Linux
Plumbers
Conference 2022

>> Dublin, Ireland / September 12-14, 2022

CXL Confidential Computing

Jérôme Glisse / Google

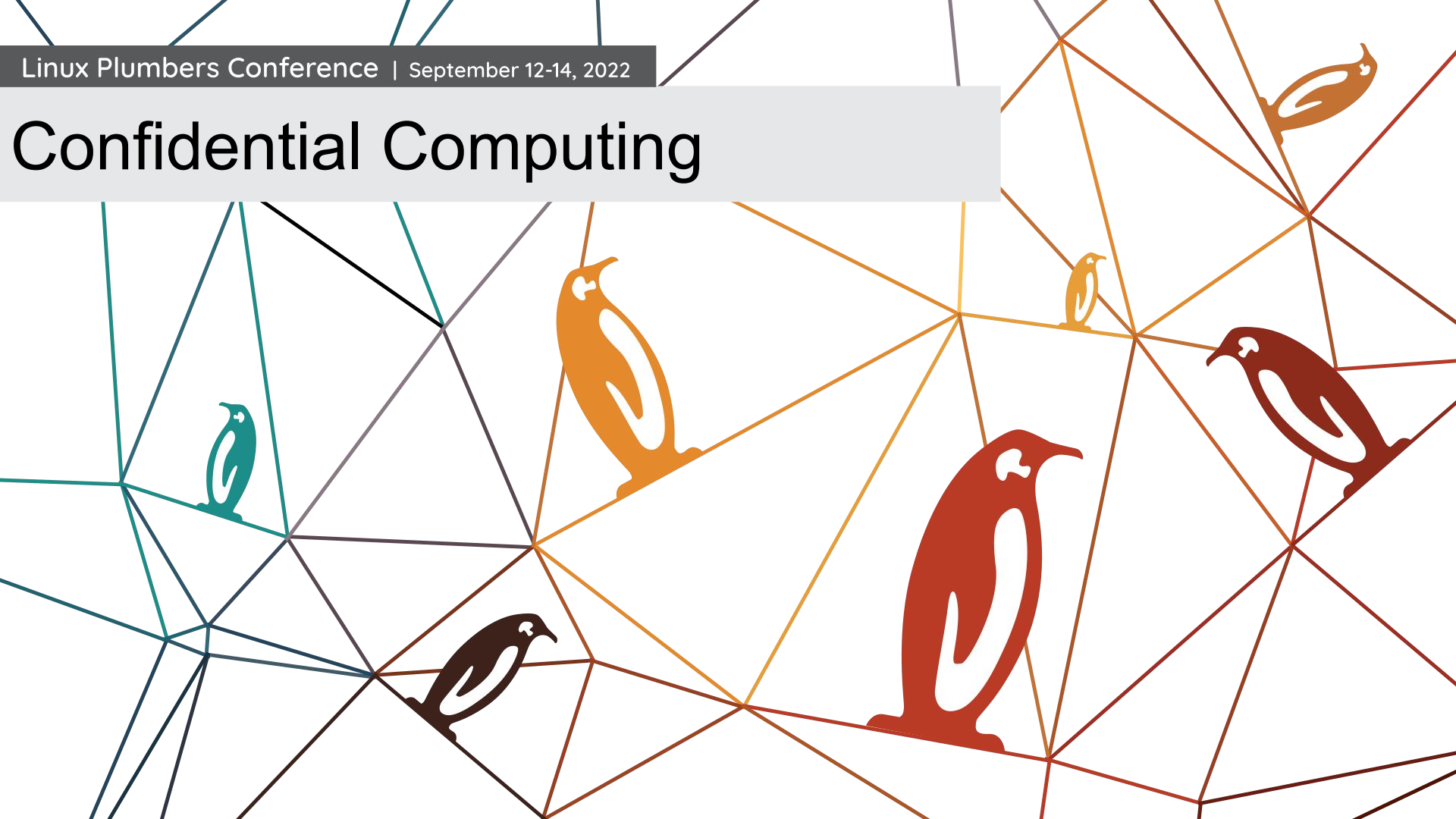
Linux Plumbers Conference 2022

>> Dublin, Ireland / September 12-14, 2022

Agenda

- Confidential Computing
- Device & Confidential Computing
- CXL & Confidential Computing

Confidential Computing

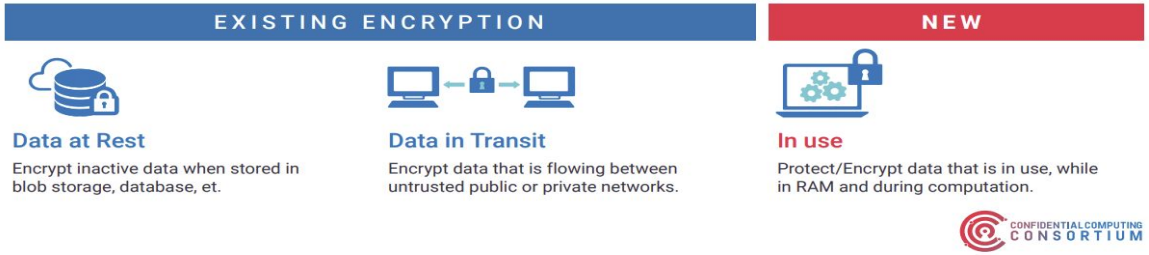


Confidential Computing

Confidential Computing protects data in use by performing computation in a hardware-based **Trusted Execution Environment (TEE)**.

These secure and isolated environments prevent unauthorized access or modification of applications and data while they are in use, thereby increasing the security level of organizations that manage sensitive and regulated data.

[From Confidential Computing Consortium]



TEE

Trusted Execution Environment (TEE) provides

- **Attestability**⇒evidence & measurements of TEE origin and current states
- **Data integrity**⇒unauthorized entities cannot alter data in a TEE
- **Data confidentiality**⇒unauthorized entities cannot view data while in a TEE
- **Code integrity**⇒unauthorized entities cannot replace or modify code in a TEE

A hardware-based TEE uses hardware-backed techniques

Threats

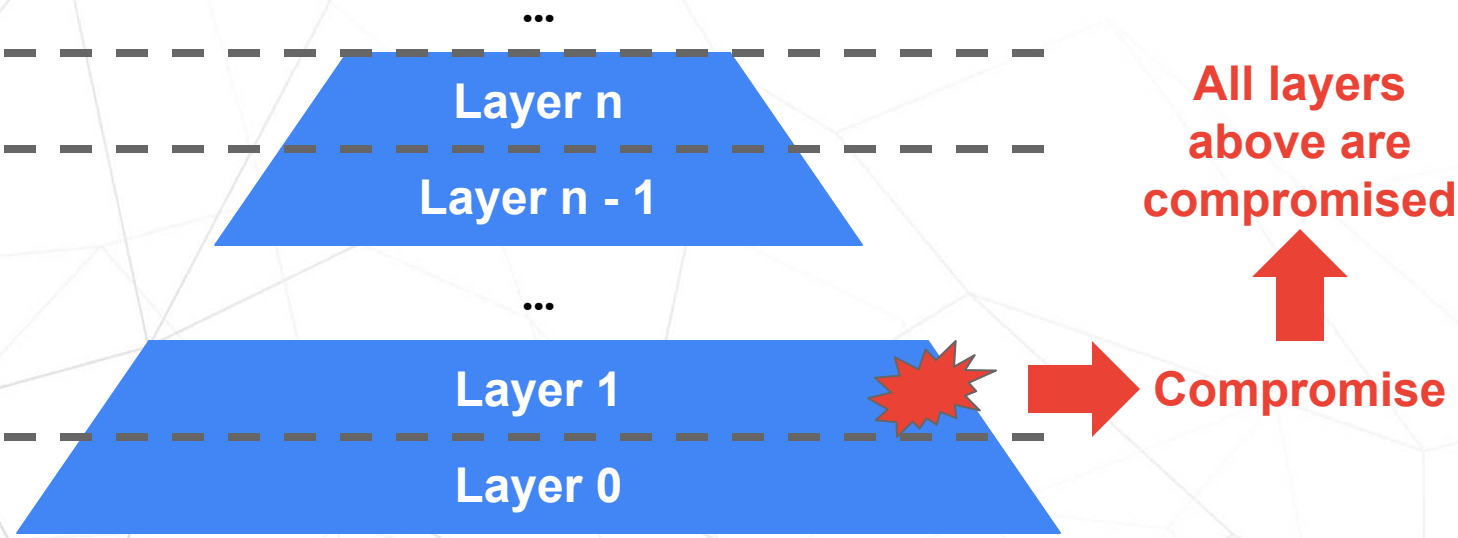
Unauthorized entities (from one application point of view)

- Other applications on the host/device
- The host operating system and hypervisor
- System administrators
- Service providers and the infrastructure owner (Cloud provider)
- Anyone else with physical access to the hardware

Optional features

- **Code Confidentiality:** unauthorized entities cannot view code
- **Authenticated Launch:** unauthorized application cannot execute
- **Recoverability: recover from potentially-compromised state**
  Recoverability generally requires that some component(s) of the TEE remain trusted

Pyramid of Trust



Hardware: lowest layer

Security is only as strong as the layers below it

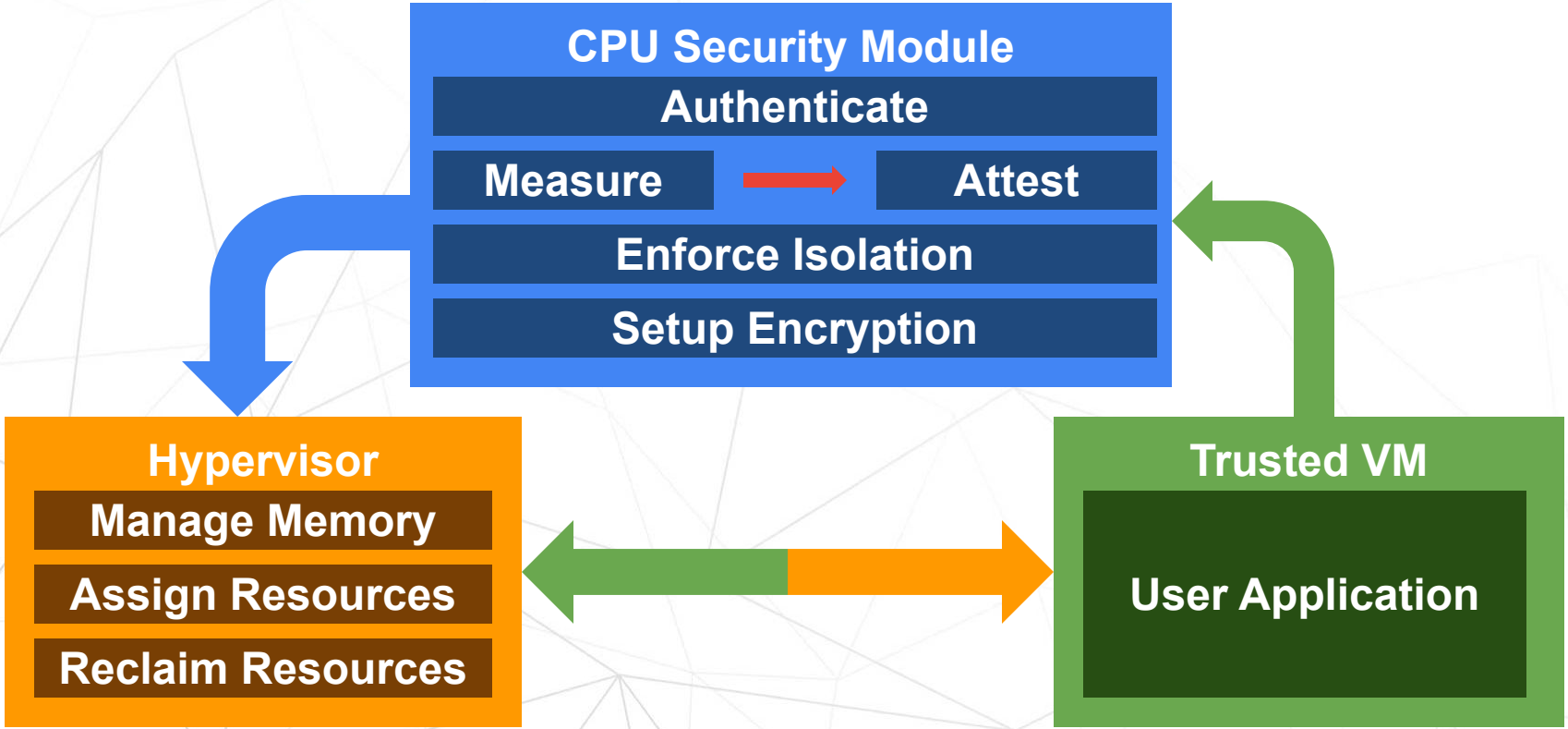
⇒Security in any layer potentially circumvented by a breach at an underlying layer
⇒Security solutions at the lowest layers possible
⇒Down to the silicon components of the hardware

Bonus

⇒Remove the operating system and device driver from Trust Base
⇒Remove service providers and their admins Trust Base

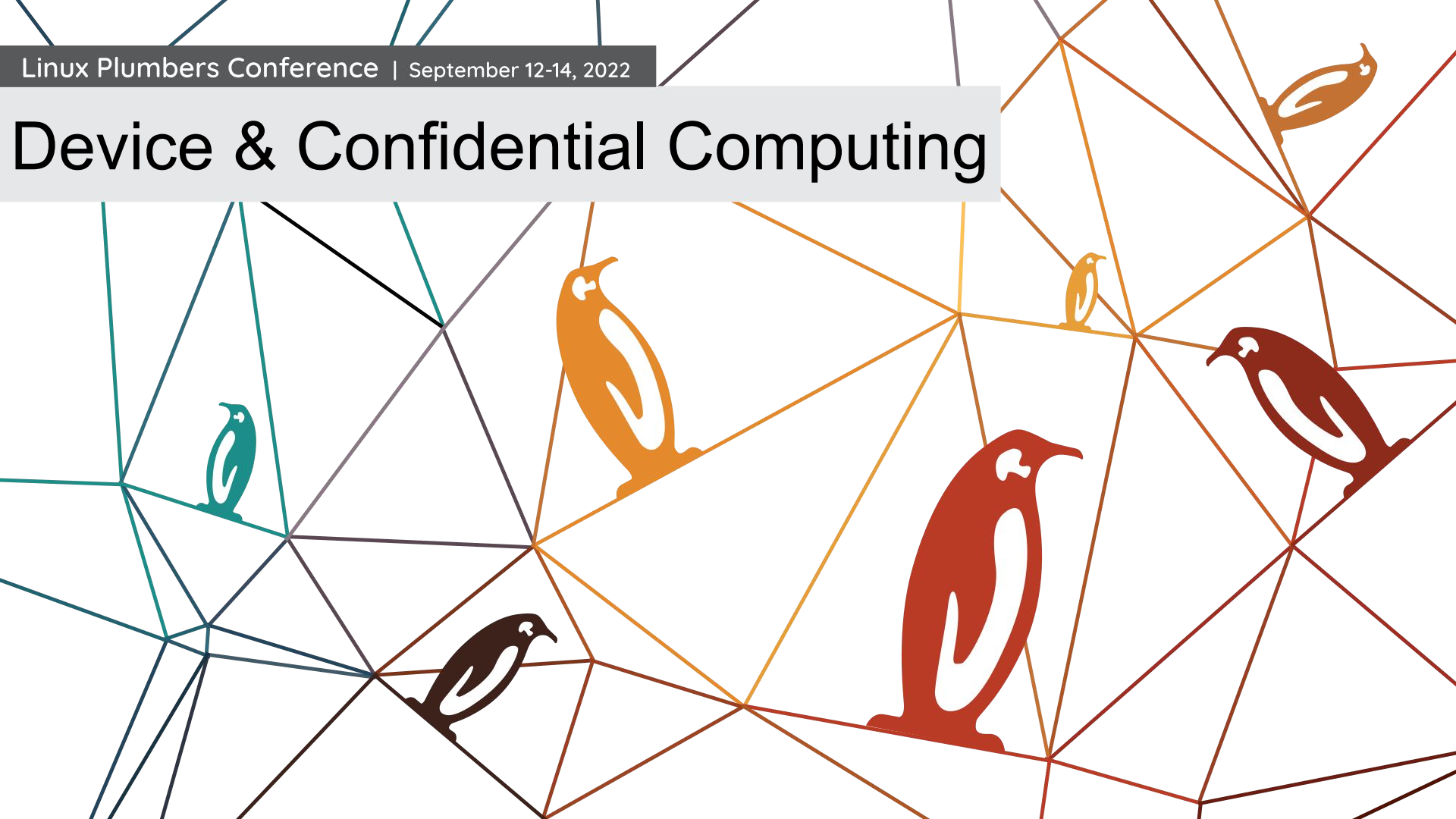
Thereby reducing exposure to potential compromise

Architecture



Linux Plumbers Conference | September 12-14, 2022

Device & Confidential Computing



TEE in a Device

Same requirements for a device than on a CPU

Trusted Execution Environment (TEE) provides

- **Attestability**⇒evidence & measurements of TEE origin and current states
- **Data integrity**⇒unauthorized entities cannot alter data in a TEE
- **Data confidentiality**⇒unauthorized entities cannot view data while in a TEE
- **Code integrity**⇒unauthorized entities cannot replace or modify code in a TEE

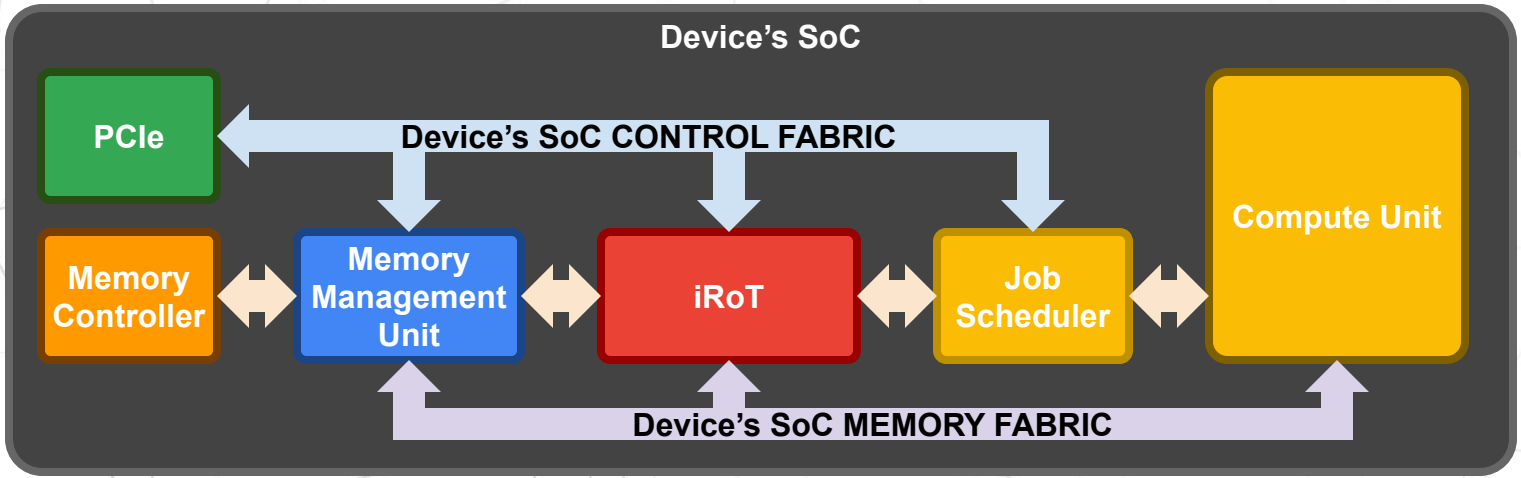
A hardware-based TEE uses hardware-backed techniques

Typical Device

Trust can not cover all of the HW Block or software within a device:

- Limit the number of Block that can access user data in plain text
- Trust limited to iRoT & Access Control Block (often the MMU)

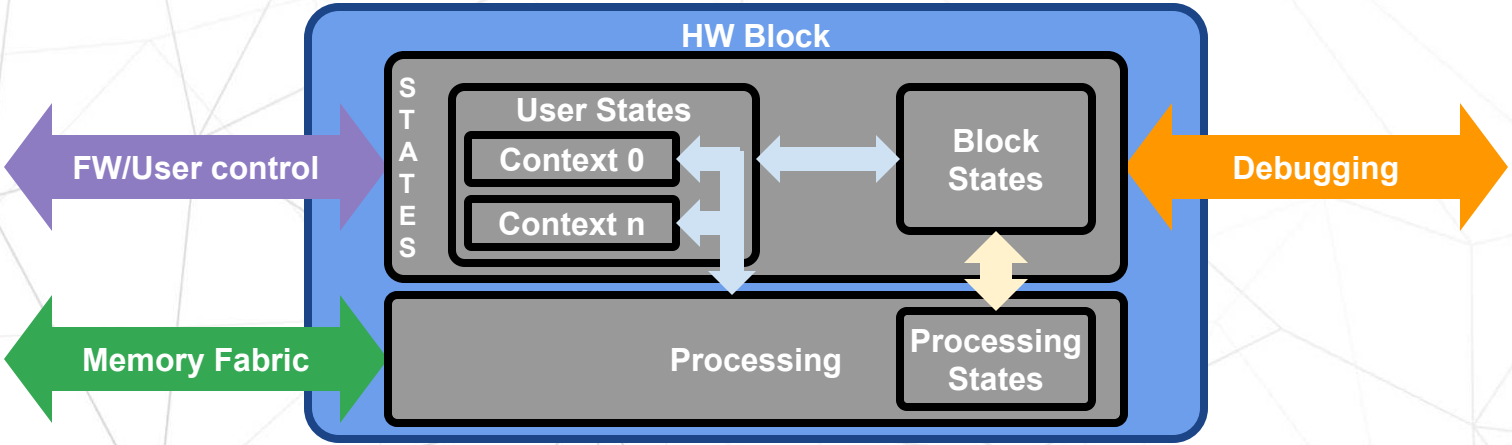
One HW Block can be used by an attacker to attack other Block



Hardware Block

For each Hardware Block we need clear boundaries:

- How to program the Block: What are the external control (registers) ?
- How the Block interact with other Block: What is its dependencies ?
- Performance monitoring: What are the relevant metrics ?
- Debugging: What HW states do we need to expose ?

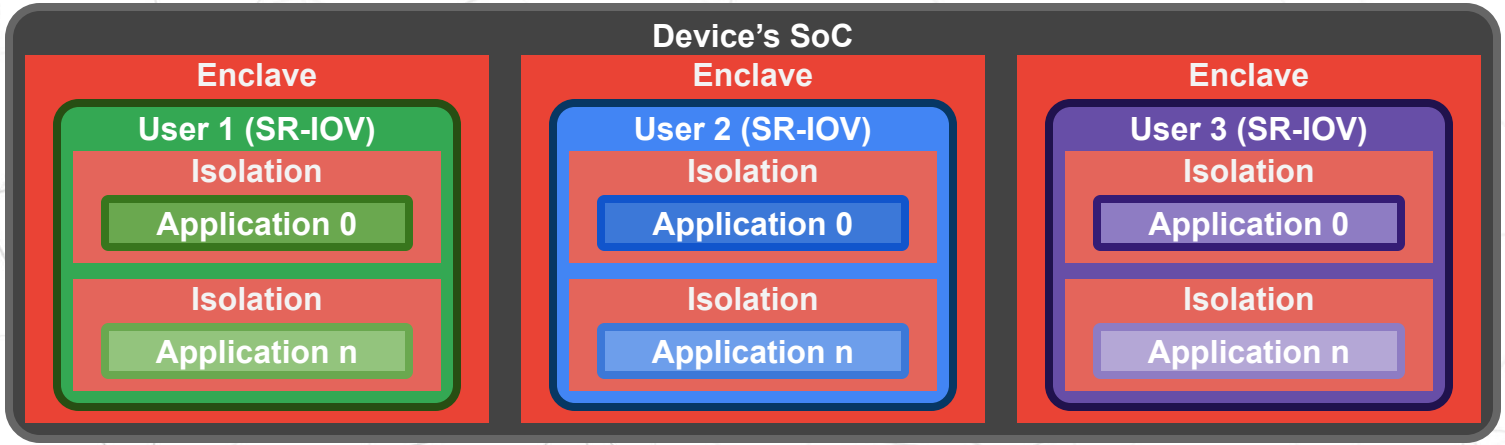


Device Enclave

Accelerator divided:

- Across user⇒Assign Accelerator Chunk (PCIe VF) to Virtual Machine (VM)
- Within a user⇒Multiple process within a Virtual Machine (VM) of one user

Isolation apply across users but also between process of one user



Device TEE Checklist

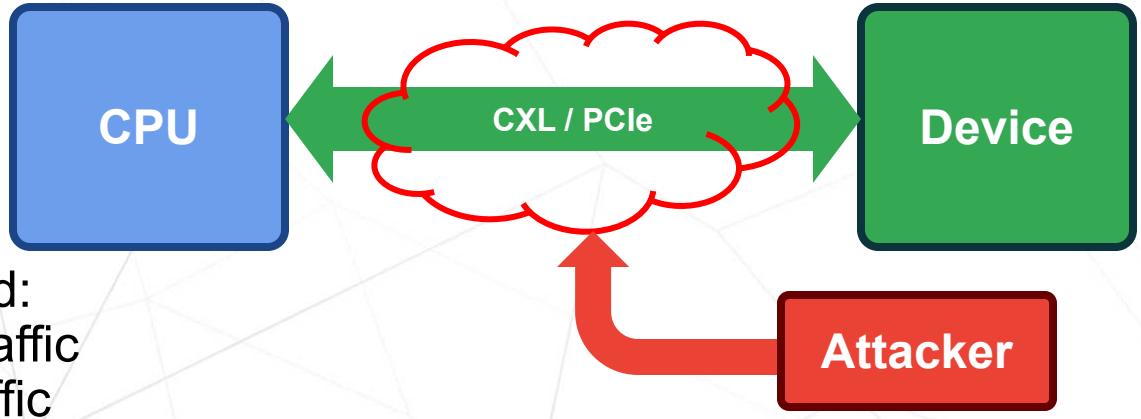
Trusted Execution Environment (TEE) provides

- device have a Root of Trust for Birth Certificate & Measurements
  ⇒**Attestability**
- device isolate device execution context from one another
  ⇒**Data integrity & confidentiality**
  ⇒**Code integrity**

This is for protection within the device

Need protection from CPU to & from device (and also device to device i.e P2P)

Protecting Traffic

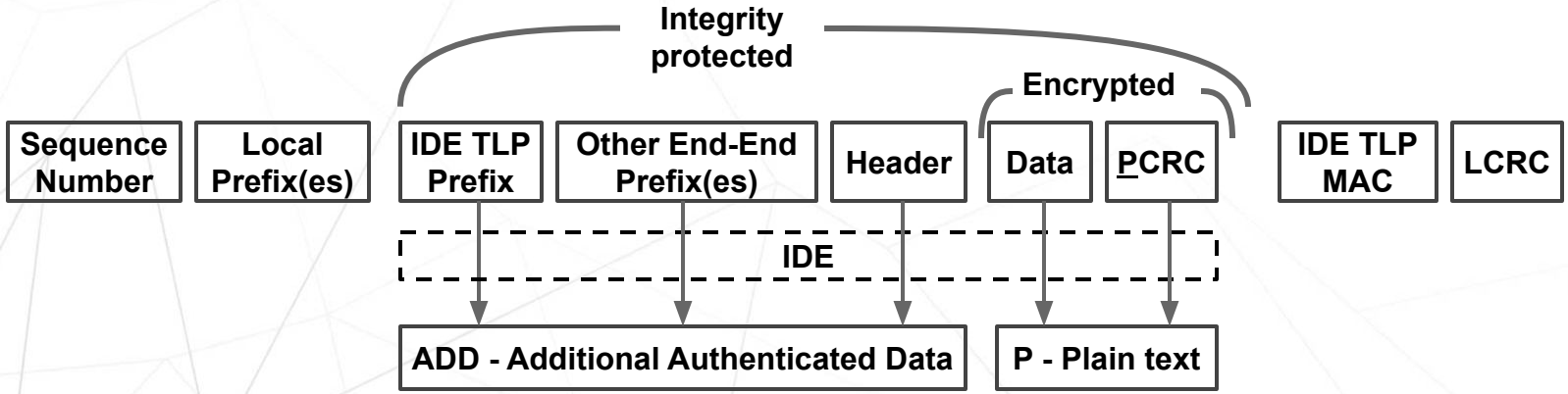


Attacker could:

1. Modify traffic
2. Inject traffic
3. Delete traffic
4. Replay traffic
5. Spy on traffic
6. Side channel

Integrity protects against first 5 while encryption protect against last two

Traffic Integrity

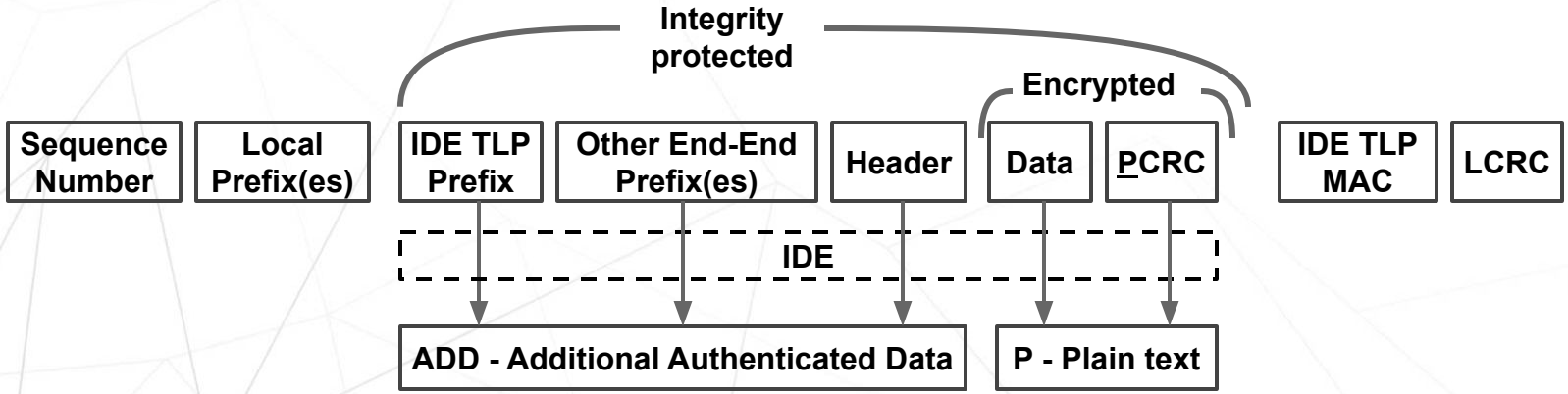


MAC == Message Authentication Code

- Crypto hash of the message + unique counter

Attacker need to know the key to generate correct MAC⇒Modify & Inject Messages Counter⇒Delete & Replay

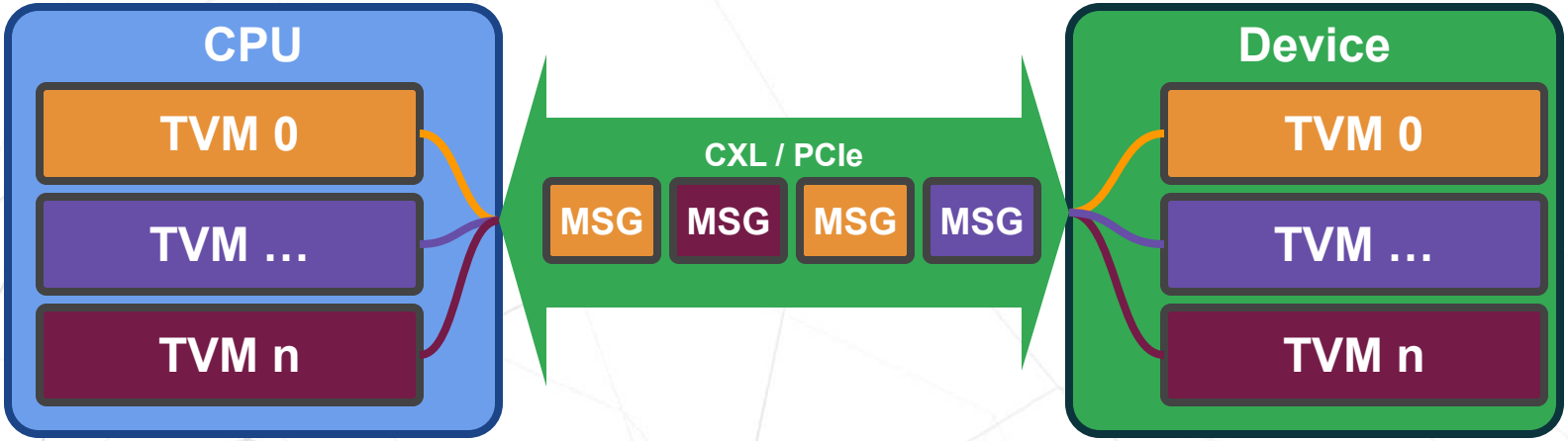
Traffic Encryption



[Encryption key] + [some counter / salt (receiver side can predict)]
⇒ Encrypt data payload (& CRC)

Attacker can not see data without knowing the key & counter / salt
Attacker will not see same ciphertext for same plaintext

Traffic Assignation

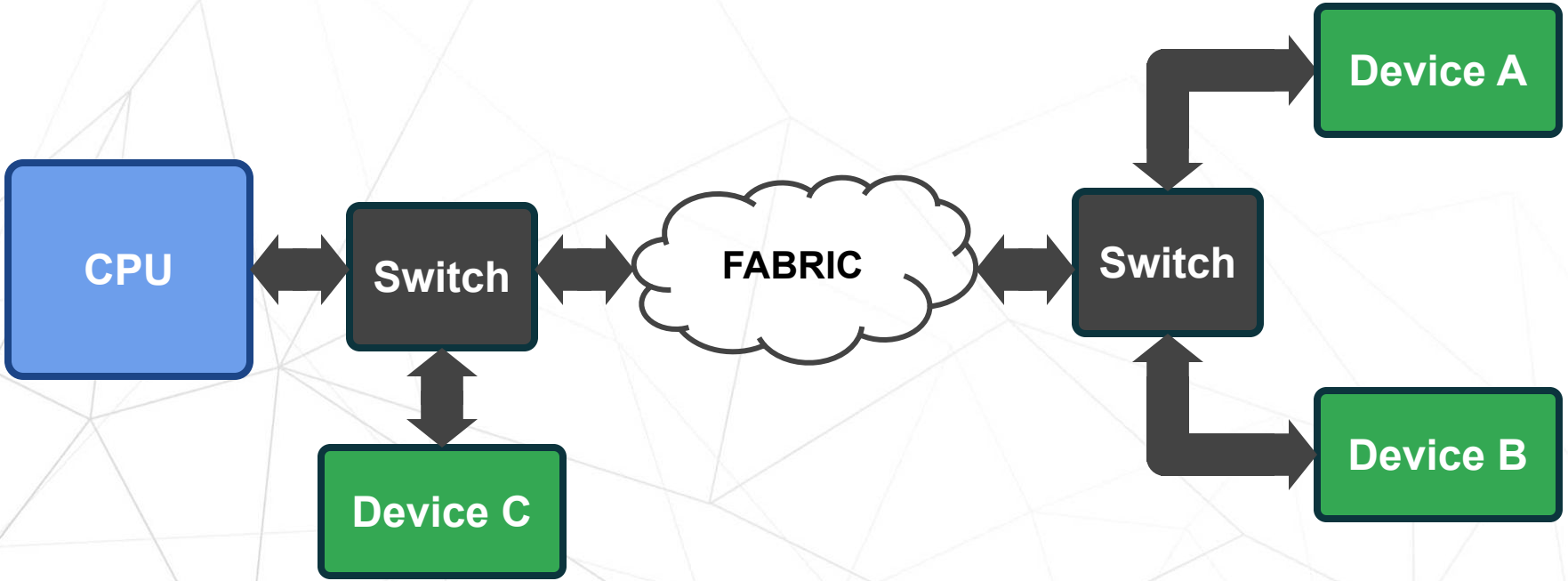


Protect Trusted Virtual Machine (TVM) End to End ⇒ Identify TVM Traffic

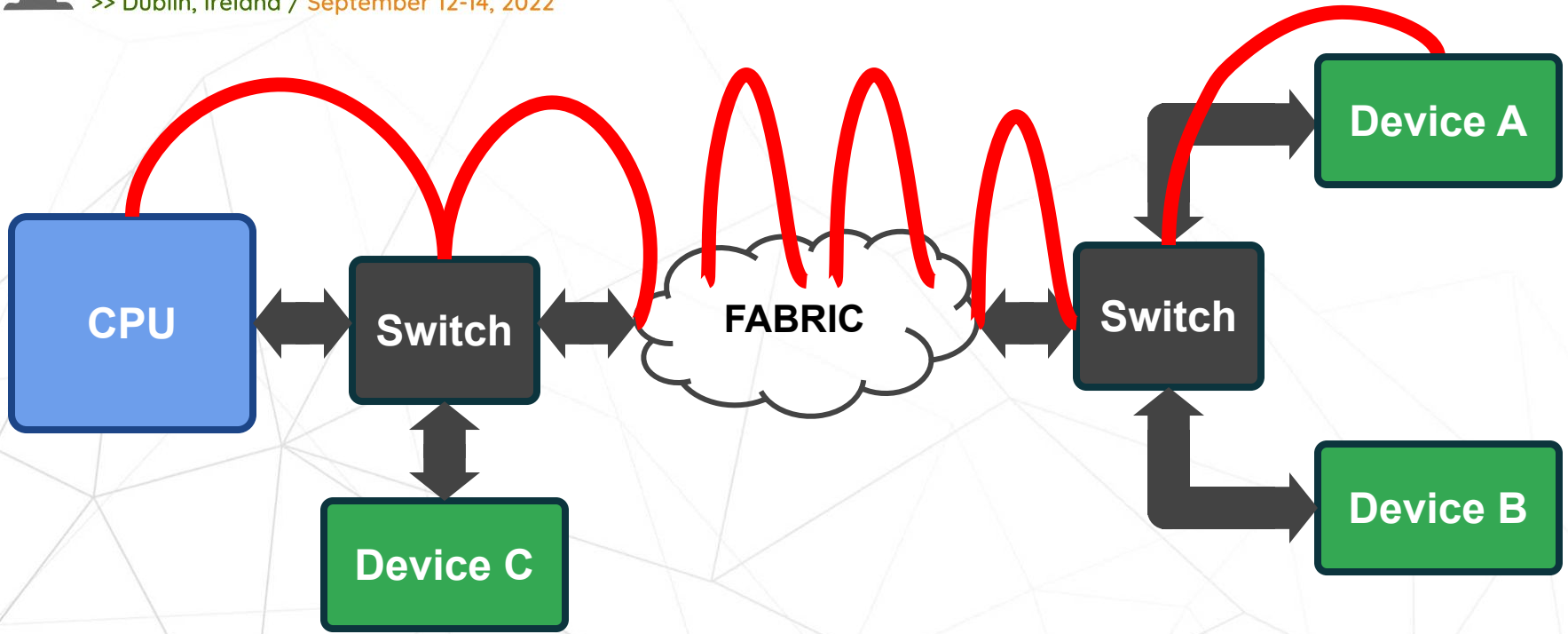
Useful for access control ⇒ a given TVM can only access specific memory (IOMMU)

Same encryption & integrity for all TVM ⇒ security risk:
Attacker can control one TVM & traffic sniffer and use it to help break traffic protection by sending specific pattern and looking for them in the traffic

Traffic Topology

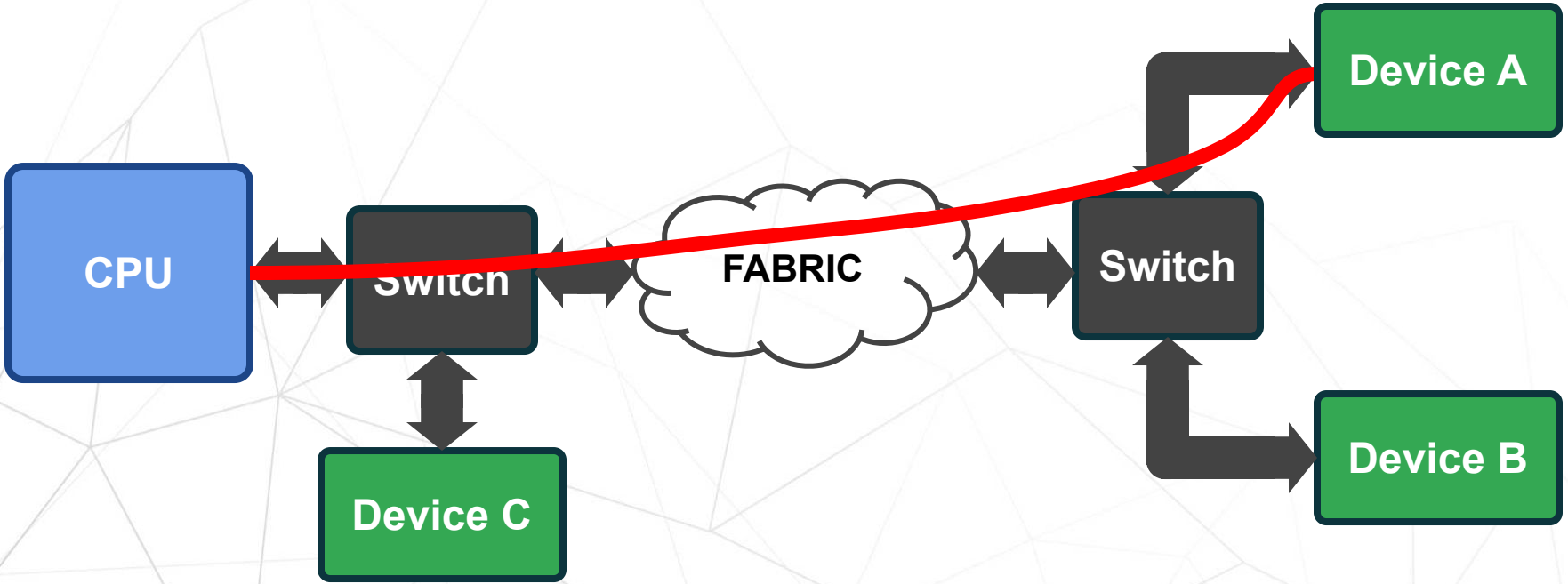


Point to Point



Integrity & Encryption between each point along a path
⇒ Have to trust each point in a path (each has access to plaintext)

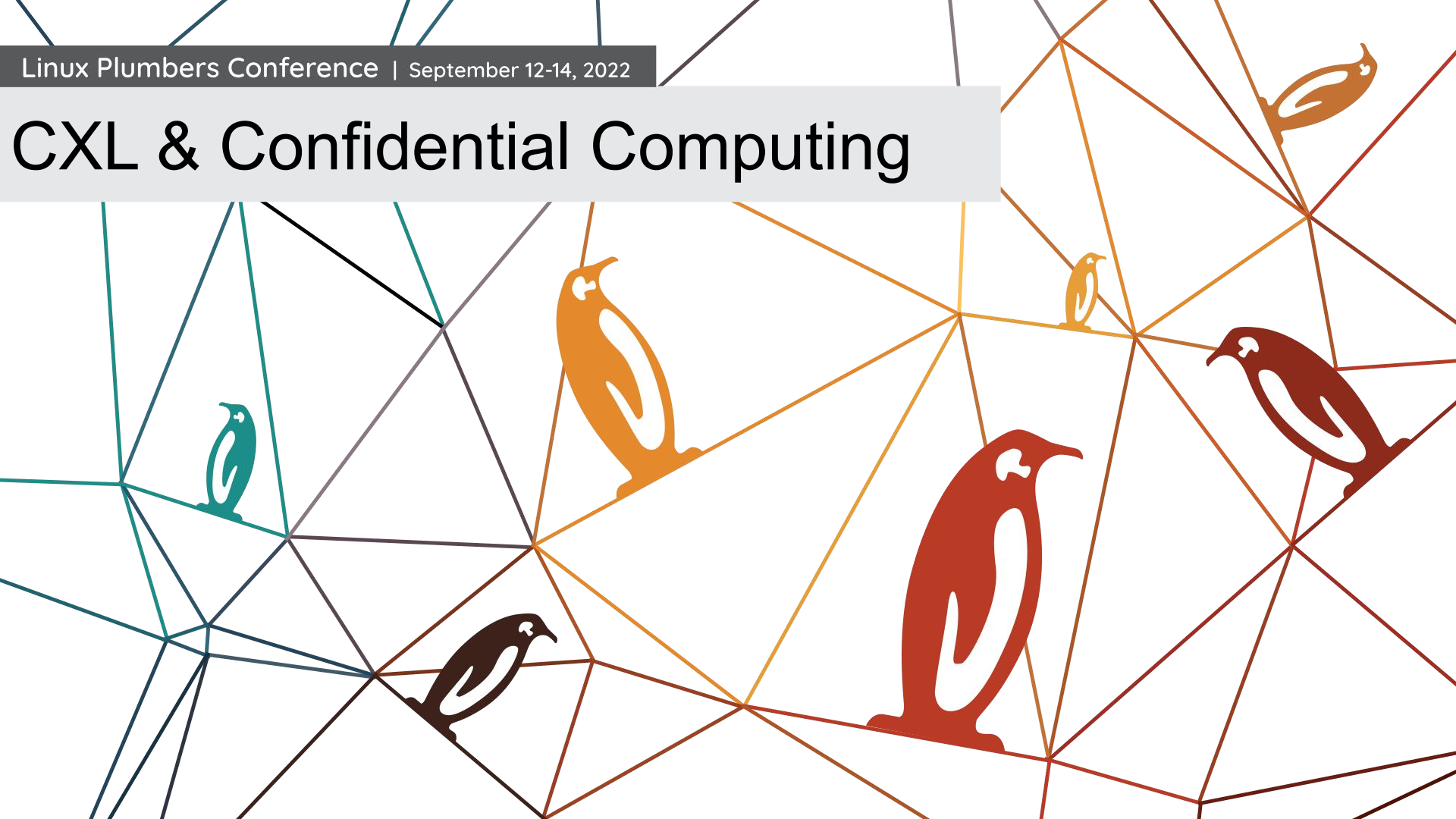
End to End



Integrity & Encryption between two end point
⇒ Have to trust end point (not the switch or anything in the path)

Linux Plumbers Conference | September 12-14, 2022

CXL & Confidential Computing



CXL Overview

CXL extends PCIe with 2 new protocols optimized for cache-coherent load/store

- Allows to add memory, through CXL, that behave like main DDR DIMM
- We can use different memory technologies through CXL memory controller
- Accelerators (GPUs, FPGAs, …) can participate in cache coherency like CPU

Every CXL message is around cache line (64 bytes) of data
⇒ Each message metadata must be as small as possible (bandwidth efficiency)
⇒ Very few fields
  - Opcode
  - Physical Address
  - Few ancillary bits

CXL Physical Address

CXL works with physical address ⇒ No IOMMU

⇒ Must trust device that can read / write / snoop
⇒ No central access control: each device responsible for it

CXL Traffic Identification

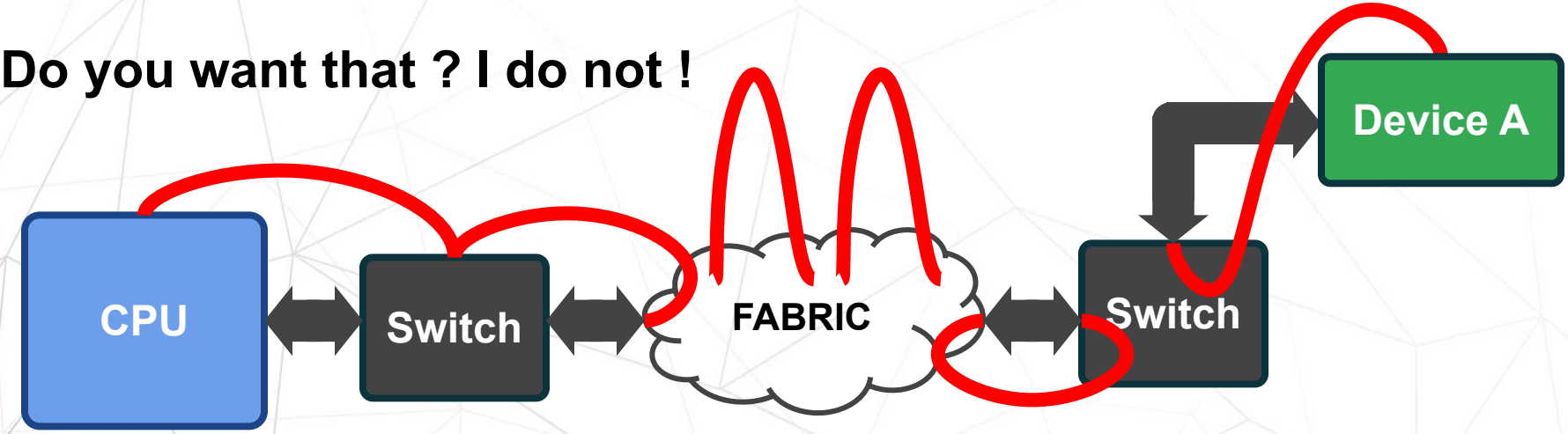
CXL no traffic identification (PCIe VF)

⇒ Traffic can not be assign to specific context (Trust Virtual Machine)
⇒ Access control at the source ? Destination can not identify context

CXL IDE

CXL IDE (Integrity and Data Encryption): point to point only
⇒ Trust every points in a path

Do you want that ? I do not !



Minimize TCB == Minimize number of chips you have to trust

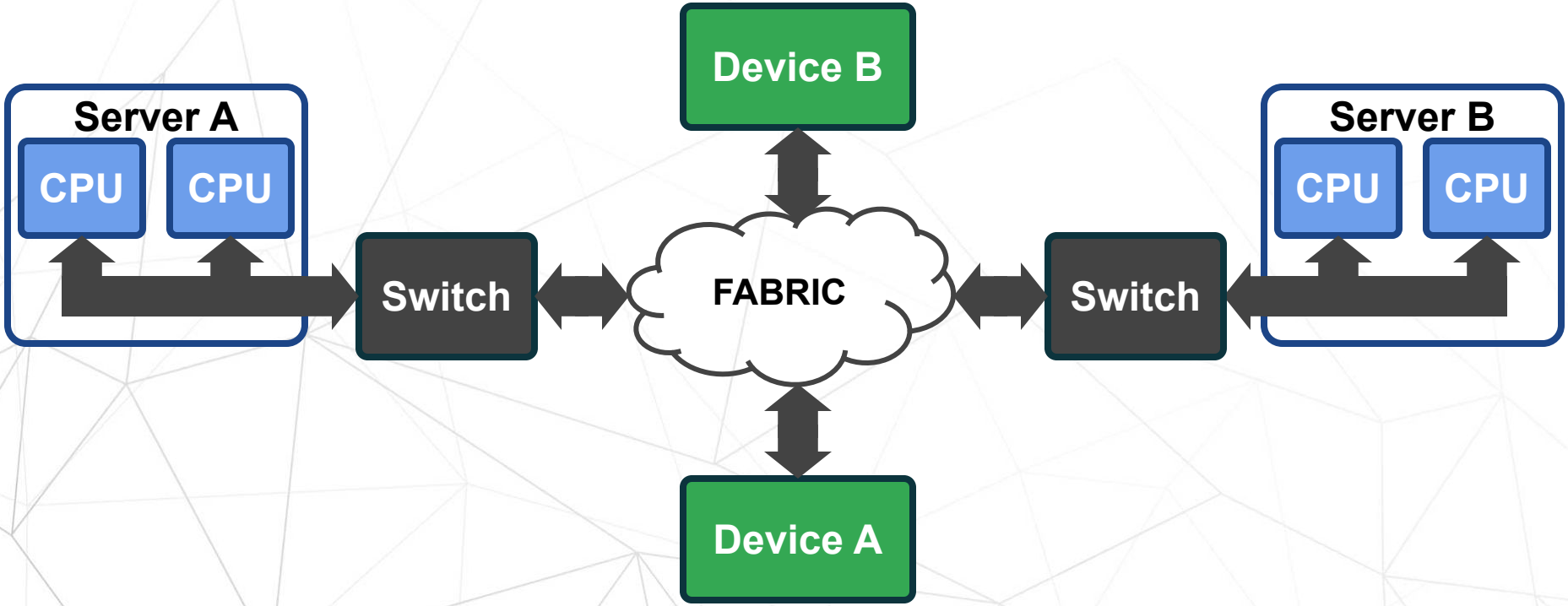
CXL: The rogue Device

All CXL memory device so far have a CPU within the CXL SoC

How do you feel about your memory having compute core ?

⇒ Attacker take control of the CXL memory controller
(rogue firmware, firmware bug, …)
⇒ Escape memory of one TVM through another

CXL Multi-Host (*CXL 3.0*)



Linux Plumbers Conference 2022

>> Dublin, Ireland / September 12-14, 2022

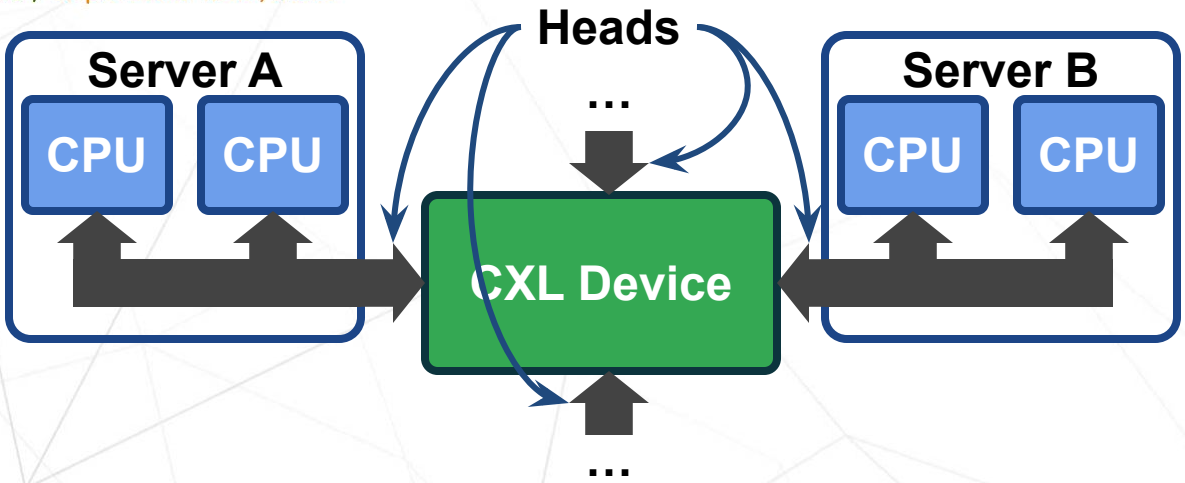
The rogue Fabric Manager

Fabric Manager control routing and device assignation

Device can have multiple concurrent host/server

Attacker controlling one host using the Fabric Manager or device

CXL Multi-Head (*CXL 3.0*)



One device ⇒ Multiple Host/Server
⇒Memory pooling
⇒Memory sharing

Any Other Issues ?

Anything else is scaring people with CXL ?

Thank You \o/