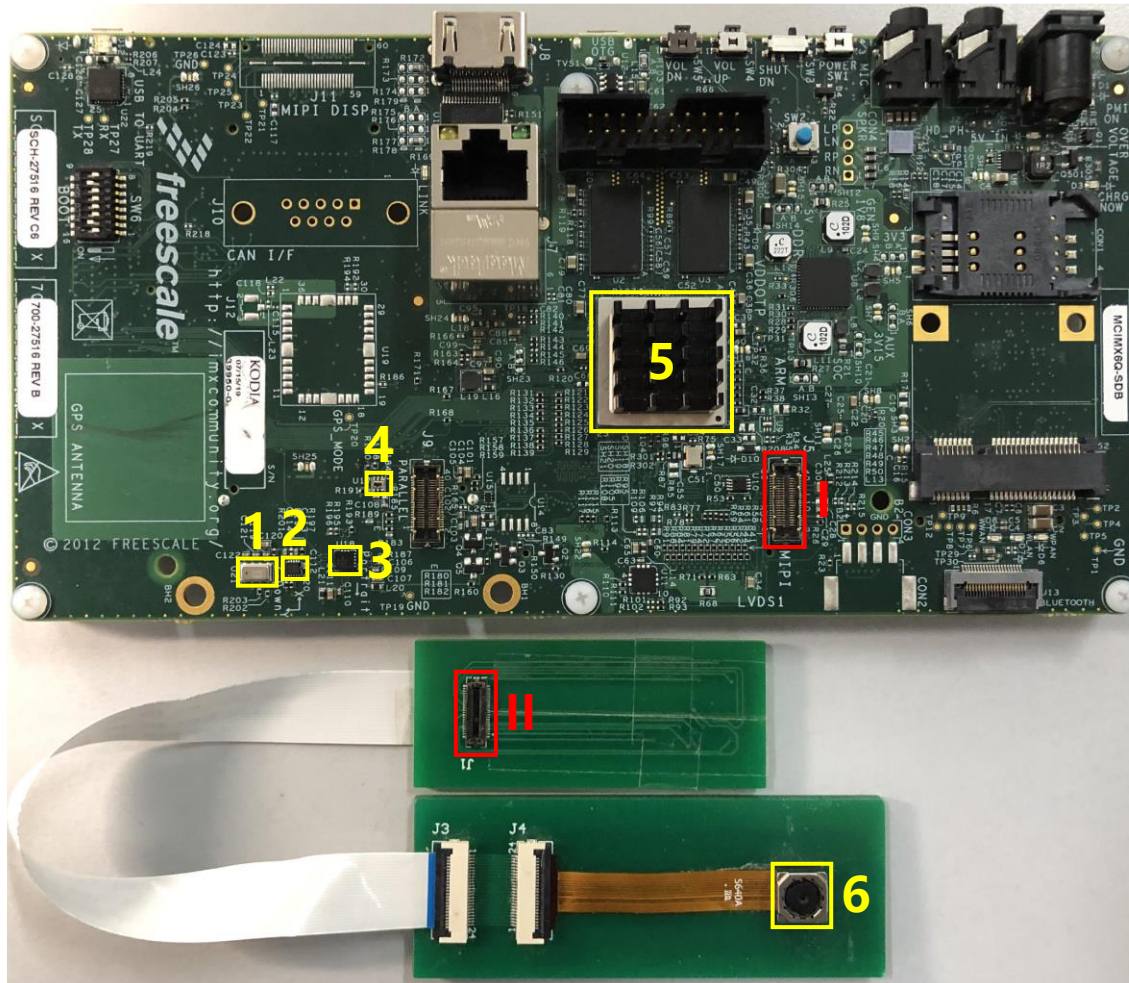
Driver Running

Trusted Untrusted Sandbox



- **Driver manager** with **session manager** coordinates SW driver execution
- Dependency function calls from SW driver go through IED gate and Linux kernel subsystem function calls are redirected to NW Linux kernel

Evaluation Platform



IMX6Q SABRES D

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

1. Barometer
2. Magnetometer
3. Accelerometer
4. Ambient Light Sensor
5. Thermal Sensor/Image Processing Unit
6. Camera Sensor

- I. MIPI CSI Socket
- II. MIPI CSI Plug

Complexity of Secure Driver Generation

SW Driver LoC Statistics

Device	Original Driver	SW Driver LoC				
	LoC	UNT	DEL	CHA	ADD	MOD rate
Image Processing Unit	10,742	10,706	18	18	372	3.80%
Ambient Light Sensor	794	748	4	42	75	15.24%
Magnetometer	537	523	11	3	84	18.25%
Accelerometer	510	498	4	8	64	14.90%
Barometer	325	312	5	8	85	30.15%
Thermal Sensor	779	773	4	2	78	10.78%

UNT: untouched code, **DEL**: deleted code, **CHA**: changed code, **ADD**: newly added code.

MOD rate = $(DEL+CHA+ADD)/(Original\ Driver\ LoC)$.

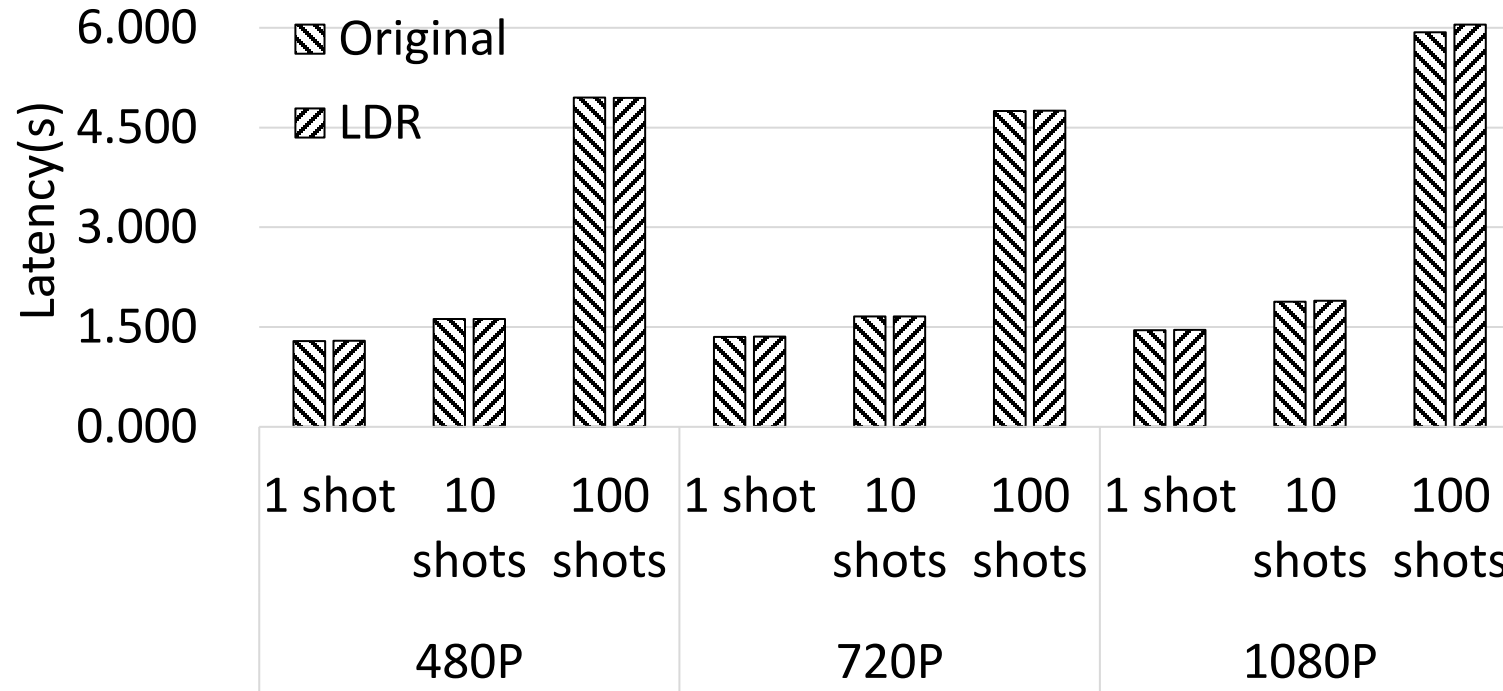
- Amount of engineering efforts for SW driver generation is relatively **small**.

Sensor Sampling Performance

Device	Default Sampling Rate	LDR without IED		LDR with IED	
		Sampling Rate	Difference	Sampling Rate	Difference
Magnetometer	80	83.56	+4.4%	84.82	+6.0%
Accelerometer	800	809.61	+1.2%	800.03	+0.0%
Ambient Light Sensor	11.1	11.47	+3.4%	11.17	+0.6%
Barometer	1	1.09	+8.7%	1.05	+5.3%

- Using SW drivers inside of the SW to read sensor data from secure peripherals **does not degrade sensor performance**

Image Capturing Performance



- For video frame capturing, there is almost **no difference in performance** between SW IPU driver inside of LDR and original Linux driver

Video Streaming Performance

Resolution	Driver	FPS	Duplicated Frames per 100 Frames	Dropped Frames per 100 Frames	Stream Speed	Speed Penalty
480P	Original	25	0	9	0.9999×	-
	LDR	25	0	9	0.9999×	0.00%
720P	Original	24.6	0	19.5	0.9945×	-
	LDR	24	8.65	4.95	0.9703×	-2.43%

- For video streaming, SW IPU driver inside of LDR can push **smooth video streams with low latency**
 - At 480P resolution, LDR has no impact on stream speed
 - At 720P resolution, LDR only reduces stream speed by 2.43%.

Performance Comparison with Driverlets & MyTEE

Sensor Data Sampling Rate Comparison with MyTEE Approaches

Device	Default Sampling Rate	LDR Sampling Rate	MyTEE Sampling Rate	MyTEE vs LDR
Accelerometer	800	800.03	610.19	-23.73%
Magnetometer	80	84.82	82.76	-2.43%
Ambient Light Sensor	11.1	11.17	11.17	0.00%
Barometer	1	1.05	1.10	4.76%

- Our approach outperforms Driverlets since record & replay approach of Driverlets incurs high runtime overhead
- Our approach outperforms MyTEE for high-performance devices such as a sensor device with high sampling rate

Conclusion

- A new **“Twin-Driver” approach** reusing existing NW Linux kernel drivers inside of secure OS
- A novel **Linux Driver Runtime (LDR)** for SW drivers that reuses existing secure OS library functions and redirects Linux kernel subsystem function calls
- A prototype on IMX6Q SABRESD development board and **6** adapted drivers with **low runtime overhead**



安徽工业大学
ANHUI UNIVERSITY OF TECHNOLOGY



Q&A

Thanks!