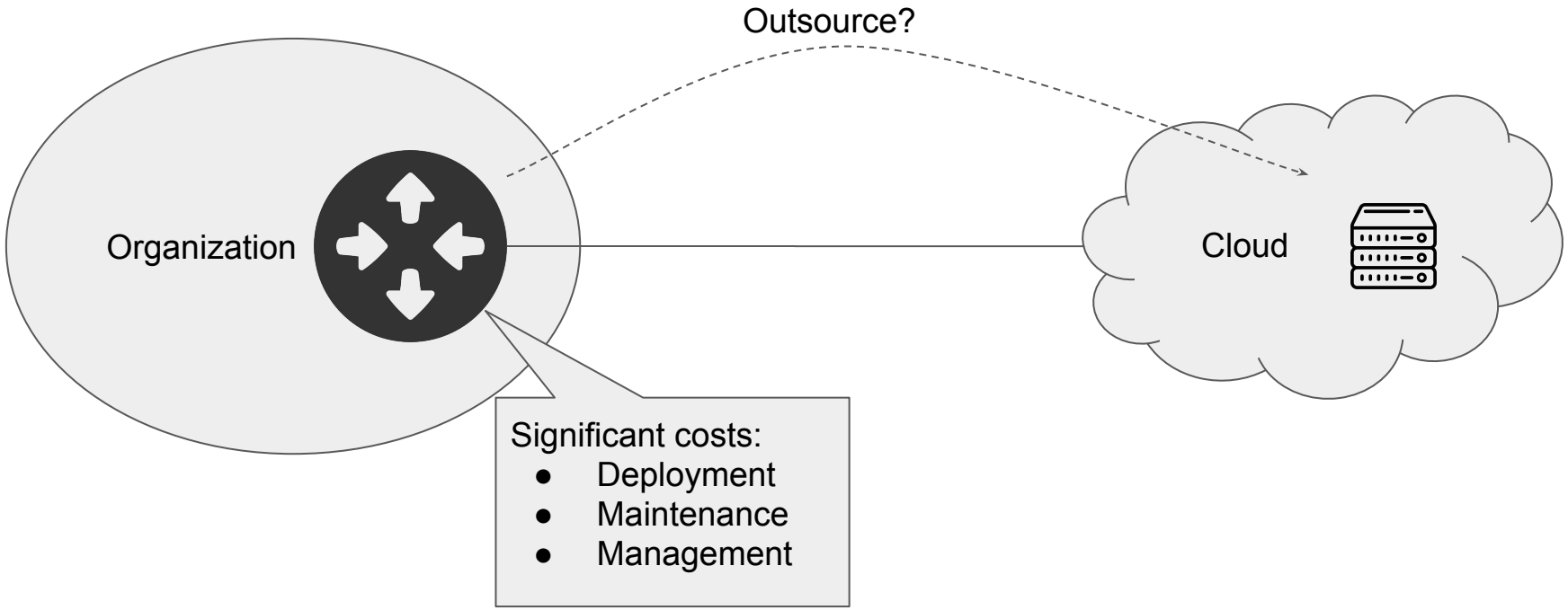
ShieldBox

Secure Middleboxes using Shielded Execution

Bohdan Trach, Alfred Krohmer, Franz Gregor, Sergei Arnautov,
Pramod Bhatotia, Christof Fetzer

ACM SIGCOMM SOSR 2018

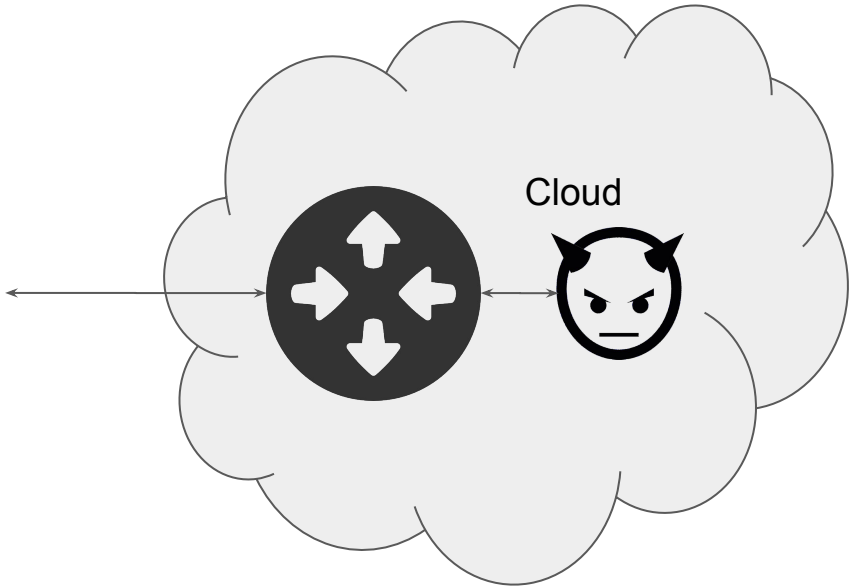
Middleboxes in the Cloud



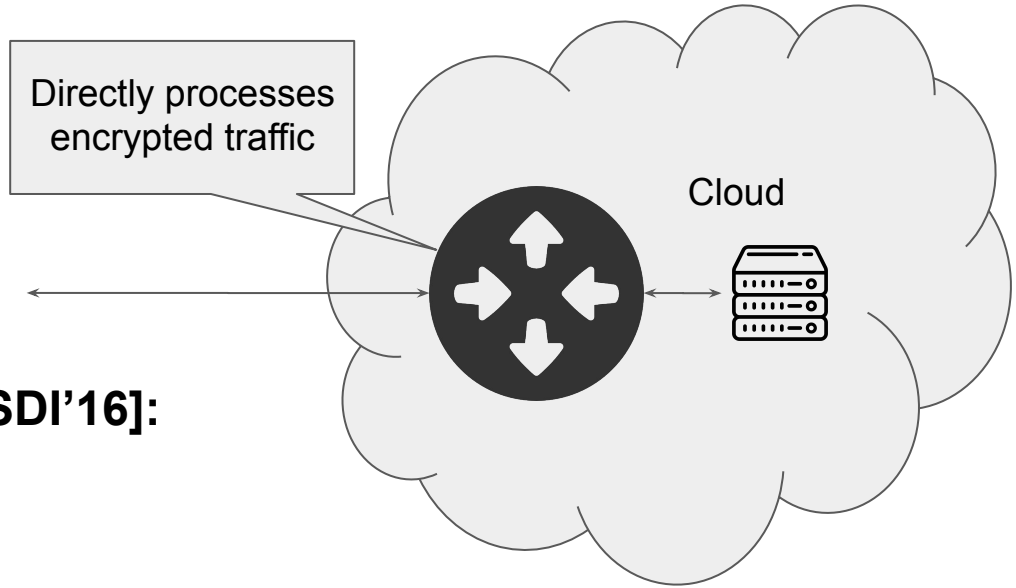
Security Issues

- Cheap computation resources ☺
- NFV advances ☺
- Low trust environment ☹

  - Observe private data
  - Extract encryption keys
  - Learn configuration



State-Of-the-Art Systems



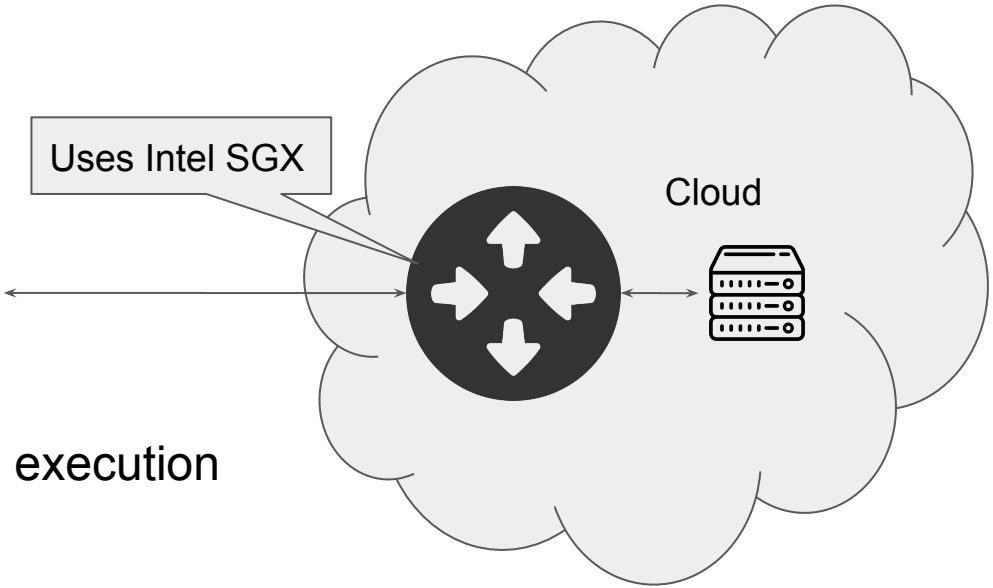
Blindbox [SIGCOMM'15], Embark [NSDI'16]:

- High performance overhead ☹
- Offer limited functionality ☹

Problem Statement

How to securely outsource middleboxes to the untrusted cloud without sacrificing performance while supporting a wide range of NFs?

ShieldBox



ShieldBox:

- Middlebox framework with shielded execution

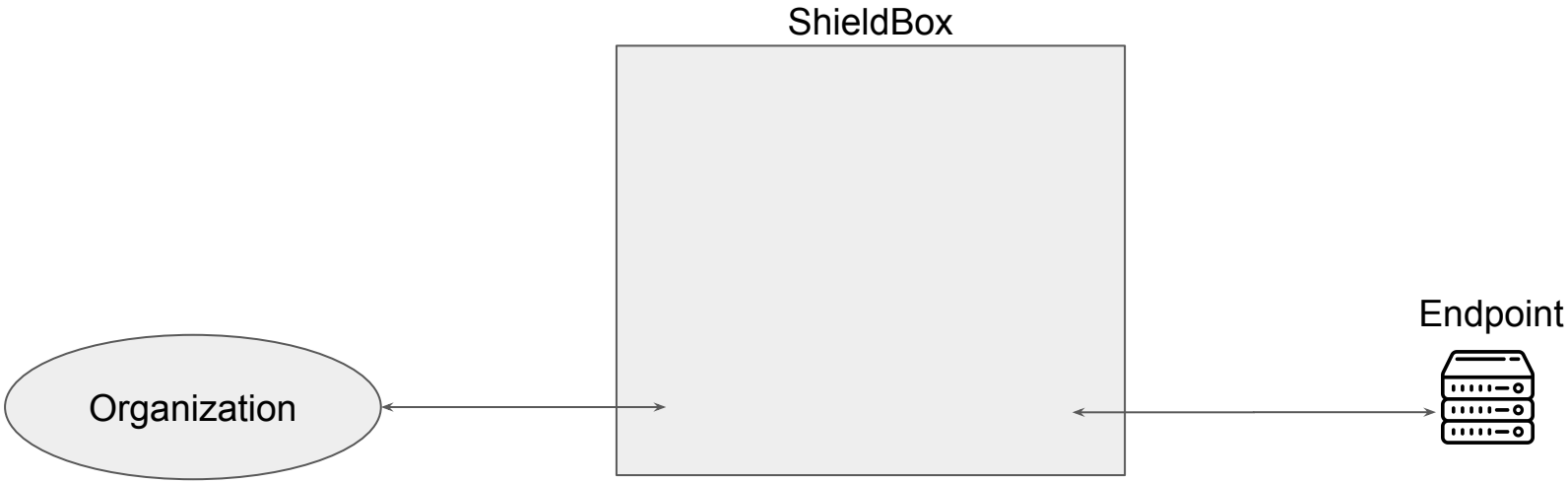
Design Goals

- **Security** - strong confidentiality and integrity guarantees
- **Performance** - near-native throughput and latency
- **Generality** - supports a wide range of NFs
- **Transparency** - portable, configurable, and verifiable architecture

Outline

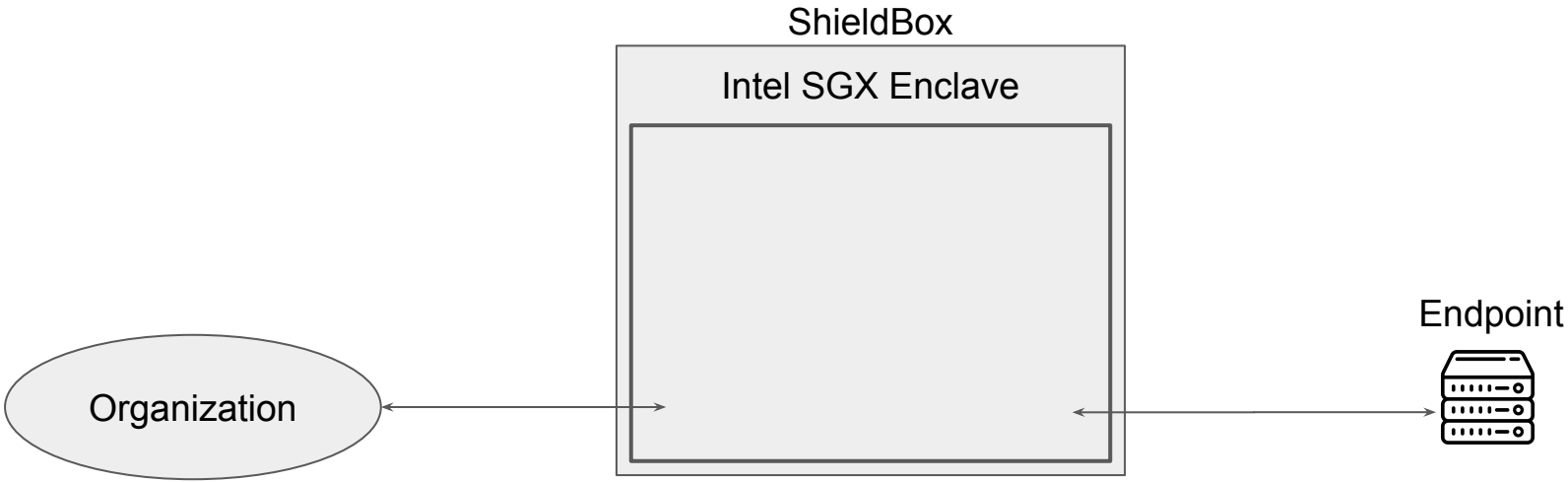
- ~~Motivation~~
- Design
- Evaluation
- Summary

System Overview



	Security	
	Performance	
	Usability	

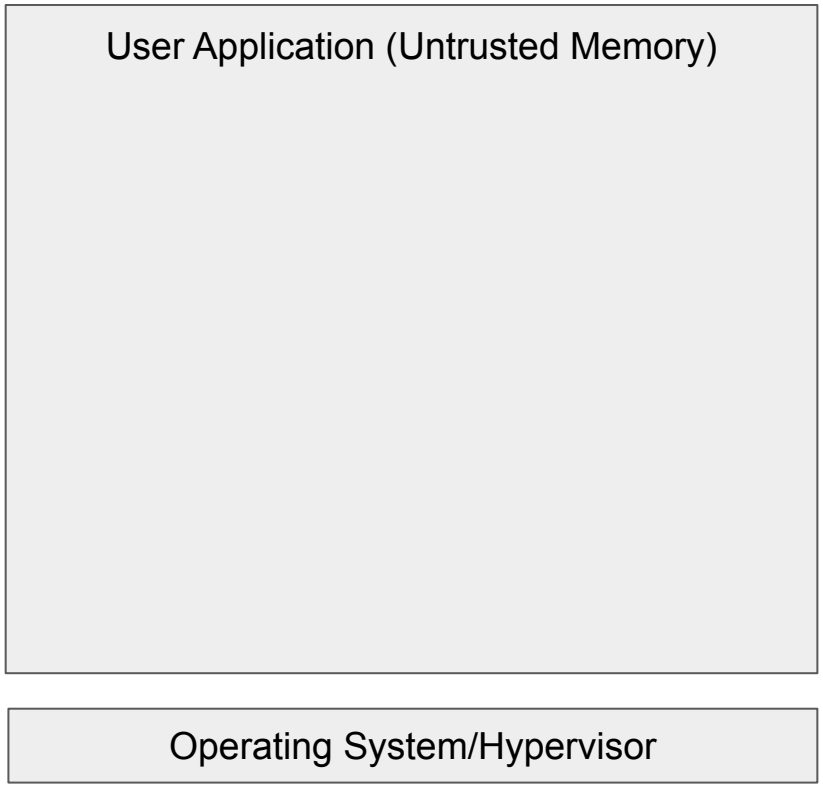
System Overview



✔	Security	Intel SGX
	Performance	
	Usability	

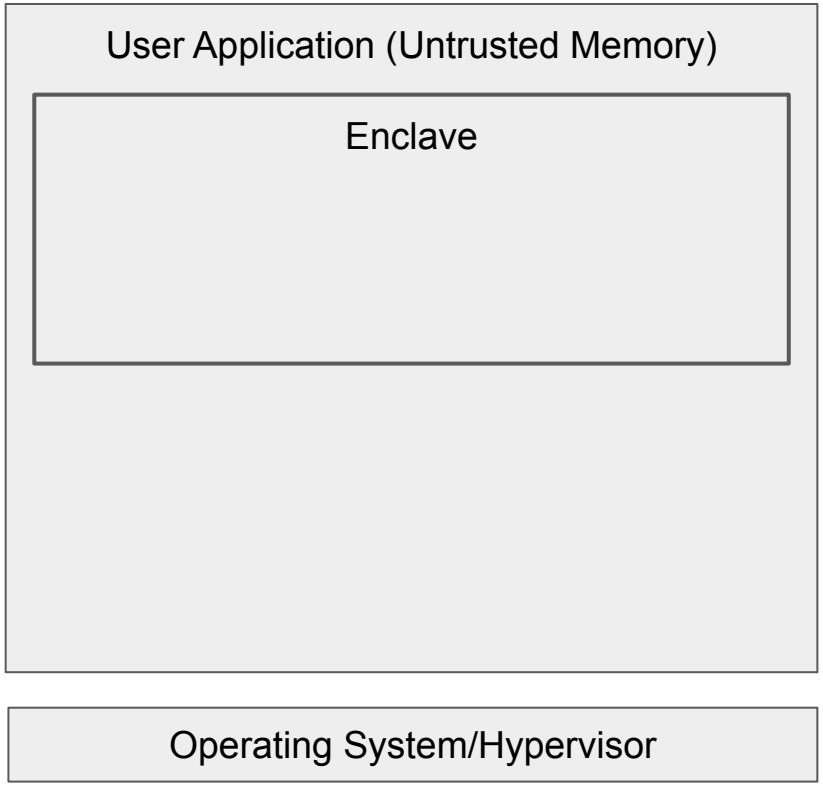
Background: Intel SGX

Intel SGX allows creation and management of **enclaves**.



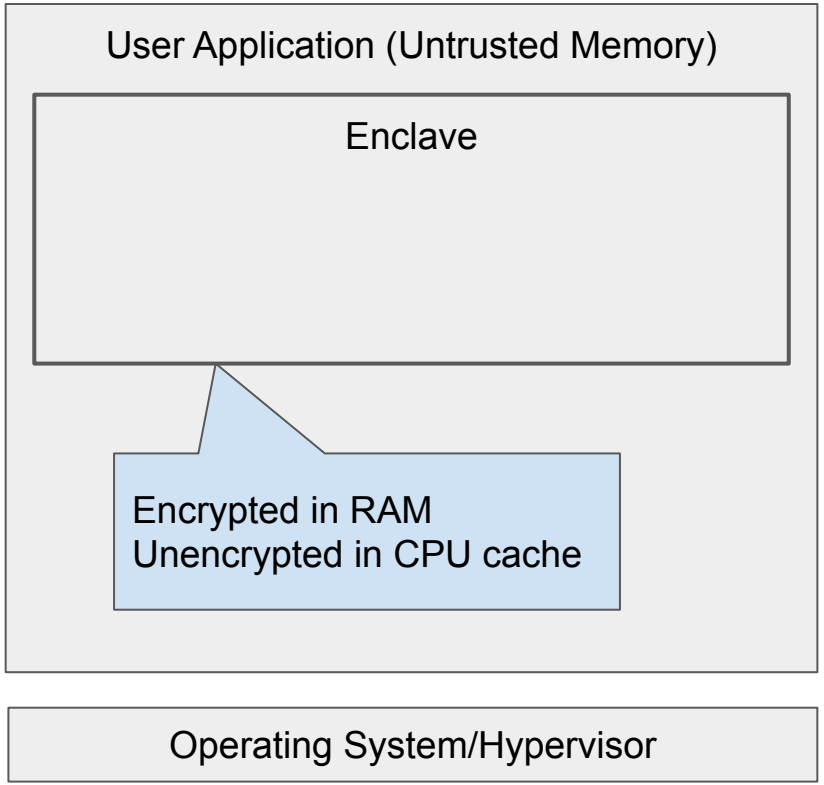
Background: Intel SGX

Intel SGX allows creation and management of **enclaves**.



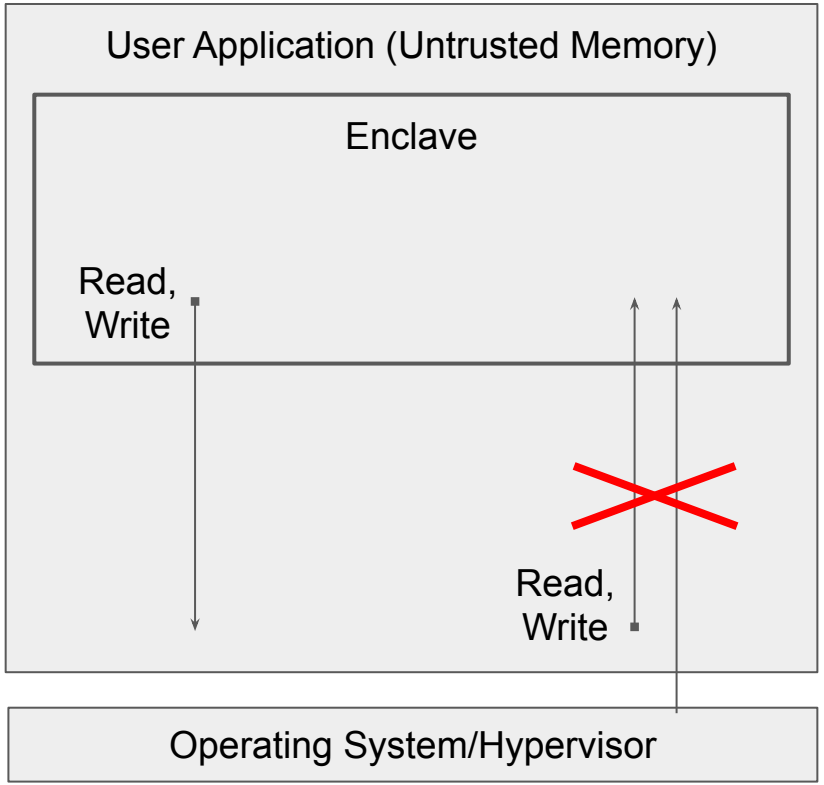
Background: Intel SGX

Intel SGX allows creation and management of **enclaves**.



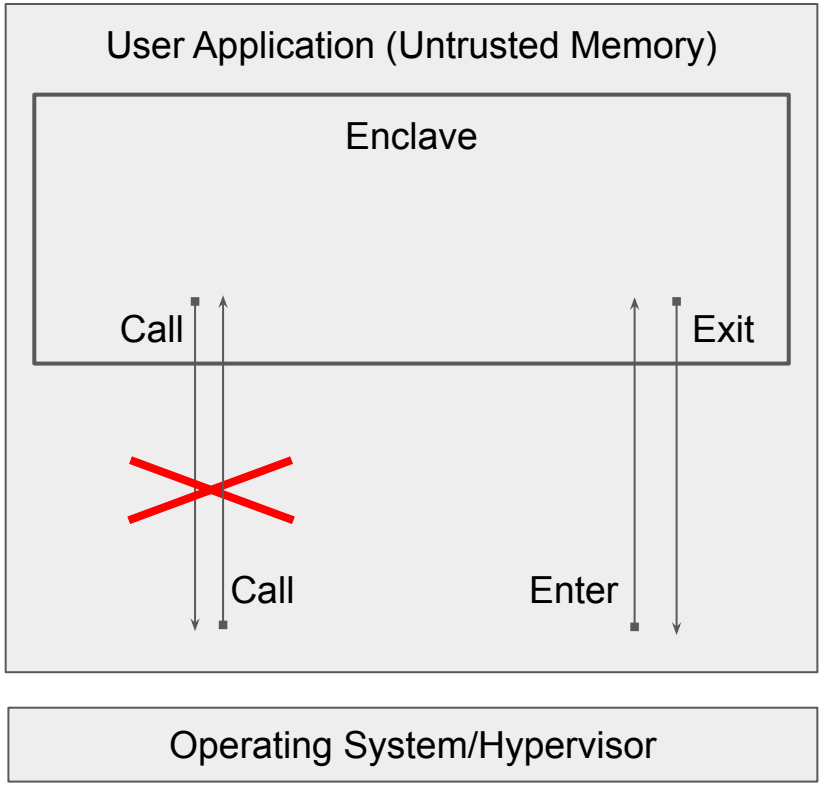
Background: Intel SGX

Intel SGX allows creation and management of **enclaves**.



Background: Intel SGX

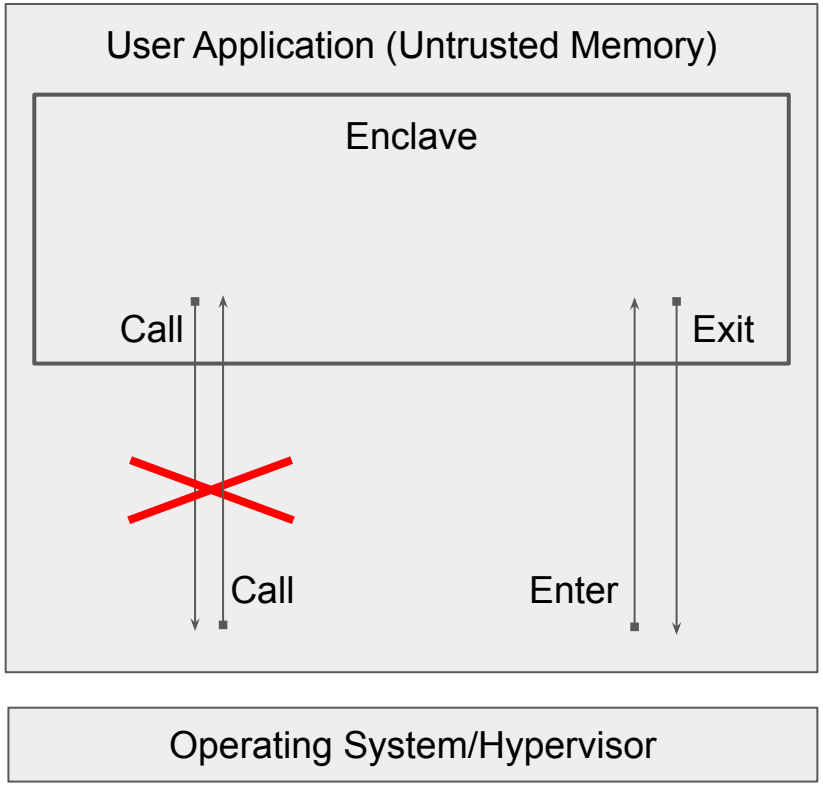
Intel SGX allows creation and management of **enclaves**.



Background: Intel SGX

Intel SGX allows creation and management of **enclaves**.

- Restrictions on allowed instructions:
  - `syscall`
  - `rdtsc`

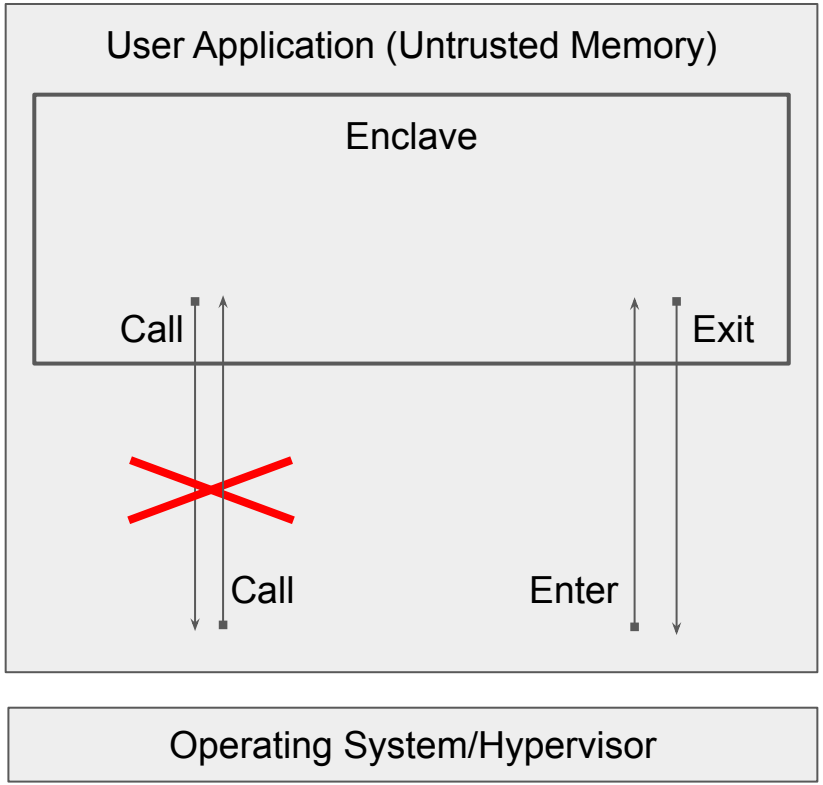


Background: Intel SGX

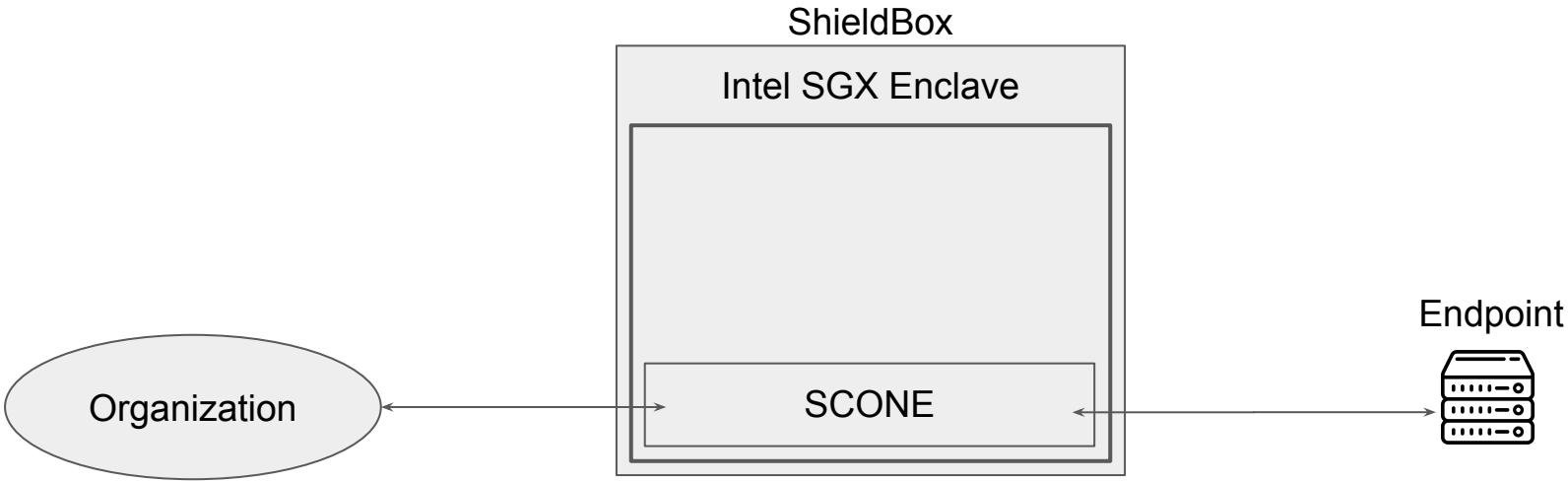
Intel SGX allows creation and management of **enclaves**.

- Restrictions on allowed instructions:
  - `syscall`
  - `rdtsc`

- High overheads for:
  - Secure memory paging
  - Enclave entry/exit

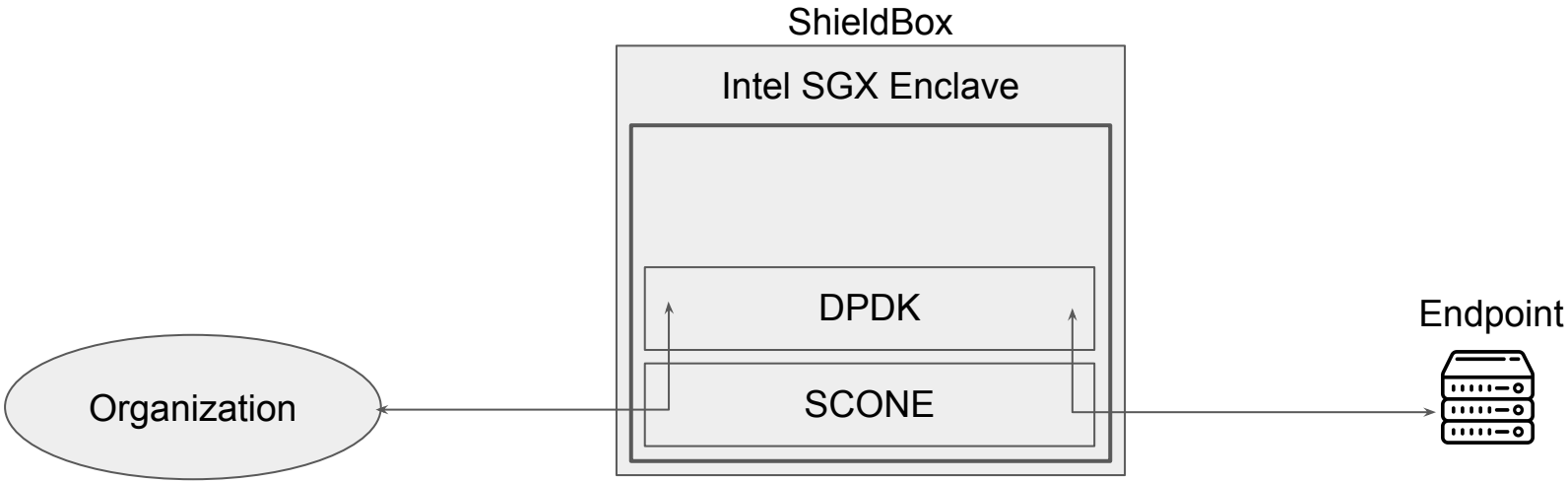


System Overview



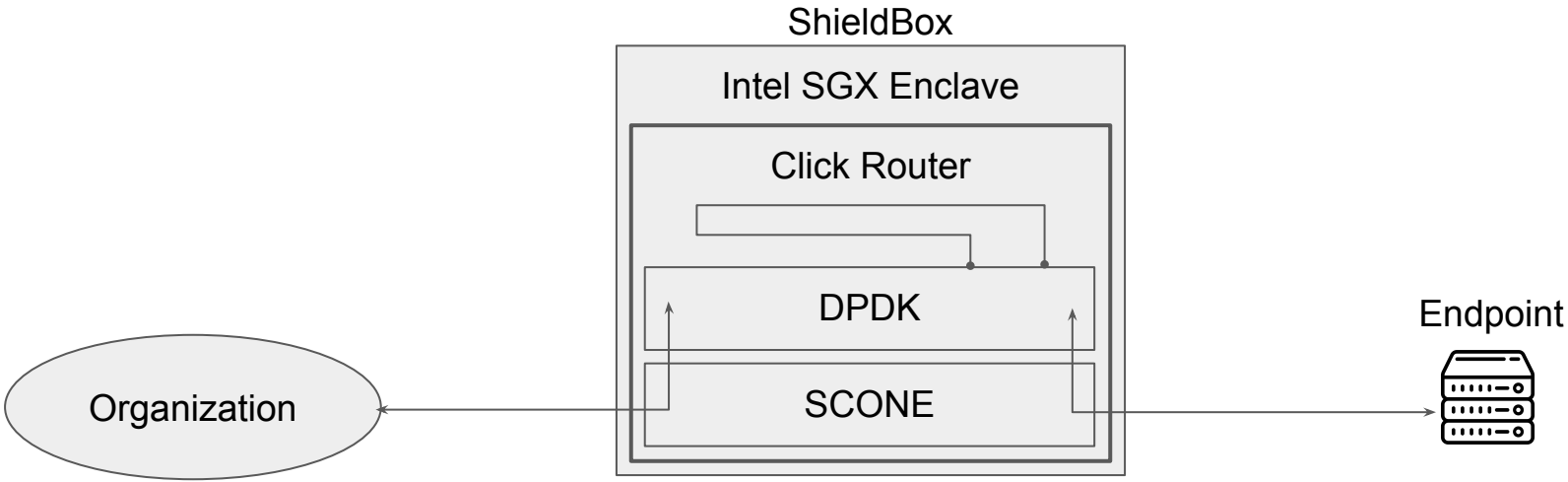
✔	Security	Intel SGX & SCONE [OSDI'16]
	Performance	
	Usability	

System Overview



✔	Security	Intel SGX & SCONE [OSDI'16]
✔	Performance	DPDK
	Usability	

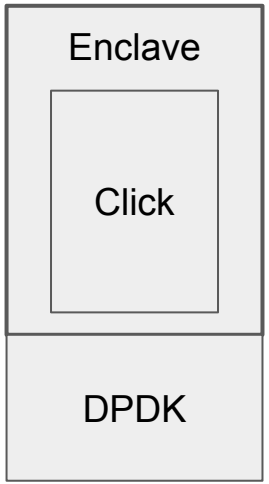
System Overview



✔	Security	Intel SGX & SCONE [OSDI'16]
✔	Performance	DPDK
✔	Usability	Click Router [TOCS'00]

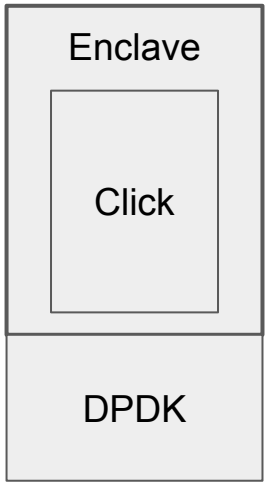
Partitioning ShieldBox

Partitioning ShieldBox

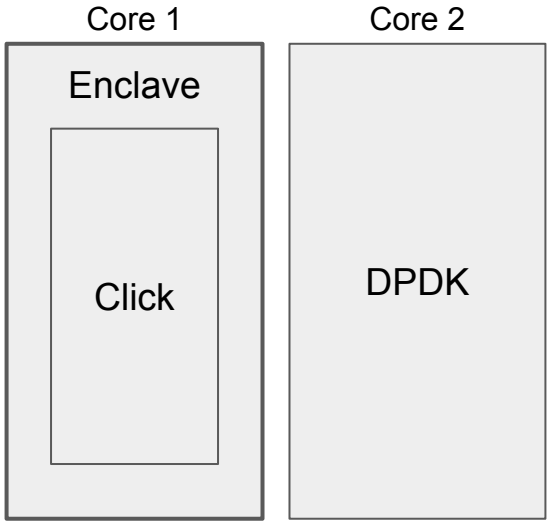


1. DPDK outside - ecalls
   - High overhead ☹
   - Reengineering ☹
   - Reduced TCB ☺

Partitioning ShieldBox

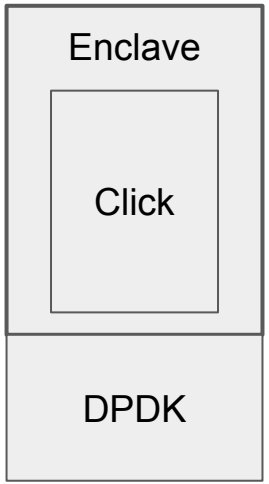


1. DPDK outside - ecalls
   - High overhead ☹
   - Reengineering ☹
   - Reduced TCB ☺

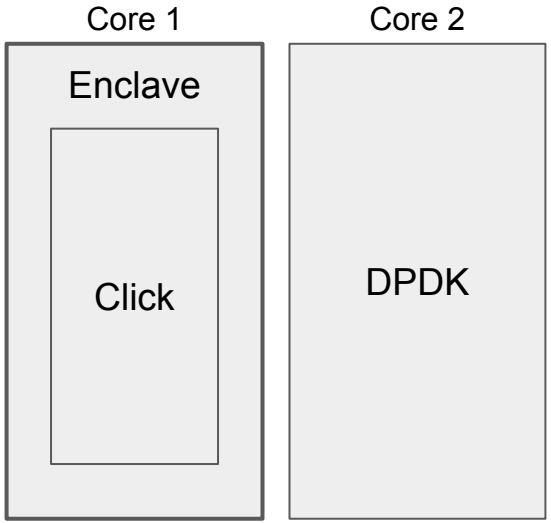


2. DPDK outside - sibling core
   - Overhead in some cases ☹
   - Reengineering ☹
   - Reduced TCB ☺

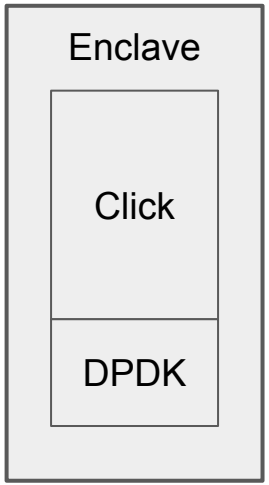
Partitioning ShieldBox



1. DPDK outside - ecalls
   - High overhead ☹
   - Reengineering ☹
   - Reduced TCB ☺



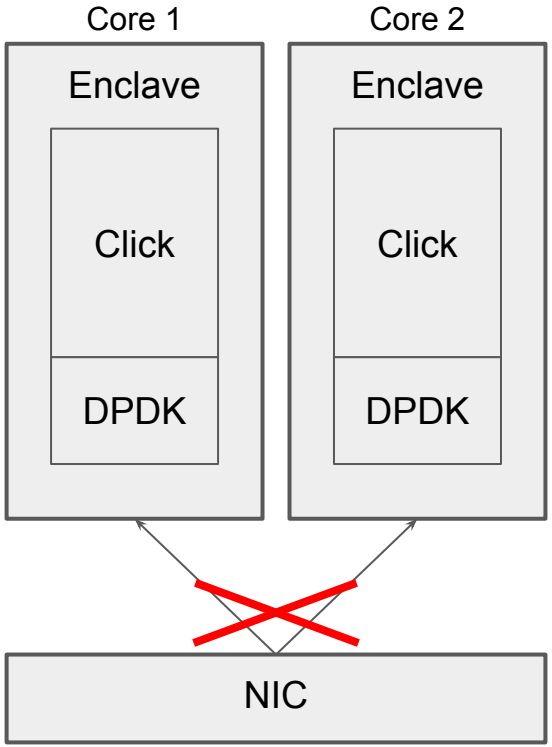
2. DPDK outside - sibling core
   - Overhead in some cases ☹
   - Reengineering ☹
   - Reduced TCB ☺



3. DPDK inside enclave
   - Low overhead ☺
   - No reengineering ☺
   - Increased TCB ☹

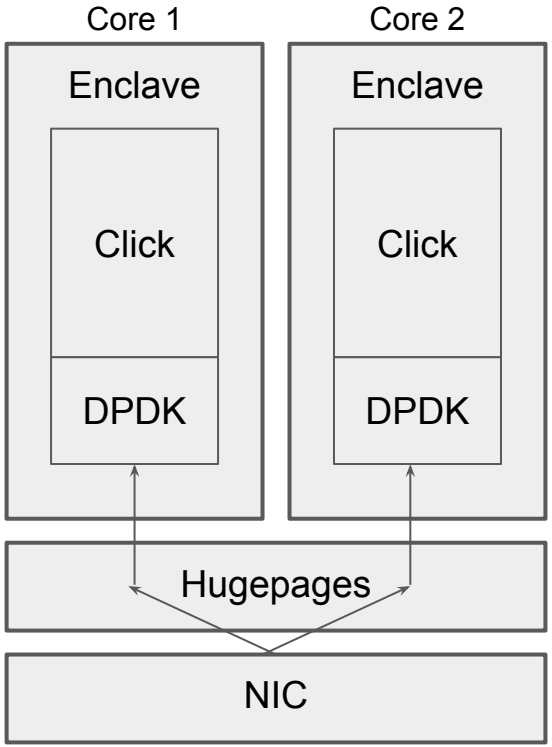
Partitioning ShieldBox - DPDK

- NIC can’t deliver packets directly to enclave.



Partitioning ShieldBox - DPDK

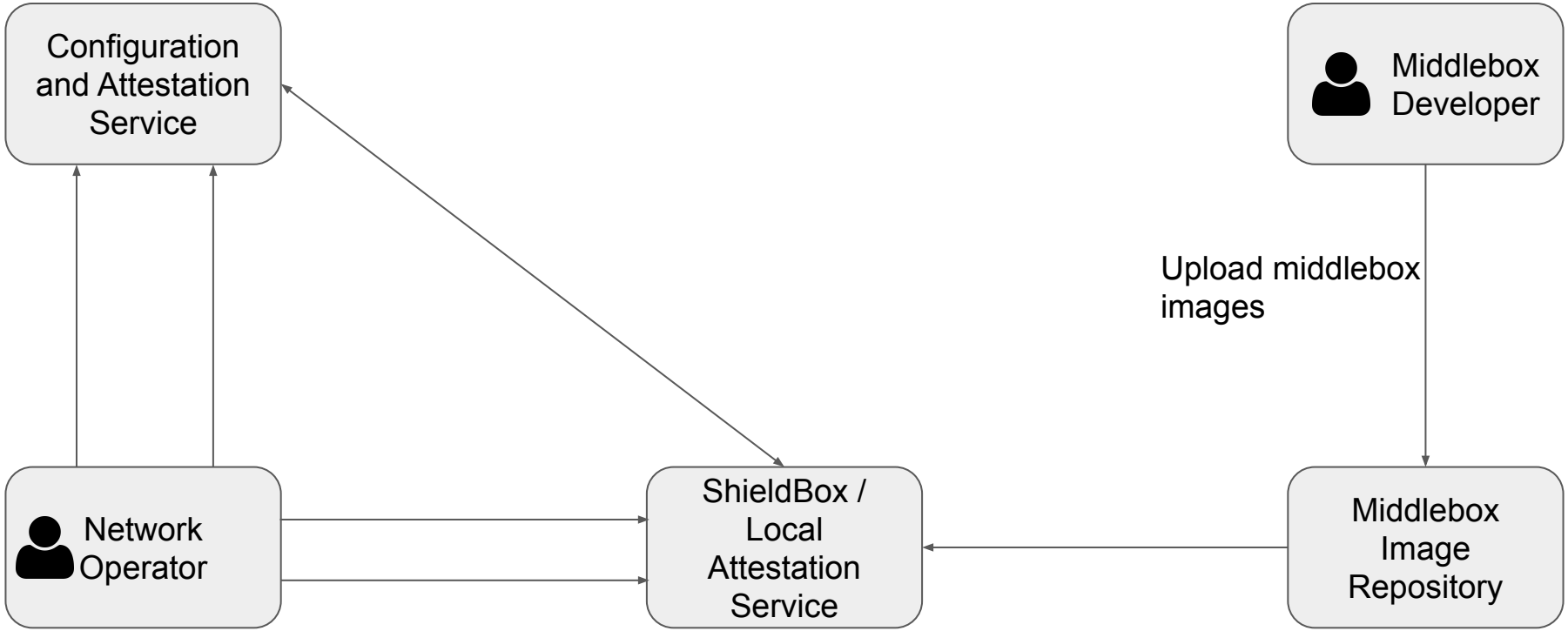
- NIC can’t deliver packets directly to enclave:
  - Allocate hugepage memory outside
  - Packets and mbufs delivered to hugepages



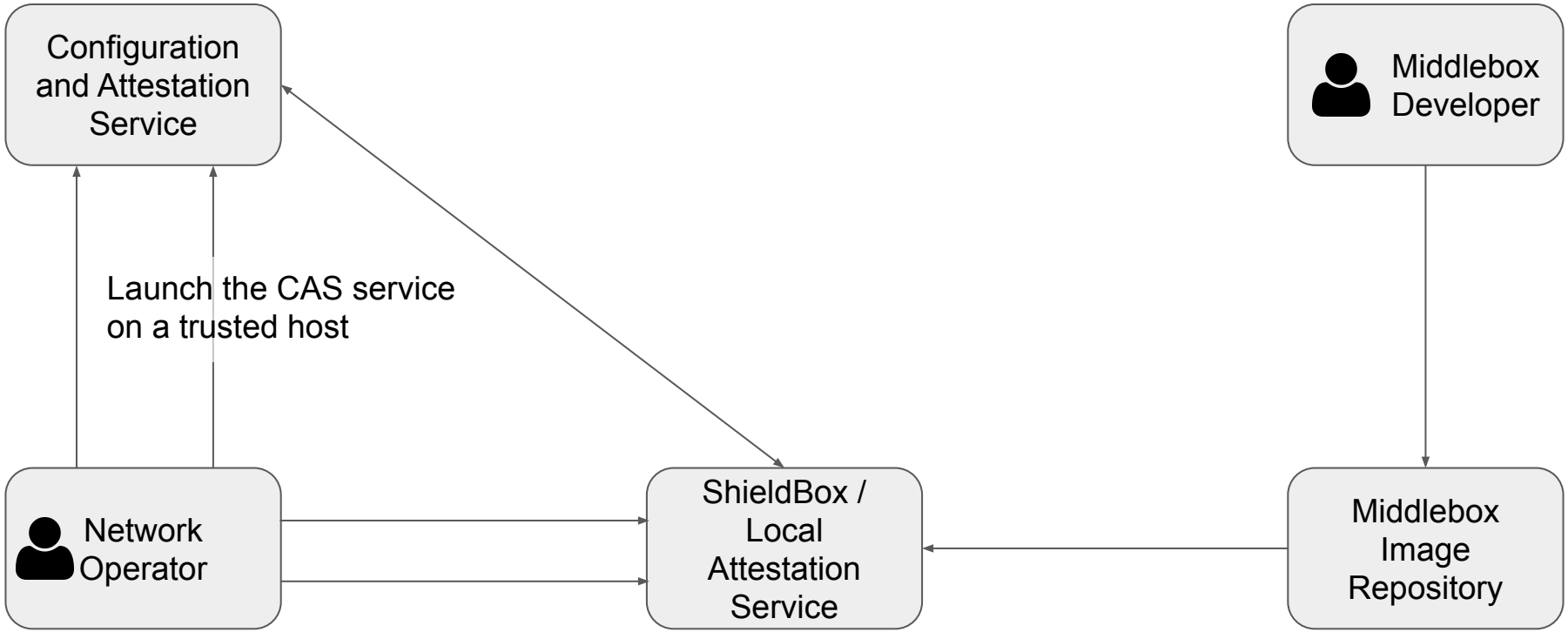
ShieldBox Features

- **Security**
  - Iago Attack Protection
  - New Elements
  - Remote Attestation and Configuration System
- **Performance**
  - On-NIC Time Source
  - Optimizations over standard Click
- **Features**
  - Middlebox State Persistence
  - Network Function Chaining
  - New Elements

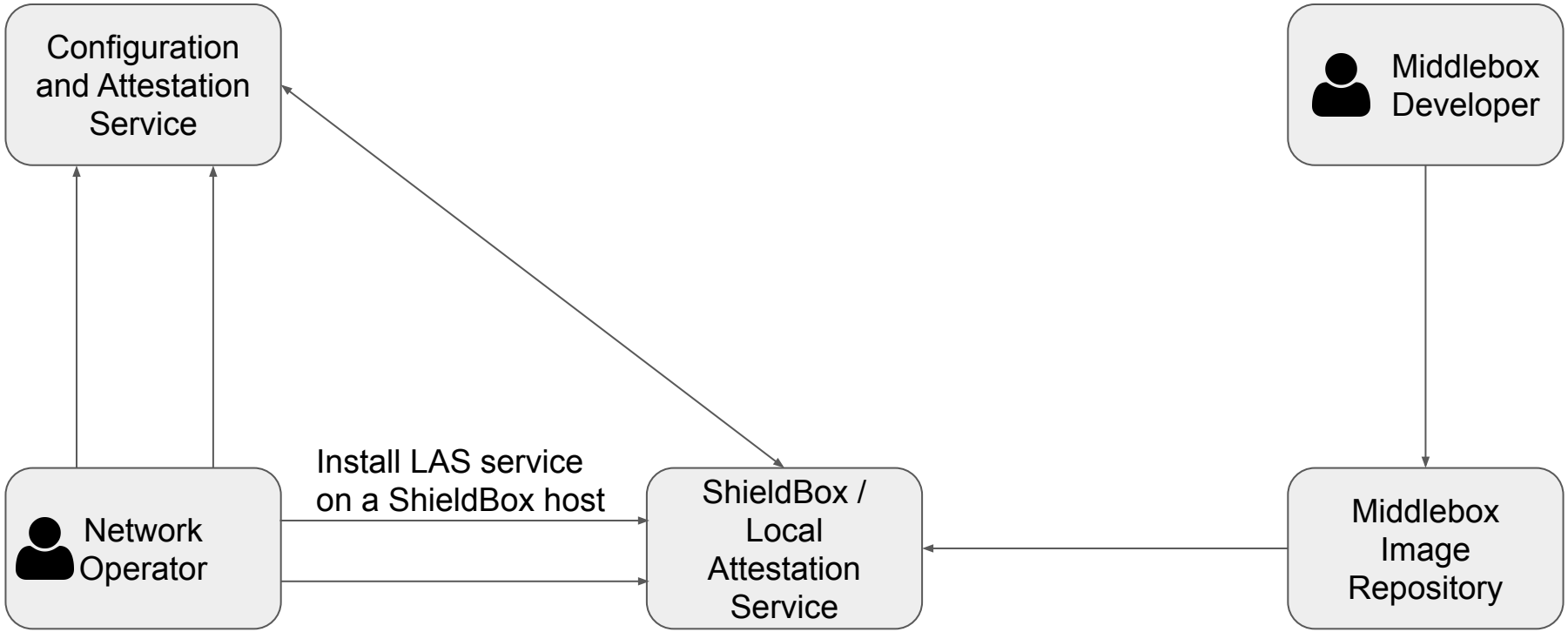
System Workflow with Remote Attestation



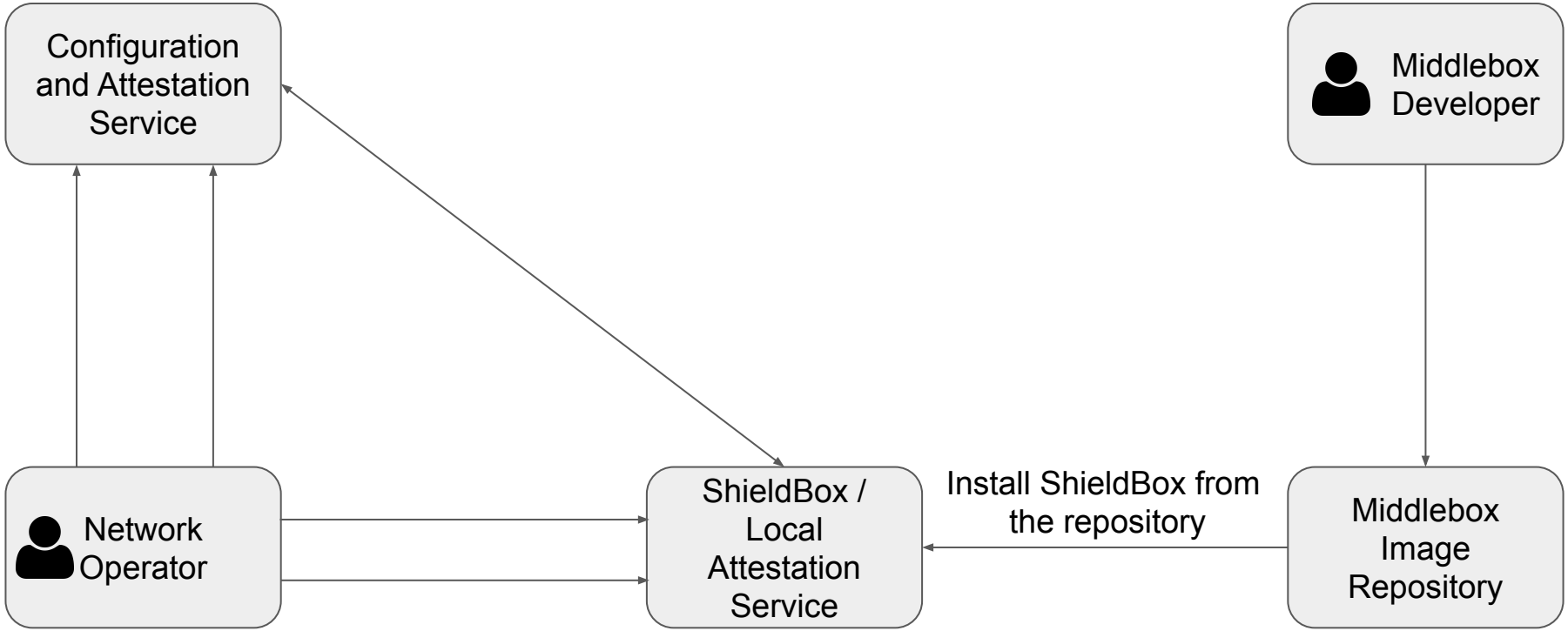
System Workflow with Remote Attestation



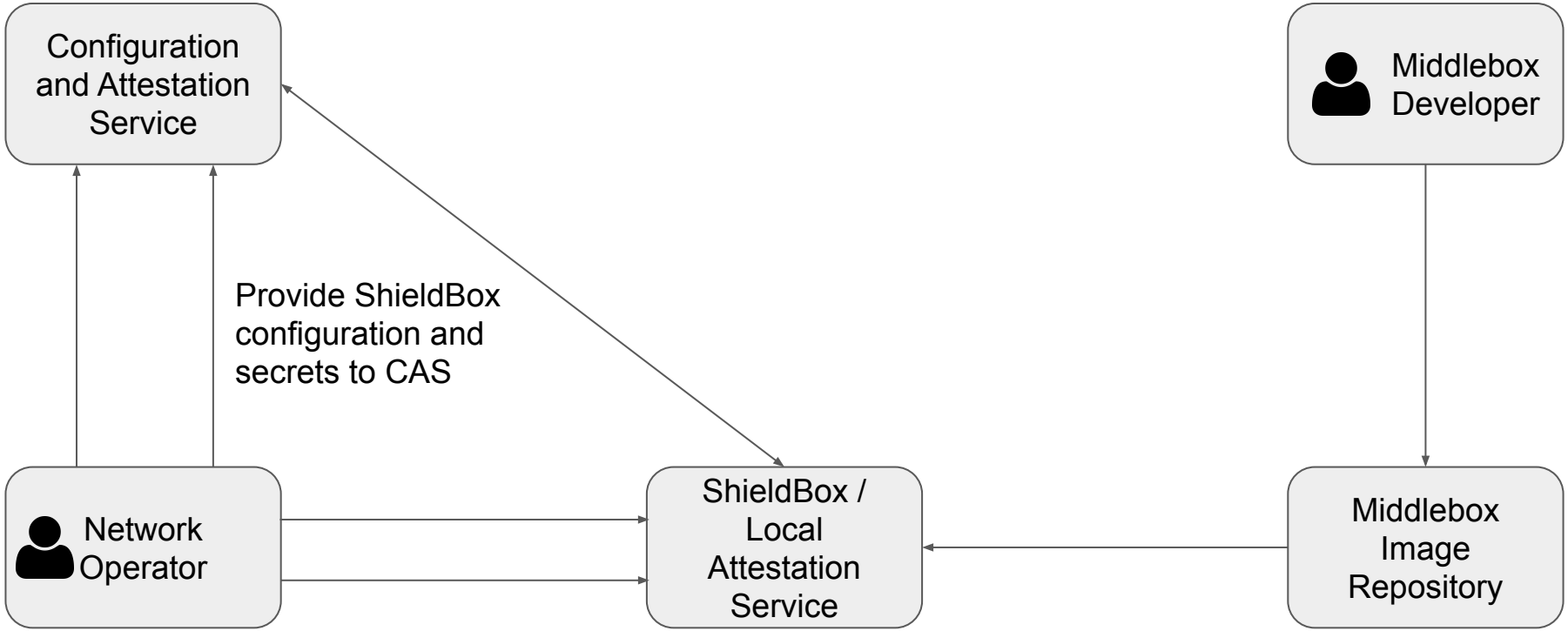
System Workflow with Remote Attestation



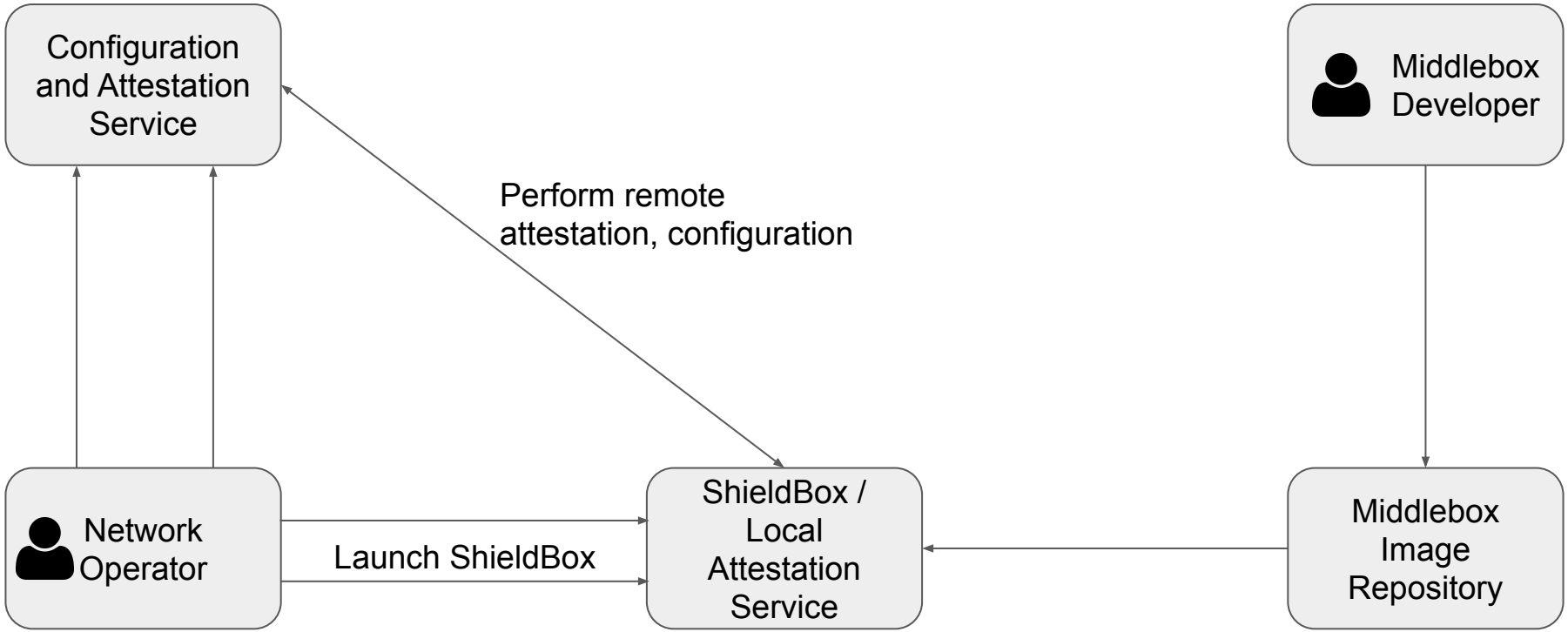
System Workflow with Remote Attestation



System Workflow with Remote Attestation

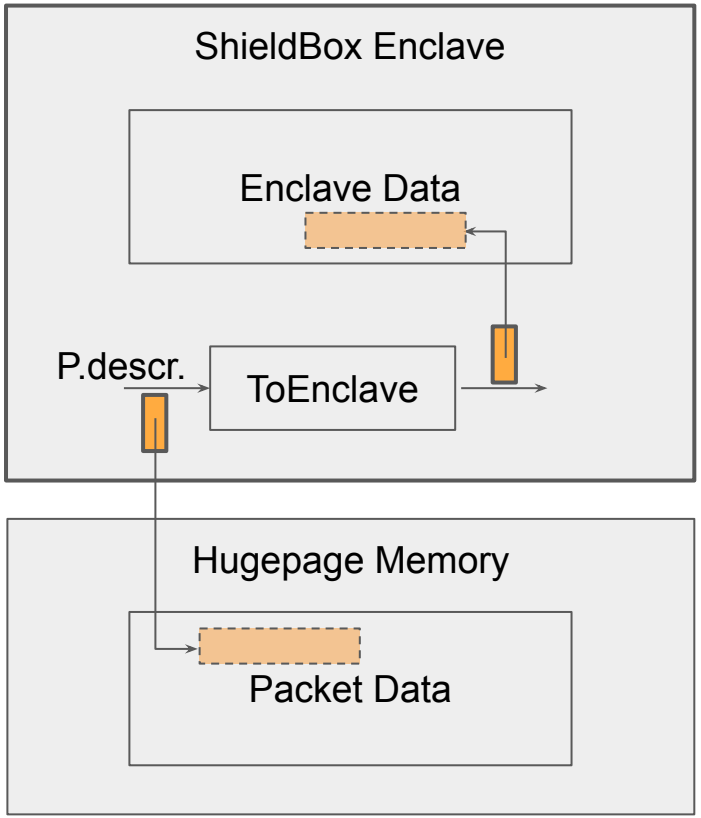


System Workflow with Remote Attestation



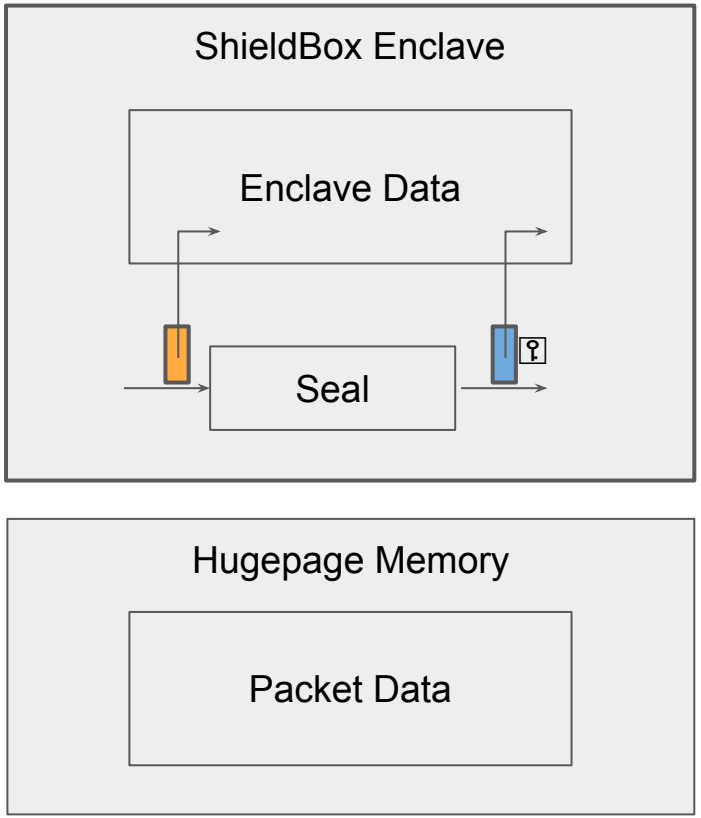
New Elements

- ToEnclave:
  - Copies packet data into enclave



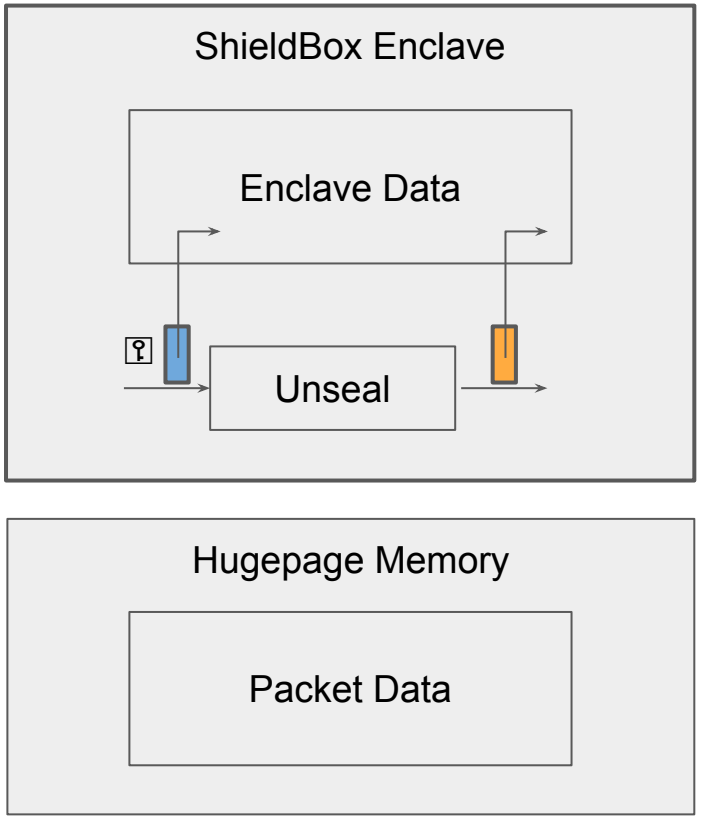
New Elements

- ToEnclave:
  - Copies packet data into enclave
- Seal:
  - Encrypts packet using AES-GCM



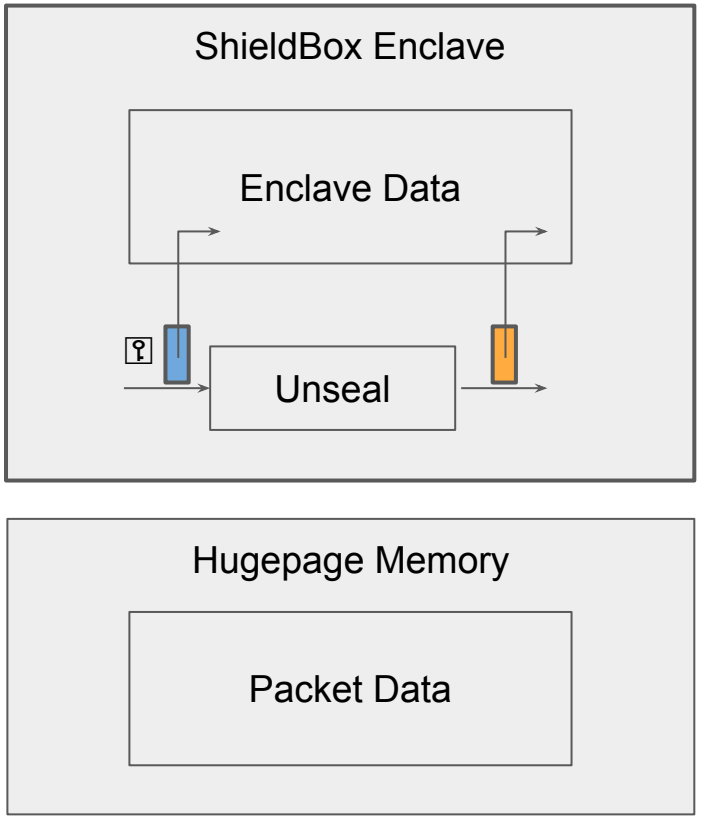
New Elements

- ToEnclave:
  - Copies packet data into enclave
- Seal:
  - Encrypts packet using AES-GCM
- Unseal:
  - Decrypts an AES-GCM encrypted packet



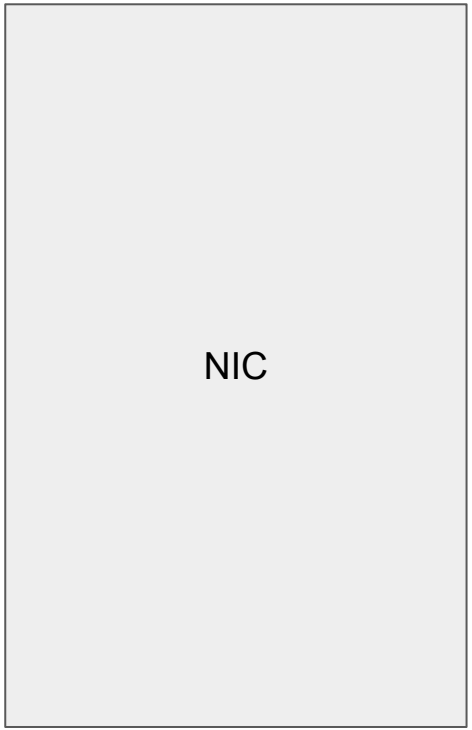
New Elements

- ToEnclave:
  - Copies packet data into enclave
- Seal:
  - Encrypts packet using AES-GCM
- Unseal:
  - Decrypts an AES-GCM encrypted packet
- HyperScan, DPDKRing, StateFile:
  - See paper!

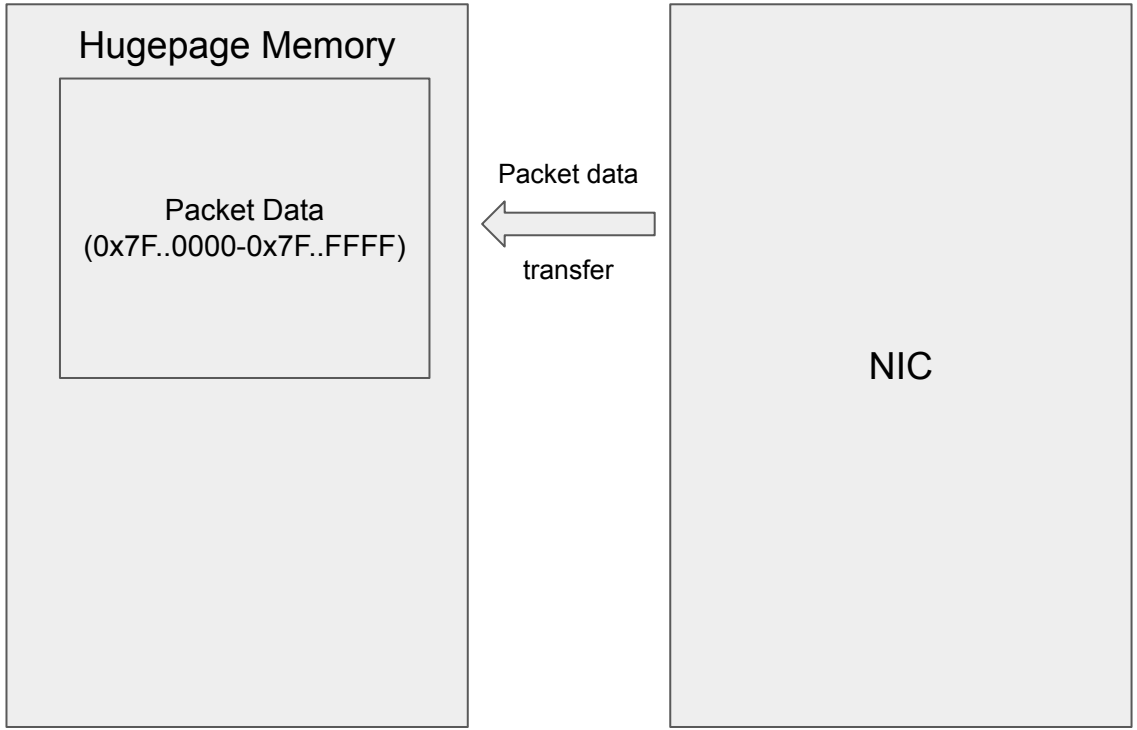


Iago Attack Protection

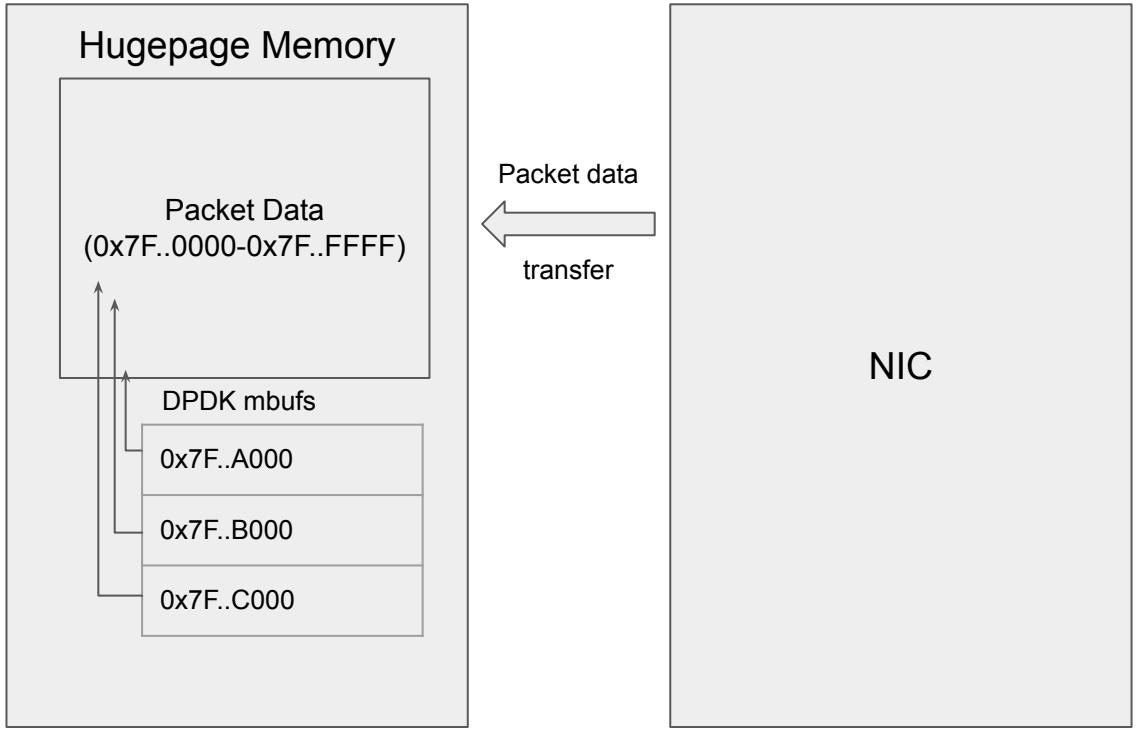
Iago Attack Protection



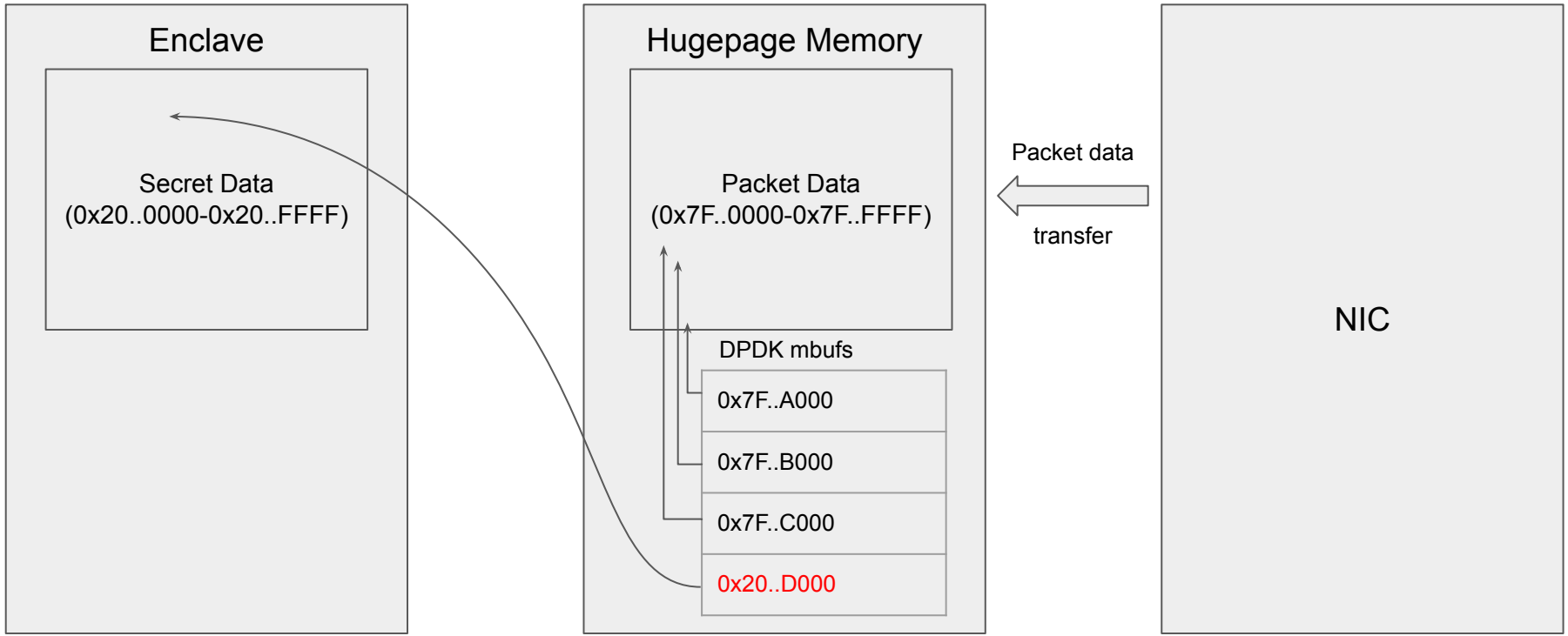
Iago Attack Protection



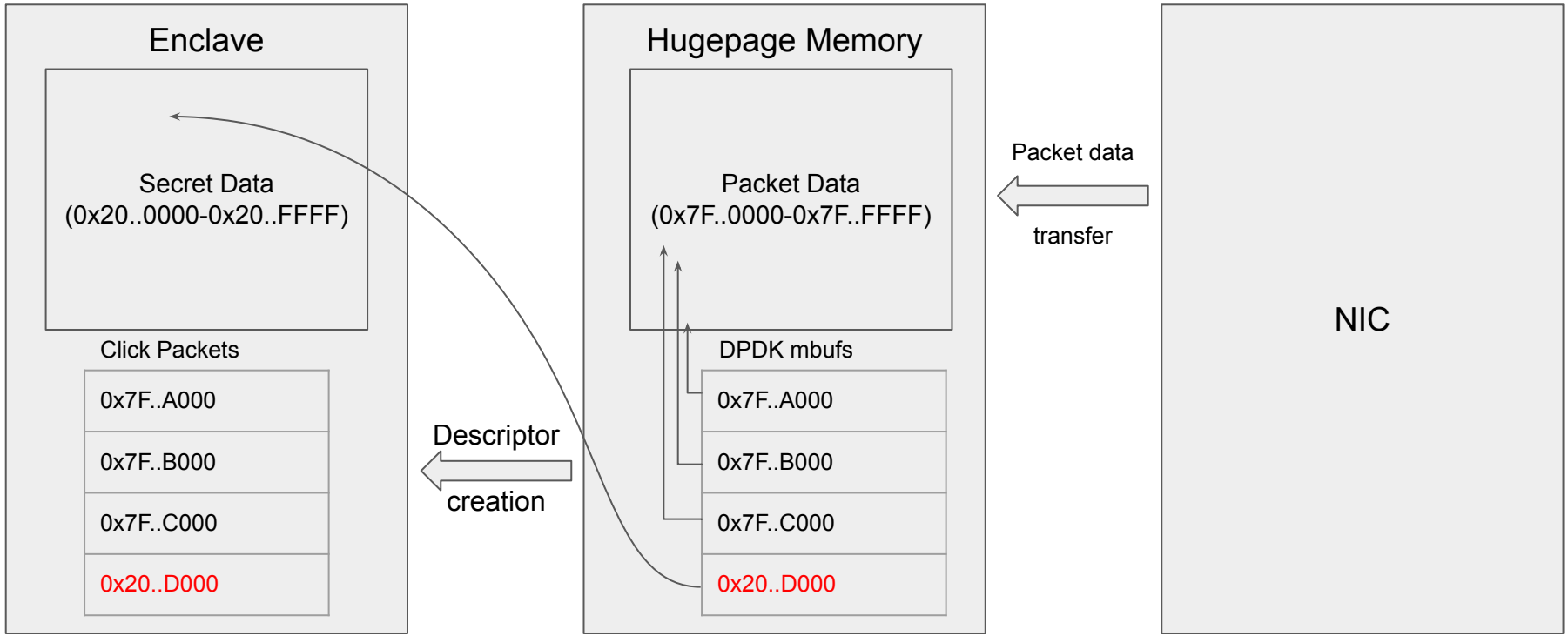
Iago Attack Protection



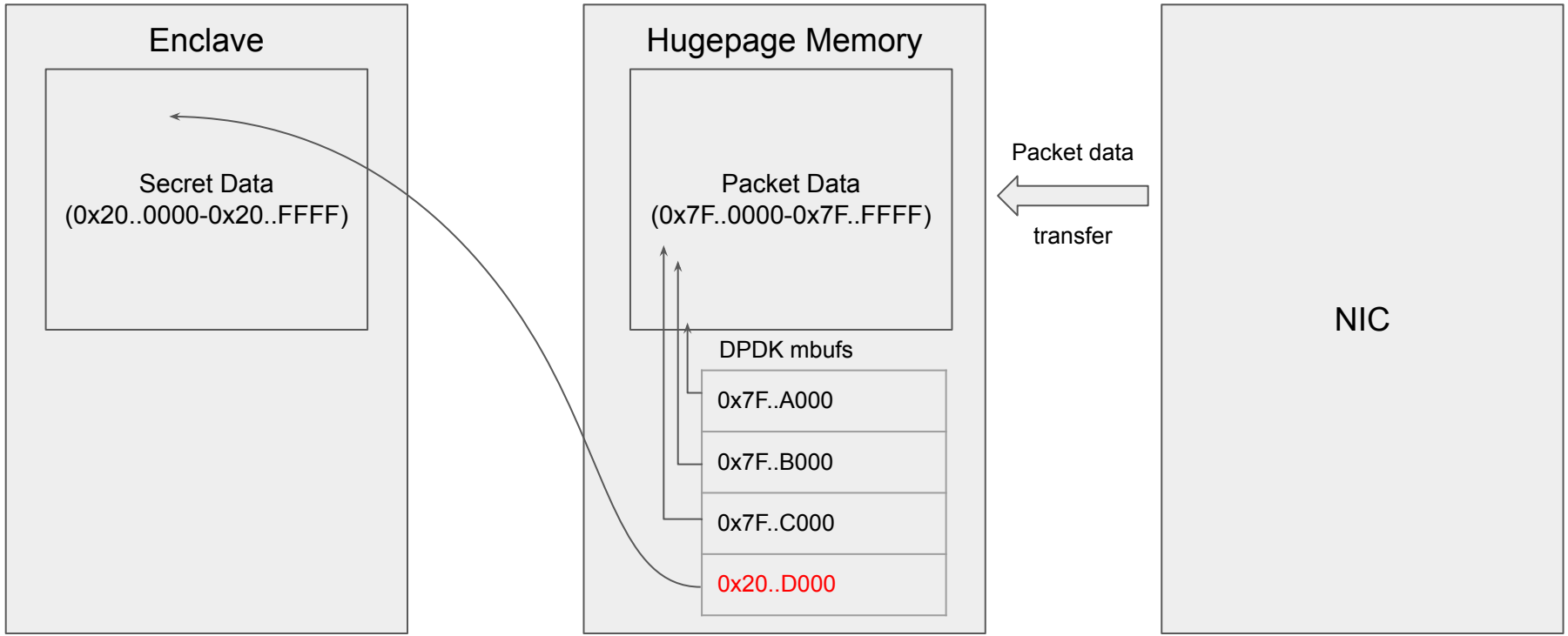
Iago Attack Protection



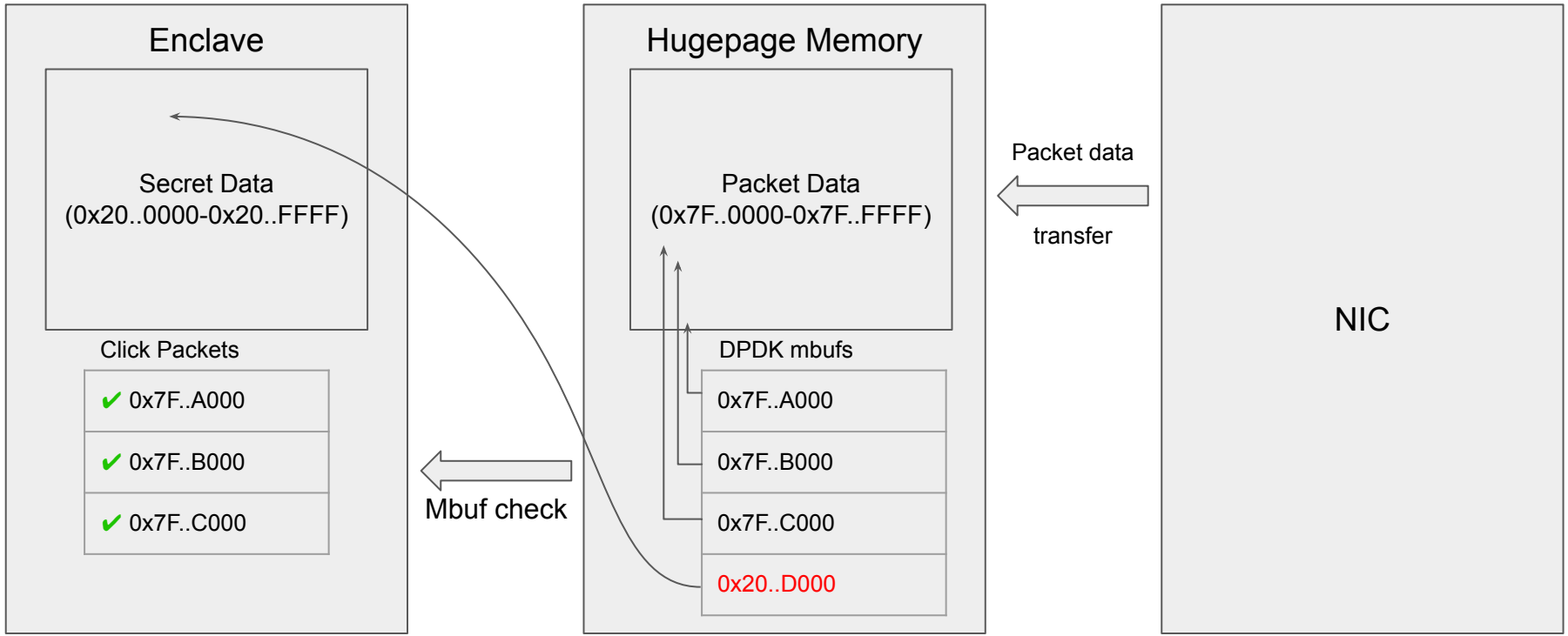
Iago Attack Protection



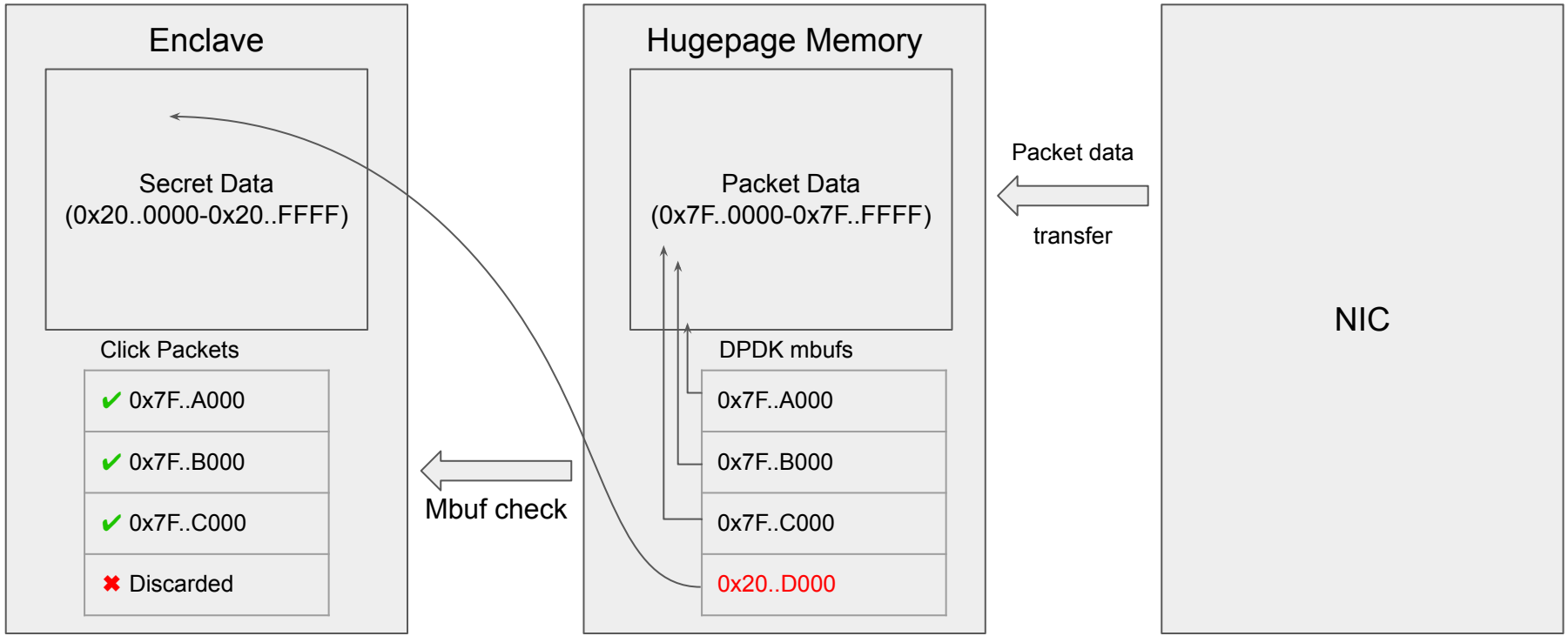
Iago Attack Protection



Iago Attack Protection

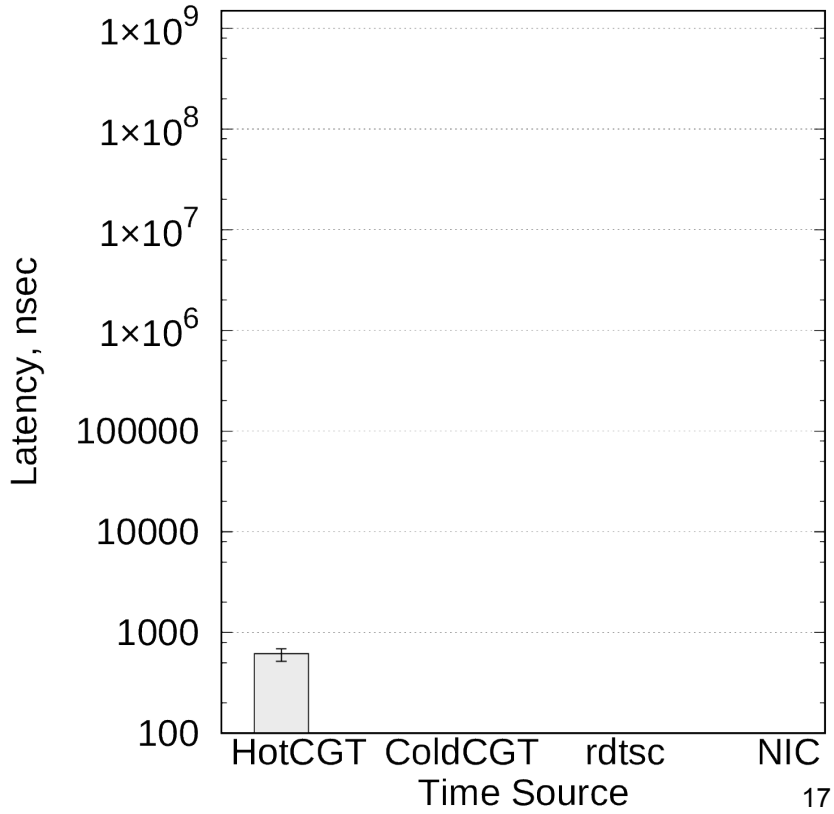


Iago Attack Protection



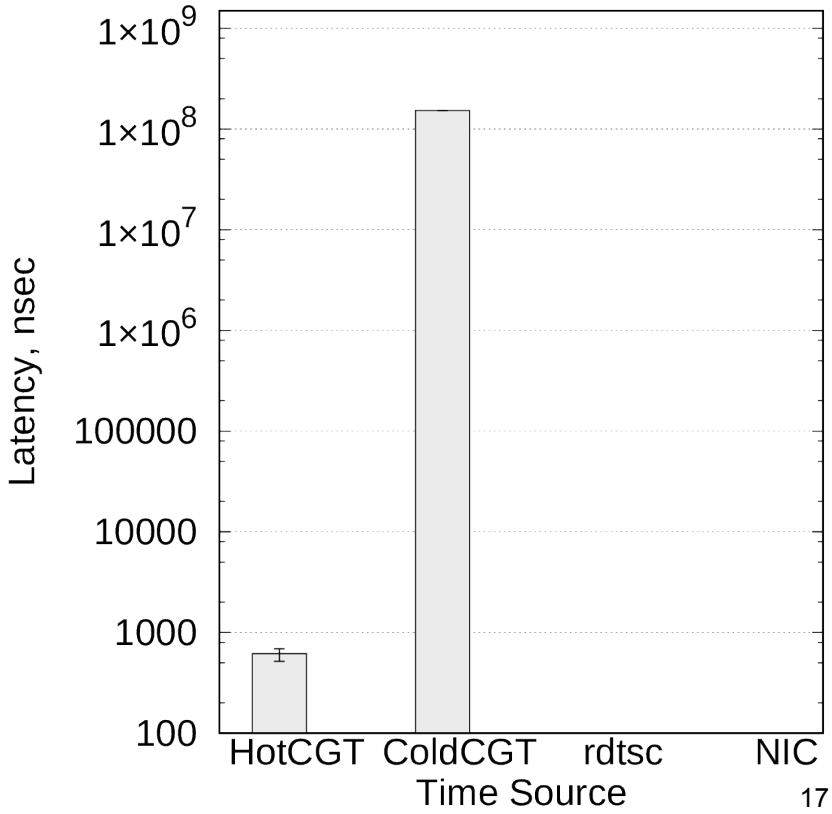
On-NIC Time Source

- `clock_gettime`
  - Hot sthread: reduce performance



On-NIC Time Source

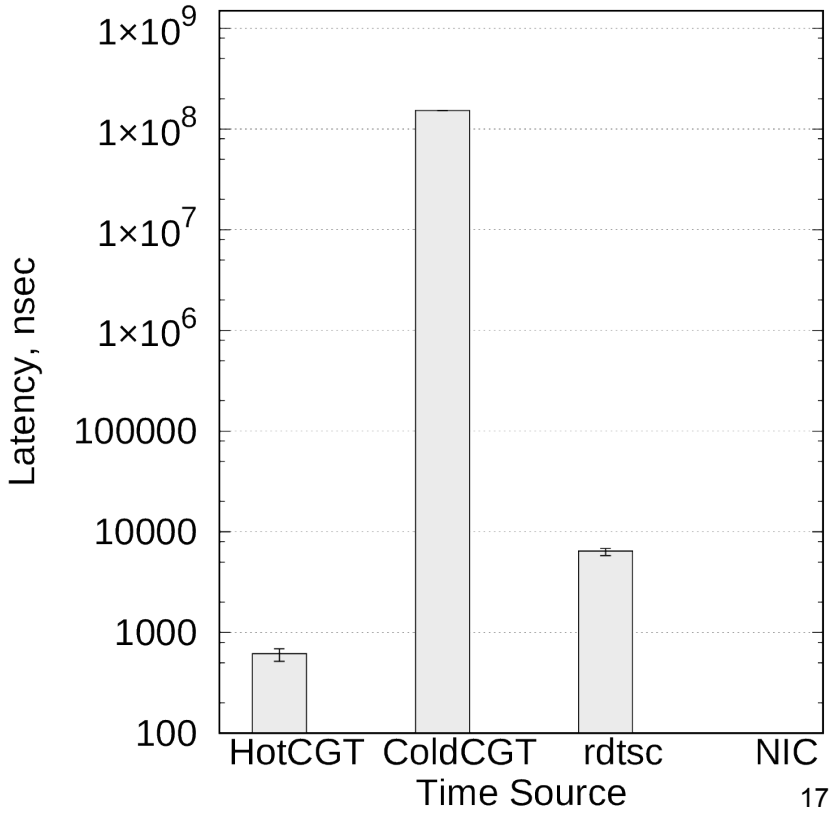
- `clock_gettime`
  - Hot sthread: reduce performance
  - Cold sthread: huge overhead



On-NIC Time Source

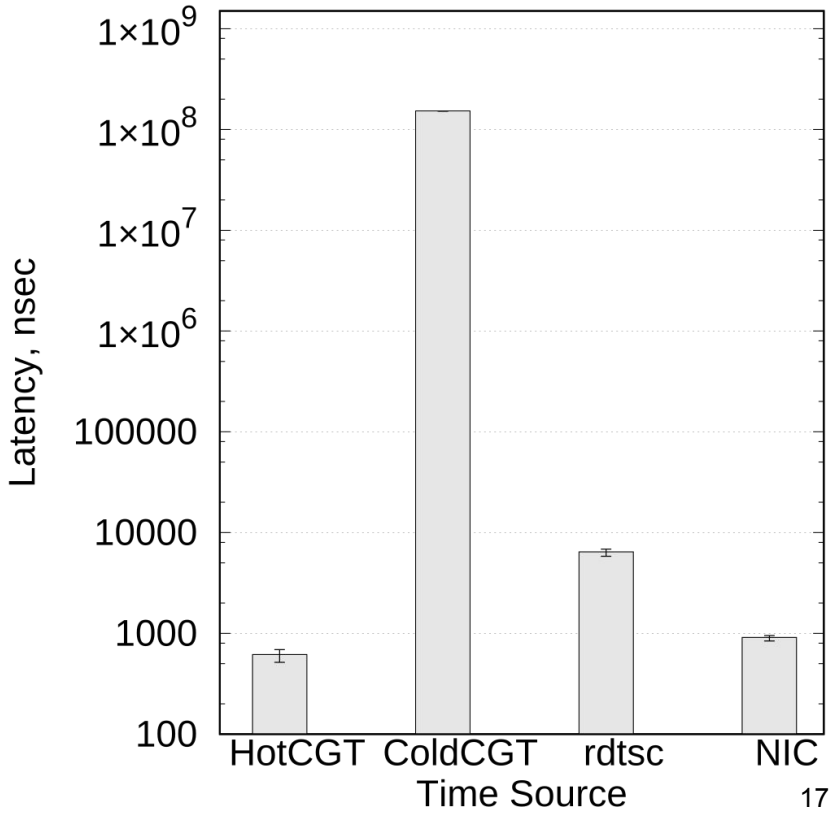
- `clock_gettime`
  - Hot sthread: reduce performance
  - Cold sthread: huge overhead

- `rdtsc`
  - Causes enclave exit
  - Performance loss due to TLB flush



On-NIC Time Source

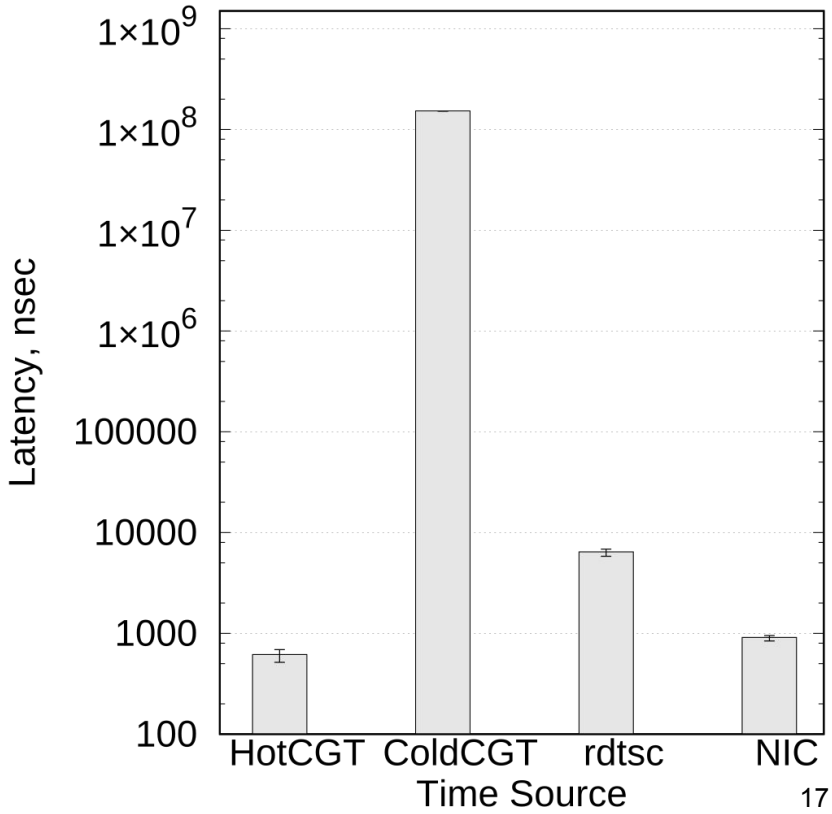
- `clock_gettime`
  - Hot sthread: reduce performance
  - Cold sthread: huge overhead
- `rdtsc`
  - Causes enclave exit
  - Performance loss due to TLB flush
- On-NIC Timer
  - Acceptable performance



On-NIC Time Source

- `clock_gettime`
  - Hot sthread: reduce performance
  - Cold sthread: huge overhead
- `rdtsc`
  - Causes enclave exit
  - Performance loss due to TLB flush
- On-NIC Timer
  - Acceptable performance

All of these time sources are untrusted.



Outline

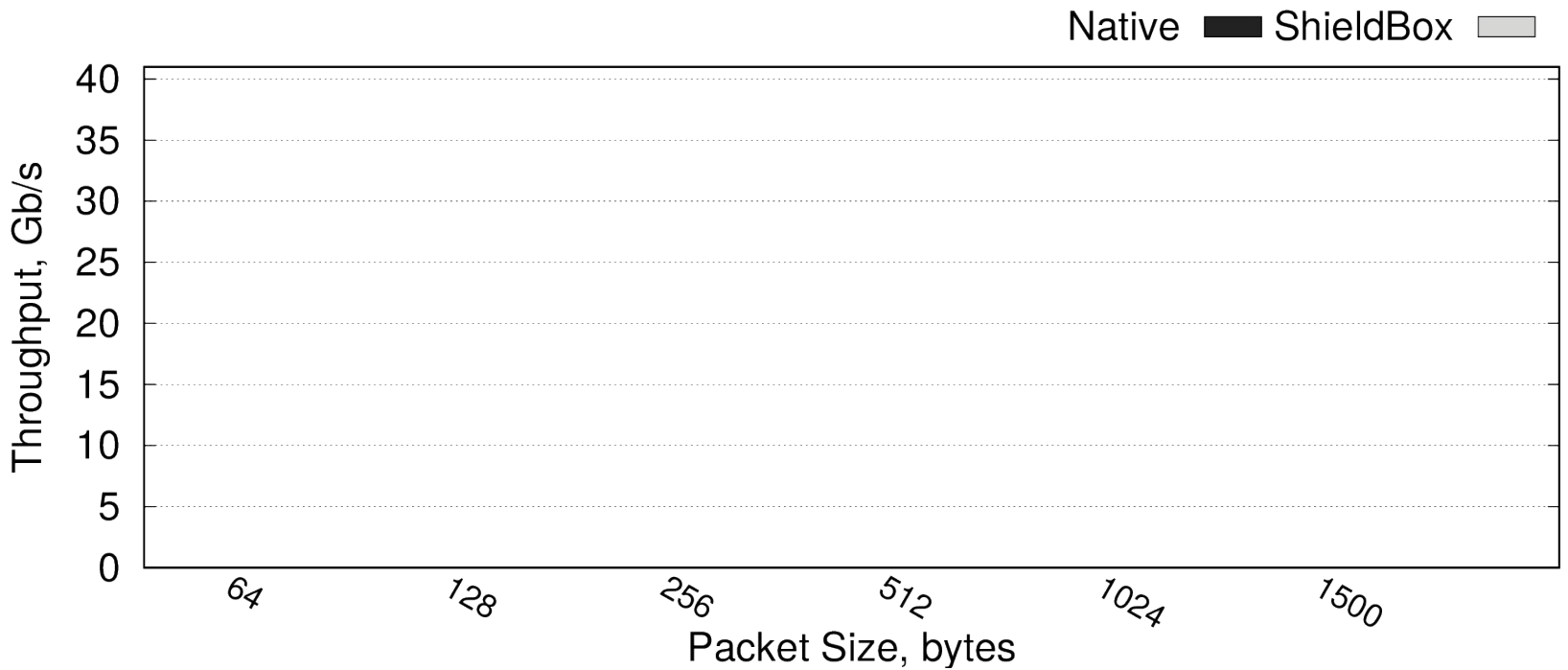
- ~~Motivation~~
- ~~Design~~
- Evaluation
- Summary

Evaluation

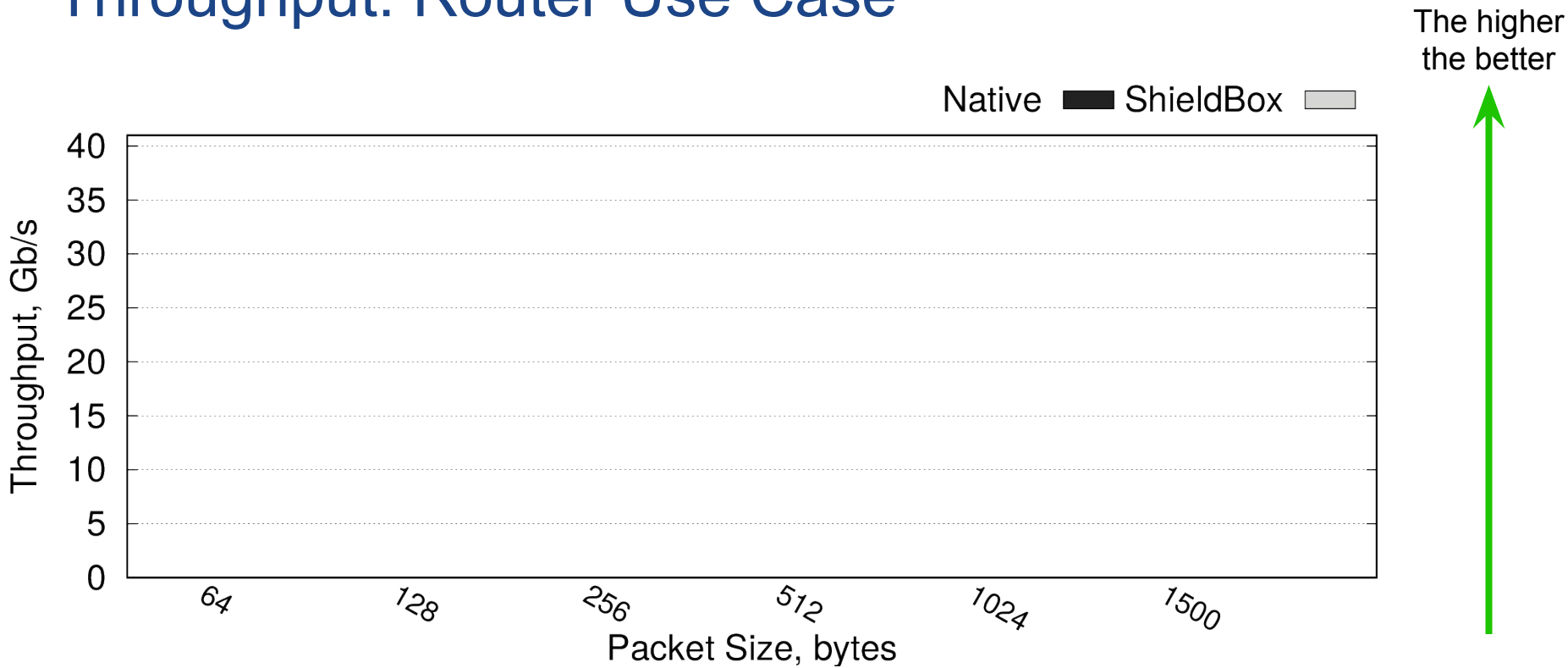
- What is the throughput and latency of our system?
- What is the influence of ToEnclave element on the performance?
- Other questions: see in the paper.

Throughput: Router Use Case

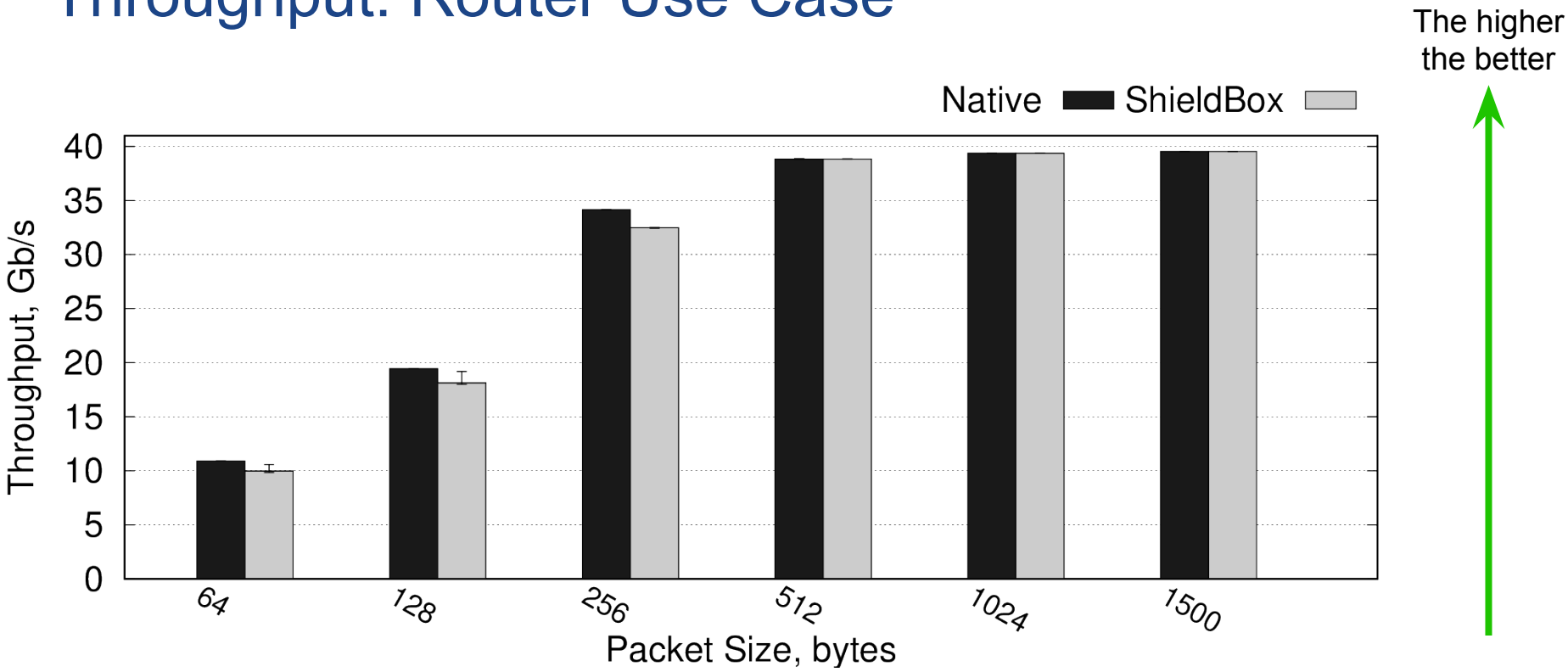
Throughput: Router Use Case



Throughput: Router Use Case



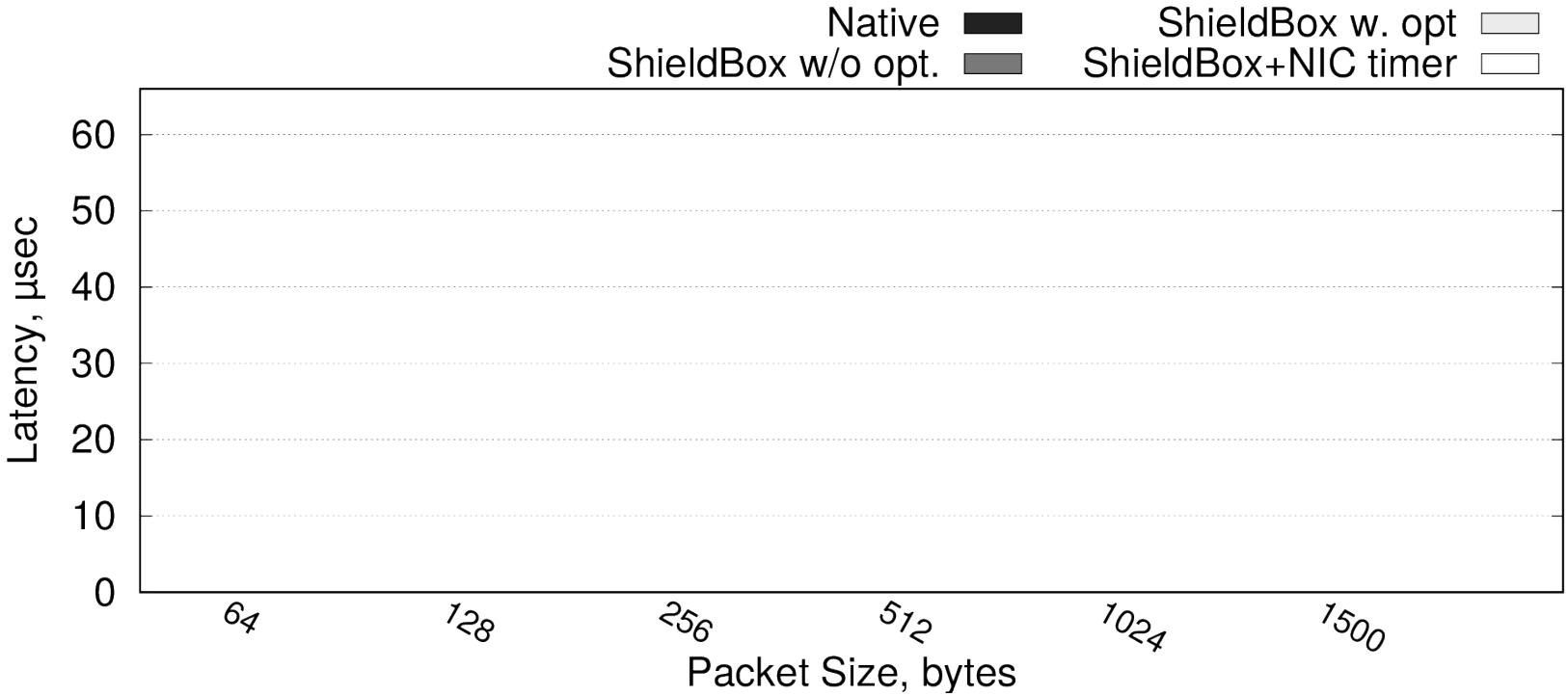
Throughput: Router Use Case



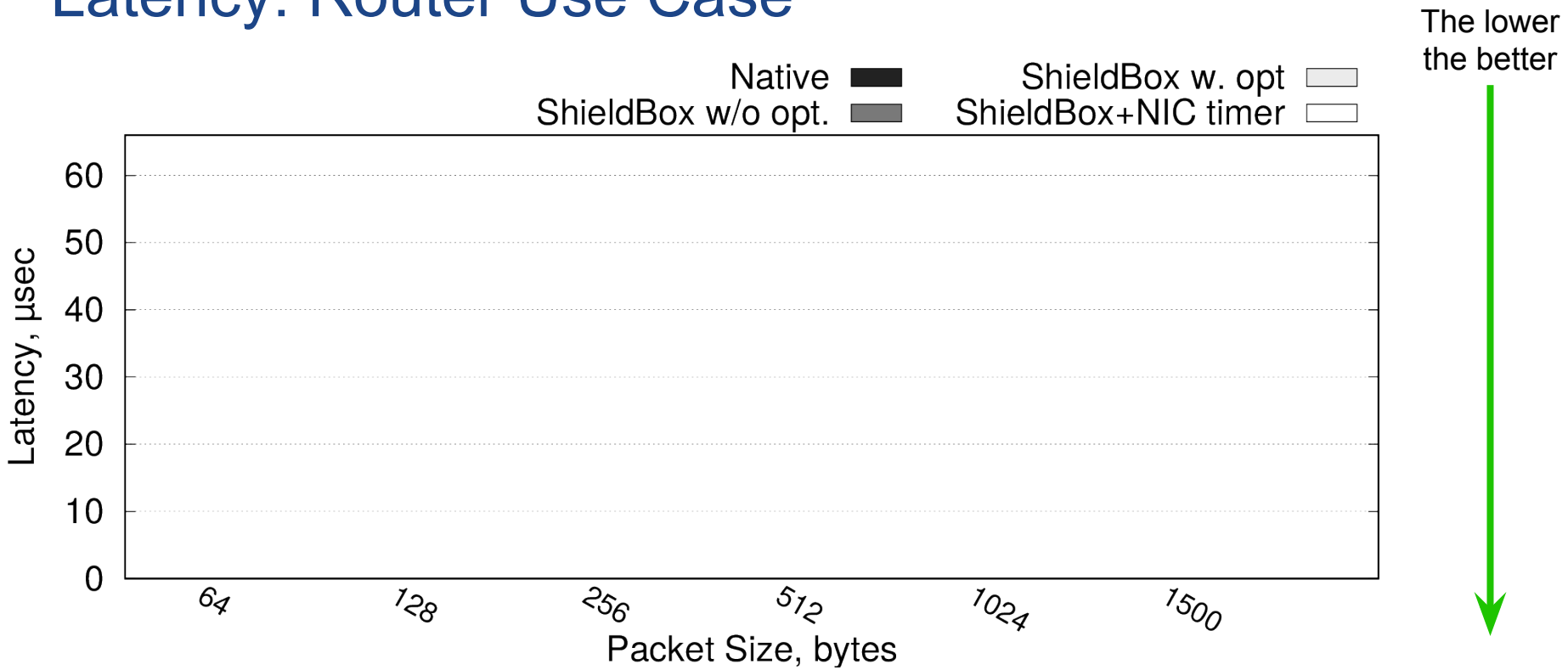
Line rate at most common packet size

Latency: Router Use Case

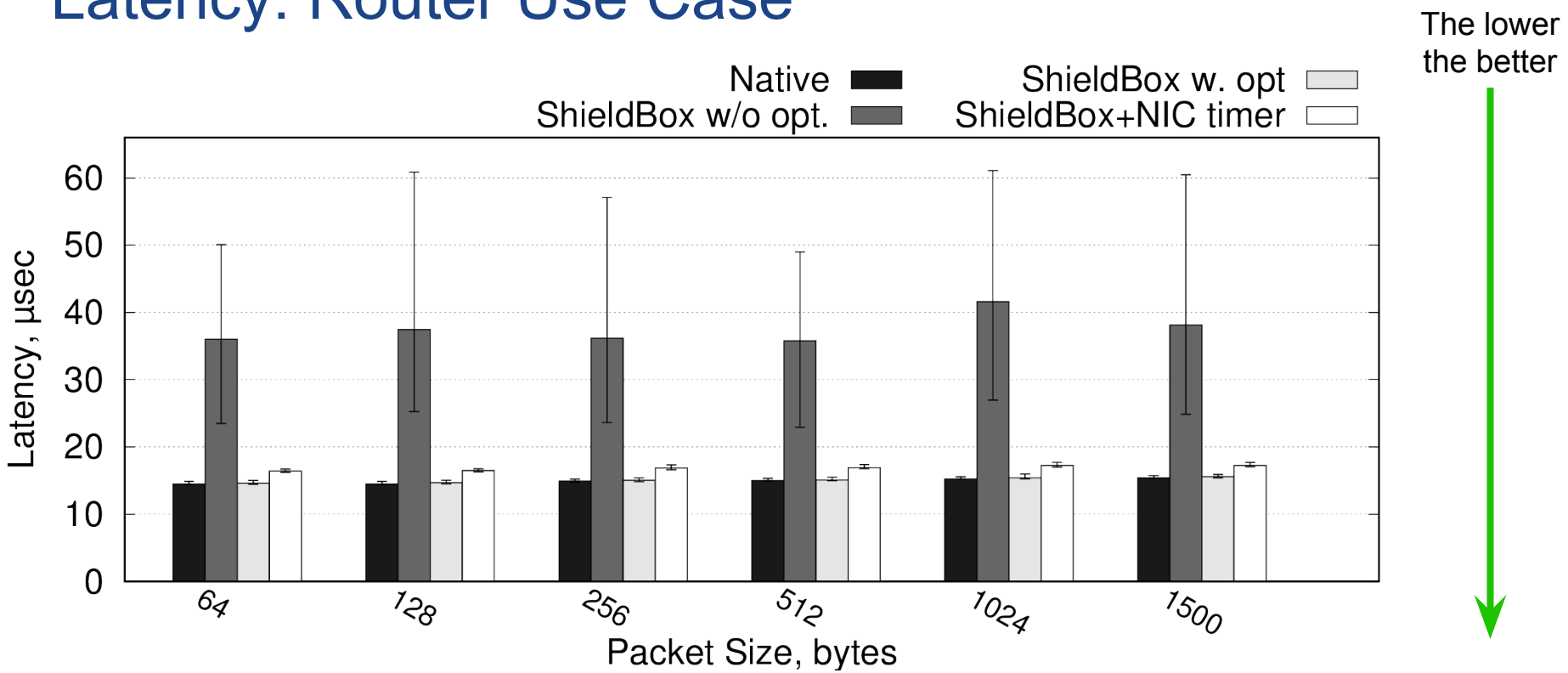
Latency: Router Use Case



Latency: Router Use Case



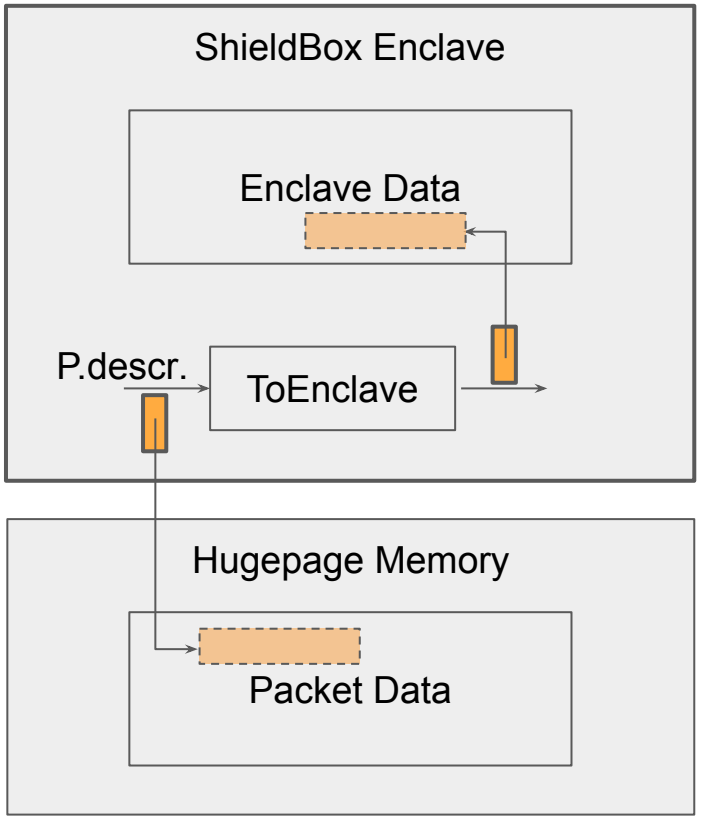
Latency: Router Use Case



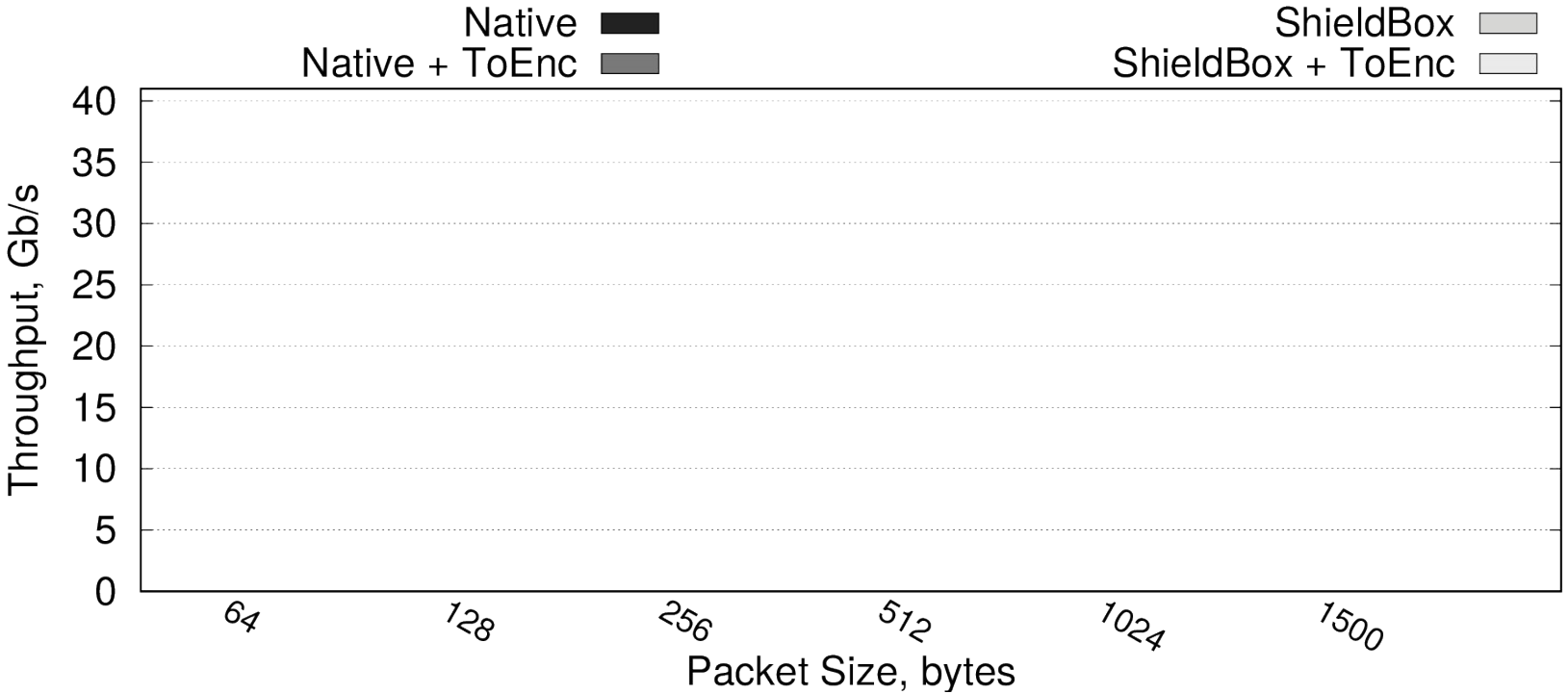
Optimizations bring latency to native level

ToEnclave Influence: EtherMirror

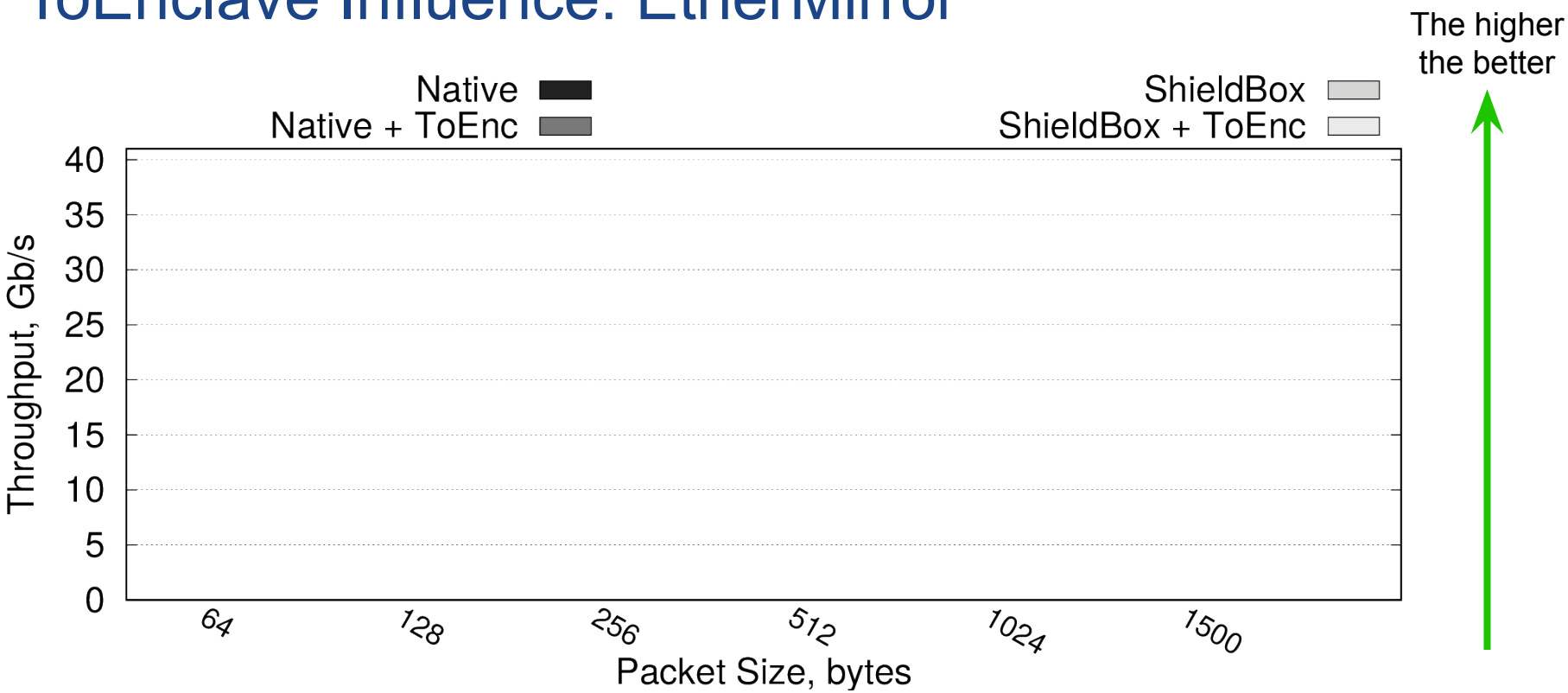
- Cheap NF→ worst-case example



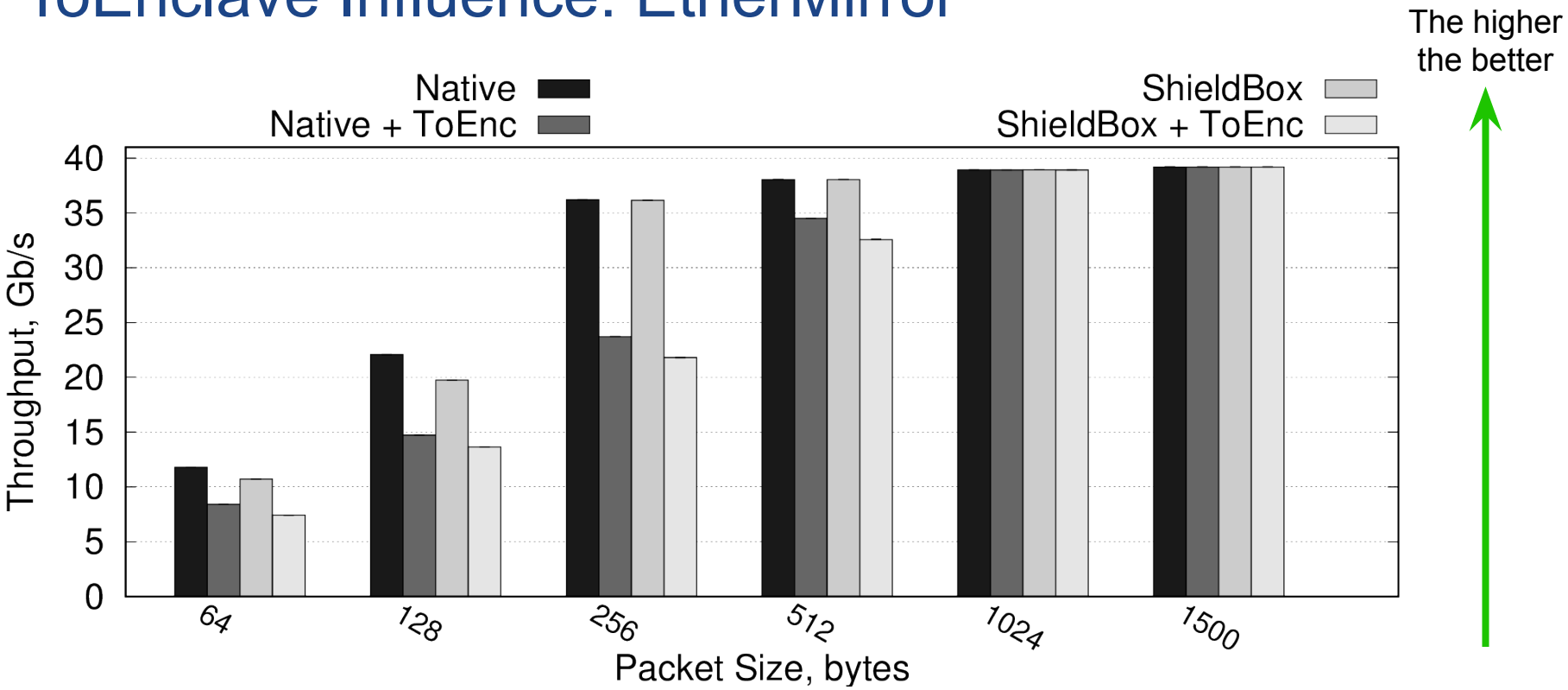
ToEnclave Influence: EtherMirror



ToEnclave Influence: EtherMirror



ToEnclave Influence: EtherMirror



~15% throughput reduction due to the extra memory copy

Outline

- ~~Motivation~~
- ~~Design~~
- ~~Evaluation~~
- Summary

Summary

- Cloud:
  - Abundant computational resources ☺
  - Limited trust to platform ☹

Summary

- Cloud:
  - Abundant computational resources ☺
  - Limited trust to platform ☹

- TEEs allow construction of middleboxes in the cloud:
  - Achieve end-to-end trust
  - Flexible frameworks for NF construction available

Summary

- **High performance:**
  - Line rate with typical Network Functions by using DPDK
  - Minimal overhead from ToEnclave element

Summary

- **High performance:**
  - Line rate with typical Network Functions by using DPDK
  - Minimal overhead from ToEnclave element
- **Secure:**
  - End-to-end trusted NF system with Intel SGX and SCONE
  - Enables use of modern cryptography

Summary

- **High performance:**
  - Line rate with typical Network Functions by using DPDK
  - Minimal overhead from ToEnclave element
- **Secure:**
  - End-to-end trusted NF system with Intel SGX and SCONE
  - Enables use of modern cryptography
- **Practical:**
  - Allows construction of wide range of Network Functions
  - Easy management using Dockerfiles from SCONE remote configuration

Summary

- **High performance:**
  - Line rate with typical Network Functions by using DPDK
  - Minimal overhead from ToEnclave element
- **Secure:**
  - End-to-end trusted NF system with Intel SGX and SCONE
  - Enables use of modern cryptography
- **Practical:**
  - Allows construction of wide range of Network Functions
  - Easy management using Dockerfiles from SCONE remote configuration

Thank You!

Funding

This project was funded by the European Union's Horizon 2020 program under grant agreements No. 645011 (SERECA), No. 690111 (SecureCloud), and No. 690588 (SELIS)

Funding

This project was funded by the European Union's Horizon 2020 program under grant agreements No. 645011 (SERECA), No. 690111 (SecureCloud), and No. 690588 (Selis)

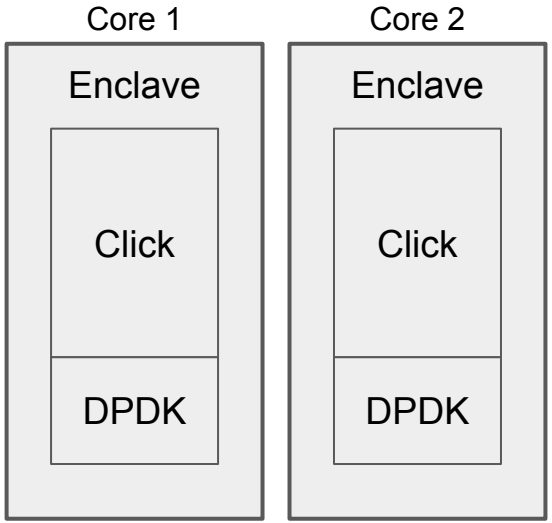
Design Challenges

- Modern middleboxes are expected to run at line speeds (10-40Gb/s)
- Overheads of 10x from asynchronous system call interface
- High impact on latency without optimizations (2-3x)
- SGX applications require a partitioning scheme

ShieldBox TCB

Trusted Computing Base:

- Click code and data
- DPDK code and non-hugepages data
- SCONE



ShieldBox: Security, Deployment, Limitations

Security:

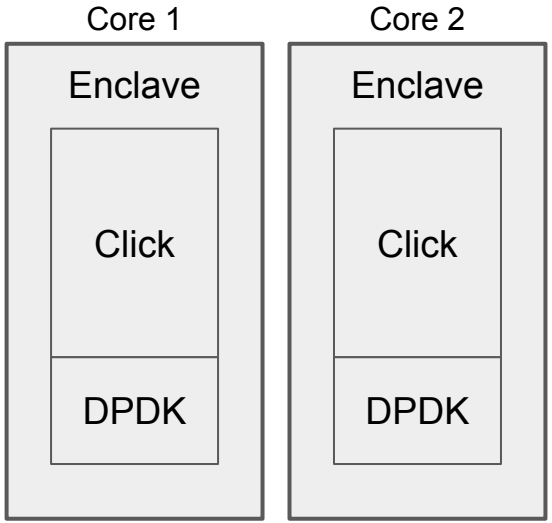
- Confidentiality for data inside enclave
- Integrity for data and processing functions

Deployment scenarios:

- Out of scope
- see APLOMB [SIGCOMM'12]

Limitations:

- No flow reassembly



Outline

- ~~Motivation~~
- ~~Design~~
- Evaluation
- Summary

DPDK and SGX Interaction

DPDK Initialization:

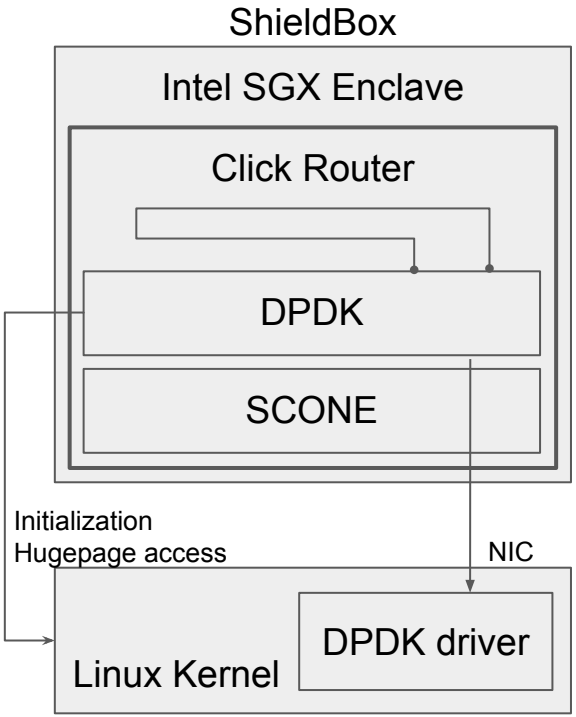
- SCONE system calls
- Minor patching for SCONE musl-libc

Obtaining huge pages:

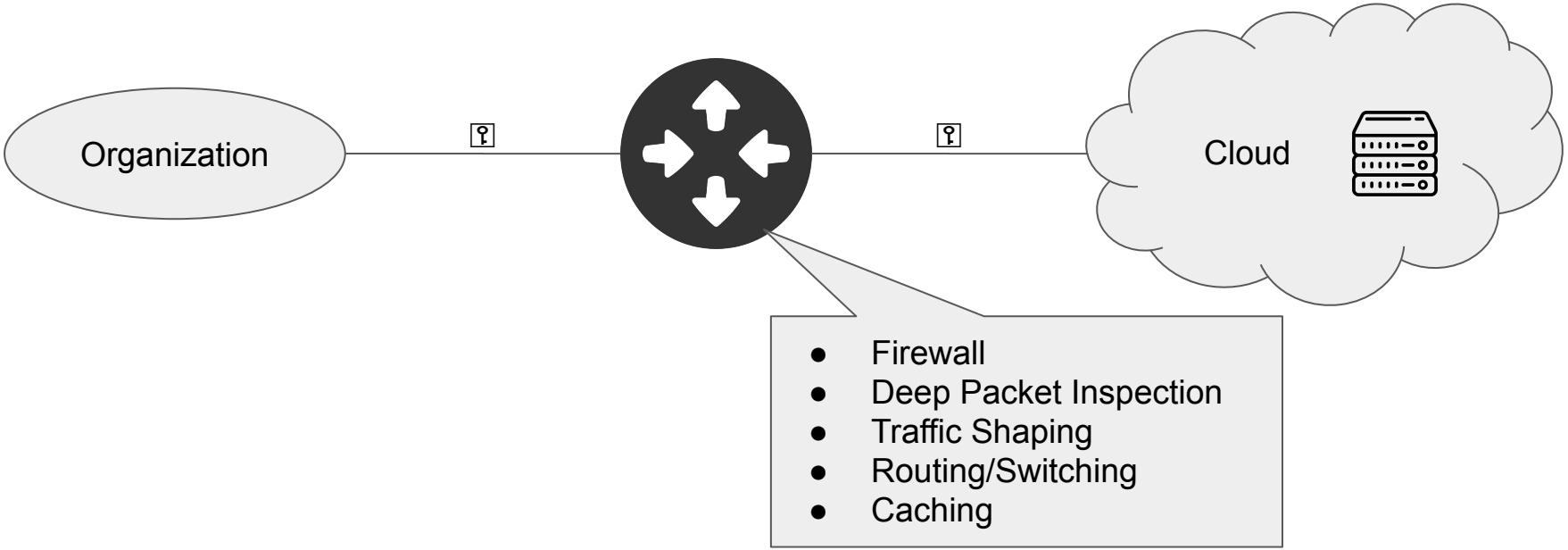
- Directly, with disabled FS shielding

NIC access:

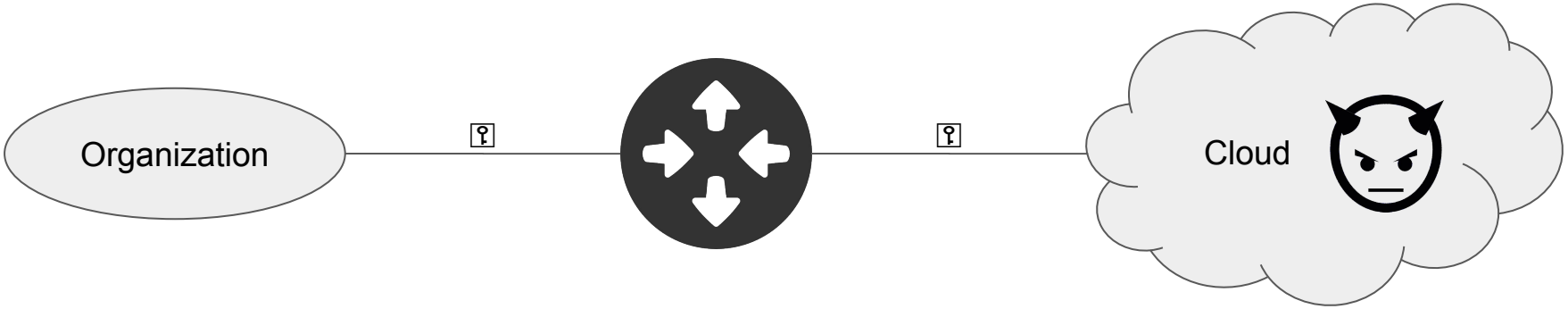
- Standard DPDK driver, disabled FS shielding



Middleboxes in the cloud



Security issues

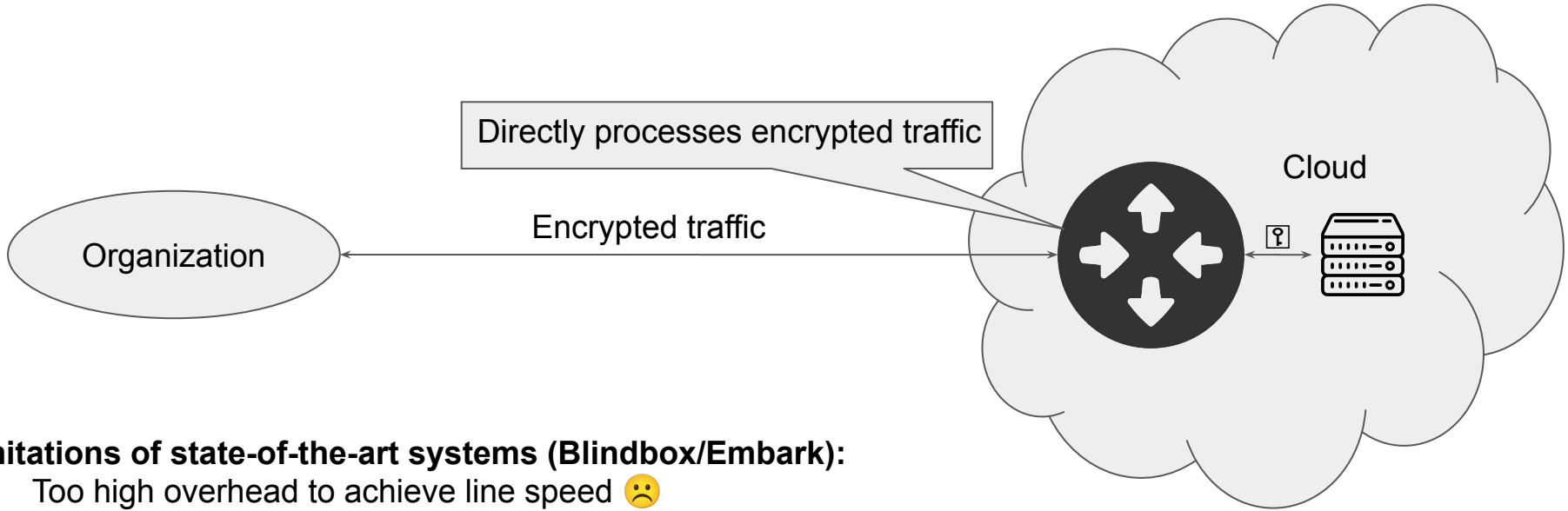


Already use cloud for computations ☺
Higher trust requirements for NFs ☹

Cheap computation resources ☺
Economy of scale ☺
Low trust environment ☹

- Observe private data
- Extract encryption keys
- Learn configuration

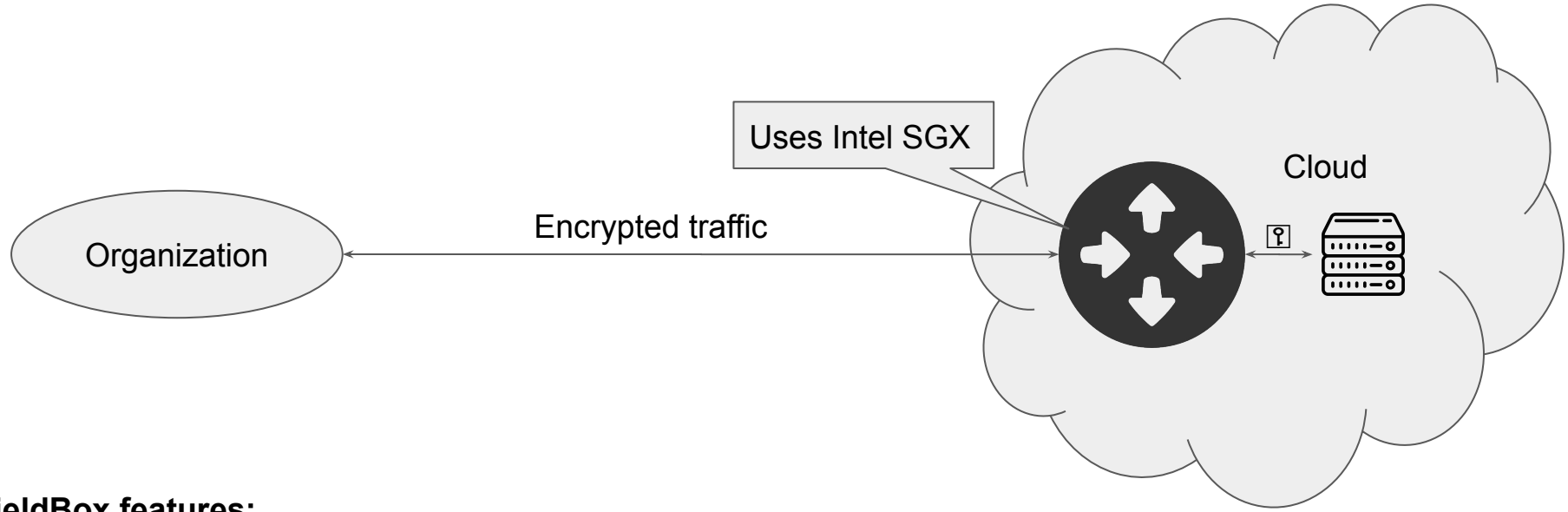
State-of-the-art



Limitations of state-of-the-art systems (Blindbox/Embark):

- Too high overhead to achieve line speed ☹
- Long communication session setup time ☹
- Require protocol modifications ☹
- Offer limited functionality ☹

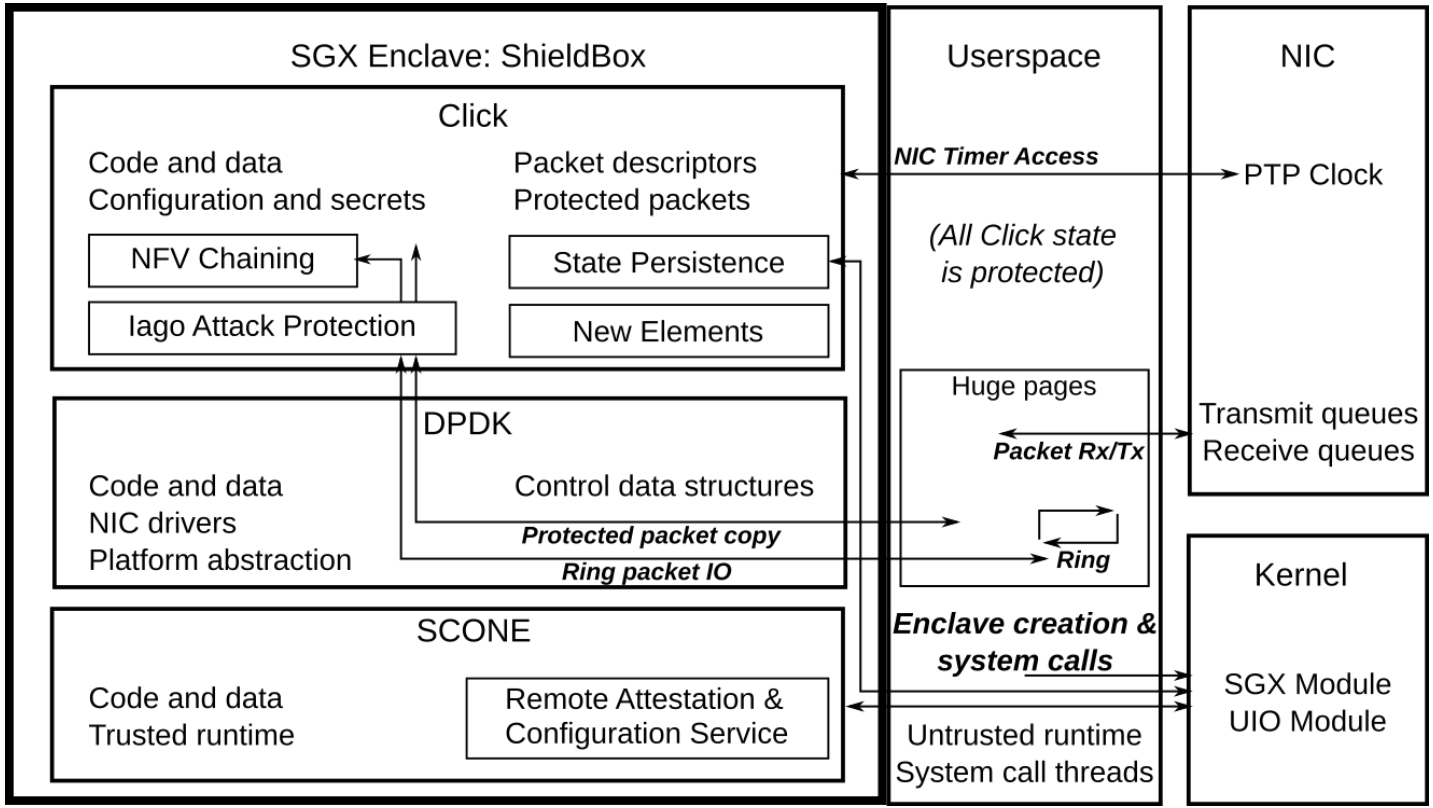
ShieldBox



ShieldBox features:

- Middlebox framework with shielded execution

System overview



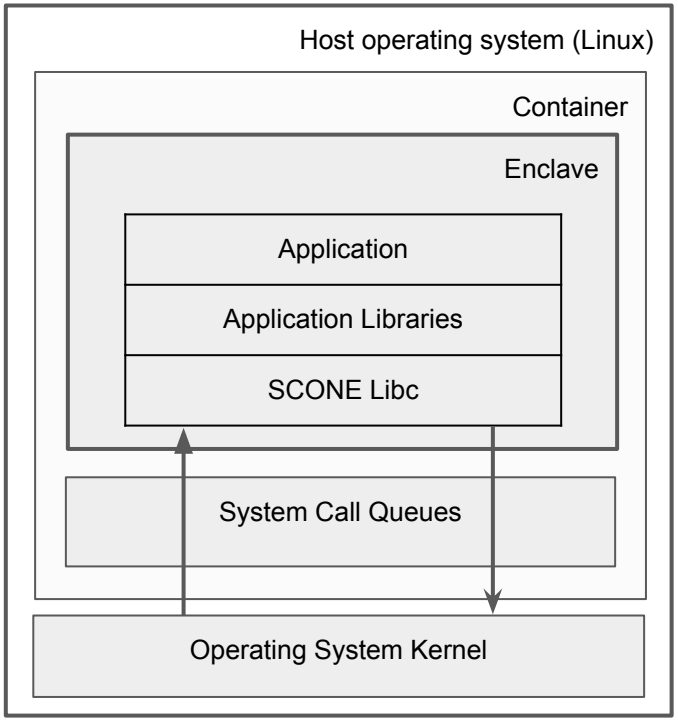
SCONE

Secure **Con**tainer Framework

Runs unmodified POSIX applications inside enclave:

- Memory management
- System calls
- Userspace threading

Provides remote attestation and configuration service.



Usage Scenarios

Developer writes a Click program and distributes it to ShieldBox via CAS

- High-level, easy-to-use API

When necessary Click element is missing, developer to ShieldBox via C++ API

- Low-level, flexible API

Operator can monitor the operation of the system using ControlSocket

Evaluation Setup

System Under Test (ShieldBox):

- Intel Xeon E3-1270 v5 (3.6 GHz, 4 cores, 8 HT)
- 32GB RAM

Load Generator:

- Intel Xeon D-1540 (2 GHz, 8 cores, 16 HT)
- 32GB RAM

Connection: Intel XL710 40Gb NIC between the machines