

Trusted Execution Environment (TEE) on Open-source RISC-V Processor System

Authors: Trong-Thuc Hoang, Ckristian Duran, Akira Tsukamoto,
Kuniyasu Suzuki, and Cong-Kha Pham

Outline

1. Introduction: RISC-V
2. Trusted Execution Environment
3. TEE-Hardware System
4. Results & Conclusion

Outline

1. Introduction: RISC-V
2. Trusted Execution Environment
3. TEE-Hardware System
4. Results & Conclusion

1. Introduction: RISC-V (1/4)

What is RISC-V?

- RISC-V is an open-source Instruction Set Architecture (ISA)
- What is ISA? *i386* and *AMD64* are ISA
- Comparing RISC-V to i386/AMD64 is like comparing Linux to Windows
- Originate from UC Berkeley, but now is maintained by RISC-V Foundation



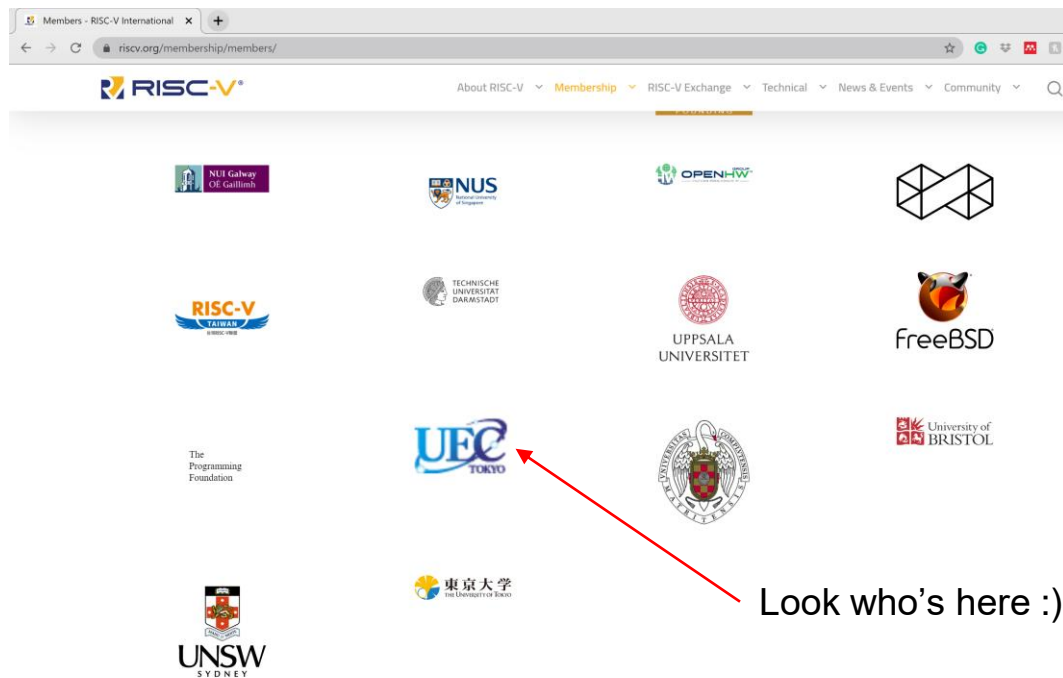
RISC-V Foundation: 235+ Members



1. Introduction: RISC-V (2/4)

What is RISC-V?

- RISC-V is an open-source Instruction Set Architecture (ISA)
- What is ISA? *i386* and *AMD64* are ISA
- Comparing RISC-V to i386/AMD64 is like comparing Linux to Windows
- Originate from UC Berkeley, but now is maintained by RISC-V Foundation



1. Introduction: RISC-V (3/4)

What is RISC-V?

- RISC-V is an open-source Instruction Set Architecture (ISA)
- What is ISA? *i386* and *AMD64* are ISA
- Comparing RISC-V to *i386/AMD64* is like comparing Linux to Windows
- Originate from UC Berkeley, but now is maintained by RISC-V Foundation

What is RISC-V capable of?

- RISC-V supports 32-bit, 64-bit, and 128-bit addressing
- Core ISA is just *Integer*, but it has many extensions: *Multiplication*, *Atomic*, *Floating-point*, *Double floating-point*, and *Compressed*
- When designing your system, you can combine any of those extensions. The most common are *RV64IMAFDC* and *RV32IMAFDC*
- Open ISA → anybody can custom their own CPU → highly customize processor to fit any specific requirement

RISC-V Foundation guarantees the open-ness of the ISA, and also maintains its toolchain (for example, assembler, linker, compiler, etc.)

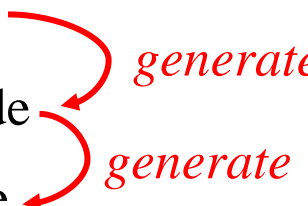
1. Introduction: RISC-V (4/4)

**RISC-V revolutionizes not only the open ISA,
but also the way of hardware coding.**

“*old school*” hardware coding

- RTL (Verilog/VHDL) code

“*RISC-V style*” hardware coding

- Chisel code
 - FIRRTL code
 - Verilog code
- 

To clarify for someone unfamiliar with Chisel-to-Verilog scheme:

1. Chisel coding is not a *programming* language, it is a *description* language
2. The generated Verilog code from Chisel is a TRUE RTL code
3. With proper settings, that Verilog codes after generated can be brought directly to VLSI tools (Synopsys/Cadence) to make a chip

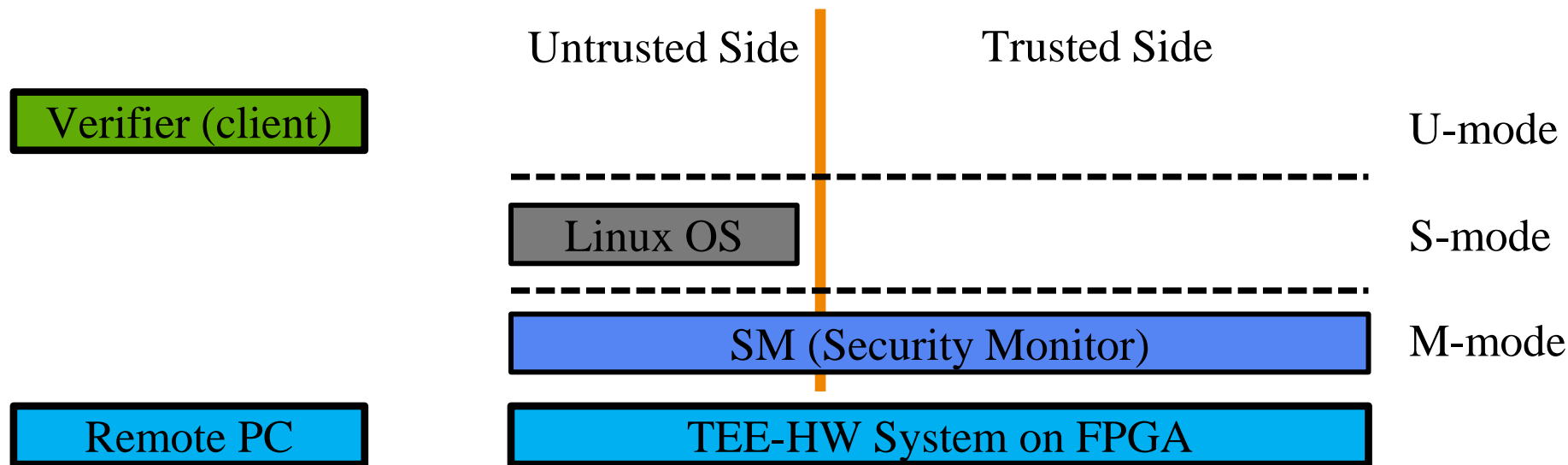
Outline

1. Introduction
- 2. Trusted Execution Environment**
3. TEE-Hardware System
4. Results & Conclusion

2. Trusted Execution Environment (1/5)

TEE in-action (*using Keystone: A TEE Framework*)

Remote PC connects to FPGA via Serial (*UART*) terminal or a TCP connection

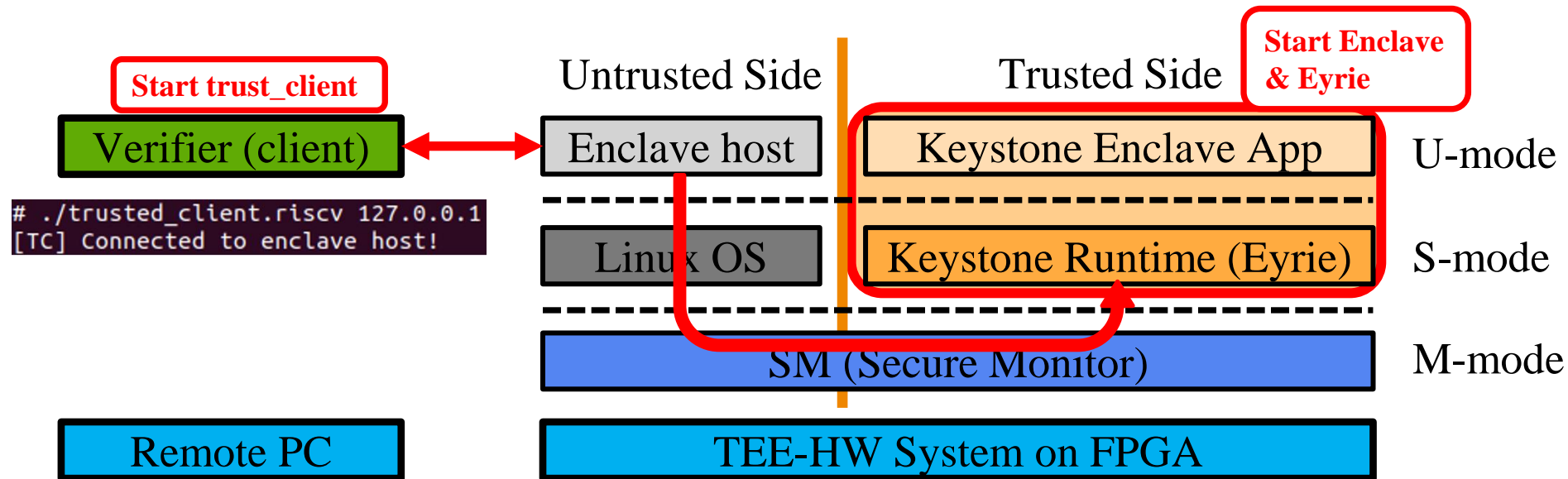


TEE (*Keystone in this case*) creates the Trusted-Side based on the chain-of-trust across multiple operating layers.

It allows client to create and operate an Enclave App in the Trusted Side.

2. Trusted Execution Environment (2/5)

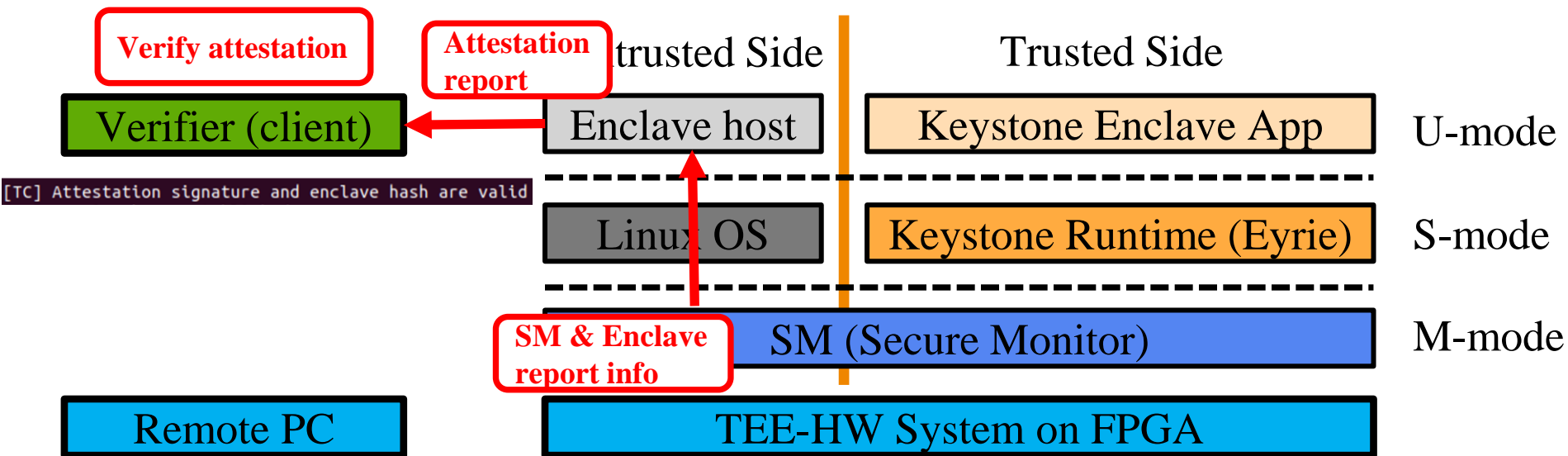
TEE in-action (using Keystone: A TEE Framework)



1. Connection with the Enclave host

2. Trusted Execution Environment (3/5)

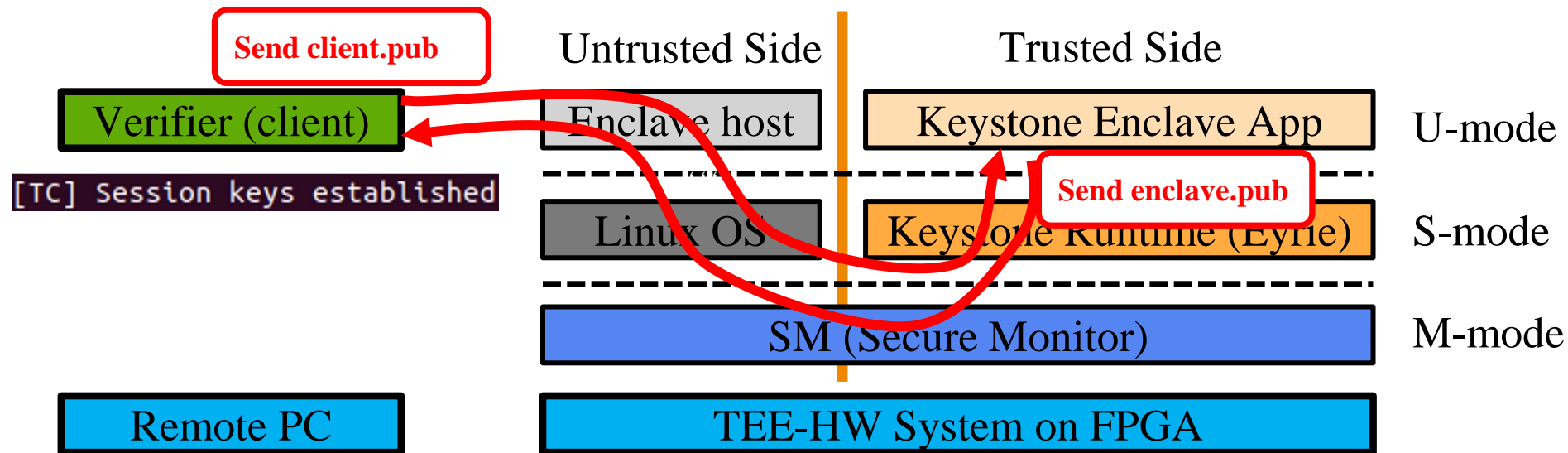
TEE in-action (using Keystone: A TEE Framework)



1. Connection with the Enclave host
2. Verify attestation report

2. Trusted Execution Environment (4/5)

TEE in-action (using Keystone: A TEE Framework)

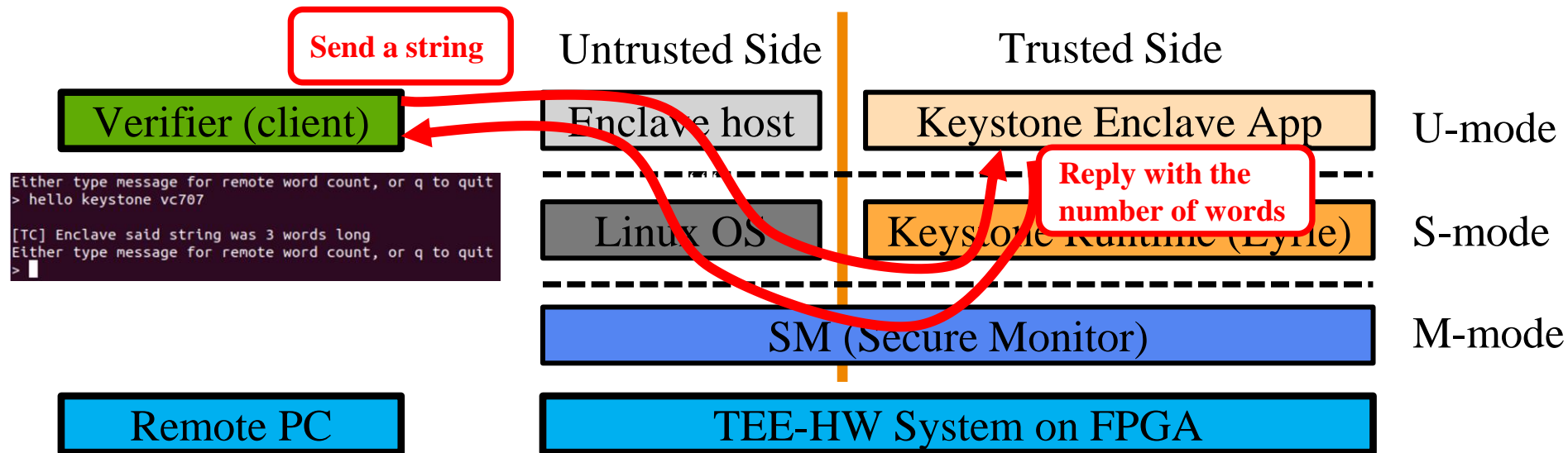


1. Connection with the Enclave host
2. Verify attestation report
3. Exchange communication keys

2. Trusted Execution Environment (5/5)

TEE in-action (using Keystone: A TEE Framework)

Keystone demo: (1) client sends strings, then (2) request calculation from the Enclave, finally (3) the Enclave replies with the number of words



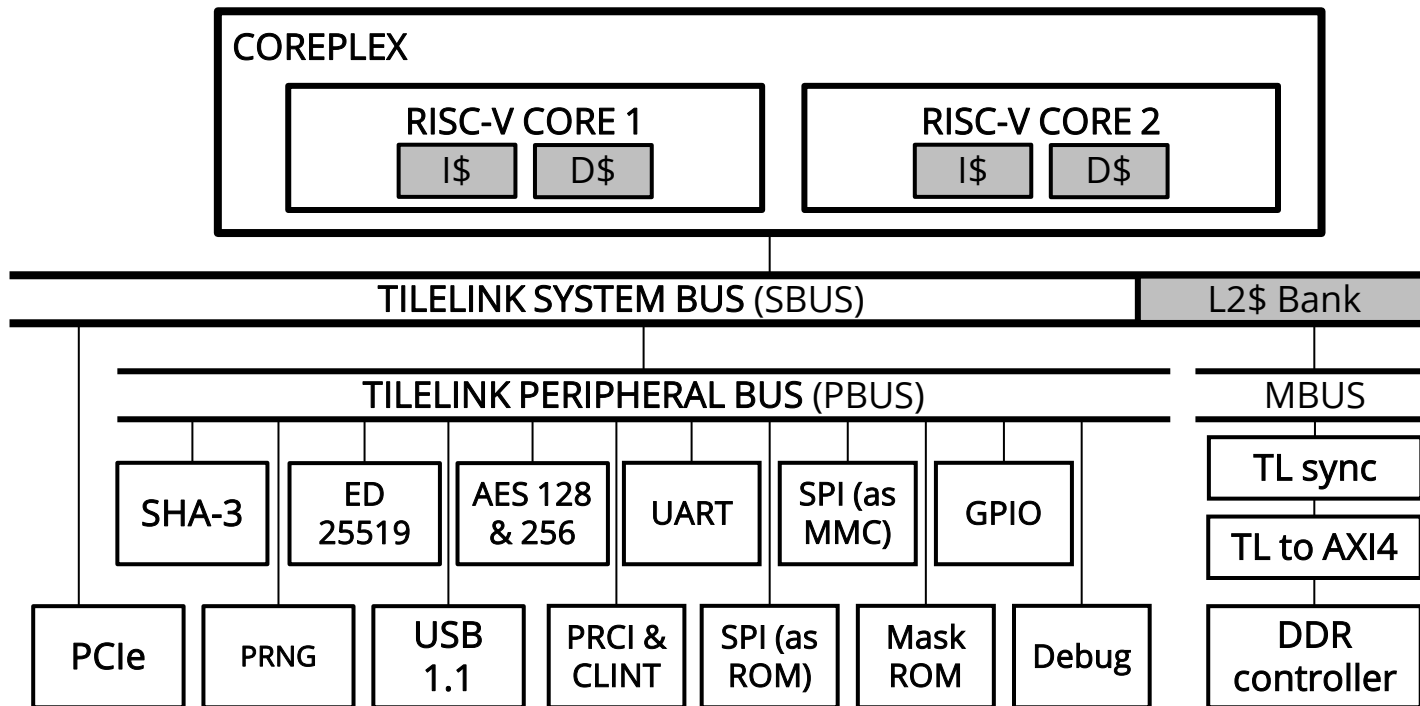
1. Connection with the Enclave host
2. Verify attestation report
3. Exchange communication keys
4. Client's app runs on the established TEE

Outline

1. Introduction
2. Trusted Execution Environment
- 3. TEE-Hardware System**
4. Results & Conclusion

3. TEE-Hardware System (1/6)

System Architecture:



- Not fixed at dual-core, can increase/decrease the number of cores as you wanted
- Available cores: **Rocket-chip** (in-of-order core) and **BOOM** (out-of-order core)
- Some hardware modules can be easily included/excluded to/from the system

3. TEE-Hardware System (2/6)

Variable	Available option	Description
BOARD	<ul style="list-style-type: none"> - VC707 - DE4 - TR4 	Select the FPGA board
ISACONF	<ul style="list-style-type: none"> - RV64GC - RV64IMAC - RV32GC - RV32IMAC 	Select the ISA
MBUS	<ul style="list-style-type: none"> - MBus64 - MBus32 	Select the bit-width for the memory bus
BOOTSRC	<ul style="list-style-type: none"> - BOOTROM - QSPI 	Select the boot source
PCIE	<ul style="list-style-type: none"> - WPCIE - WoPCIE 	<ul style="list-style-type: none"> - Include PCIe module in the system - Remove PCIe module from the system
DDRCLK	<ul style="list-style-type: none"> - WSepaDDRClk - WoSepaDDRClk 	<ul style="list-style-type: none"> - Separate DDR-clock with System-clock - Not separate DDR-clock with System-clock
HYBRID	<ul style="list-style-type: none"> - Rocket - Boom - RocketBoom - BoomRocket 	<ul style="list-style-type: none"> - Two Rocket cores - Two Boom cores - Rocket core 1st, Boom core 2nd - Boom core 1st, Rocket core 2nd

In the Makefile system, these variables are available.

Example usage:

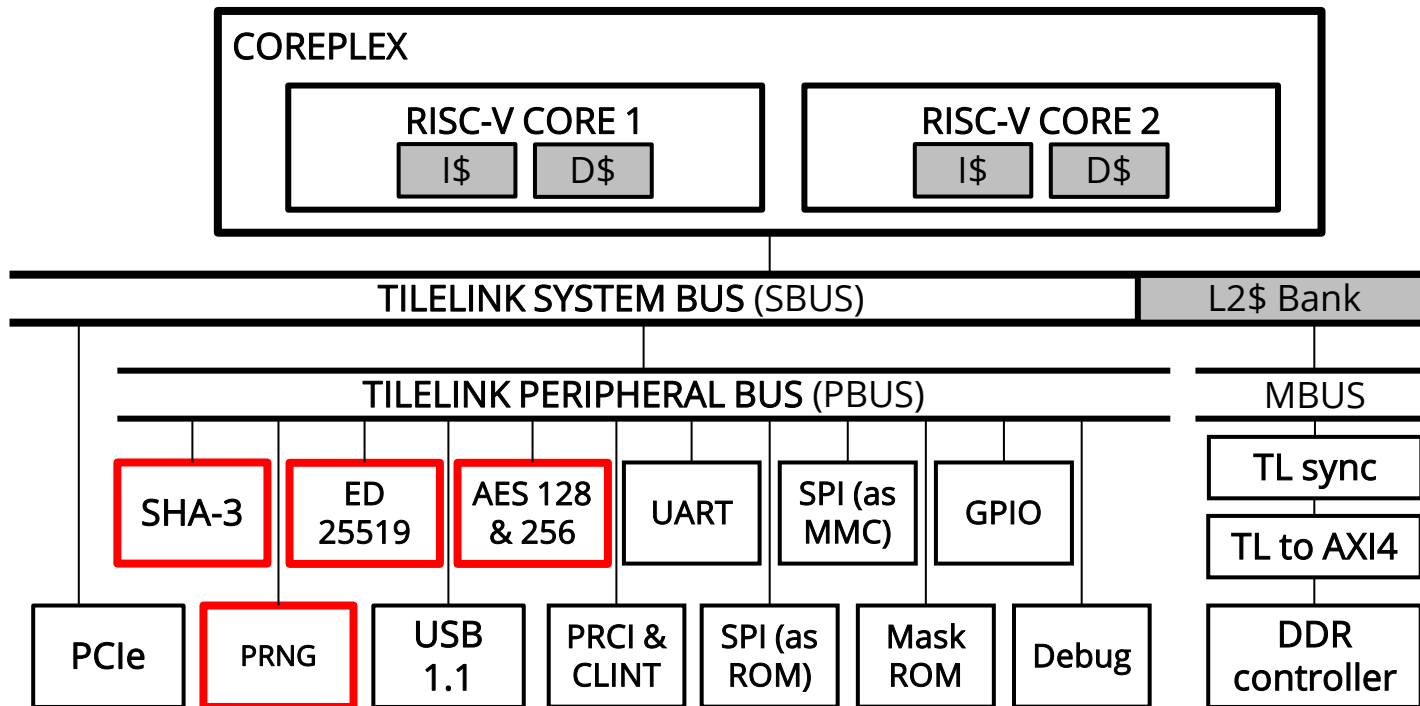
```
BOARD=VC707
ISACONF=RV64GC
MBUS=MBus64
BOOTSRC=BOOTROM
PCIE=WoPCIE
DDRCLK=WoSepaDDRClk
HYBRID=Rocket
```


3. TEE-Hardware System (3/6)

Summary table of FPGA logic utilization (*on VC707*) with various core configurations:

ISACONF	HYBRID		FPGA logic utilization (LUT) (<i>on VC707</i>)	
	Core0	Core1		
RV64GC	Boom	Boom	160,873	52.99%
	Rocket	Rocket	96,571	31.81%
	Boom	Rocket	128,708	42.39%
	Rocket	Boom	128,719	42.40%
RV64GC	Rocket	Rocket	96,571	31.81%
RV64IMAC			72,007	23.72%
RV32GC			89,356	29.43%
RV32IMAC			65,899	21.71%

3. TEE-Hardware System (4/6)



- Crypto-cores:**
- SHA-3 512
 - Ed25519 (*genkey and certification*)
 - AES-128/256
 - PRNG (*Pseudo-random generator*)

3. TEE-Hardware System (5/6)

Crypto-cores on Stratix-IV FPGA

	SHA-3	AES-128/256	Ed25519	
			Mult	Sign
ALUT	8,108	3,195	2,737	3,969
Registers	2,790	2,854	4,778	4,617
Fmax (MHz)	100	100	100	100
Memory	0	0	8KB	0
DSP block	0	0	48	0

3. TEE-Hardware System (6/6)

Crypto-cores in ASIC (*ROHM-180nm*)

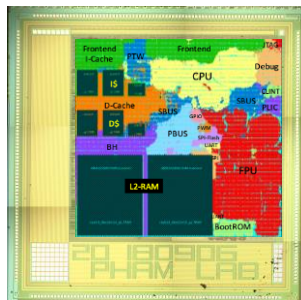
	SHA-3	AES-128/256	Ed25519	
			Mult	Sign
Size	1,150 μm \times 1,150 μm	808.96 μm \times 806.4 μm	1,694.72 μm \times 1,693.44 μm	1,346.56 μm \times 1,345.68 μm
Gate-count (NAND)	102,500	50,560	222,432	140,442
Fmax (MHz)	104	90	106	91
Power (mW)	42.745	37.566	53.061	80.894

Outline

1. Introduction
2. Trusted Execution Environment
3. TEE-Hardware System
4. Results & Conclusion

6. Results & Conclusion (1/3)

ROHM180 5.0×5.0mm



RV64GC
Rocket-
chip(×1)

64-bit MCU

2019

Sep.

Aug



RV64GC
Rocket-
chip(×4)
*Add:*Linux-
compatible
ROHM180
5.0×7.0mm

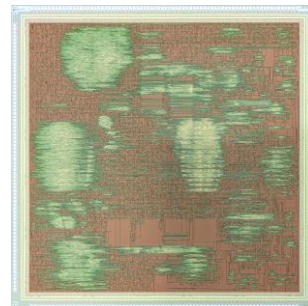
TEE-HW

Oct

2020

Jan.

ROHM180 5.0×5.0mm

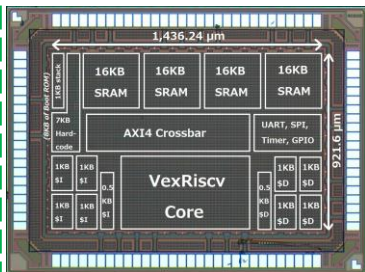


RV64GC
BOOM(×1)
Add:
BOOM
core

TEE-HW

Jun.

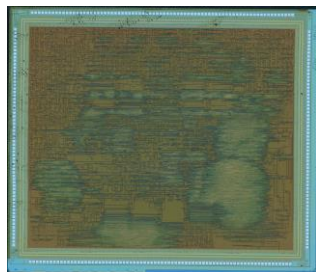
32-bit MCU



RV32IM
VexRiscv(×1)

SOTB65 2.0×1.5mm

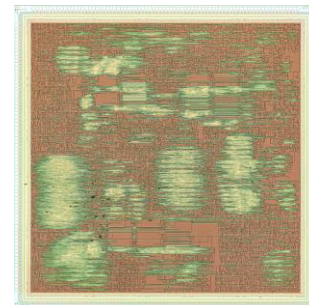
TEE-HW



ROHM180 5.0×5.0mm

RV64GC
Rocket-
chip(×2)
Add:
Crypto
cores

TEE-HW



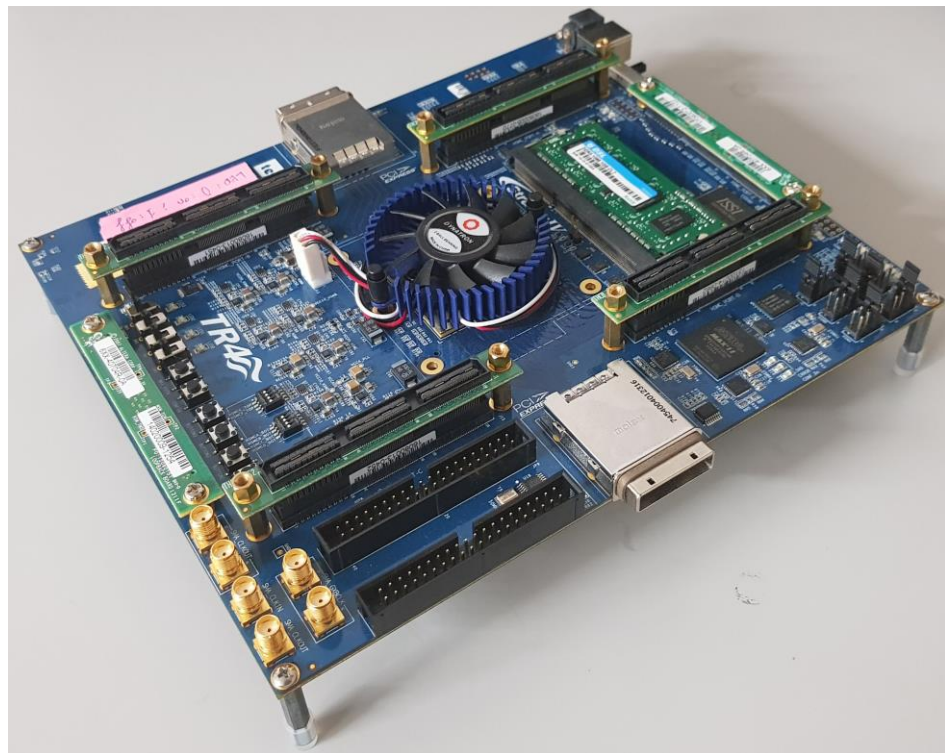
ROHM180 5.0×5.0mm

RV32IMAC
Rocket-
chip(×1)+
BOOM(×1)
Add: 32-bit
system

6. Results & Conclusion (2/3)

Solving the DDR problem (*for Linux-boot*) for the chip by:

1. Using the DIMM RAM in the TR4
2. Having the PCB (*with socket-chip*) mounted on the TR4



6. Results & Conclusion (3/3)

- We presented a system platform for Trusted Execution Environment (TEE) featuring crypto-cores accelerators.
- Completed TEE-Hardware system was developed with various configurations to fit specific needs;
such as core options, hybrid options, ISA options, etc.
- The system was implemented and tested on various FPGAs (*VC707, DE4, TR4*) and ASIC (*ROHM-180nm*).
- The execution time of the TEE with hardware accelerators dropped significantly compared to software.



国立大学法人
電気通信大学



THANK YOU FOR YOUR LISTENING

Acknowledgement

This work is based on results obtained from a project commissioned by the New Energy and Industrial Technology Development Organization (NEDO).

The chip tape-out is supported by VLSI Design and Education Center (VDEC), the University of Tokyo in collaboration with Synopsys, Inc., Cadence Design Systems, Inc., Renesas Electronics Corp., and Nippon Systemware Co., Ltd.