IceClave: A Trusted Execution Environment for In-Storage Computing

Luyi Kang*, Yuqi Xue*, Weiwei Jia*, Xiaohao Wang, Jongryool Kim‡, Changhwan Youn‡, Myeong Joon Kang‡, Hyung Jin Lim‡, Bruce Jacob†, Jian Huang

*Co-primary authors.
In-Storage Computing: A Promising Technique for I/O-Intensive Applications

Host-based Computing
In-Storage Computing: A Promising Technique for I/O-Intensive Applications

Host-based Computing

Host

CPU

DRAM

SSD

I/O Bottleneck!
In-Storage Computing: A Promising Technique for I/O-Intensive Applications

Host-based Computing

Host

CPU

DRAM

SSD

I/O Bottleneck!

In-Storage Computing

Host

In-Storage Processor

DRAM Buffer

Flash Memory
In-Storage Computing: A Promising Technique for I/O-Intensive Applications

Host-based Computing

- Host
- CPU
- DRAM
- SSD

I/O Bottleneck!

In-Storage Computing

- Host
- In-Storage Processor
- DRAM Buffer
- Flash Memory
In-Storage Computing: A Promising Technique for I/O-Intensive Applications

In-storage computing offers an effective solution to alleviate the I/O bottleneck.
SSD Architecture for In-Storage Computing

SSD Controller
SSD Architecture for In-Storage Computing

- **SSD Controller**
- **Embedded Cores**
- **Internal Bus**
- **Memory Controller**
- **Off-chip DRAM**
- **PCIe Interface**
- **Host**
SSD Architecture for In-Storage Computing

- SSD Controller
- Embedded Cores
- Internal Bus
- Memory Controller
- Off-chip DRAM
- PCIe Interface
- Host
- Flash Controller
SSD Architecture for In-Storage Computing

- SSD Controller
- Embedded Cores
- PCIe Interface
- Internal Bus
- Memory Controller
- Off-chip DRAM
- Flash Controller
- Flash Controller
- Flash Controller
- Flash
- Flash
- Flash
- Flash
- Flash
- Flash
- Flash
- Flash
- Channel
SSD Architecture for In-Storage Computing

- **Host**
- **PCIe Bus**
- **SSD**
- **Flash Translation Layer**
- **Block I/O**
- **NAND Flash**
SSD Architecture for In-Storage Computing

- Host
- PCIe Bus
- SSD
- Flash Translation Layer
  - Address Translation
  - NAND Flash
- Block I/O
SSD Architecture for In-Storage Computing

- Host
- PCIe Bus
- SSD
- Flash Translation Layer
  - Address Translation
  - Garbage Collection
- Garbage Collection
- Block I/O
- NAND Flash
SSD Architecture for In-Storage Computing

- Host
- PCIe Bus
- SSD

Block I/O
Flash Translation Layer
- Address Translation
- Garbage Collection
- Wear Leveling

NAND Flash
State-of-the-Art Frameworks for In-Storage Computing

Flow-based Programming

SSDlet → SSDlet → ...

SSD Processor

MapReduce-based Framework
State-of-the-Art Frameworks for In-Storage Computing

Flow-based Programming

SSDlet → SSDlet → ...

SSD Processor

MapReduce-based Framework

RPC-based Offloading

Host

RPC

Lightweight OS

SSD Processor

User App

User App

User App
State-of-the-Art Frameworks for In-Storage Computing

- **Flow-based Programming**
  - SSDlet → SSDlet → …
  - SSD Processor

- **MapReduce-based Framework**

- **RPC-based Offloading**
  - Host
  - Lightweight OS
  - User App → User App → User App
  - SSD Processor

- **FPGA Accelerator**
  - SSD Controller → Flash Chips

- **Industry SmartSSD**
State-of-the-Art Frameworks for In-Storage Computing

Most of the existing frameworks focus on performance and programmability
State-of-the-Art Frameworks for In-Storage Computing

Flow-based Programming
- SSDlet → SSDlet → ...

MapReduce-based Framework

RPC-based Offloading
- Host → SSHD Processor

Industry SmartSSD
- FPGA Accelerator
- SSD Controller
- Flash Chips

Most of the existing frameworks focus on performance and programmability.

Few of them consider security as the first-class citizen.
Why Should We Secure In-Storage Computing?
Why Should We Secure In-Storage Computing?

- In-Storage App 1
- In-Storage App 2
- In-Storage App 3

- Flash Translation Layer
- NAND Flash
Why Should We Secure In-Storage Computing?

In-Storage App 1  In-Storage App 2  In-Storage App 3

Flash Translation Layer

NAND Flash

Attack co-located programs
Why Should We Secure In-Storage Computing?

- Flash Translation Layer
- NAND Flash
- In-Storage App 1
- Data Breach
- In-Storage App 3

Attack co-located programs
Why Should We Secure In-Storage Computing?

- Attack co-located programs
- Attack FTL

Diagram:
- In-Storage App 1
- Data Breach
- In-Storage App 3
- Flash Translation Layer
- NAND Flash
Why Should We Secure In-Storage Computing?

Attack co-located programs

- In-Storage App 1
- Data Breach
- Data Leakage & Data Loss
- NAND Flash
- Attack FTL

In-Storage App 3
Why Should We Secure In-Storage Computing?

- Attack co-located programs
- Attack FTL
- Physical Attacks

Data Leakage & Data Loss

NAND Flash

In-Storage App 1

Data Breach

In-Storage App 3

In-Storage App 2

- Attack co-located programs
- Attack FTL
- Physical Attacks
Why Should We Secure In-Storage Computing?

- Attack co-located programs
- Data Breach
- In-Storage App 1
- Data Leakage & Data Loss
- In-Storage App 3
- Attack FTL
- Physical Attacks
Why Should We Secure In-Storage Computing?

It is desirable to build a secure in-storage computing environment!
Existing TEEs Do Not Work For In-Storage Computing

Intel SGX is not available in storage processors
Existing TEEs Do Not Work For In-Storage Computing

Intel SGX is not available in storage processors

Unclear how to apply ARM TrustZone to in-storage computing
IceClave: A Trusted Execution Environment for In-Storage Computing

Diagram showing the integration of In-Storage Apps (App 1, App 2, and App 3) with the Flash Translation Layer, SSD Processors, Flash Chips, and Off-chip DRAM.
IceClave: A Trusted Execution Environment for In-Storage Computing

Protecting FTL from malicious in-storage apps
IceClave: A Trusted Execution Environment for In-Storage Computing

Protecting FTL from malicious in-storage apps

Security isolation between in-storage apps
IceClave: A Trusted Execution Environment for In-Storage Computing

- Protecting FTL from malicious in-storage apps
- Security isolation between in-storage apps
- Securing data against physical attacks

Diagram:
- In-Storage App 1
- In-Storage App 2
- In-Storage App 3
- Flash Translation Layer
- SSD Processors
- Flash Chips
- Memory Encryption
- Encrypted Flash I/O
- Off-chip DRAM
IceClave Design Challenges

Bare-metal Environment
IceClave Design Challenges

Bare-metal Environment

Efficient Flash Access
IceClave Design Challenges

- Bare-metal Environment
- Efficient Flash Access
- Limited Resources in SSD Device
Threat Model

In-Storage App 1
In-Storage App 2
In-Storage App 3

Flash Translation Layer

Host
SSD Processors
Off-chip DRAM
Flash Chips

Embedded Processor
SSD
Processors
In-Storage App 1
In-Storage App 2
In-Storage App 3

Flash Translation Layer

Host
SSD Processors
Off-chip DRAM
Flash Chips

Embedded Processor
SSD
Processors
In-Storage App 1
In-Storage App 2
In-Storage App 3

Flash Translation Layer

Host
SSD Processors
Off-chip DRAM
Flash Chips
Threat Model

In-storage applications can be malicious

In-Storage App 1

In-Storage App 2

Flash Translation Layer

Host

SSD Processors

Flash Chips

Off-chip DRAM
Threat Model

In-storage applications can be malicious

Cloud platform operator may conduct physical attacks
Threat Model

In-storage applications can be malicious

Flash Translation Layer

In-Storage App 1

In-Storage App 2

Host runtime and host-SSD offloading channel is trusted

Cloud platform operator may conduct physical attacks

Host

SSD Processors

Off-chip DRAM

Flash Chips

In-storage applications can be malicious
Protecting Flash Translation Layer

Protecting FTL from malicious in-storage apps

Security isolation between in-storage apps

Securing data against physical attacks
Protecting Flash Translation Layer

- In-Storage App
- Flash Translation Layer

Isolated

Protecting FTL from malicious in-storage apps
Protecting Flash Translation Layer

Secure

Normal
Protecting Flash Translation Layer

Flash Translation Layer

In-Storage App 1

In-Storage App 2

Secure

Normal

Secure

Normal
Protecting Flash Translation Layer

- Flash Translation Layer
- Address Mapping Table
- In-Storage App 1
- In-Storage App 2
- Secure
- Normal

Diagram outlines the relationship between the Flash Translation Layer and the Address Mapping Table, showing how they interact with the In-Storage Apps 1 and 2.
Protecting Flash Translation Layer

Naively applying TrustZone partitioning incurs significant performance penalty!
Protecting Flash Translation Layer

Naively applying TrustZone partitioning incurs significant performance penalty!
Protecting Flash Translation Layer

Naively applying TrustZone partitioning incurs significant performance penalty!
Protecting Flash Translation Layer

- Flash Translation Layer
- Address Mapping Table
- In-Storage App 1
- In-Storage App 2

Secure
Normal

Protected
Normal
Protecting Flash Translation Layer

- Flash Translation Layer
- Address Mapping Table
- In-Storage App 1
- In-Storage App 2
- Secure
- Protected
- Normal
- Read-Only
- R/W

Secure mode for Flash Translation Layer and Address Mapping Table protects the data, while Normal mode allows read-only access.
Isolating In-Storage Applications

- Protecting FTL from malicious in-storage apps
- Security isolation between in-storage apps
- Securing data against physical attacks
Isolating In-Storage Applications

Security isolation between in-storage apps
Isolating In-Storage Applications

Secure

Normal

Flash Translation Layer

In-Storage App 1

In-Storage App 2

Address Mapping Table

Secure

Protected

Normal

Read-Only

R/W

Secure

Normal

Flash Translation Layer

In-Storage App 1

In-Storage App 2

Address Mapping Table

Secure

Protected

Normal
Isolating In-Storage Applications

Secure
Normal

Flash Translation Layer

In-Storage App 1

In-Storage App 2

Address Mapping Table

App 1 Allocated Memory

App 2 Allocated Memory

Secure
Protected
Normal
Isolating In-Storage Applications

- **IceClave Runtime**
- **Flash Translation Layer**
- **In-Storage App 1**
- **In-Storage App 2**
- **App 1 Metadata**
- **App 2 Metadata**
- **Address Mapping Table**
- **App 1 Allocated Memory**
- **App 2 Allocated Memory**

- Read-Only
- R/W

- Secure
- Normal

- Secure
- Protected
- Normal
Isolating In-Storage Applications

- IceClave Runtime
- Flash Translation Layer
- In-Storage App 1
- In-Storage App 2
- App 1 Metadata
- App 2 Metadata
- Address Mapping Table
- Flash Access Control
- App 1 Allocated Memory
- App 2 Allocated Memory

Read-Only → R/W

Secure → Protected → Normal
Protecting Against Physical Attacks

- Protecting FTL from malicious in-storage apps
- Security isolation between in-storage apps
- Securing data against physical attacks

Diagram:
- In-Storage App 1, In-Storage App 2, In-Storage App 3
- Flash Translation Layer
- SSD Processors
- Flash Chips
- Off-chip DRAM
- Memory Encryption
- Encrypted Flash I/O
Protecting Against Physical Attacks

Securing data against physical attacks
Protecting Against Physical Attacks

Securing data against physical attacks
Protecting Against Physical Attacks

Physical Attacks

SSD Processors

Flash Chips

Off-chip DRAM

Physical Attacks

Securing data against physical attacks
Protecting Against Physical Attacks

Securing data against physical attacks

- SSD Processors
- Flash Chips
- Off-chip DRAM

Protecting Physical Memory
Protecting Physical Memory

- Processor
  - Secure Root
    - Merkle Tree
      - Last-Level Counter Blocks
        - Data Blocks
Protecting Physical Memory

- Processor
  - Secure Root
    - Merkle Tree
      - Last-Level Counter Blocks
        - Data Blocks
Protecting Physical Memory

Split Counter Mode (ISCA’06)
Protecting Physical Memory

Split Counter Mode (ISCA’06)
Protecting Physical Memory

In-storage programs are read-intensive
Protecting Physical Memory

In-storage programs are read-intensive
Protecting Physical Memory

In-storage programs are read-intensive

State-of-the-art Split Counter Mode is not optimal for in-storage computing
Protecting Physical Memory

Read-Only Root

Processor

Split Counter Root

IceClave Hybrid Counter

Read-Only Memory

Writable Memory
Protecting Physical Memory

IceClave Hybrid Counter

- **Read-Only Root**
- **Split Counter Root**

**Processor**

**Read-Only Memory**
- 4KB
- 4KB
- 64b
- ... 64b
- 4KB

**Writable Memory**
- 64B
- 64B
- 64B
- ... 64B

Securing data against physical attacks
Protecting Against Physical Attacks

Securing data against physical attacks

- SSD Processors
- Flash Chips
- Protecting Physical Memory
- Protecting Data Access To Flash Chips
- Off-chip DRAM
Protecting Data Access To Flash Chips
Put It All Together

- Host Machine
- SSD Controller
  - PCIe Interface
  - CPU Cores
  - Internal Bus
- Memory Encryption Engine
- Off-chip DRAM
- Stream Cipher Engine
- Flash Controller
- Flash Chips
Put It All Together

Host Machine

PCIe Interface

CPU Cores

Internal Bus

Memory Encryption Engine

Off-chip DRAM

SSD Controller

Stream Cipher Engine

Flash Controller

Flash Chips
Put It All Together

Secure World
- IceClave Runtime
- Flash Translation Layer
- Secure Monitor

Normal World
- In-Storage App 1
- In-Storage App 2

In-Storage
App 1

Host
SSD

Host Machine

PCle Interface

SSD Controller

Internal Bus

CPU Cores

Stream Cipher Engine

Flash Controller

Flash Chips

Memory Encryption Engine

Off-chip DRAM
Put It All Together

- Applications
  - IceClave Library
  - NVMe Driver

- Secure World
  - IceClave Runtime
  - Flash Translation Layer

- In-Storage
  - App 1
  - App 2

- Secure Monitor

- Normal World

- Host Machine

- SSD

- Host

- SSD Controller
  - CPU Cores
  - Stream Cipher Engine
  - Flash Controller
  - Flash Chips
  - PCIe Interface
  - Internal Bus
  - Memory Encryption Engine
  - Off-chip DRAM
IceClave Workflow

Secure
- IceClave Runtime
- Flash Translation Layer

Protected
- Mapping Table

Secure
- TEE
- Stream Cipher Engine
- Flash Controller
- Flash
IceClave Workflow

IceClave Library

Secure

Offload App

IceClave Runtime

Flash Translation Layer

Protected

Mapping Table

TEE

Stream Cipher Engine

Flash Controller

Flash
IceClave Workflow

1. Create TEE
2. Secure

- IceClave Runtime
- Flash Translation Layer

- Stream Cipher Engine
- Flash Controller
- Flash

- IceClave Library
- Protected
- Mapping Table
IceClave Workflow

IceClave Library

Secure

IceClave Runtime

Flash Translation Layer

Protected

Mapping Table

Reading Mapping Table

3

TEE

Stream Cipher Engine

Flash Controller

Flash
IceClave Workflow

IceClave Library

Secure

IceClave Runtime

Flash Translation Layer

Flash Mgmt

Protected

Mapping Table

TEE

Stream Cipher Engine

Flash Controller

Flash
IceClave Workflow

Secure

IceClave Library

Flash Translation Layer

Protected

Mapping Table

IceClave Runtime

Stream Cipher Engine

Flash Controller

Flash

TEE

6 Load Data
IceClave Workflow

1. Secure
2. Get Result

- IceClave Library
- Flash Translation Layer
- IceClave Runtime

Protected

- Mapping Table

- TEE
- Stream Cipher Engine
- Flash Controller
- Flash
IceClave Workflow

- **IceClave Library**
- **IceClave Runtime**
- **Flash Translation Layer**
- **Protected Mapping Table**
- **Secure**
- **Terminate**
- **TEE**
- **Stream Cipher Engine**
- **Flash Controller**
- **Flash**
IceClave Workflow

1. Secure Offload App
2. Create TEE
3. Reading Mapping Table
4. Missing Mapping Entry
5. Flash Mgmt
6. Load Data
7. Get Result
8. Terminate TEE

IceClave Library

Flash Translation Layer

IceClave Runtime

Protected

Mapping Table

Stream Cipher Engine

Flash Controller

Flash
IceClave Implementation

Experimental Setup

Simulator: gem5 + USIMM + SimpleSSD
Prototype: OpenSSD Cosmos+ FPGA

Synthetic Workloads: Arithmetic, Aggregate, Filter, Wordcount
Real-world Workloads: TPC-H, TPC-B, TPC-C
IceClave Overall Performance

Normalized Execution Time

- Host Load Time
- Host Compute Time
- SSD Load Time
- SSD Compute Time
- Mempry Encrypt

Left to Right:
- Aggregate
- Arithmetic
- Filter
- TPC-H Q1
- TPC-H Q3
- TPC-H Q12
- TPC-H Q14
- TPC-H Q19
- TPC-B
- TPC-C
- Wordcount
IceClave Overall Performance

Left to Right:  **Host**

- **Aggregate**
- **Arithmetic**
- **Filter**
- **TPC-H Q1**
- **TPC-H Q3**
- **TPC-H Q12**
- **TPC-H Q14**
- **TPC-H Q19**
- **TPC-B**
- **TPC-C**
- **Wordcount**
IceClave Overall Performance

- Host Load Time
- SSD Load Time
- Host Compute Time
- SSD Compute Time
- Mempry Encrypt

Left to Right: Host, Host+SGX

Tasks: Aggregate, Arithmetic, Filter, TPC-H Q1, TPC-H Q3, TPC-H Q12, TPC-H Q14, TPC-H Q19, TPC-B, TPC-C, Wordcount
IceClave Overall Performance

Left to Right:  Host  Host+SGX  ISC

- Host Load Time
- SSD Load Time
- Host Compute Time
- SSD Compute Time
- Memory Encrypt
IceClave Overall Performance
IceClave Overall Performance

- Host Load Time
- Host Compute Time
- SSD Load Time
- SSD Compute Time
- Mempry Encrypt

Normalized Execution Time

Left to Right:
- Host
- Host+SGX
- ISC
- IceClave
IceClave Overall Performance

Normalized Execution Time

- Host Load Time
- Host Compute Time
- SSD Load Time
- SSD Compute Time
- Mempry Encrypt

Left to Right: Host, Host+SGX, ISC, IceClave
IceClave Overall Performance

Left to Right: Host, Host+SGX, ISC, IceClave
IceClave Overall Performance

IceClave introduces minimal overhead while providing strong security.
IceClave Overall Performance

More evaluations in the paper!
IceClave Summary

First Trusted Execution Environment for In-Storage Computing

2.3× Faster Than Host-based Computing
Thank you!

Luyi Kang, **Yuqi Xue†**, Weiwei Jia, Xiaohao Wang, Jongryool Kim, Changhwan Youn, Myeong Joon Kang, Hyung Jin Lim, Bruce Jacob, Jian Huang

†yuqixue2@illinois.edu

Systems Platform Research Group

This presentation and recording belong to the authors. No distribution is allowed without the authors’ permission.