DmaAuth: A Lightweight Pointer Integrity-based Secure Architecture to Defeat DMA Attacks

Xingkai Wang, Wenbo Shen[✉], Yujie Bu, Jinmeng Zhou, Yajin Zhou

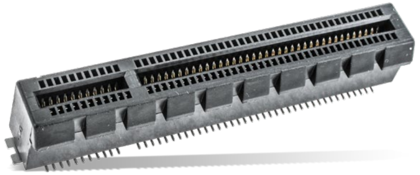
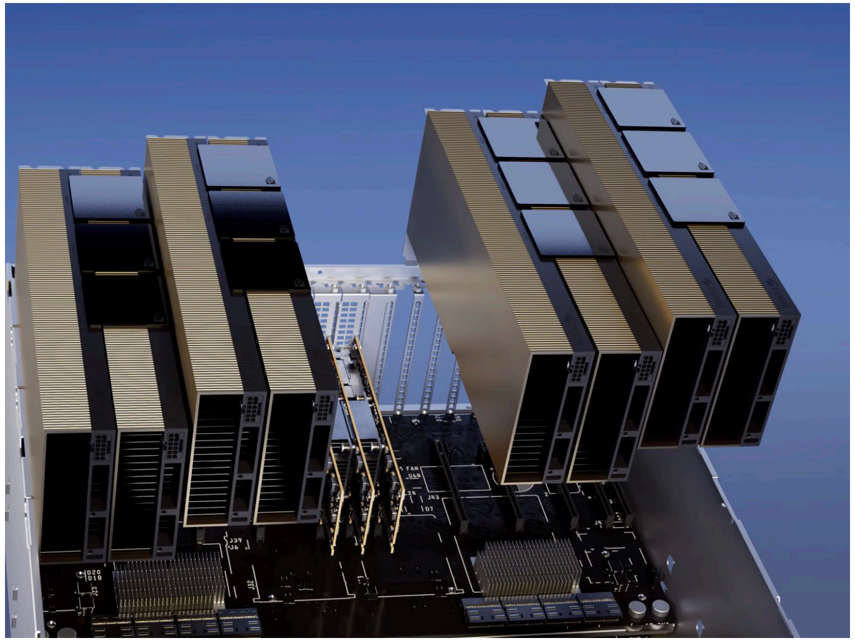
Zhejiang University

Agenda

- Motivation
- Characterization
- Design
- Implementation
- Evaluation
- Conclusion

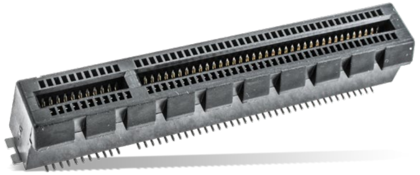
DMA Attack

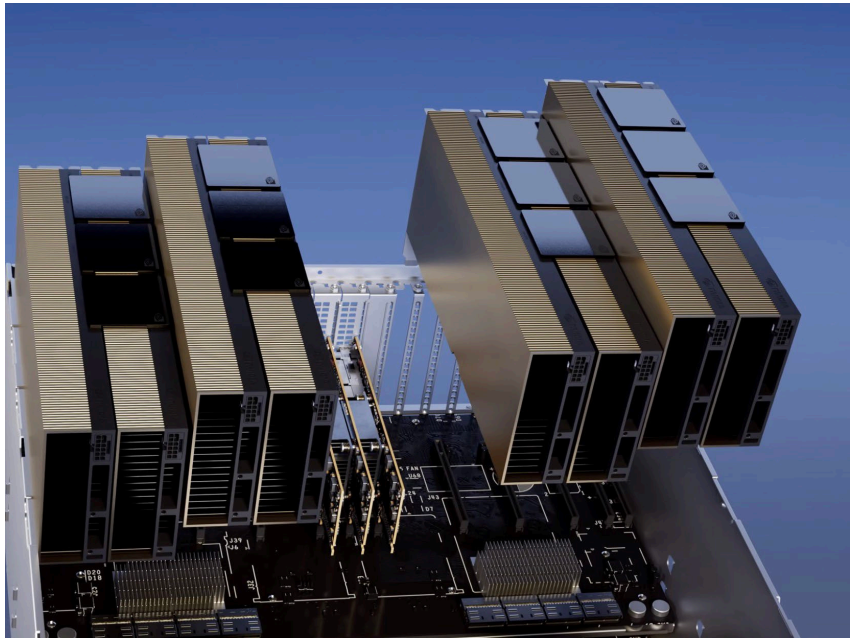
- DMA allows devices to read/write the memory.
  - Fire Wire
  - Thunderbolt
  - PCIe

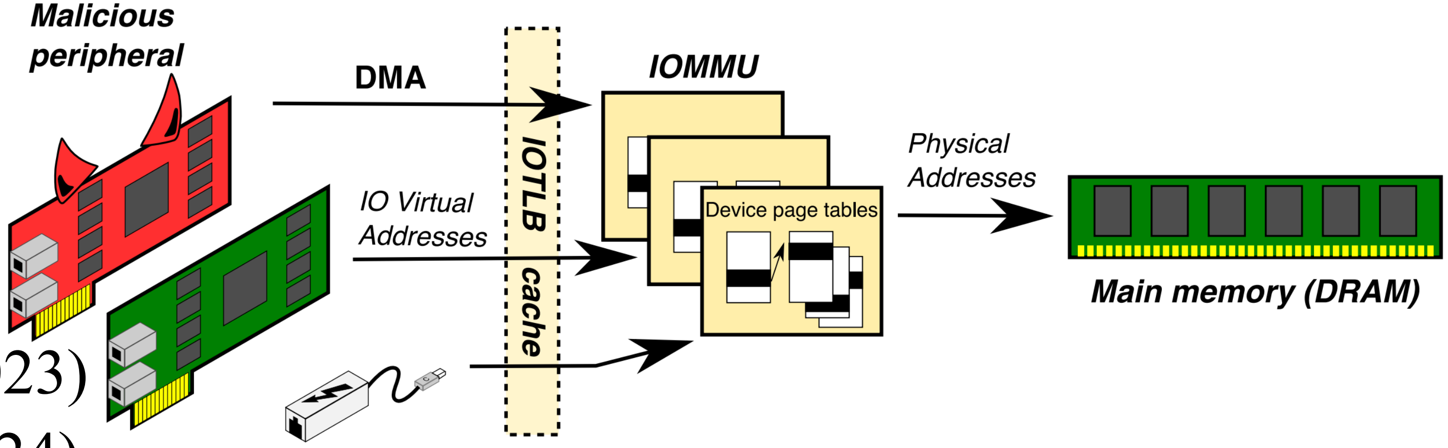
DMA Attack

- DMA allows devices to read/write the memory.
  - Fire Wire
  - Thunderbolt
  - PCIe



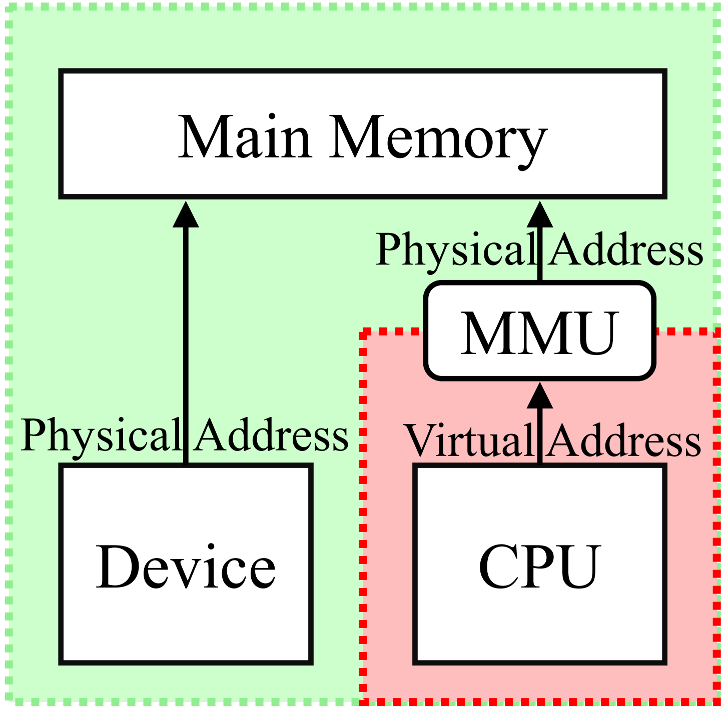


- Threat: devices can control the entire system by corrupting the memory.
  - Owned by an iPod (2005)
  - Over the Air (2017)
  - TiYunZong (2019)
  - Thunderclap (2019)
  - Make KSMA Great Again (2023)
  - The Way to Android Root (2024)



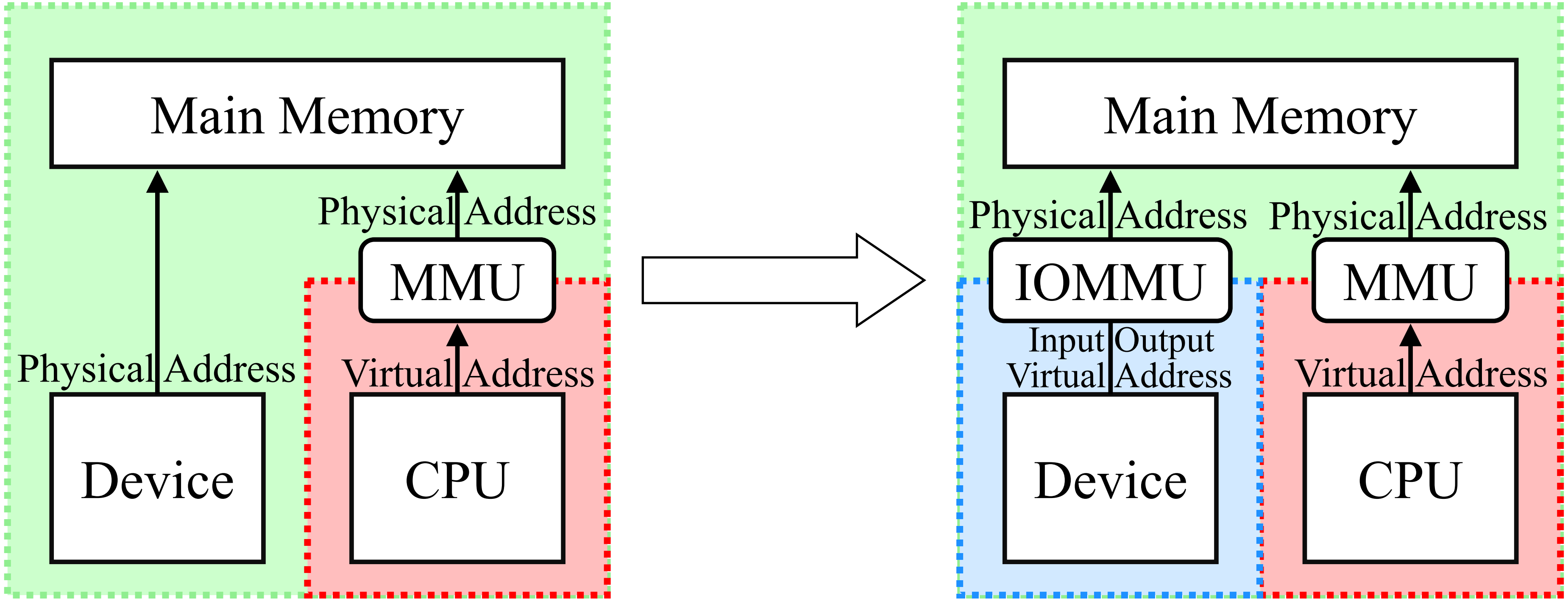
Defense: IOMMU

- Traditional systems use MMU to **virtualize** the address space for user space programs and restrict memory accesses from user space.



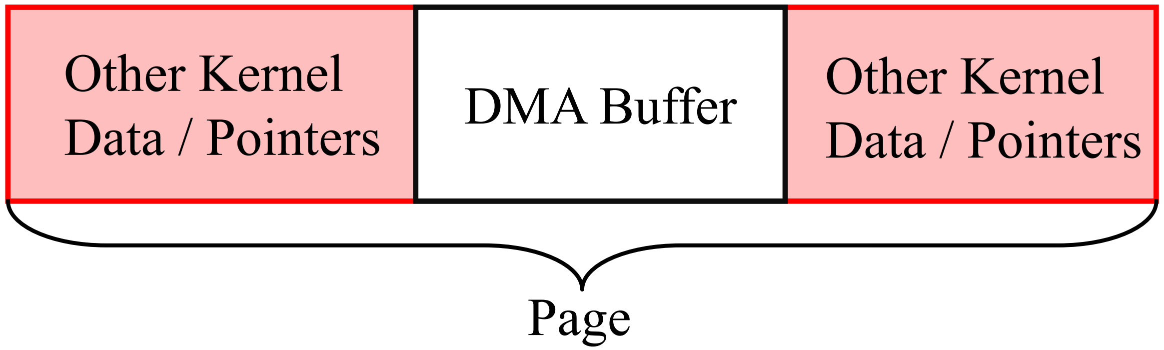
Defense: IOMMU

- IOMMU maps Physical Addresses to **Input Output Virtual addresses**, restricting memory regions accessed by devices.



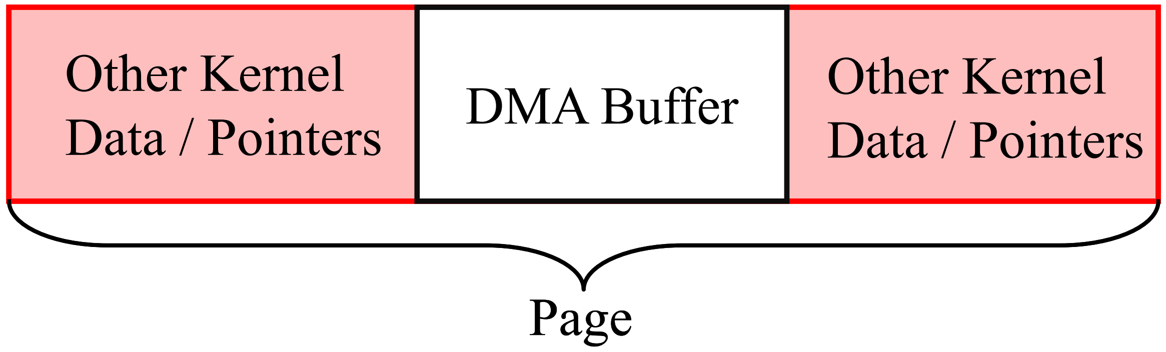
Vulnerabilities

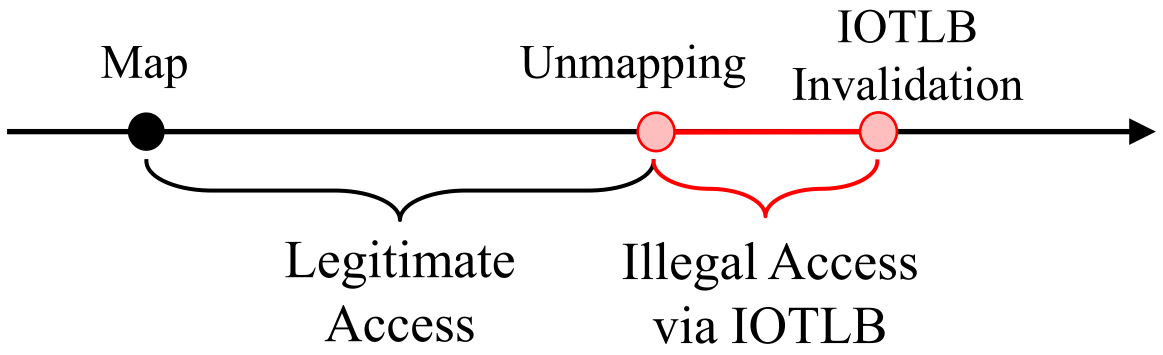
- Spatial Vulnerability
  - DMA buffers are **not** always multi-page sized.
  - Pages mapped for devices may contain other **sensitive data**.



Vulnerabilities

- Spatial Vulnerability
  - DMA buffers are **not** always multi-page sized.
  - Pages mapped for devices may contain other **sensitive data**.
- Temporal Vulnerability
  - IOTLB invalidation is **deferred** to reach acceptable overhead.
  - Devices can access **unmapped** memory in the deferred window.





Motivation

- Contemporary IOMMU cannot effectively defeat elaborate DMA attacks exploiting spatial and temporal vulnerabilities.
- There needs to be a solution with
  - Strong spatial and temporal **security** guarantees
  - **Transparency** to existing hardware
  - **Compatibility** with existing device drivers
  - **Small** throughput overhead
  - **Low** CPU time consumption

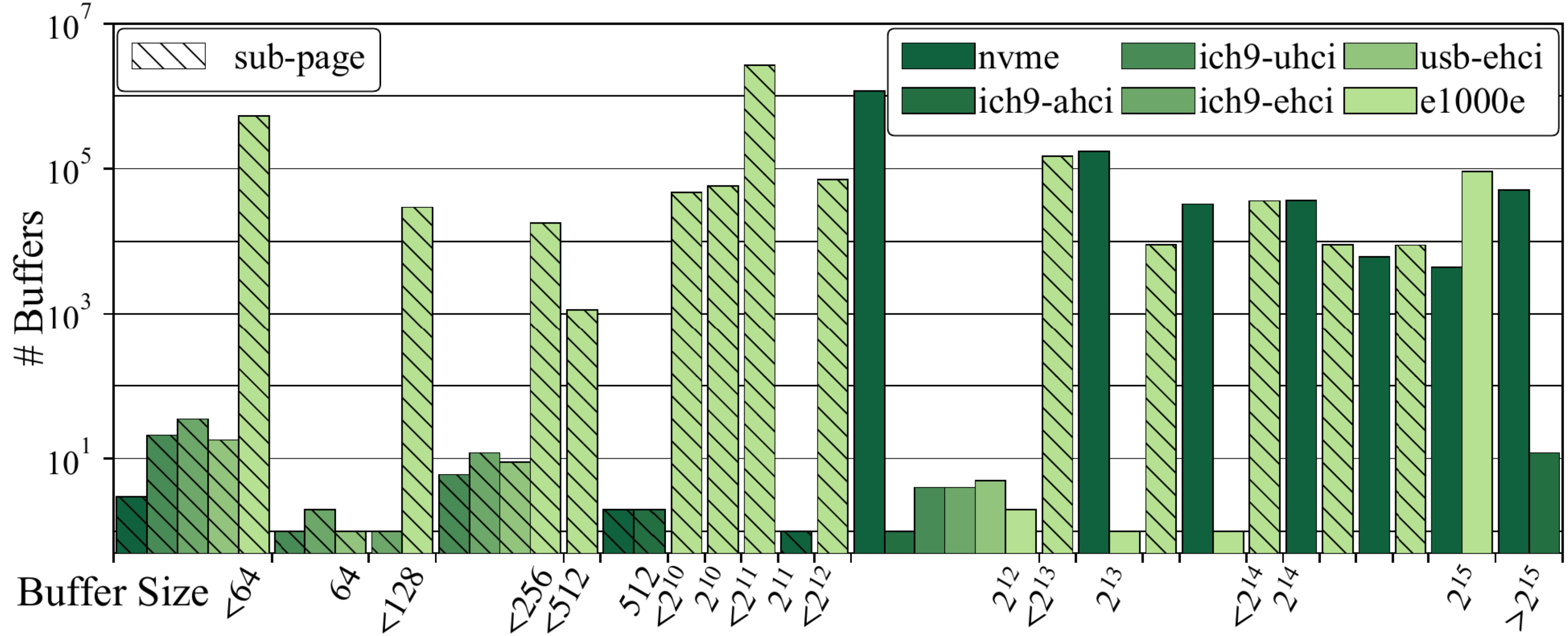
Characterization: Access Pattern

- Most (75.2%) DMAs are not using the original pointers, but with an offset added to the pointer (**pointer arithmetic**).
- The number of coexisting DMA buffers is **limited**.

Device Information		Pointer Arithmetic Statistics			Mappings Statistics	
Device	DMA Interface	With Offset	Total Access	Ratio	Coexist	Total
NVMe SSD	PCIe (`nvme`)	4406751	5943096	74.1%	154	1487516
SCSI HDD	AHCI (`ich9-ahci`)	40	67	59.7%	13	15
Mouse and Tablet	EHCI (`ich9-ehci`)	40690	40956	99.4%	6	54
Keyboard	UHCI (`ich9-uhci`)	5066871	6629284	76.4%	5	32
USB Stick	EHCI (`usb-ehci`)	35086	35372	99.2%	5	33
E1000E NIC	PCIe (`e1000e`)	11230518	14985786	74.9%	271	3744537
Total	/	20779956	27634561	75.2%	435	5232187

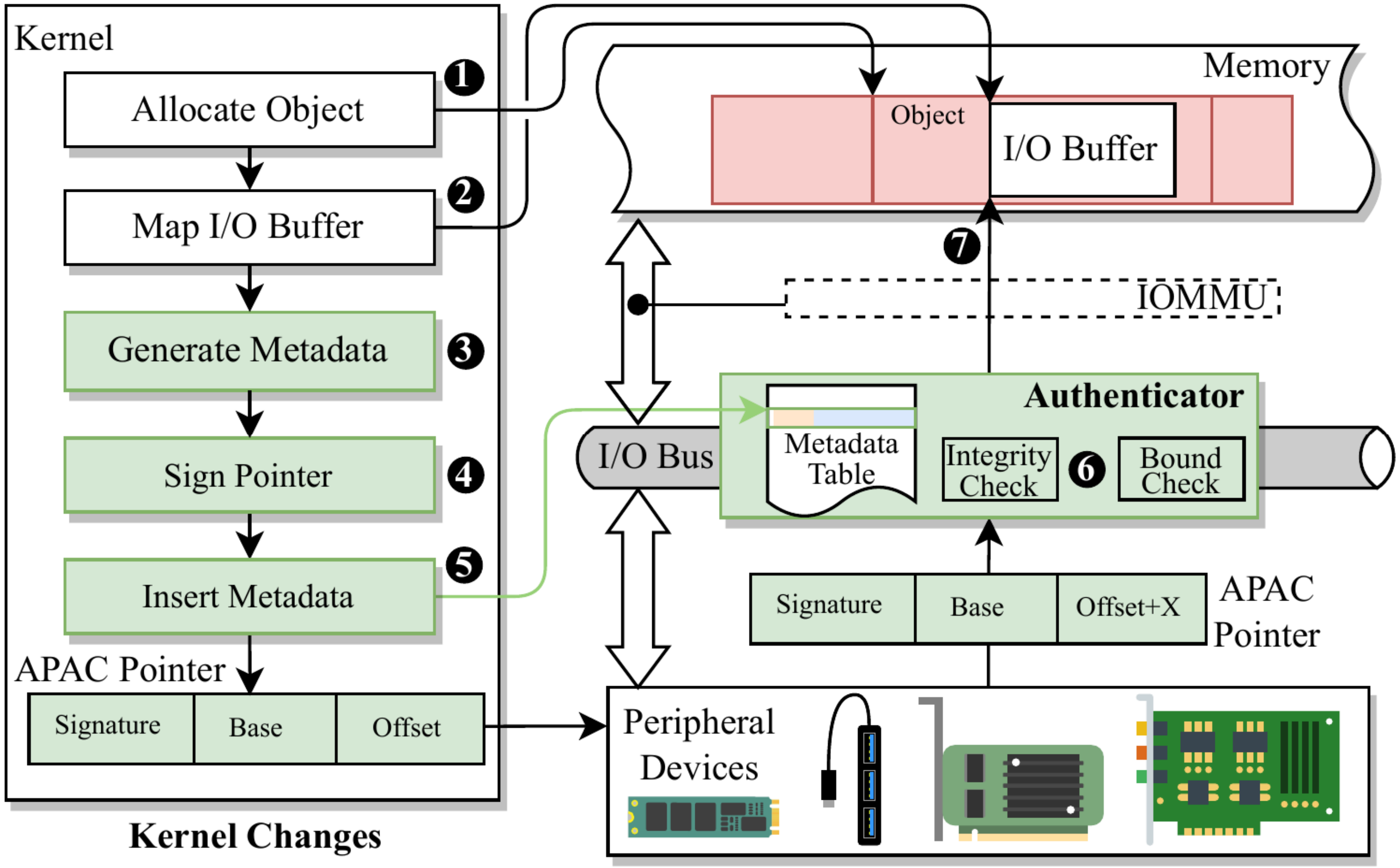
Characterization: Mapping Size

- Most (69.8%) of the DMA buffers are **not multi-page sized** and have potential spatial vulnerability.



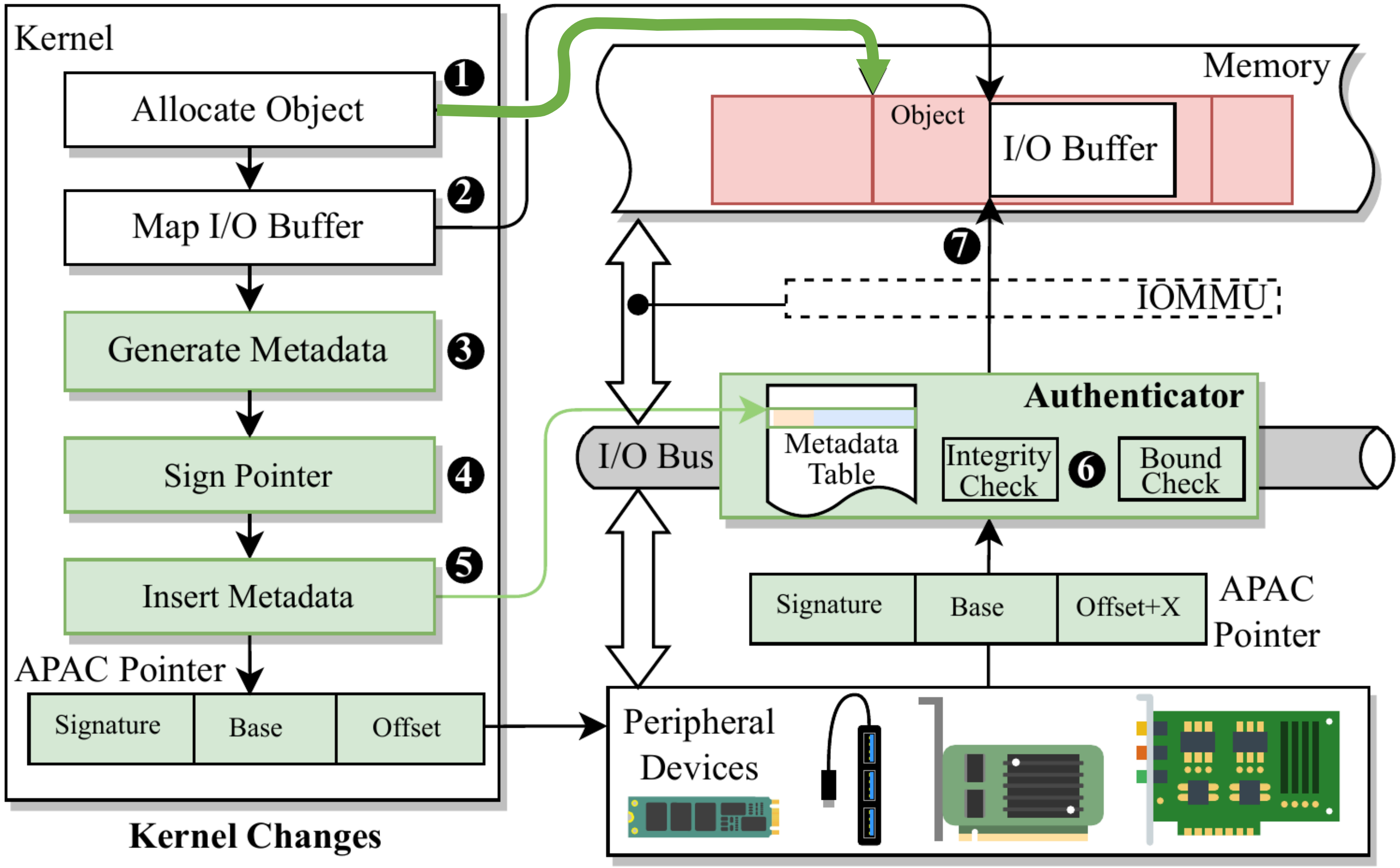
Design

- DMA Pointer Authentication
  - Keeps the key in CPU
  - Lets kernel fully control DMA pointers.
- Bound Checking
  - Records fine-grained bound information
  - Prevents all the out-of-bound DMAs



Design

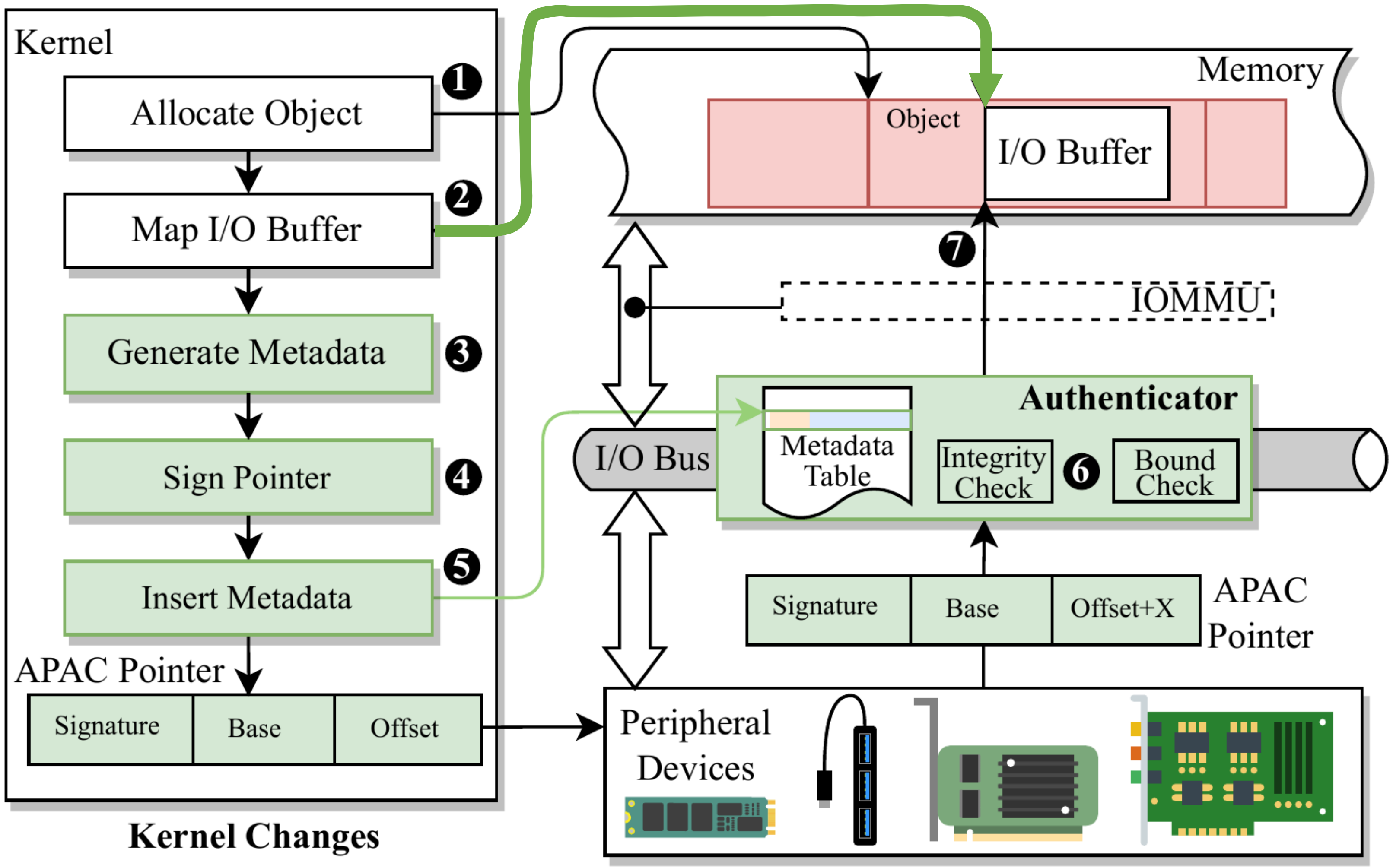
① Kernel **allocates** a object, which has a I/O buffer. But the rest of the object or the page shouldn't be accessed by DMA.



Design

① Kernel **allocates** a object, which has a I/O buffer. But the rest of the object or the page shouldn't be accessed by DMA.

② Kernel **maps** the buffer to the device explicitly a to get the DMA pointer.

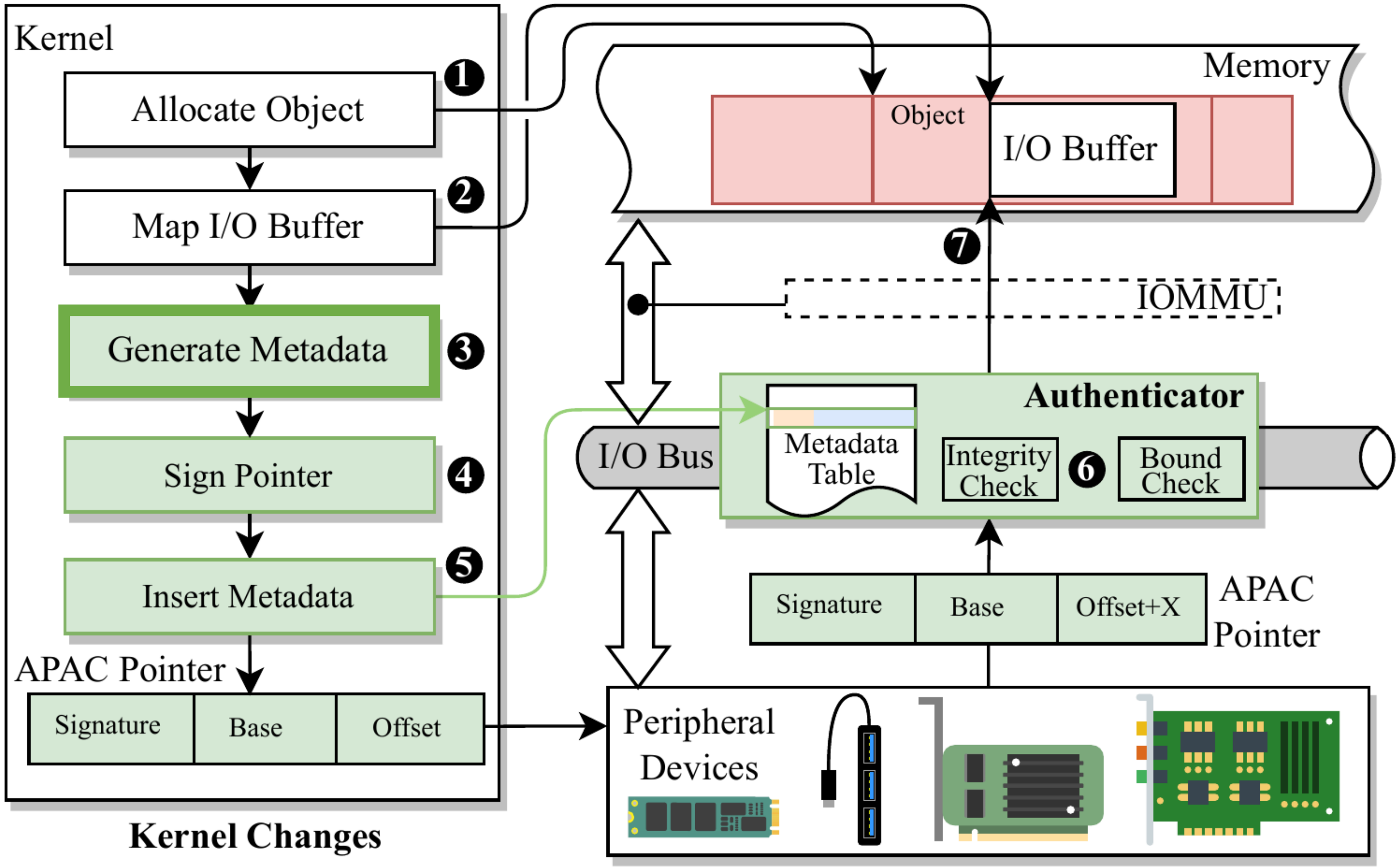


Design

① Kernel **allocates** a object, which has a I/O buffer. But the rest of the object or the page shouldn't be accessed by DMA.

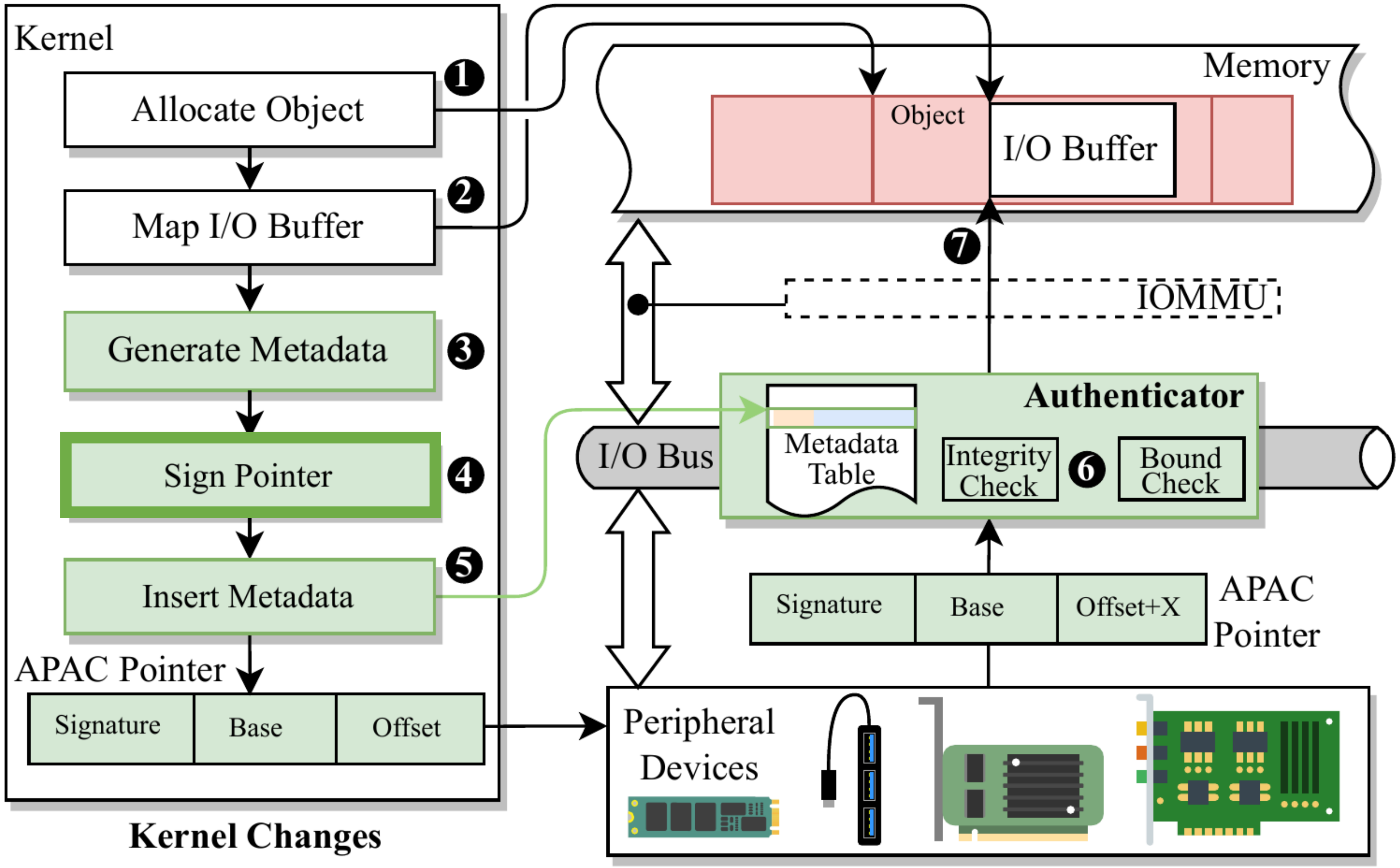
② Kernel **maps** the buffer to the device explicitly a to get the DMA pointer.

③ Metadata of the mapped I/O buffer is **generated**.



Design

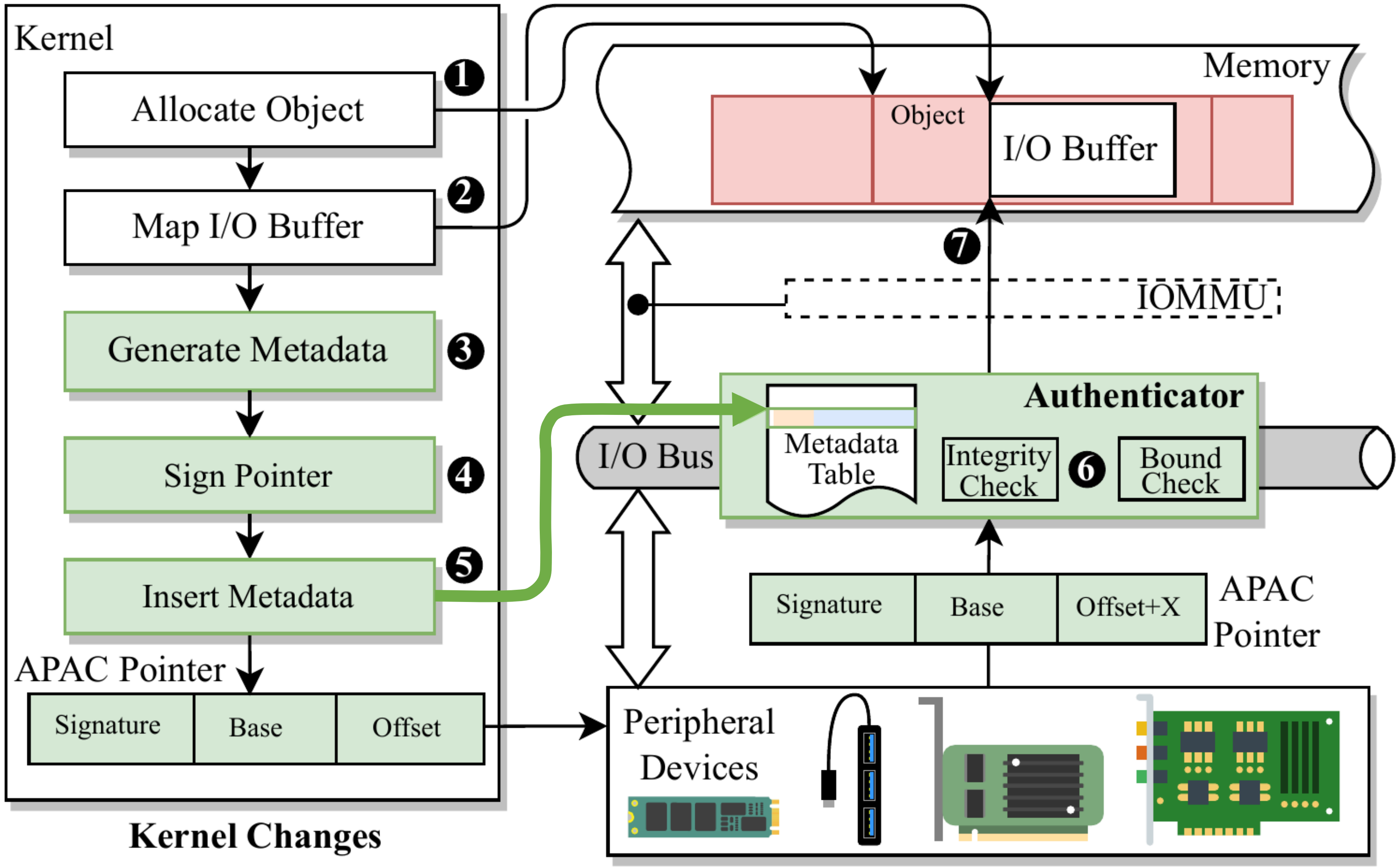
④ Kernel **signs** the DMA pointer with the corresponding metadata.



Design

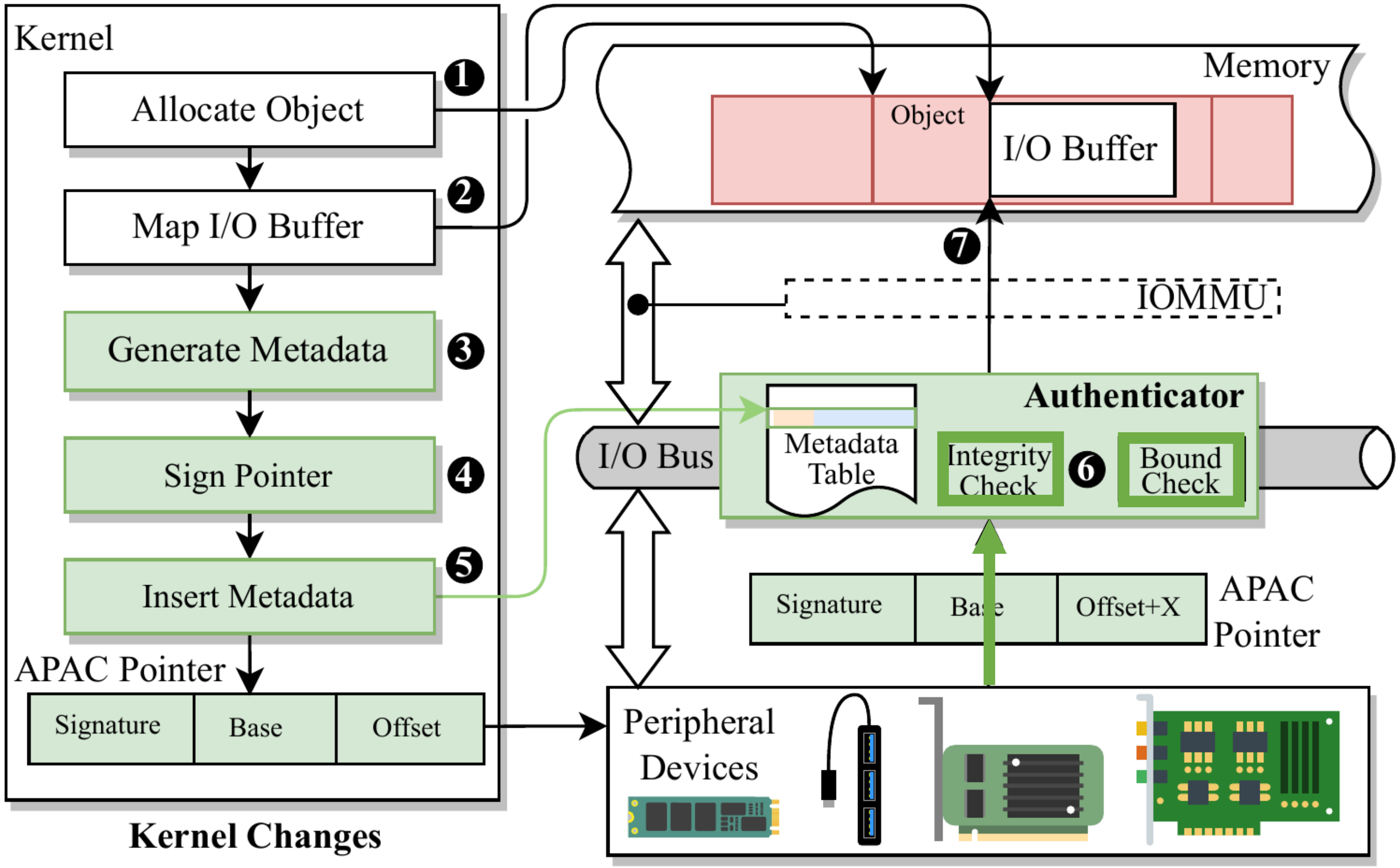
④ Kernel **signs** the DMA pointer with the corresponding metadata.

⑤ Metadata is **stored** in the hardware authenticator to be referenced when authenticating the corresponding pointer.



Design

⑥ When peripherals use the signed pointers to perform DMA, the authenticator hardware on I/O bus fetches the corresponding metadata to **check bounds** and **perform authentication**.

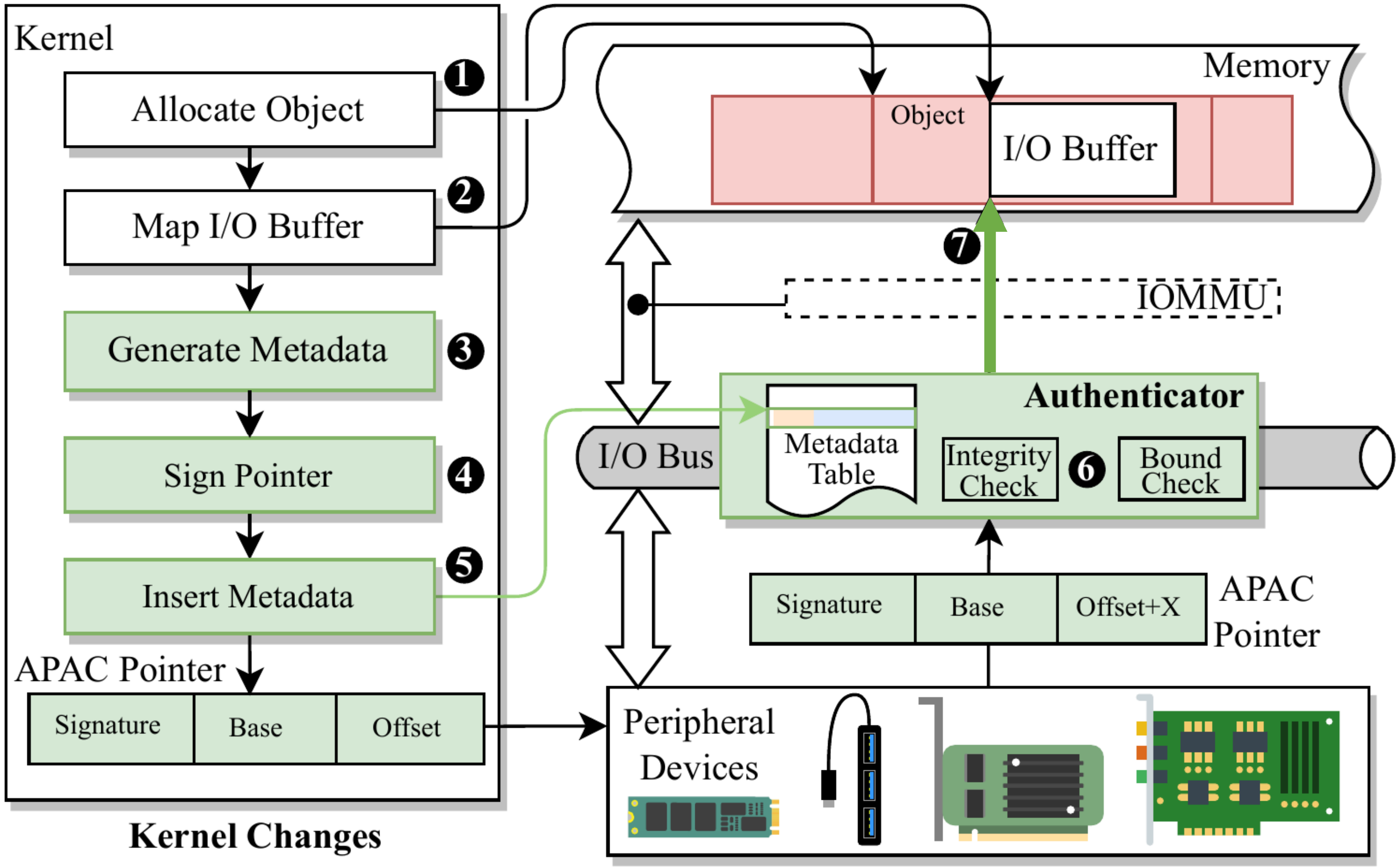


Kernel Changes

Design

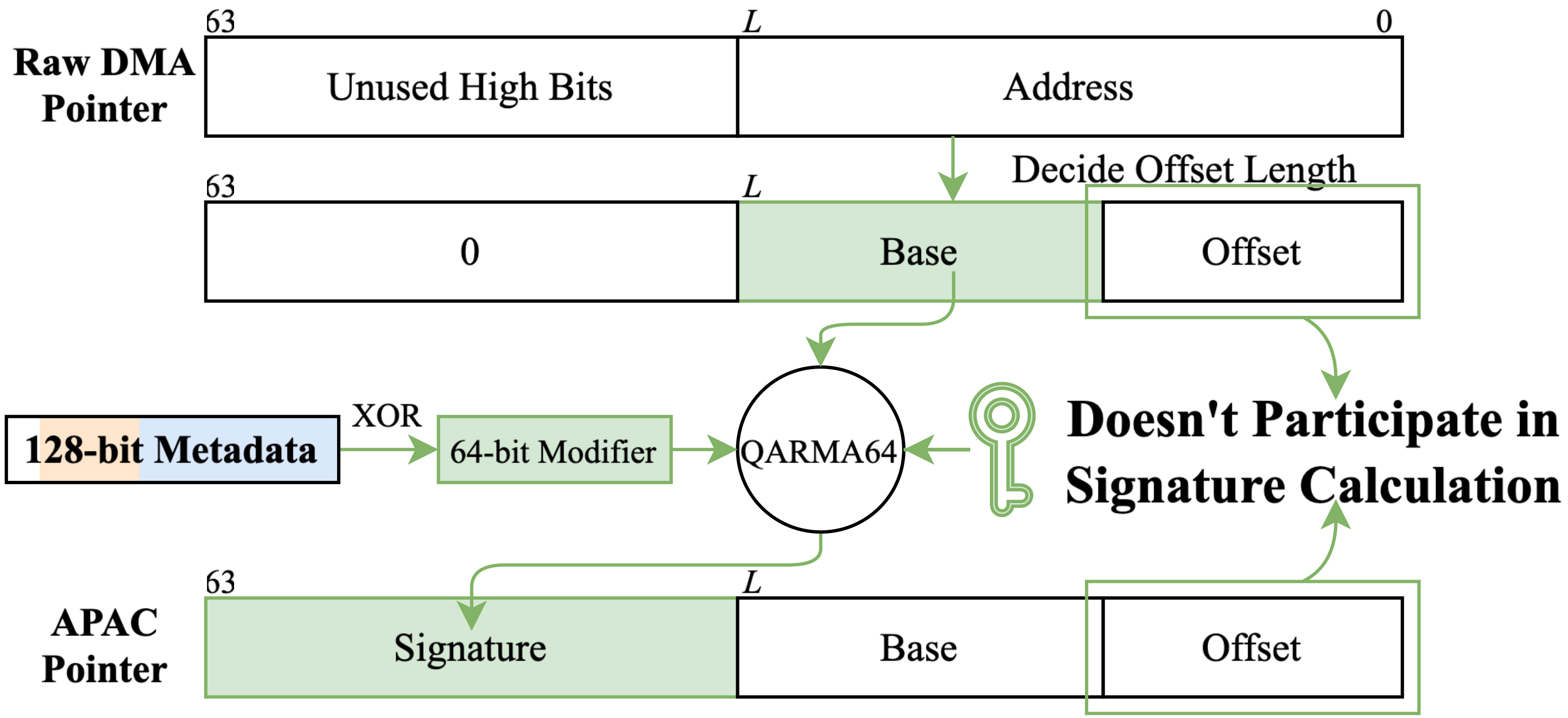
⑥ When peripherals use the signed pointers to perform DMA, the authenticator hardware on I/O bus fetches the corresponding metadata to **check bounds** and **perform authentication**.

⑦ Only legitimate DMAs can **access** the memory.



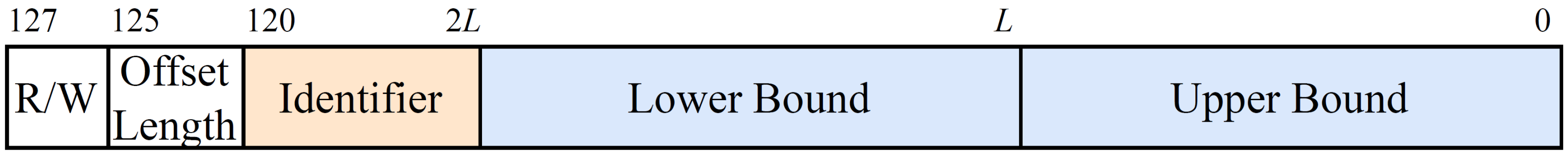
Solution to Pointer Arithmetic: APAC

- **<u>A</u>rithmetic Capable <u>P</u>ointer <u>A</u>uthenti<u>c</u>ation** signs the DMA pointer with only the high bits, allowing pointer arithmetic within the lower bits without influencing signature calculation.



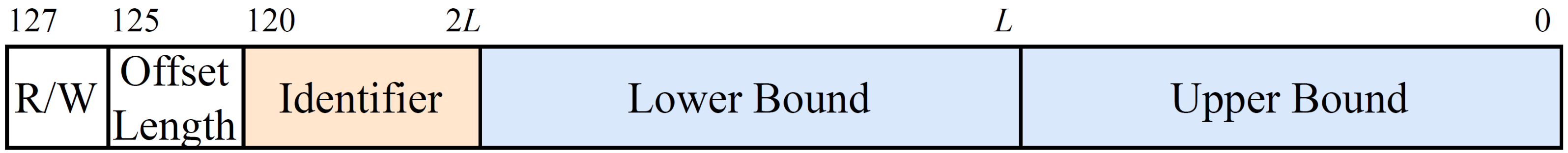
Metadata Format and Positioning

- Metadata contains the following fields:
  - Read/write permission
  - Length of the offset
  - Random identifier
  - Upper bound and Lower bound

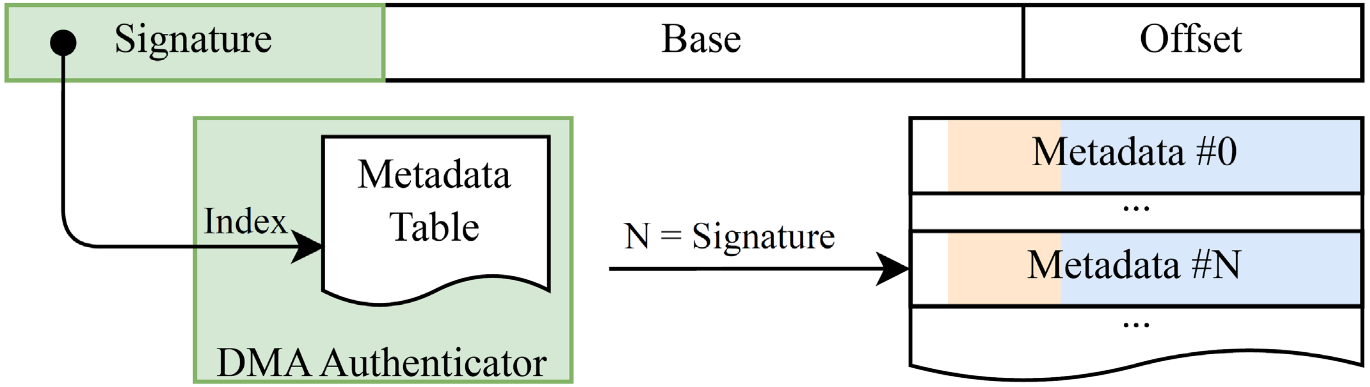


Metadata Format and Positioning

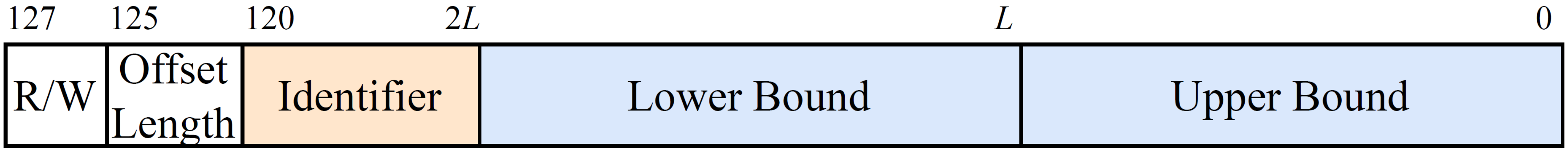
- Metadata contains the following fields:
  - Read/write permission
  - Length of the offset
  - Random identifier
  - Upper bound and Lower bound



- The metadata is stored in a dedicated area and index with the signature
  - Identifier defeats reuse and temporal attacks.
  - Write-only metadata prevents the potential metadata leakage.



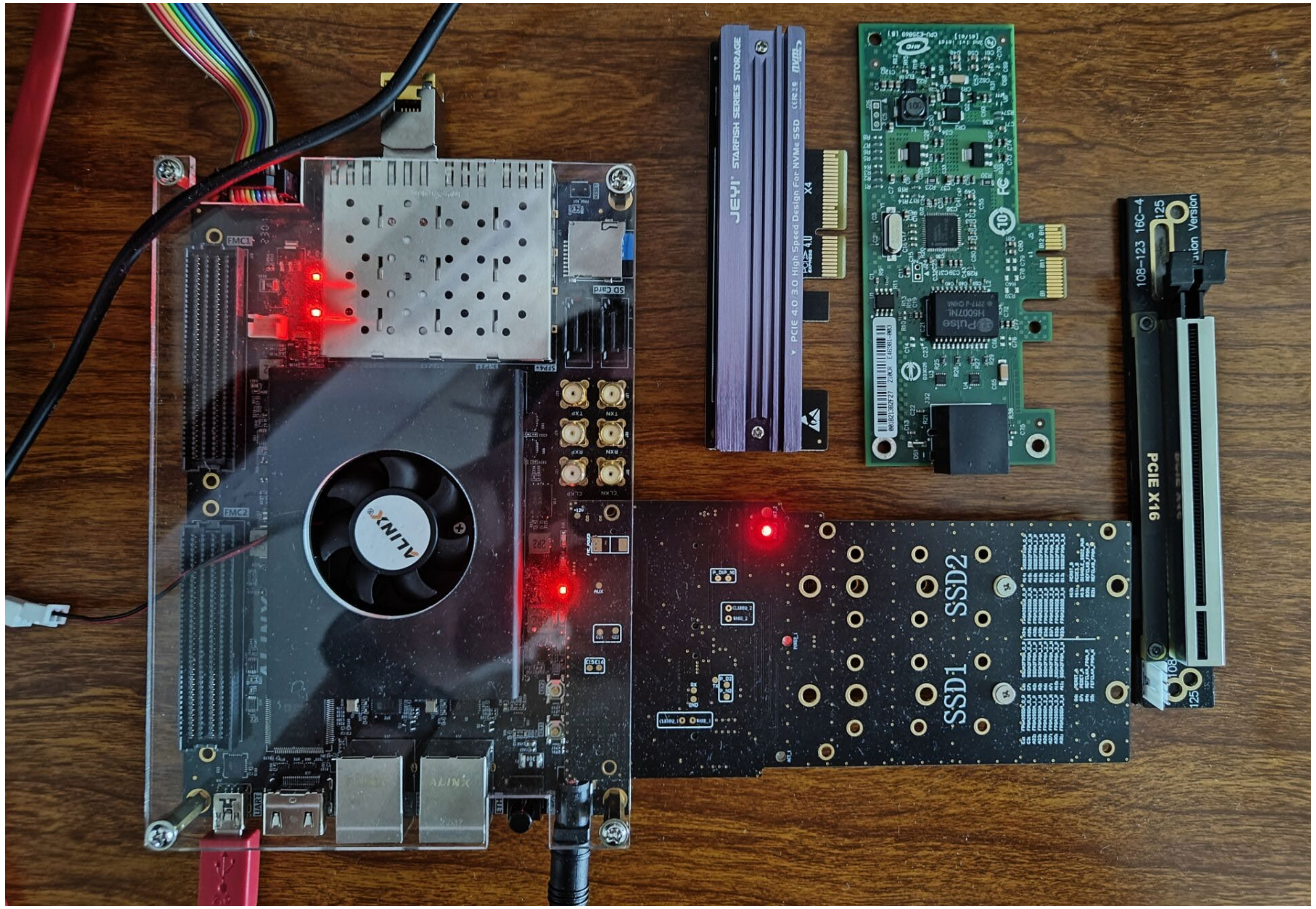
Resolving the Vulnerabilities



- Spatial Vulnerability
  - **Byte-granularity** bound information
- Temporal Vulnerability
  - Re-randomizes the **Identifier**
  - **Changes signature** hash result
  - **Immediately** invalidates outdated pointers holding the **outdated signature**

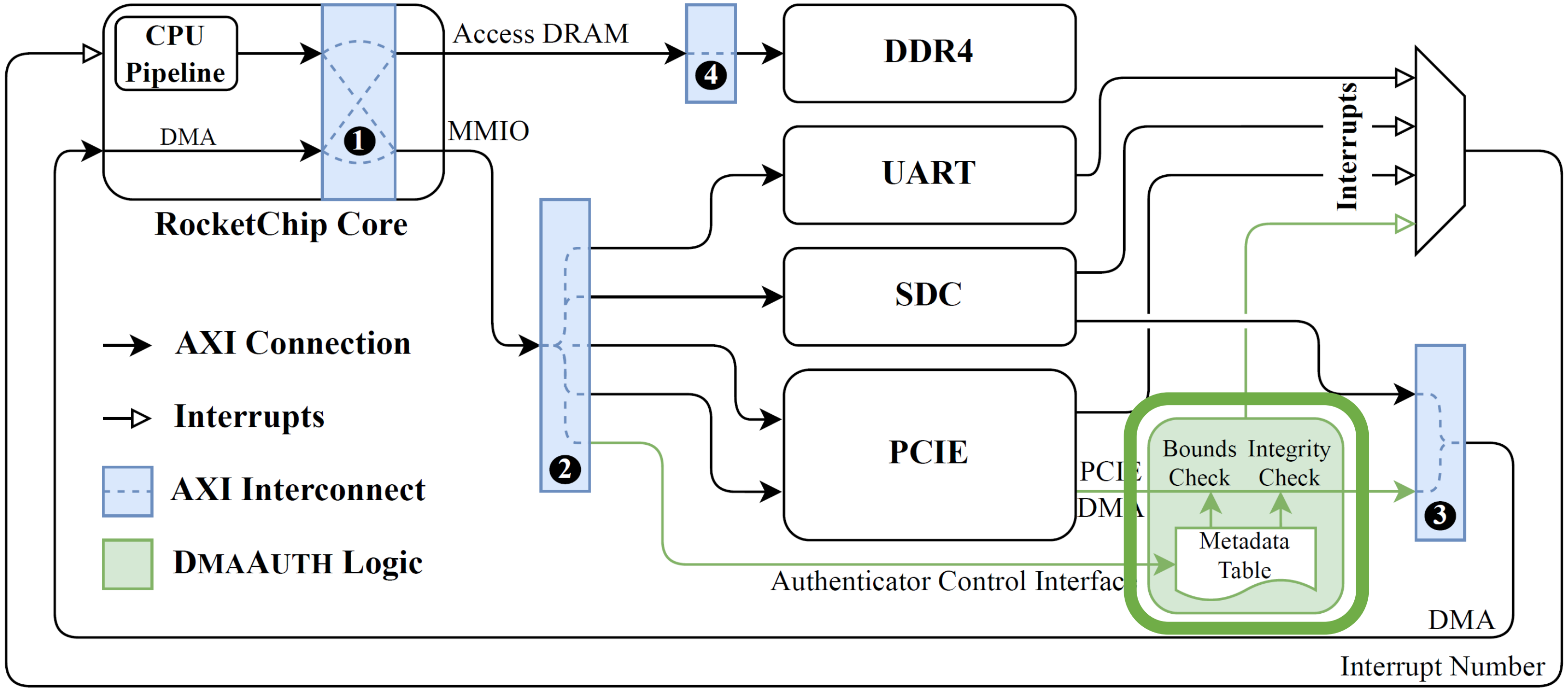
Implementation

- SoC research framework with PCIe 3.0 x8 bus
  - Customizable interconnection between PCIe bus and DRAM
  - Baseline for various hardware-software co-design
  - High performance IOMMU
    - 5.8% throughput overhead
    - 5.6% CPU time overhead
    - Comparable to IOMMUs on commercial SoCs



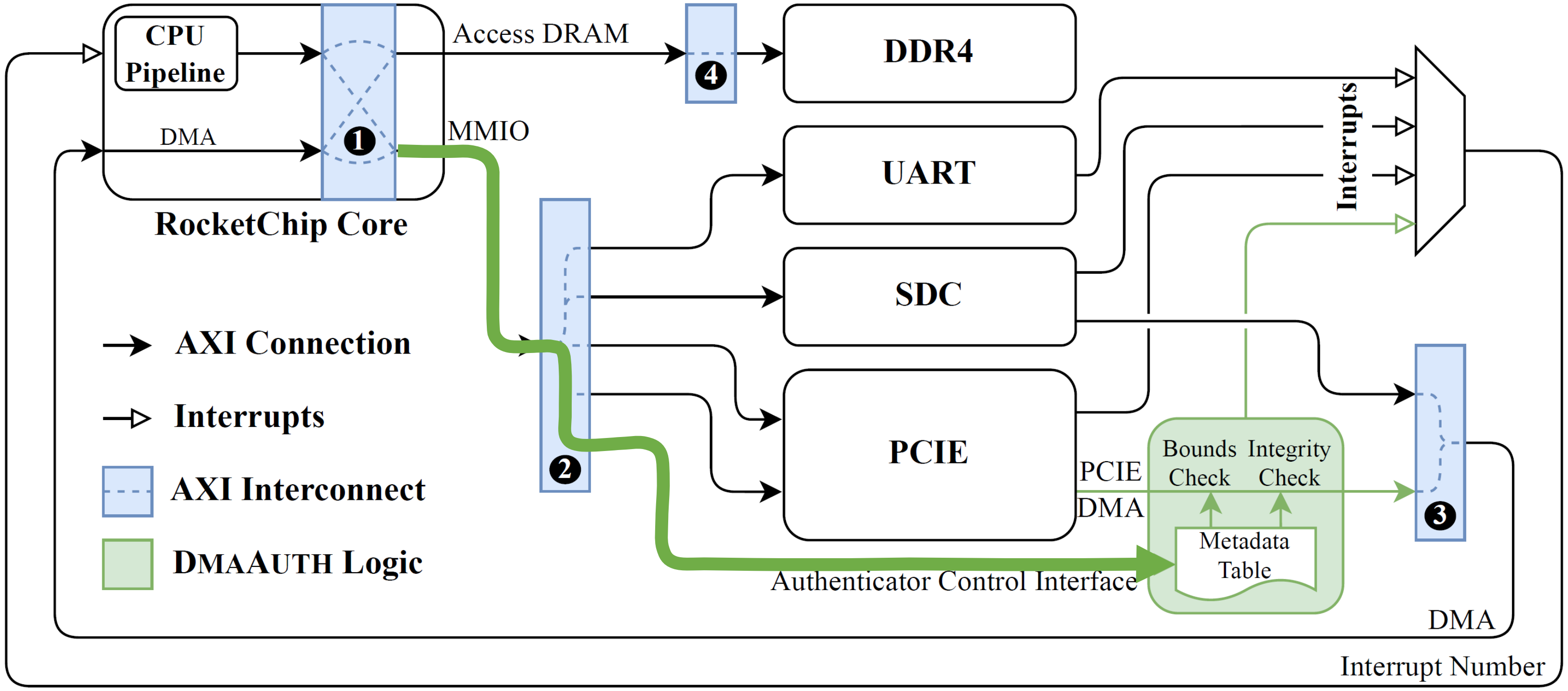
Implementation

- The hardware authenticator is put between **PCIe bus and DRAM**.



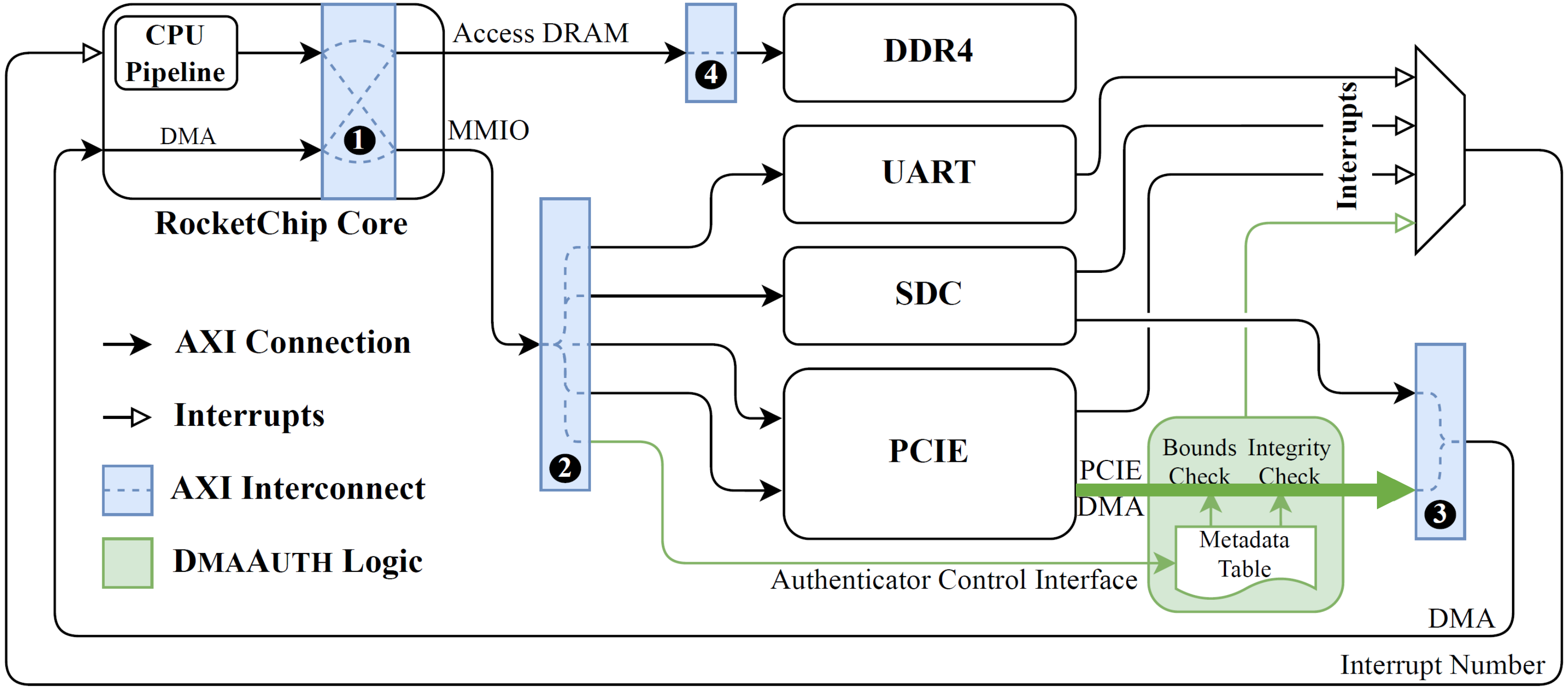
Implementation

- The hardware authenticator is put **between PCIe bus and DRAM**.
- The CPU uses **MMIO** to control the Authenticator on the bus.



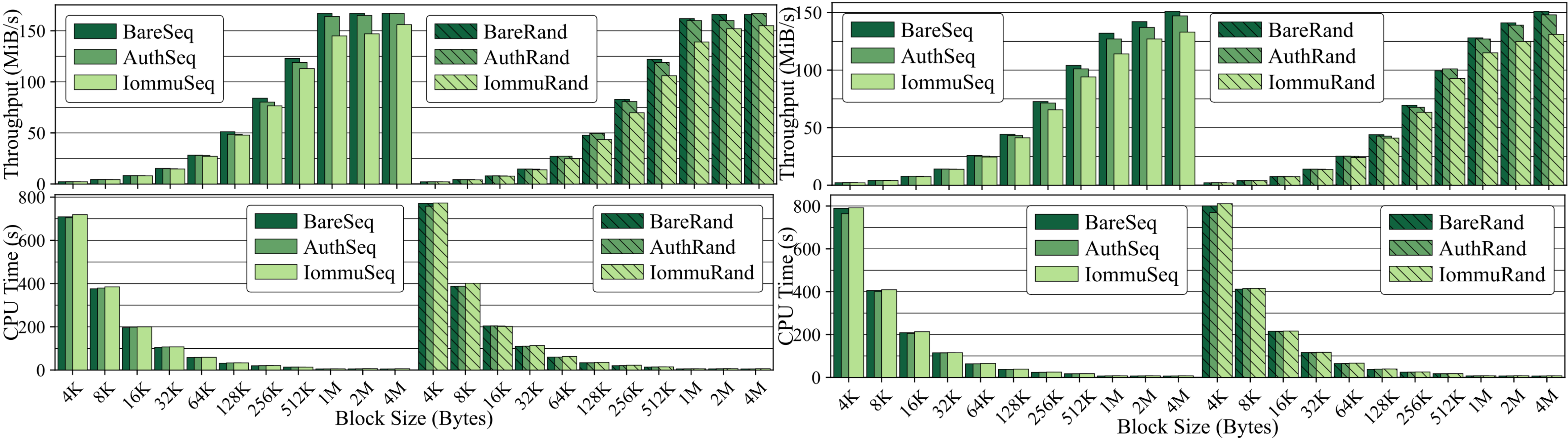
Implementation

- The hardware authenticator is put **between PCIe bus and DRAM**.
- The CPU uses **MMIO** to control the Authenticator on the bus.
- The Authenticator **intercepts** and **authenticates** the DMA transactions.



Evaluation

- DMAAUTH brings 1.0% throughput overhead, 1.8% CPU time overhead
- Significantly faster than IOMMU



Takeaways

- DMAAUTH hardware-software co-design
  - **Defeats** DMA attacks effectively
  - Is significantly **faster** than IOMMU
  - Is **transparent** to existing hardware
  - Requires **zero** driver modification
- Arithmetic Capable Pointer Authentication
  - Supports **pointer arithmetic**
  - Ensures **pointer integrity**
- PCIe-capable research framework
  - Is equipped with **high-performance IOMMU**
  - Provides **customizable** research platform

Q & A