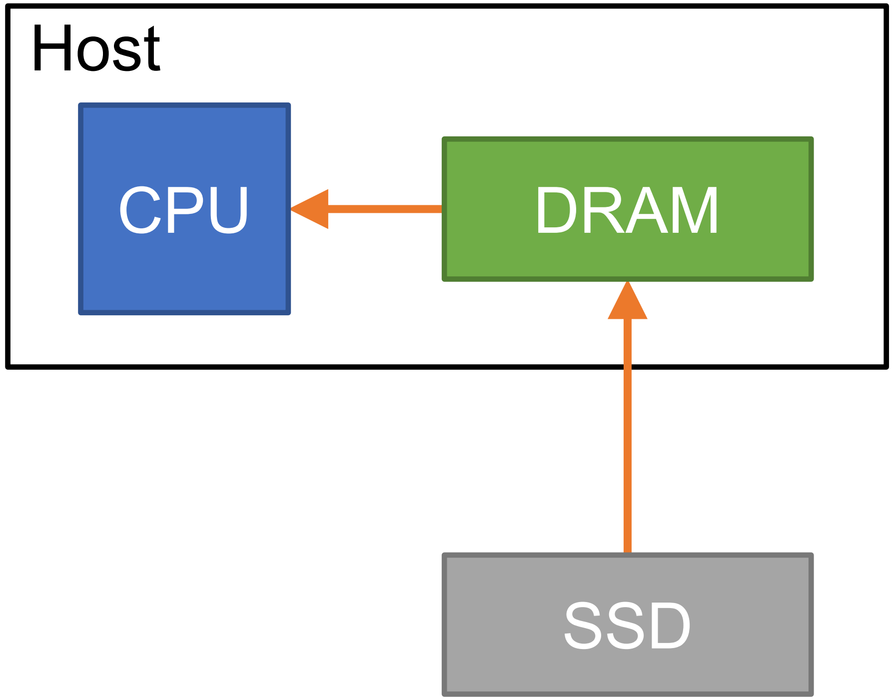
IceClave: A Trusted Execution Environment for In-Storage Computing

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.

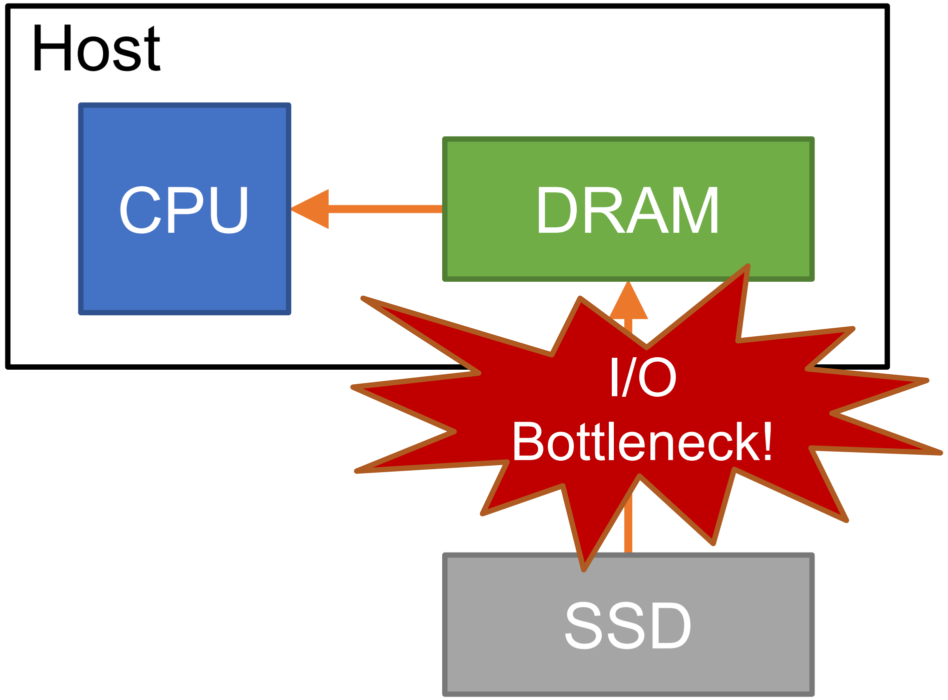
† ‡

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



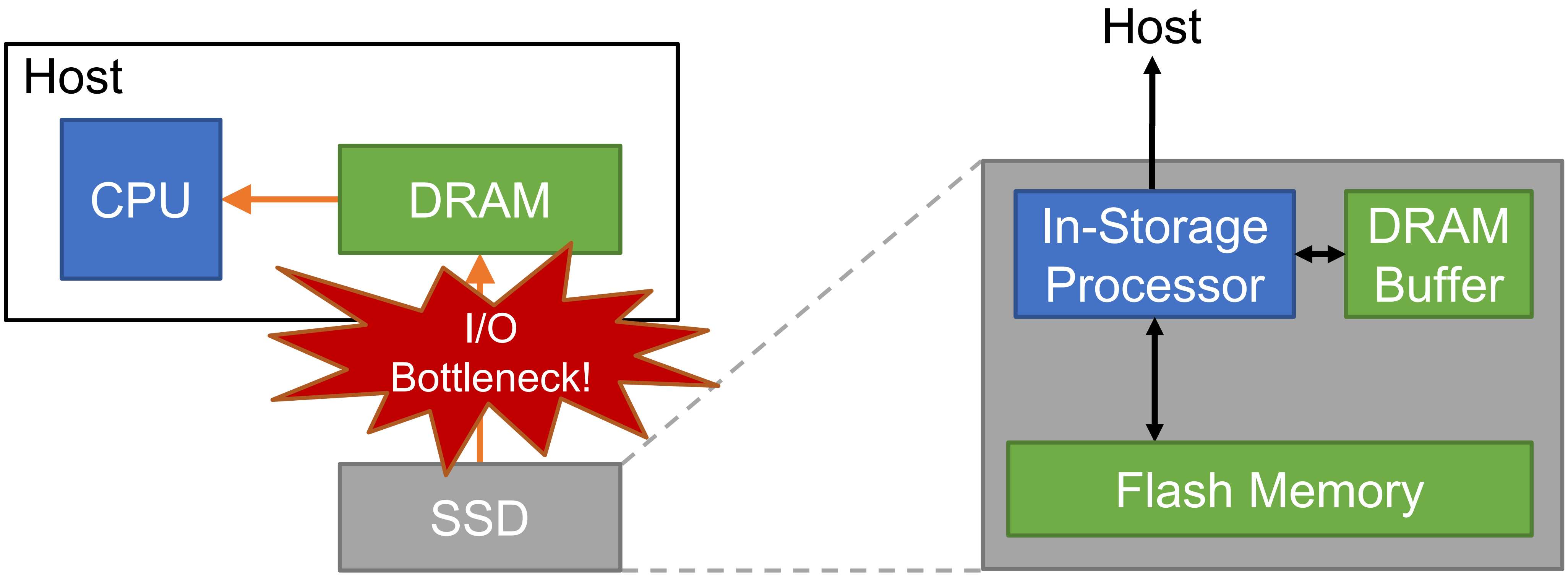
Host-based Computing

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



Host-based Computing

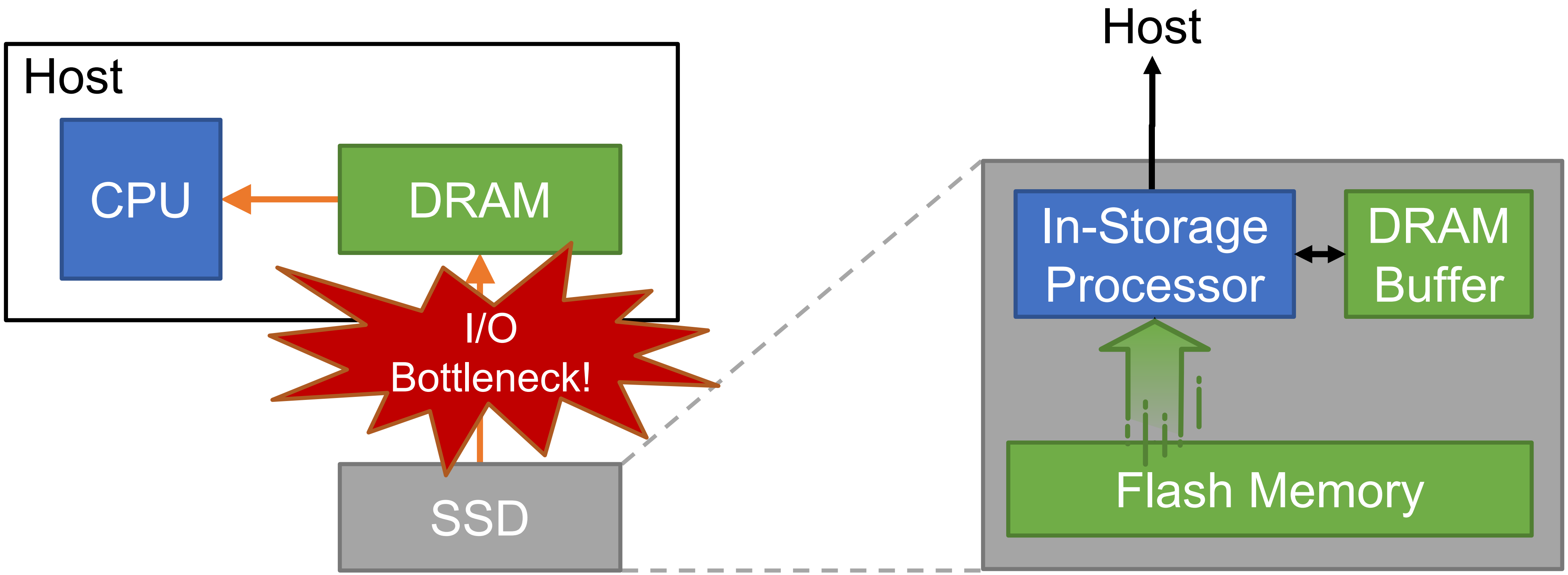
In-Storage Computing: A Promising Technique for I/O-Intensive Applications



Host-based Computing

In-Storage Computing

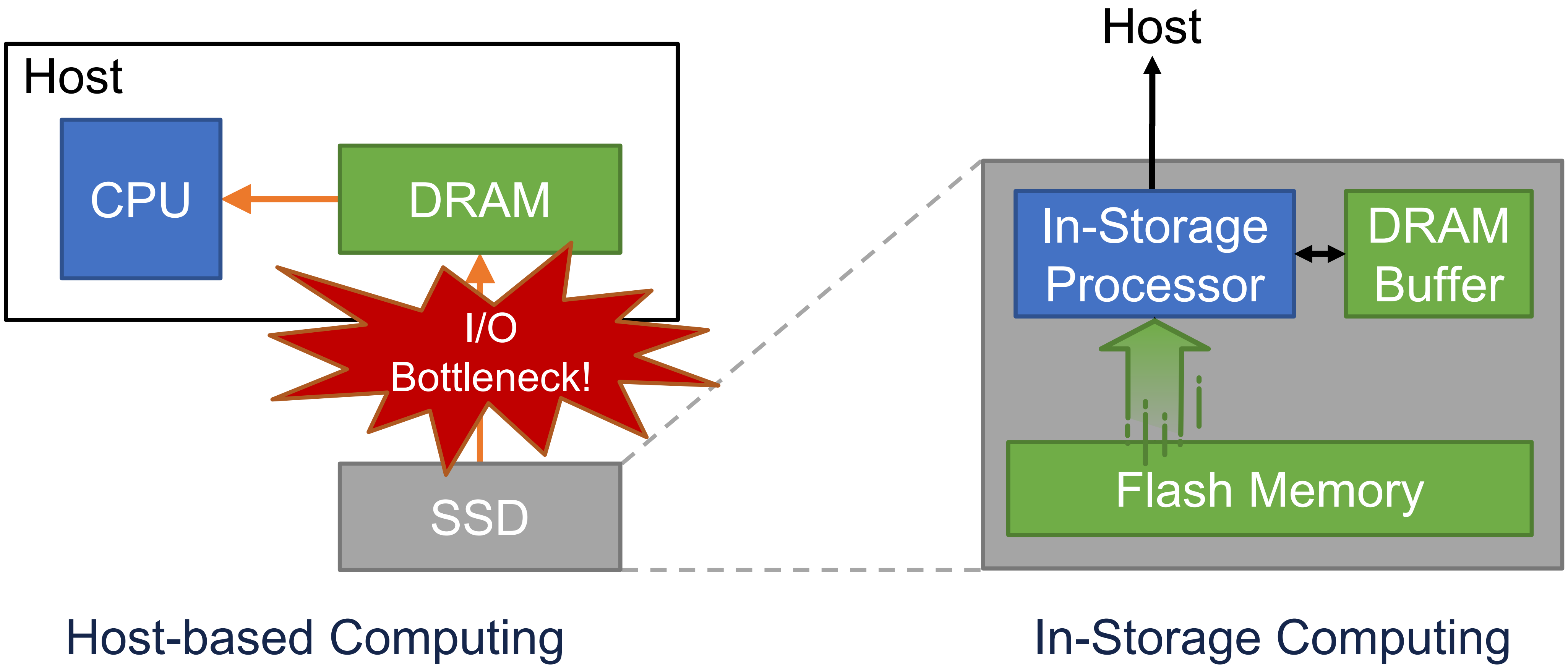
In-Storage Computing: A Promising Technique for I/O-Intensive Applications



Host-based Computing

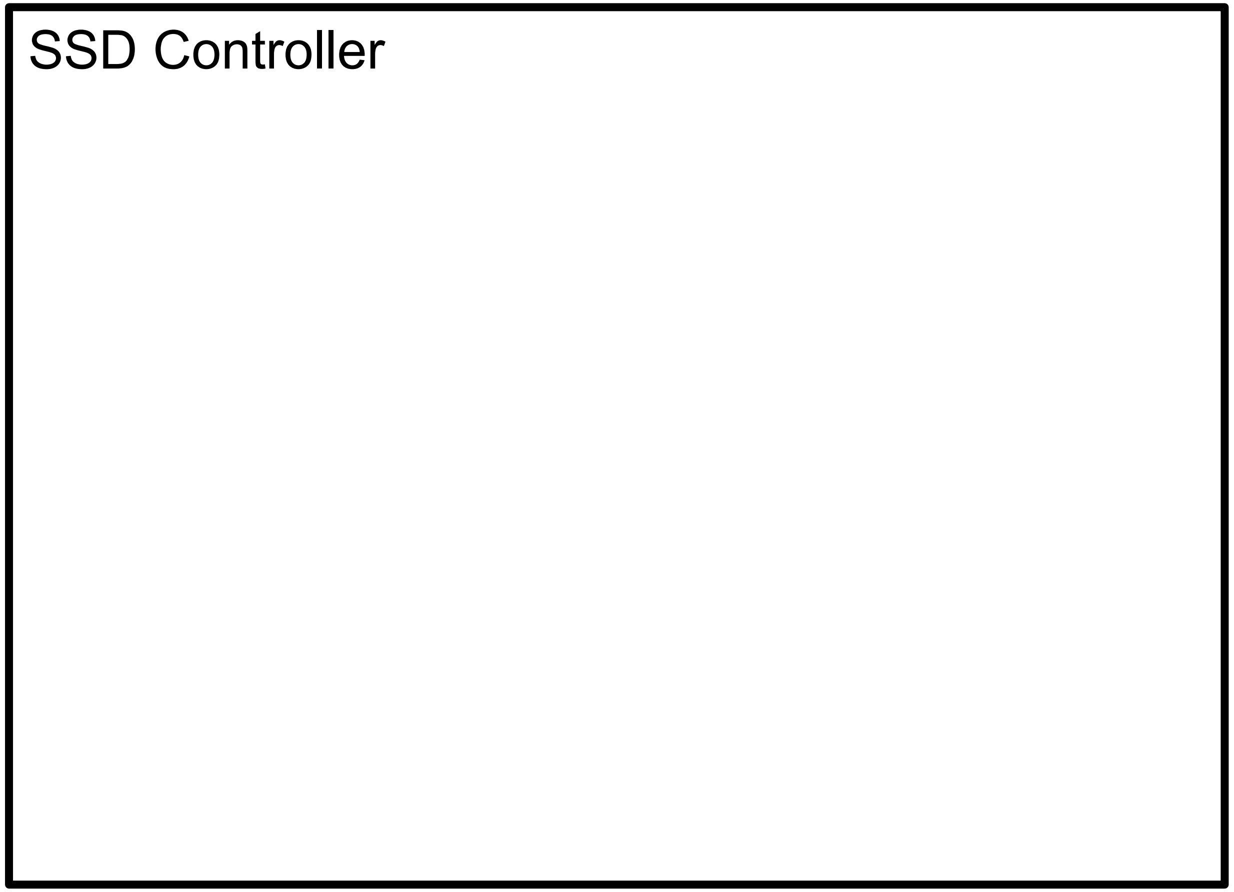
In-Storage Computing

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

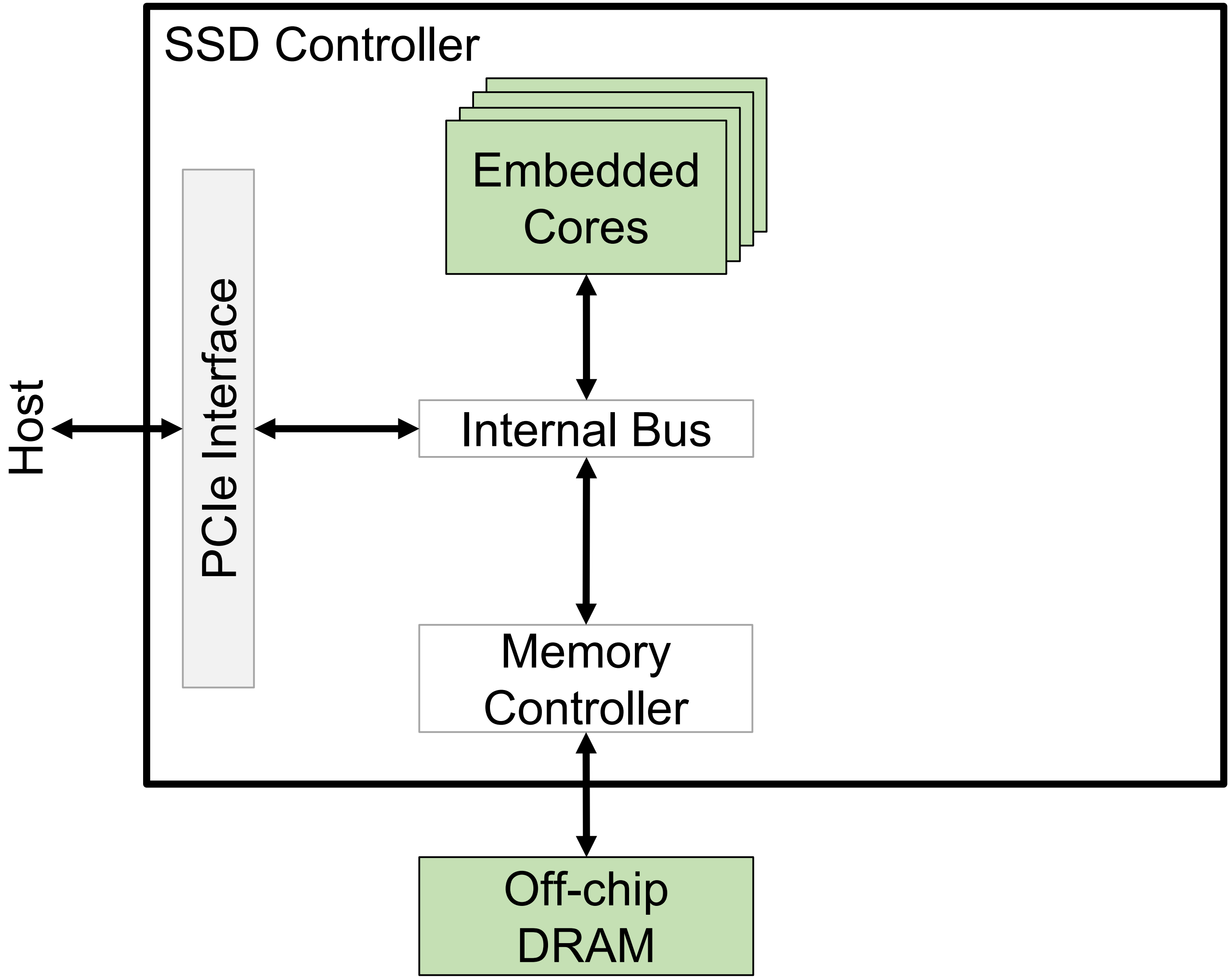


In-storage computing offers an effective solution to alleviate the I/O bottleneck

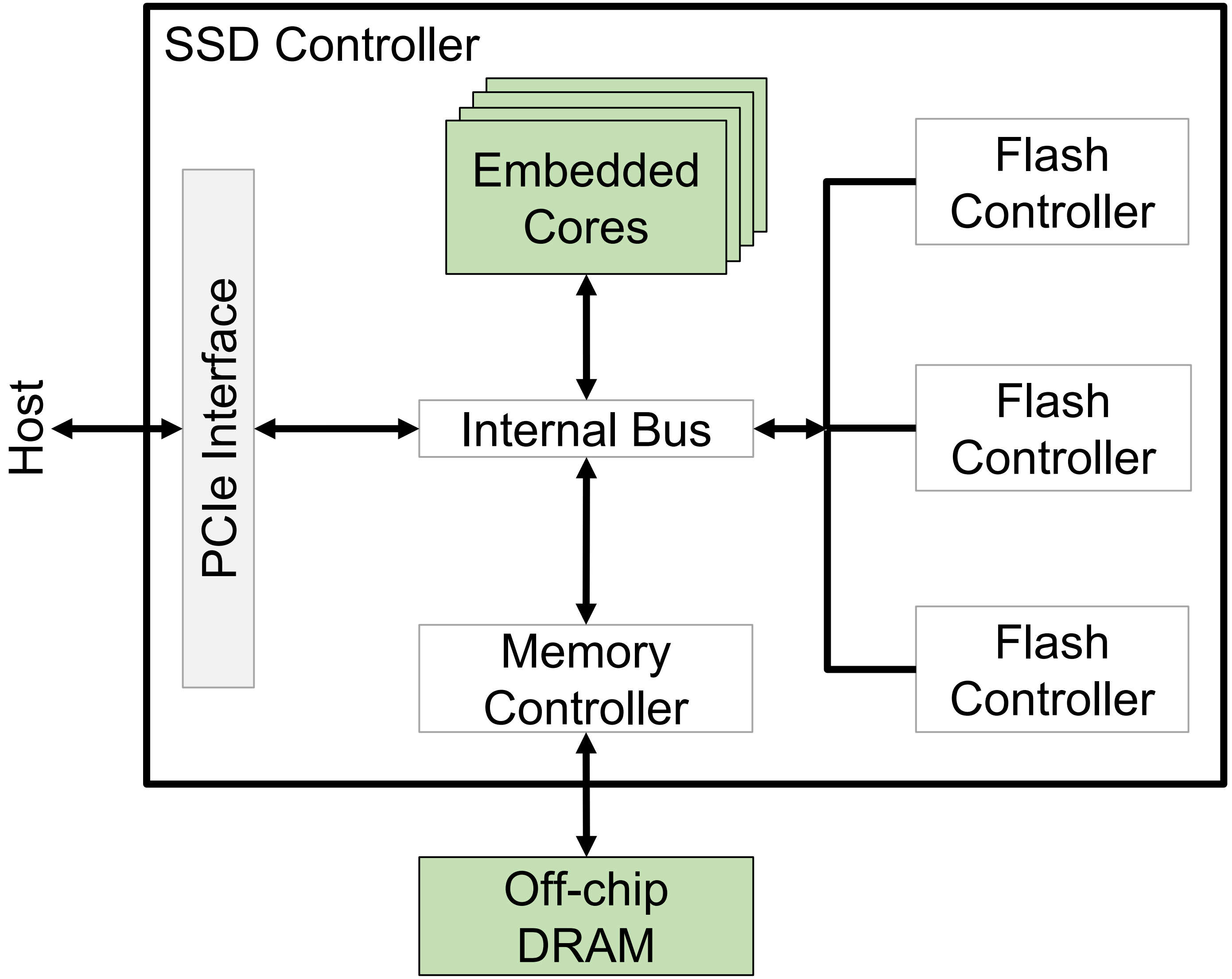
SSD Architecture for In-Storage Computing



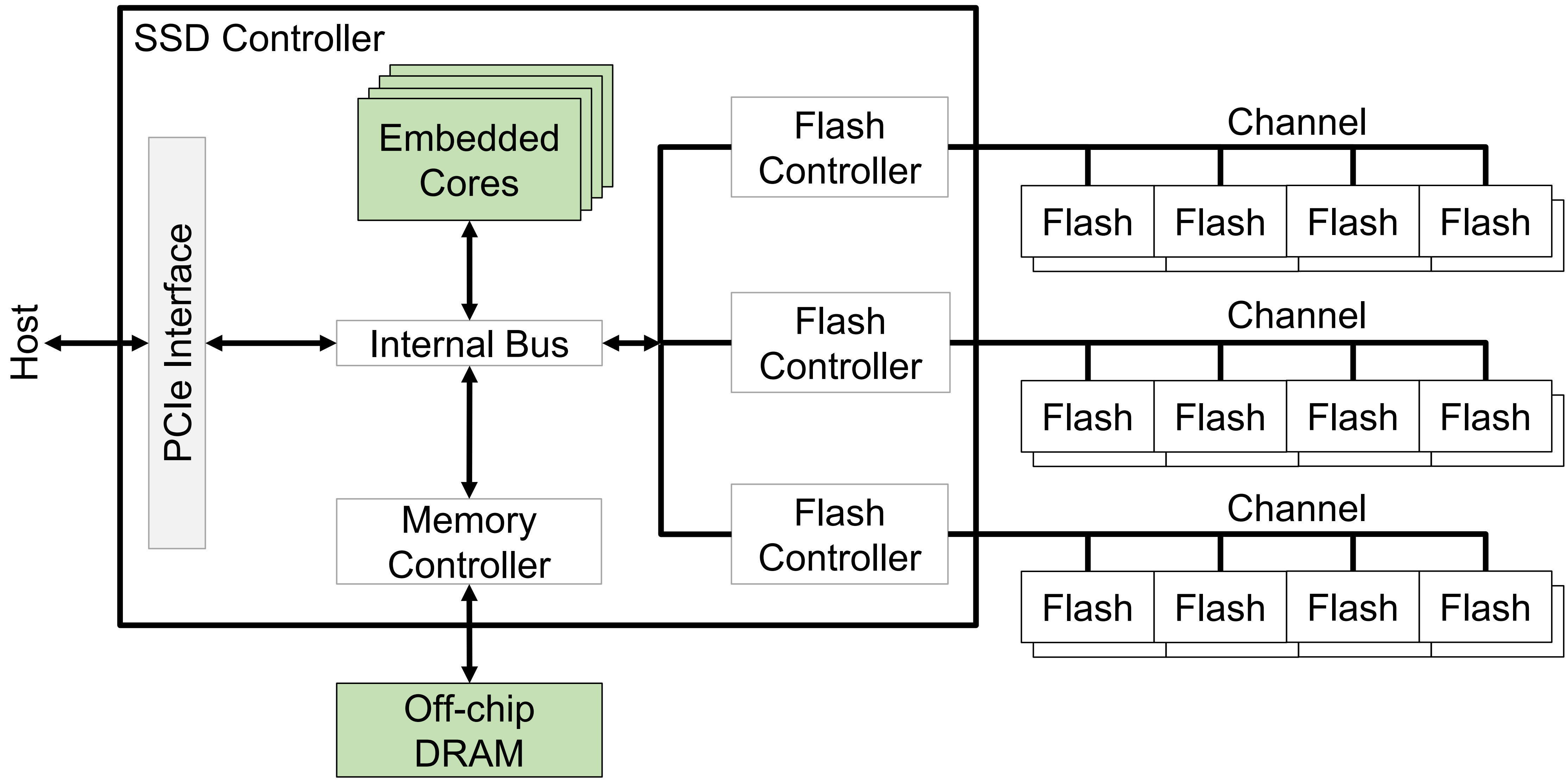
SSD Architecture for In-Storage Computing



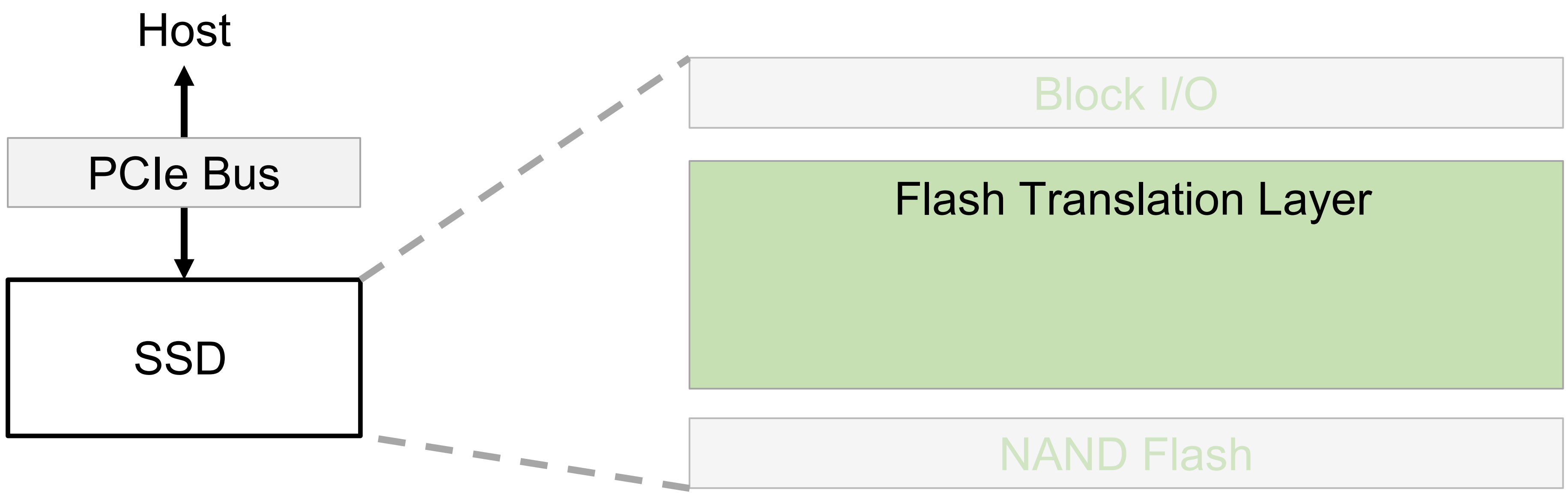
SSD Architecture for In-Storage Computing



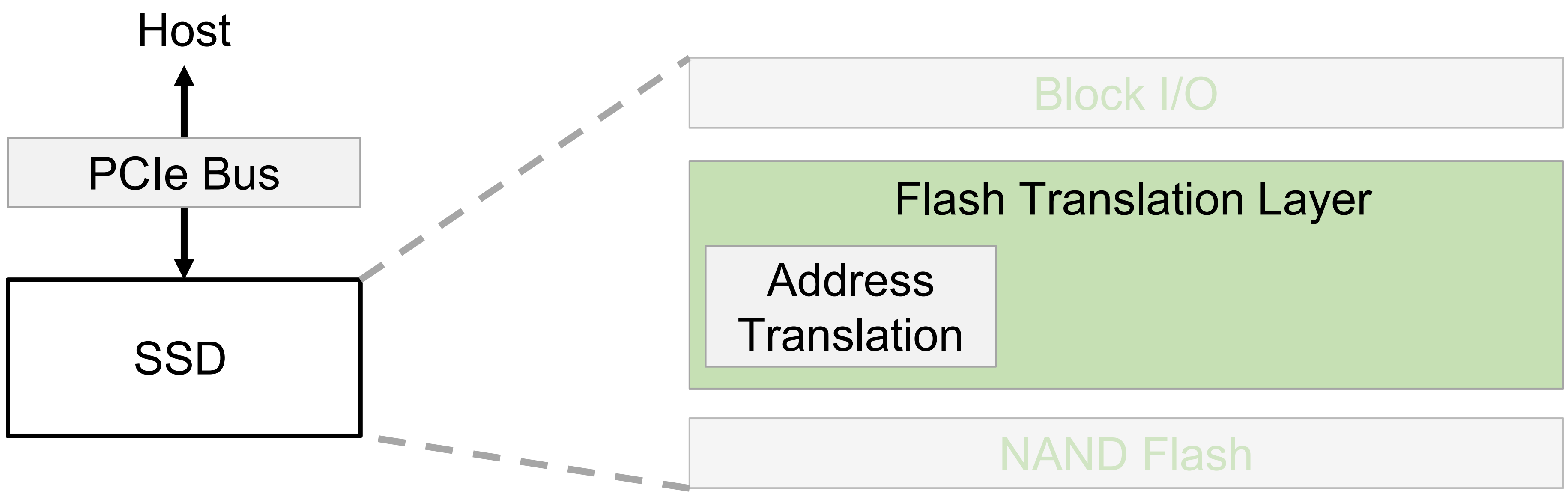
SSD Architecture for In-Storage Computing



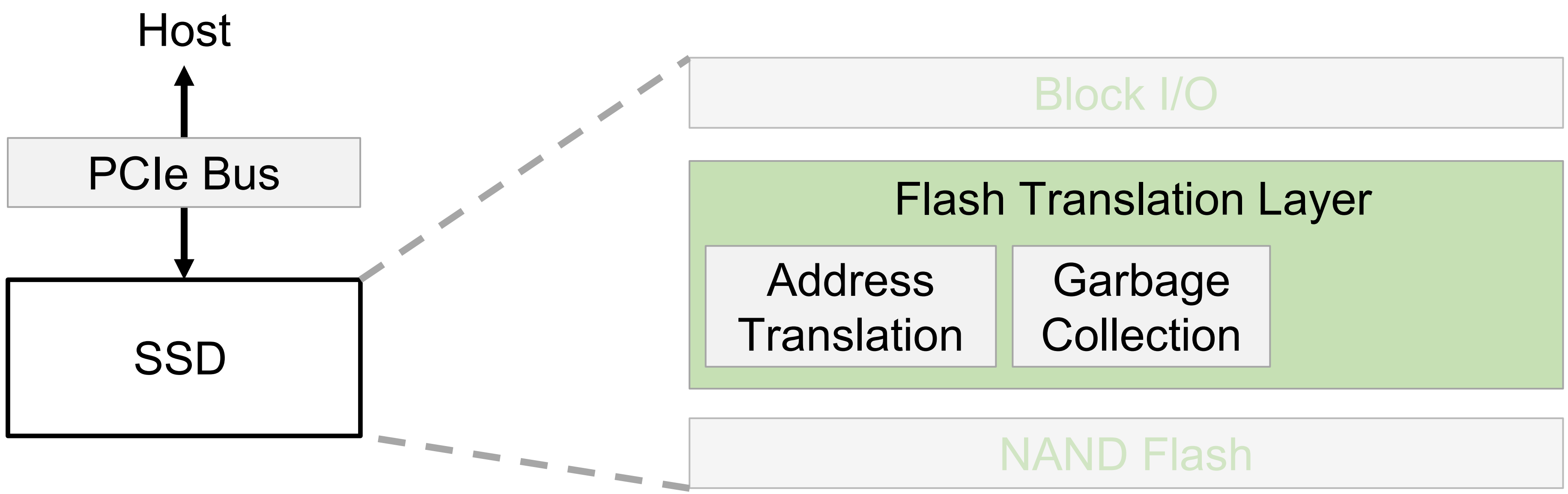
SSD Architecture for In-Storage Computing



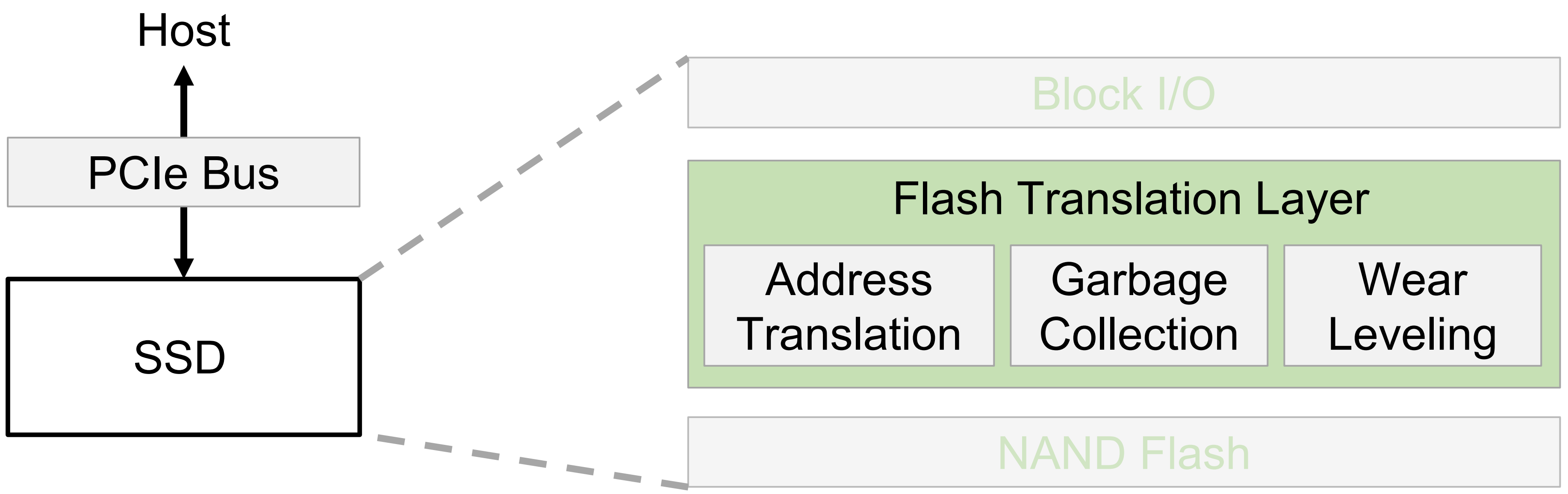
SSD Architecture for In-Storage Computing



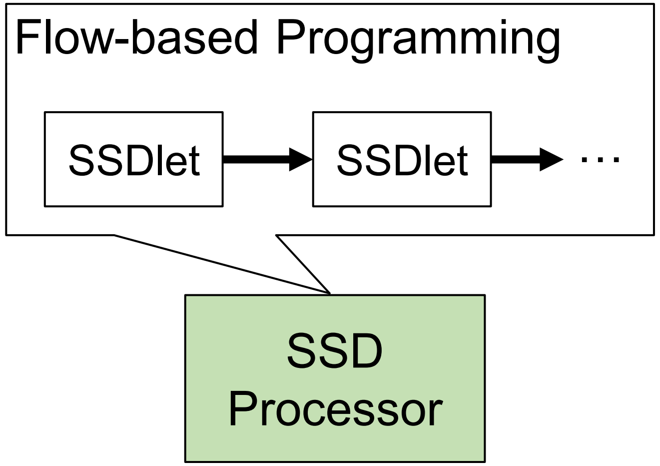
SSD Architecture for In-Storage Computing



SSD Architecture for In-Storage Computing

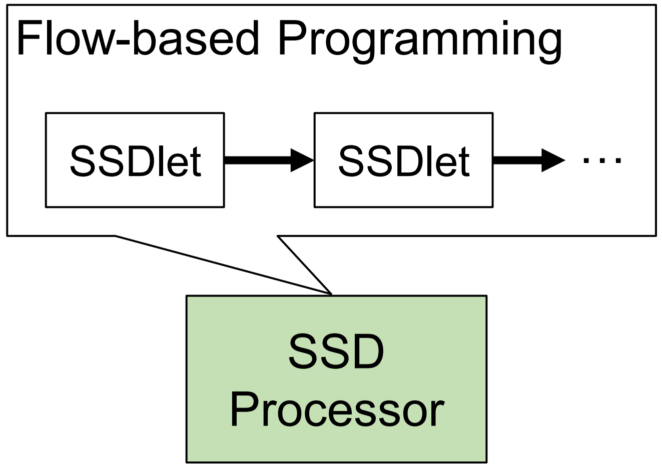


State-of-the-Art Frameworks for In-Storage Computing

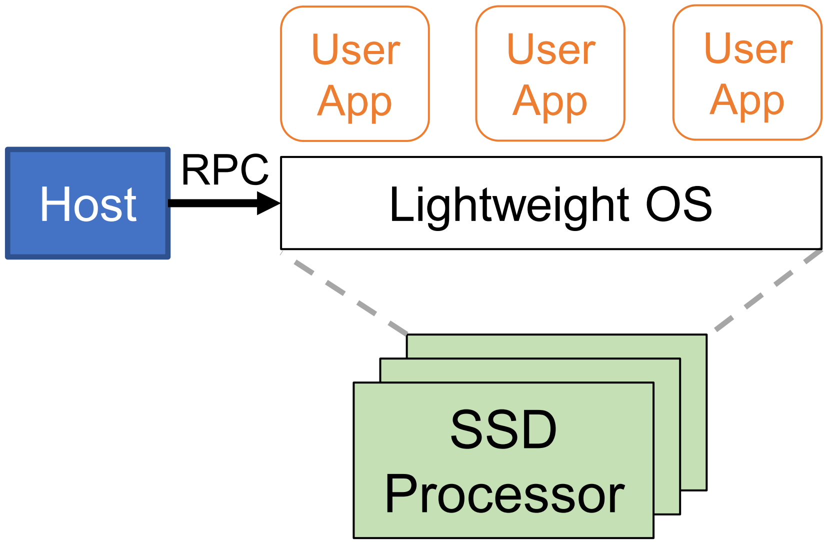


MapReduce-based Framework

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

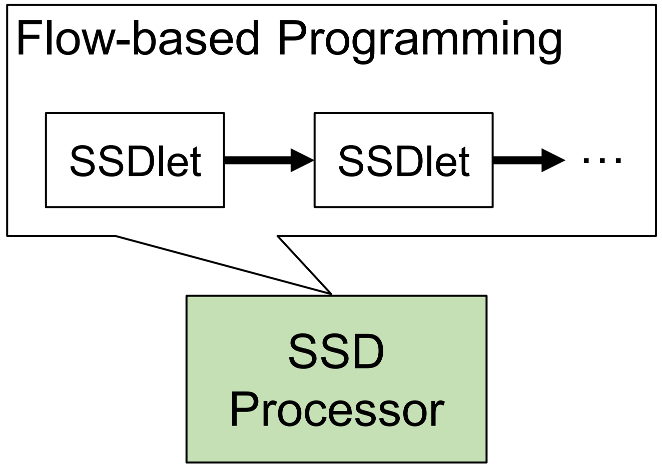


MapReduce-based Framework

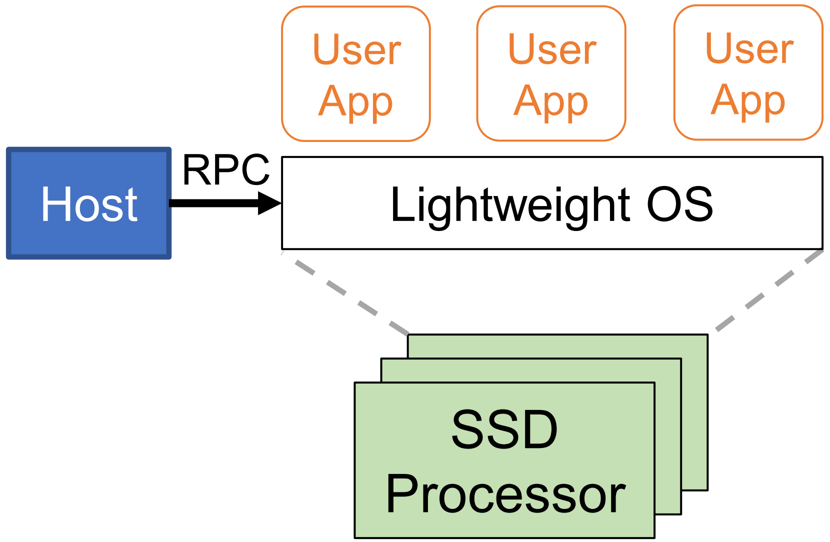


RPC-based Offloading

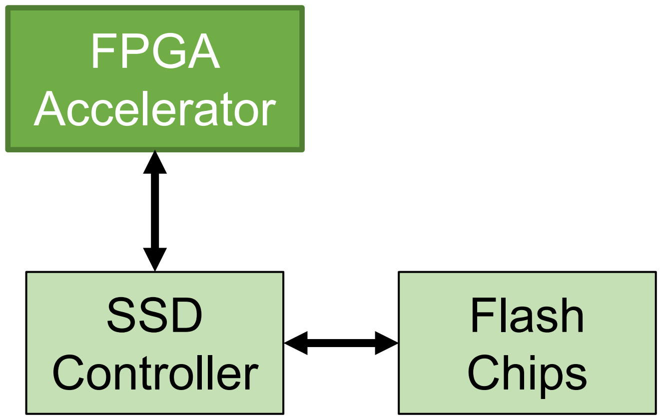
State-of-the-Art Frameworks for In-Storage Computing



MapReduce-based Framework

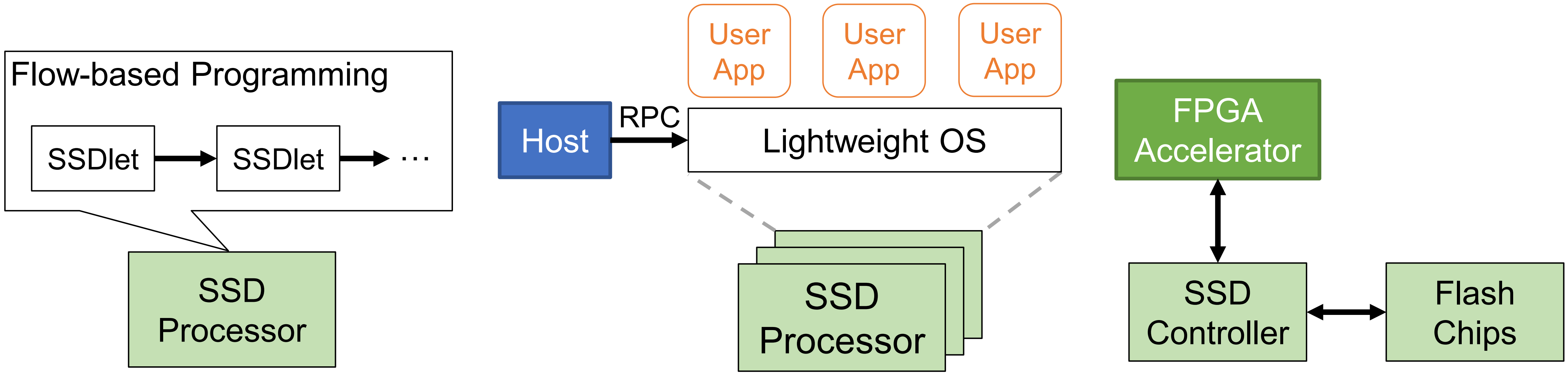


RPC-based Offloading



Industry SmartSSD

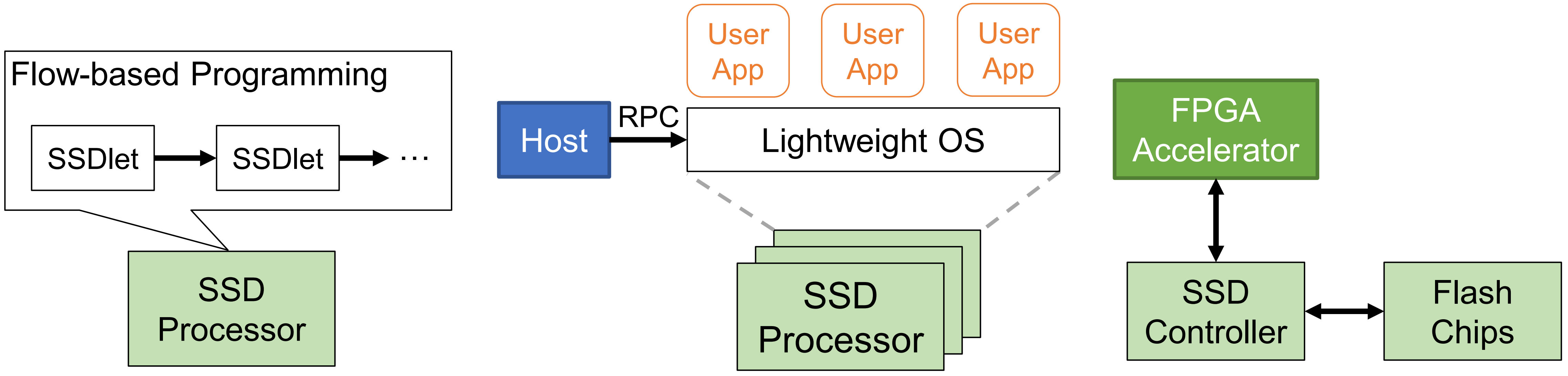
State-of-the-Art Frameworks for In-Storage Computing



MapReduce-based Framework | RPC-based Offloading | Industry SmartSSD

Most of the existing frameworks focus on performance and programmability

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



MapReduce-based Framework | RPC-based Offloading | Industry SmartSSD

Most of the existing frameworks focus on performance and programmability

Few of them consider security as the first-class citizen

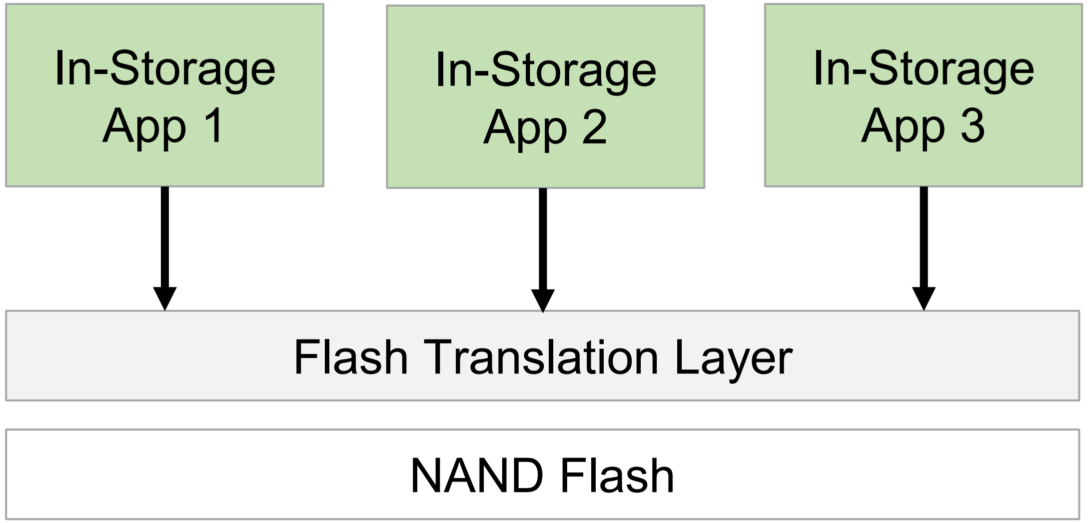
Why Should We Secure In-Storage Computing?

In-Storage App 1

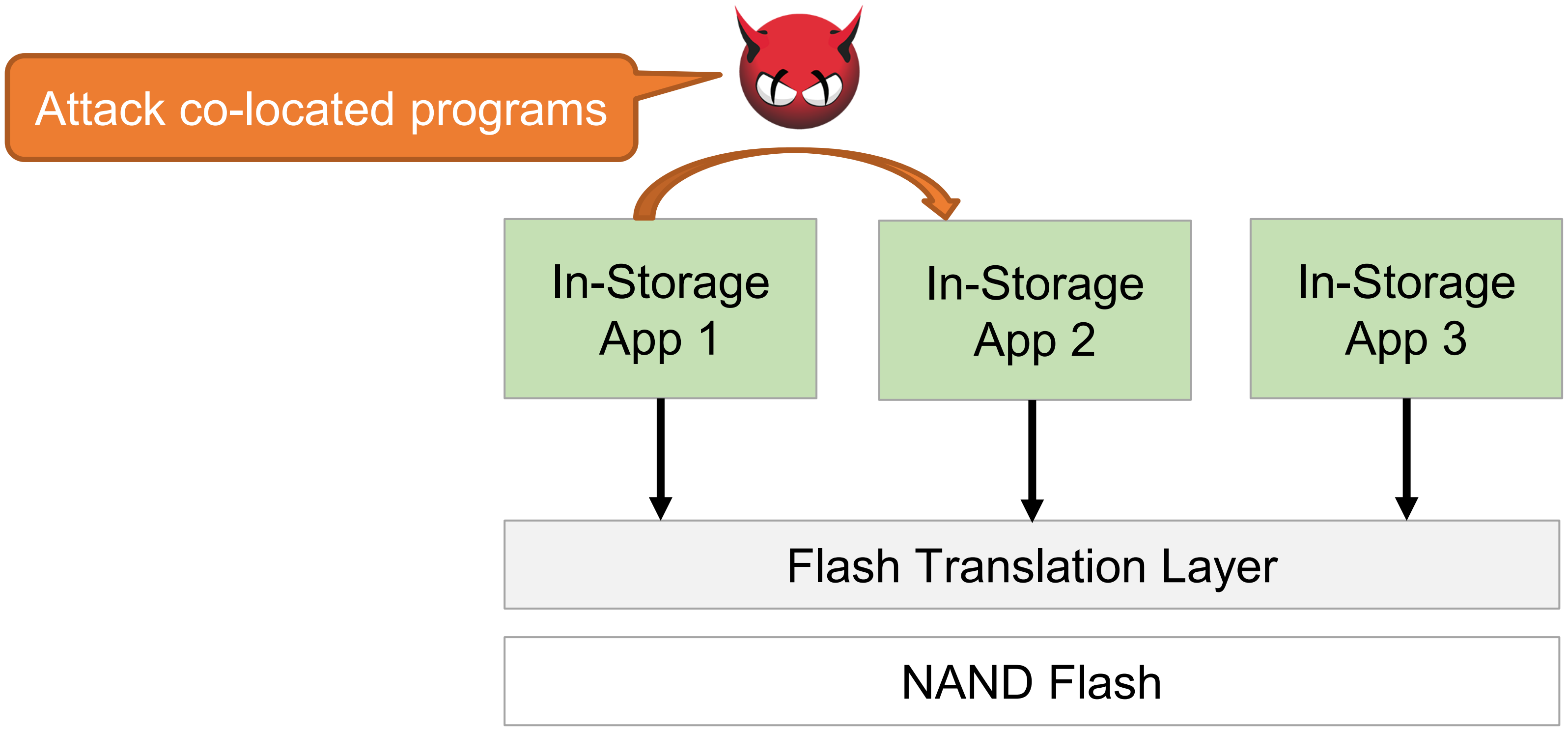
In-Storage App 2

In-Storage App 3

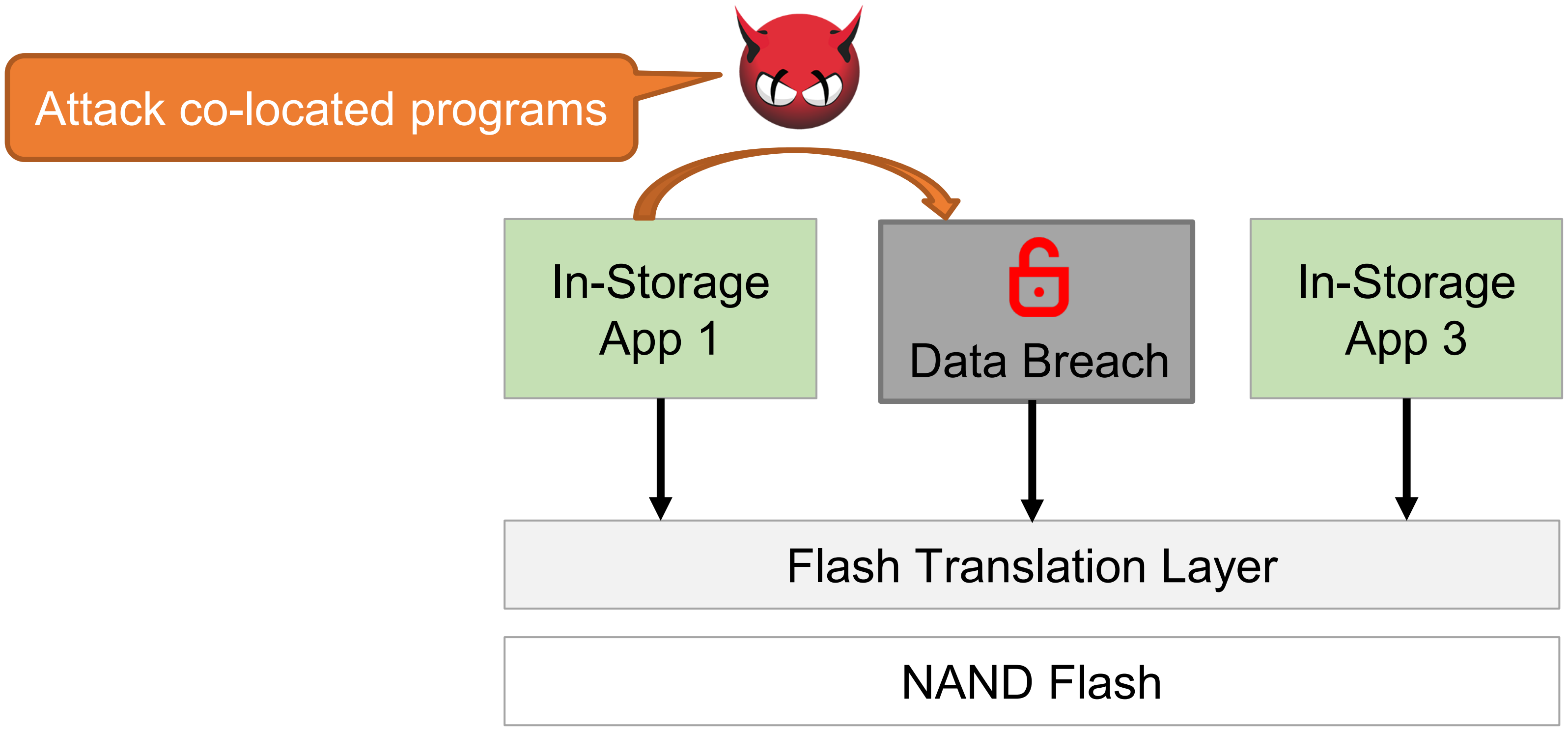
Why Should We Secure In-Storage Computing?



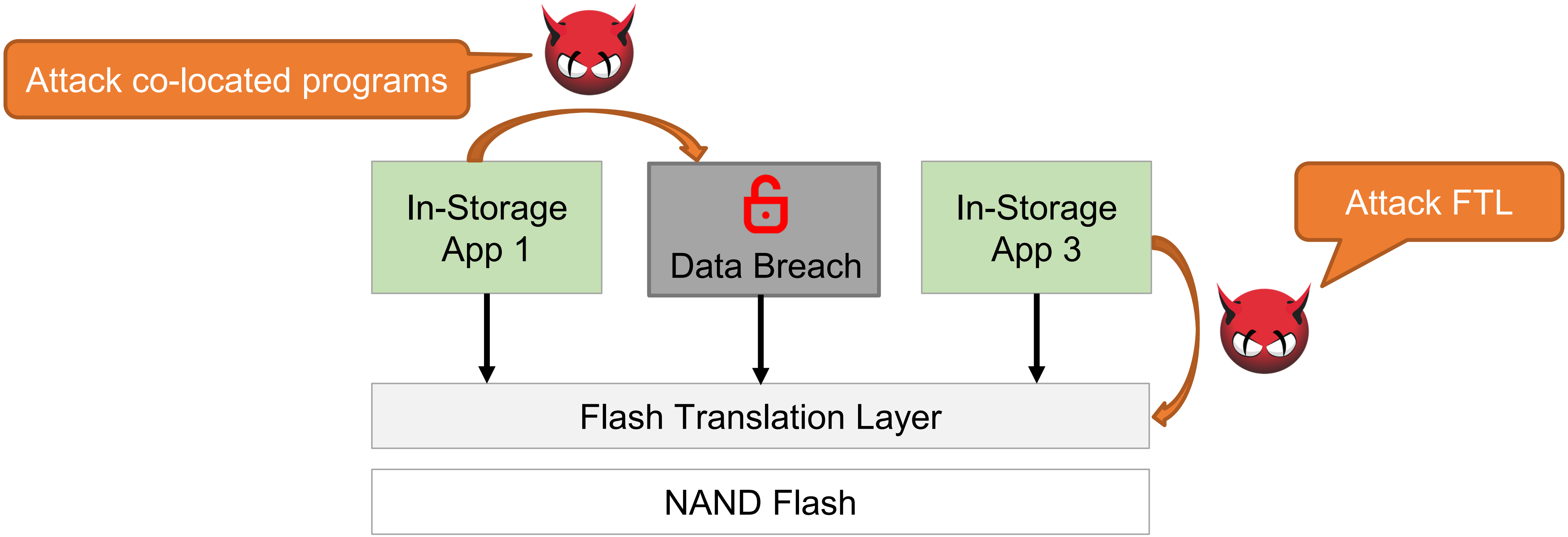
Why Should We Secure In-Storage Computing?



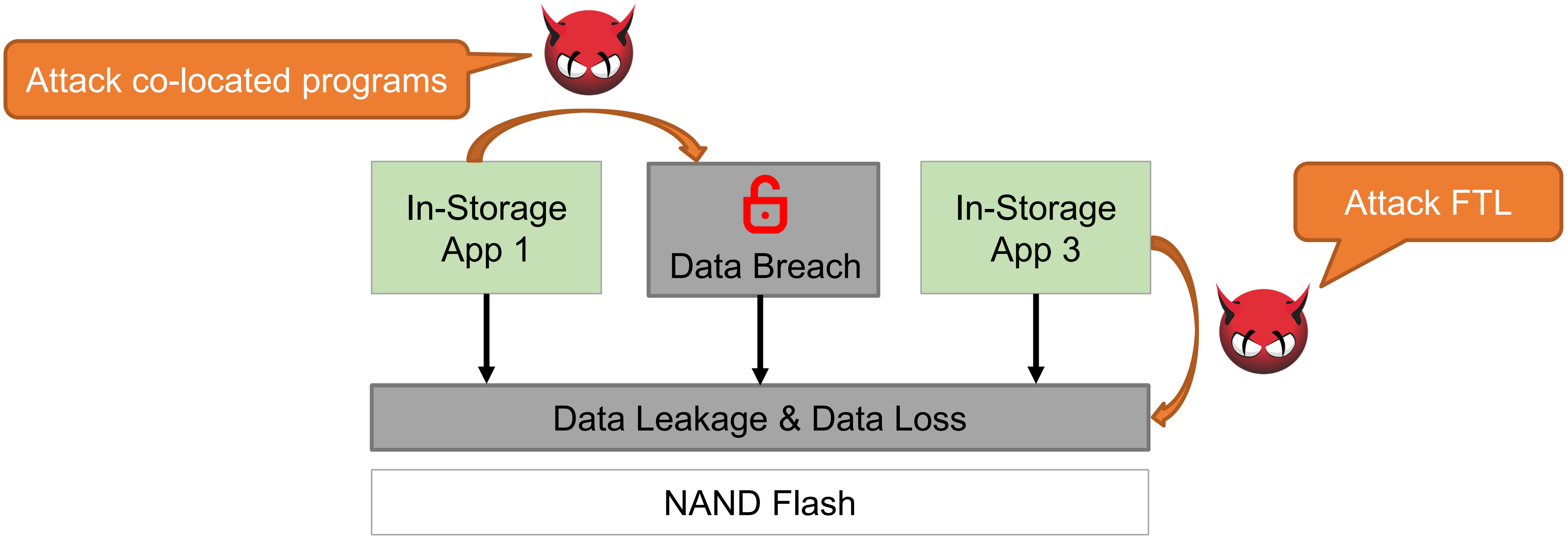
Why Should We Secure In-Storage Computing?



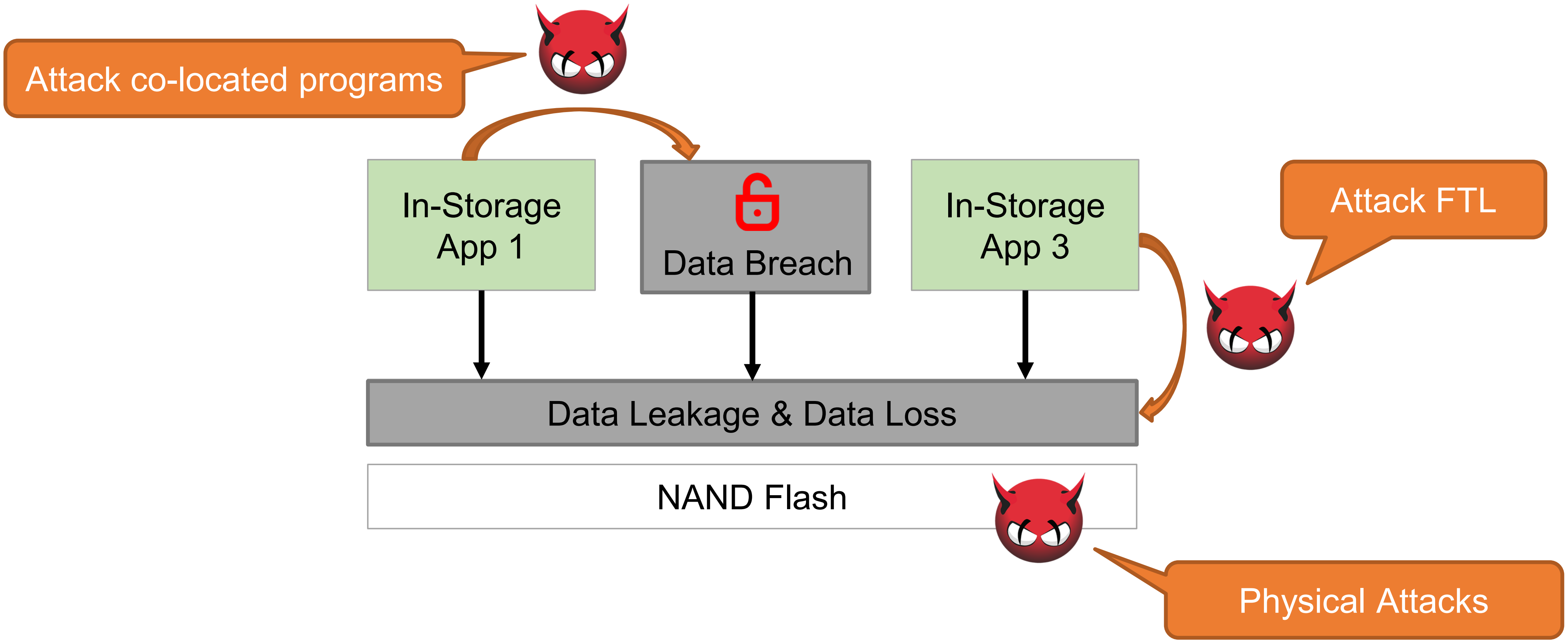
Why Should We Secure In-Storage Computing?



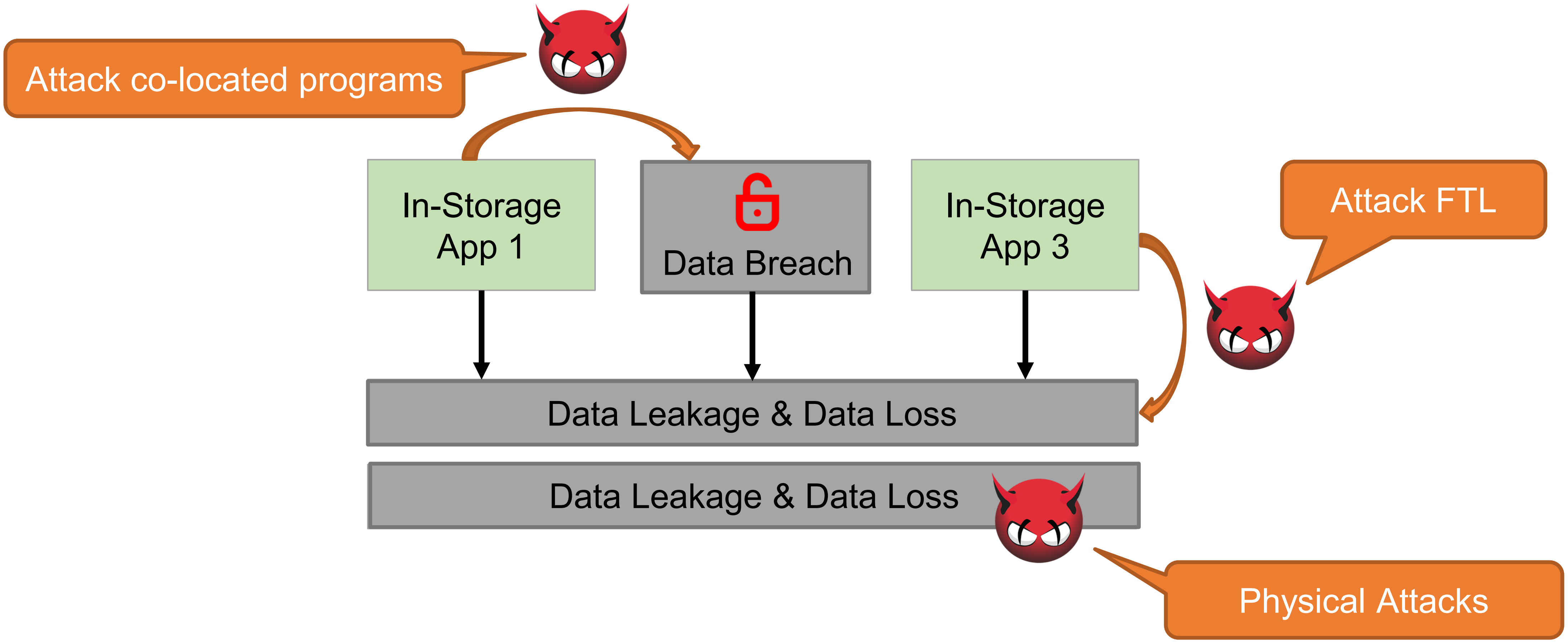
Why Should We Secure In-Storage Computing?



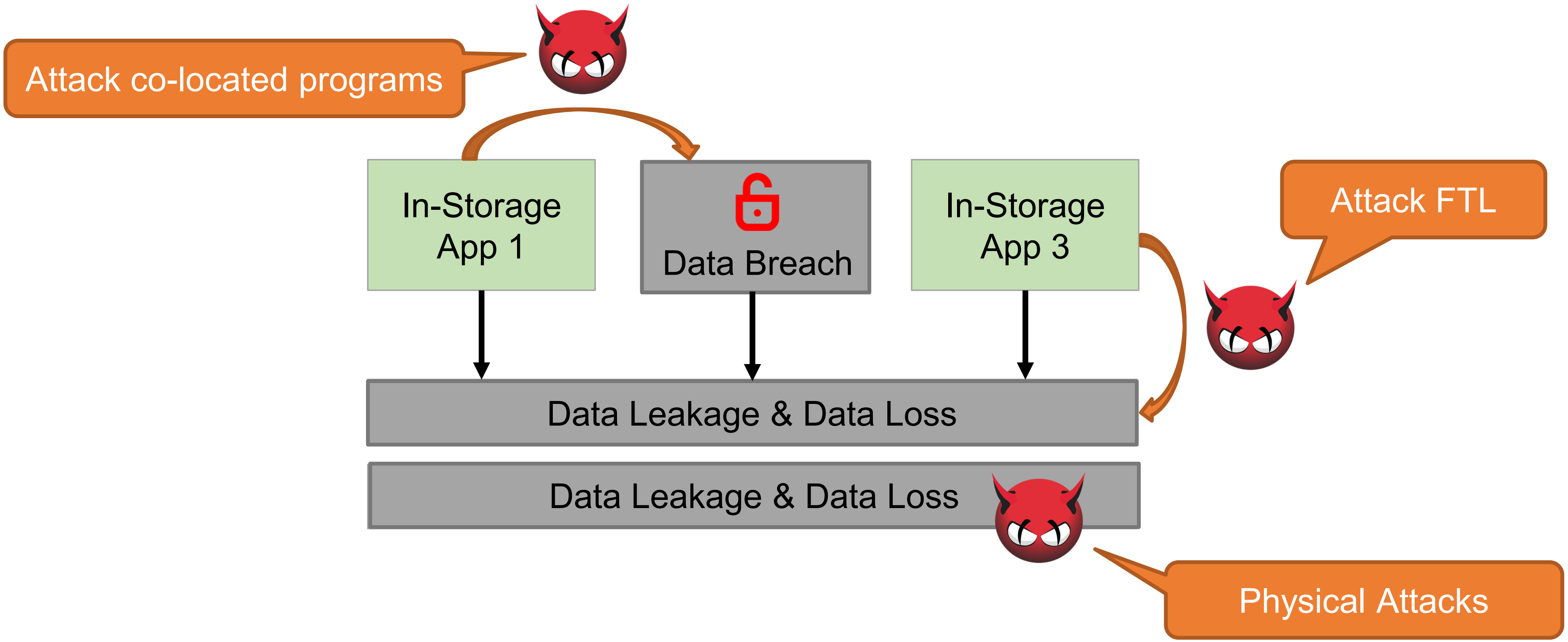
Why Should We Secure In-Storage Computing?



Why Should We Secure In-Storage Computing?



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

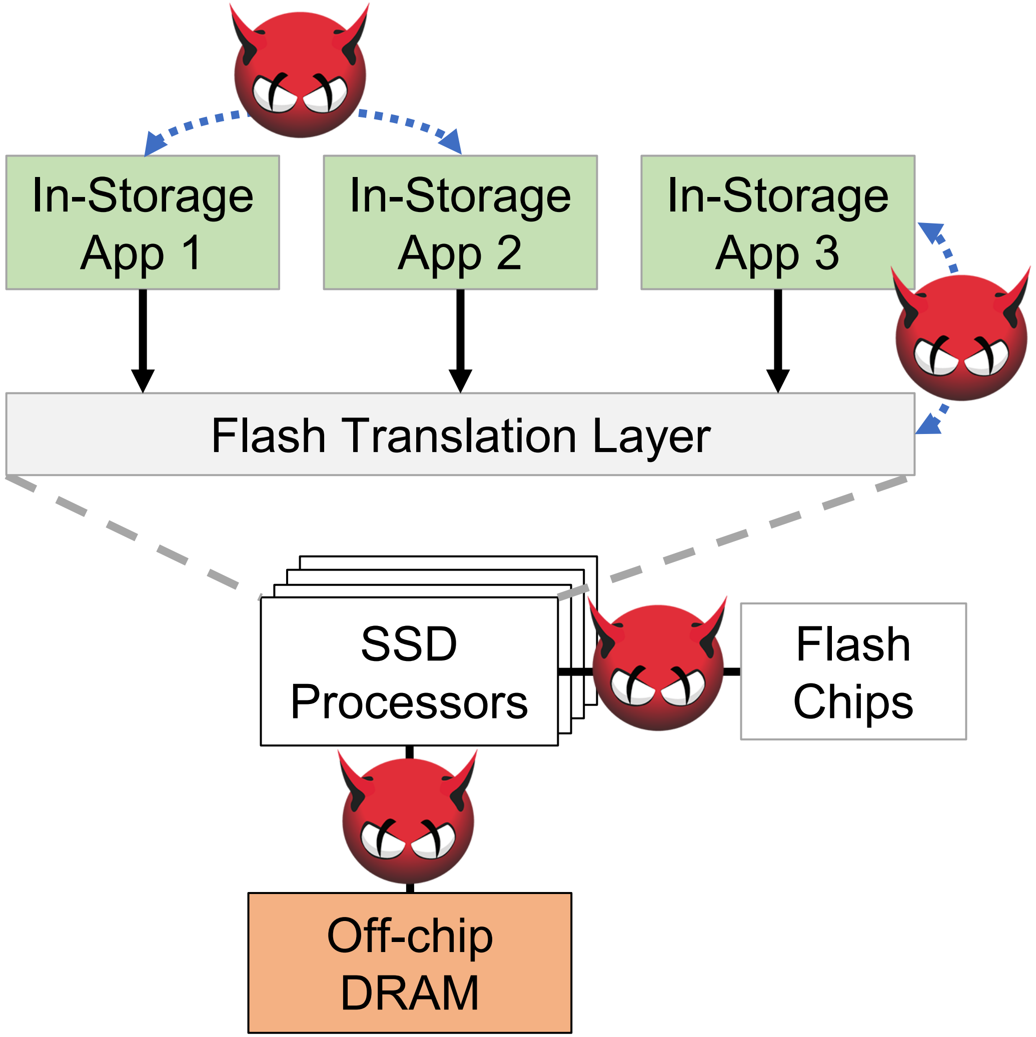
Intel SGX is not available in storage processors

Existing TEEs Do Not Work For In-Storage Computing

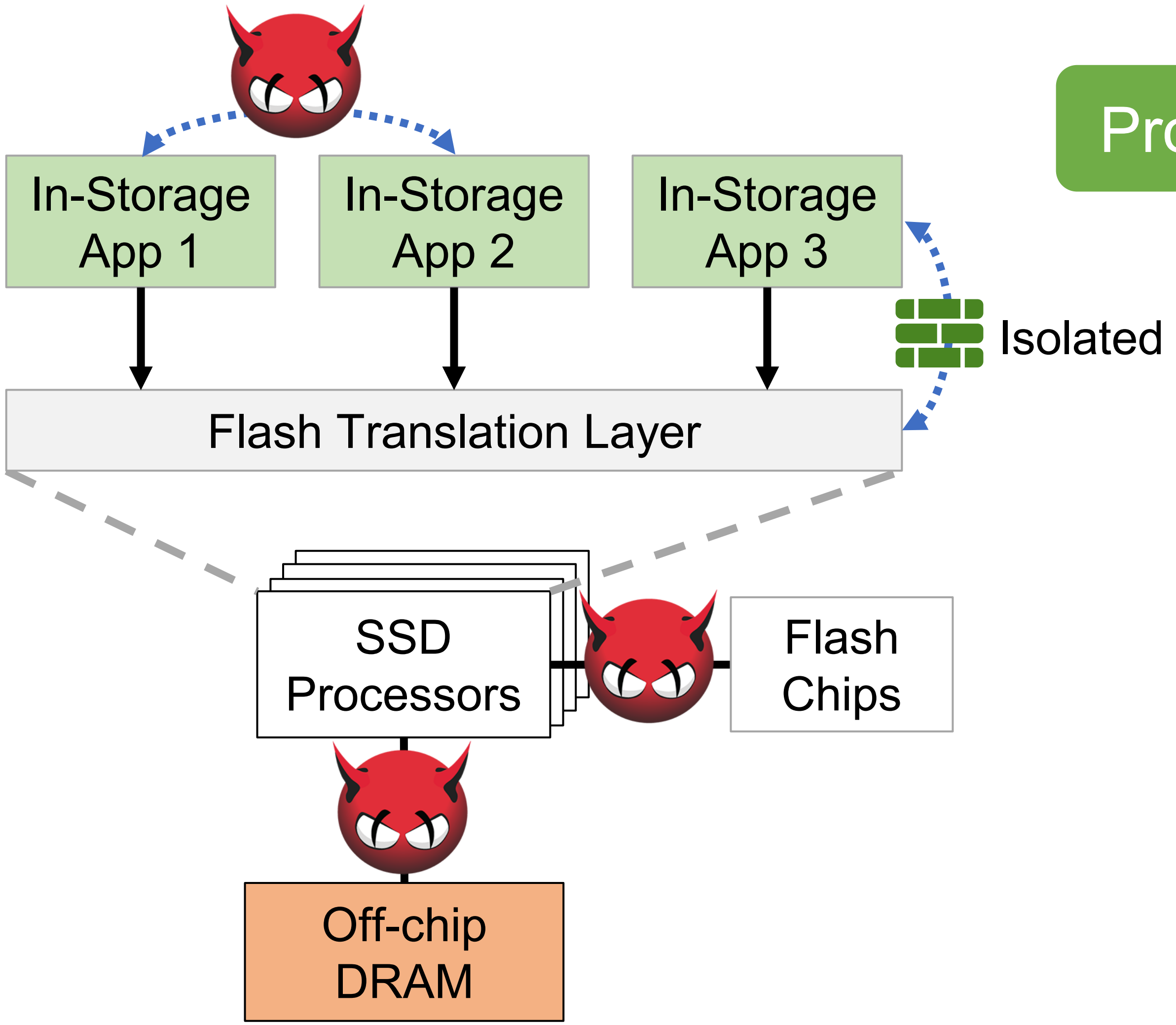
Intel SGX is not available in storage processors

Unclear how to apply ARM TrustZone to in-storage computing

IceClave: A Trusted Execution Environment for In-Storage Computing

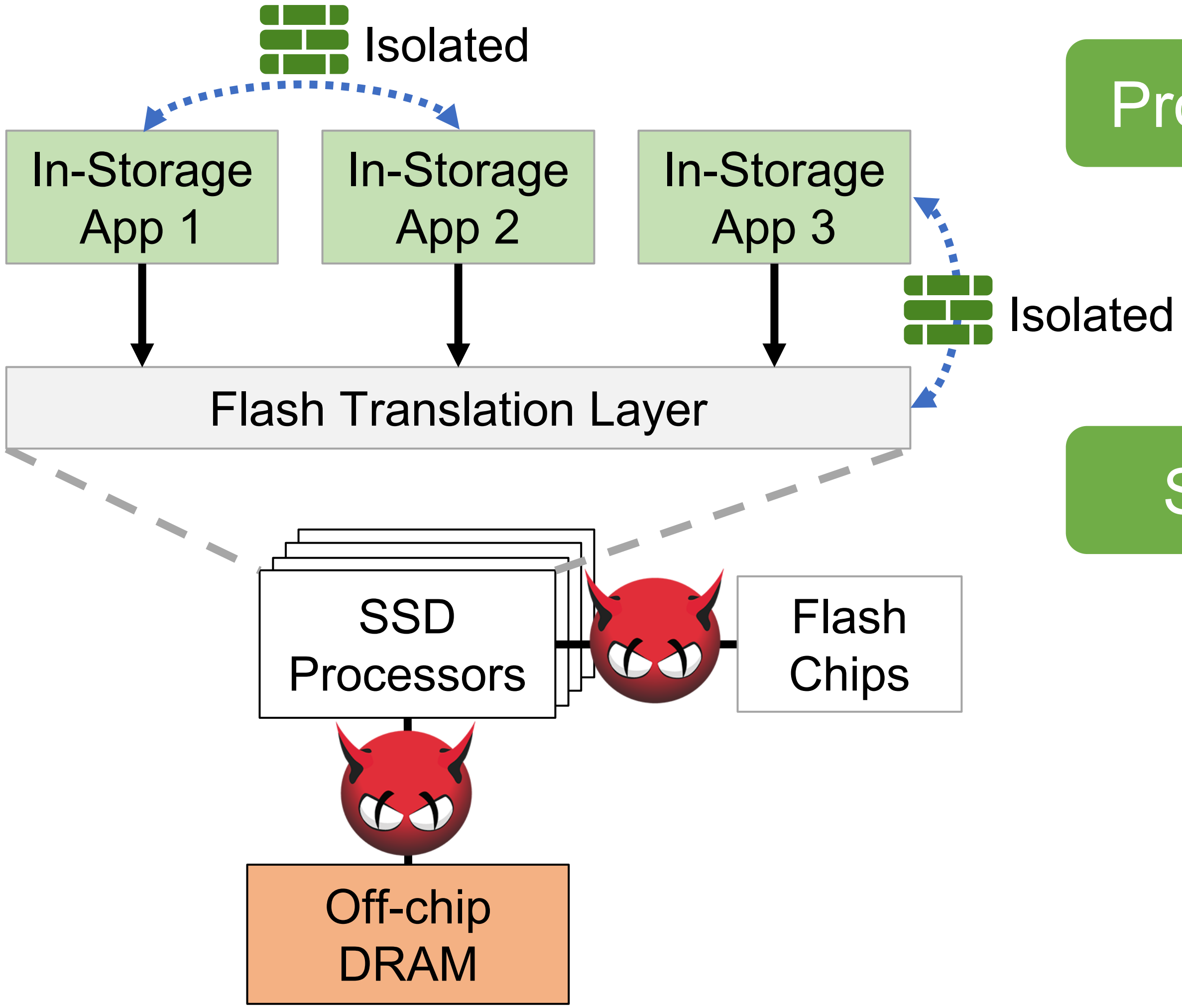


IceClave: A Trusted Execution Environment for In-Storage Computing



Protecting FTL from malicious in-storage apps

IceClave: A Trusted Execution Environment for In-Storage Computing

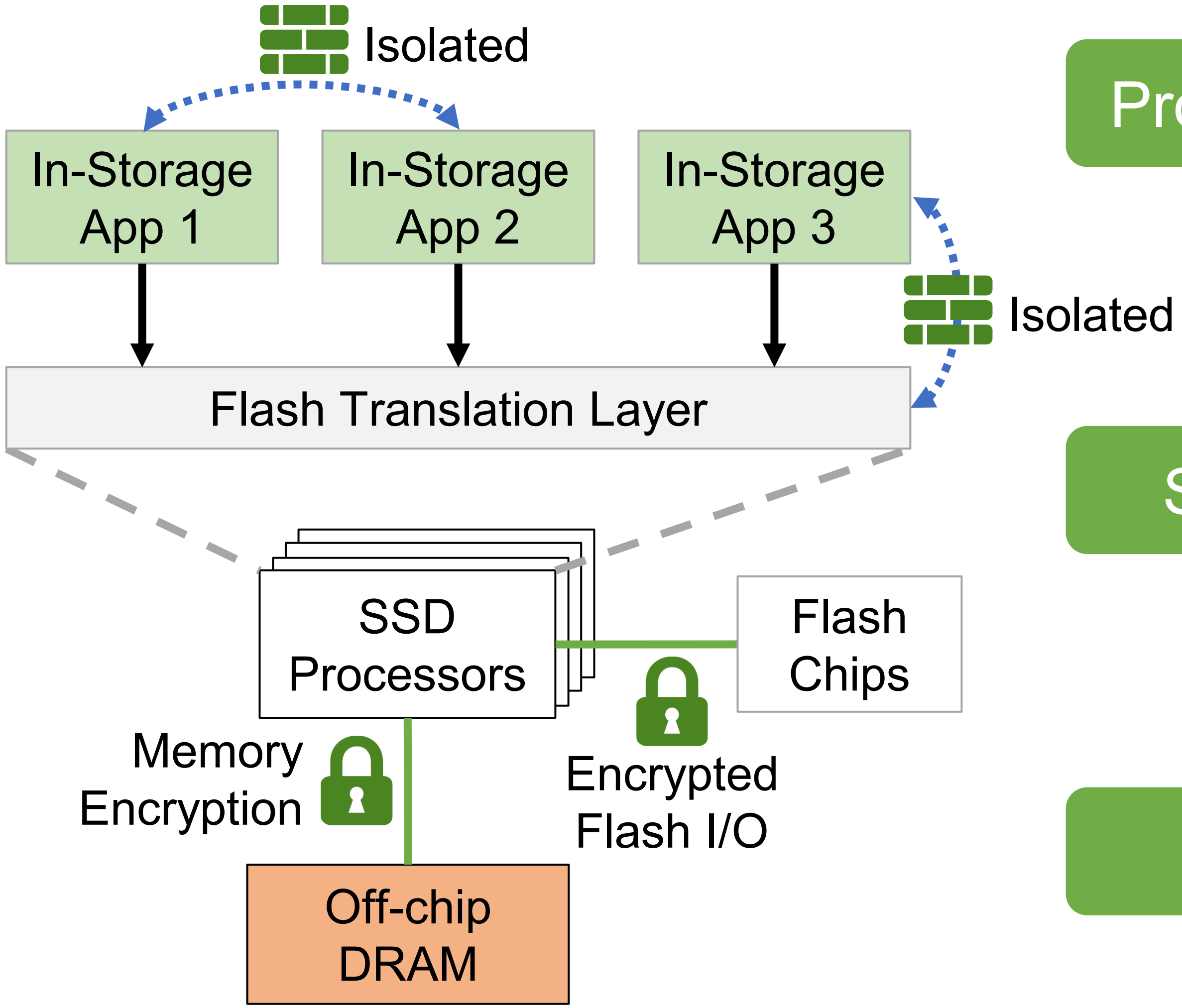


Protecting FTL from malicious in-storage apps

+

Security isolation between in-storage apps

IceClave: A Trusted Execution Environment for In-Storage Computing



Protecting FTL from malicious in-storage apps

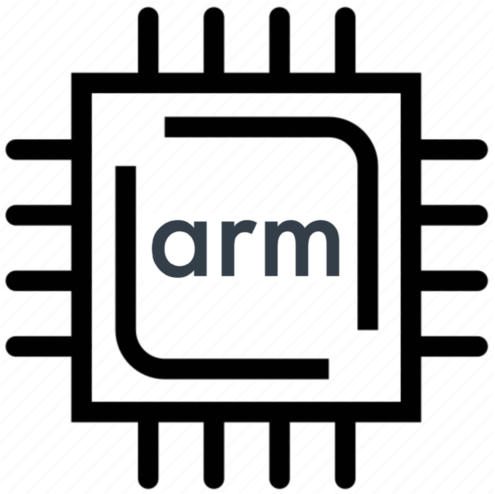
+

Security isolation between in-storage apps

+

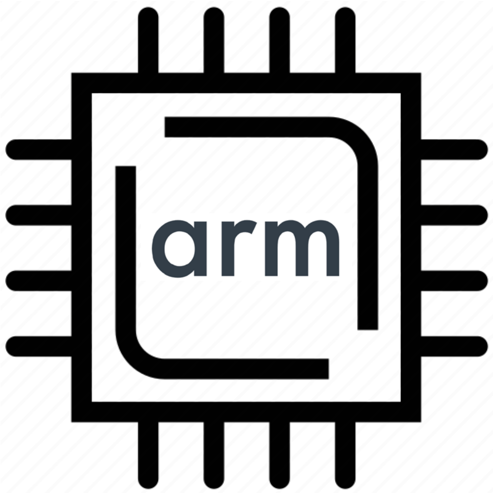
Securing data against physical attacks

IceClave Design Challenges



Bare-metal
Environment

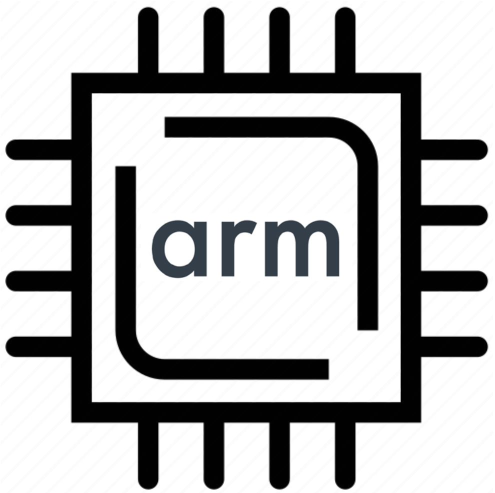
IceClave Design Challenges



Bare-metal Environment

Efficient Flash Access

IceClave Design Challenges

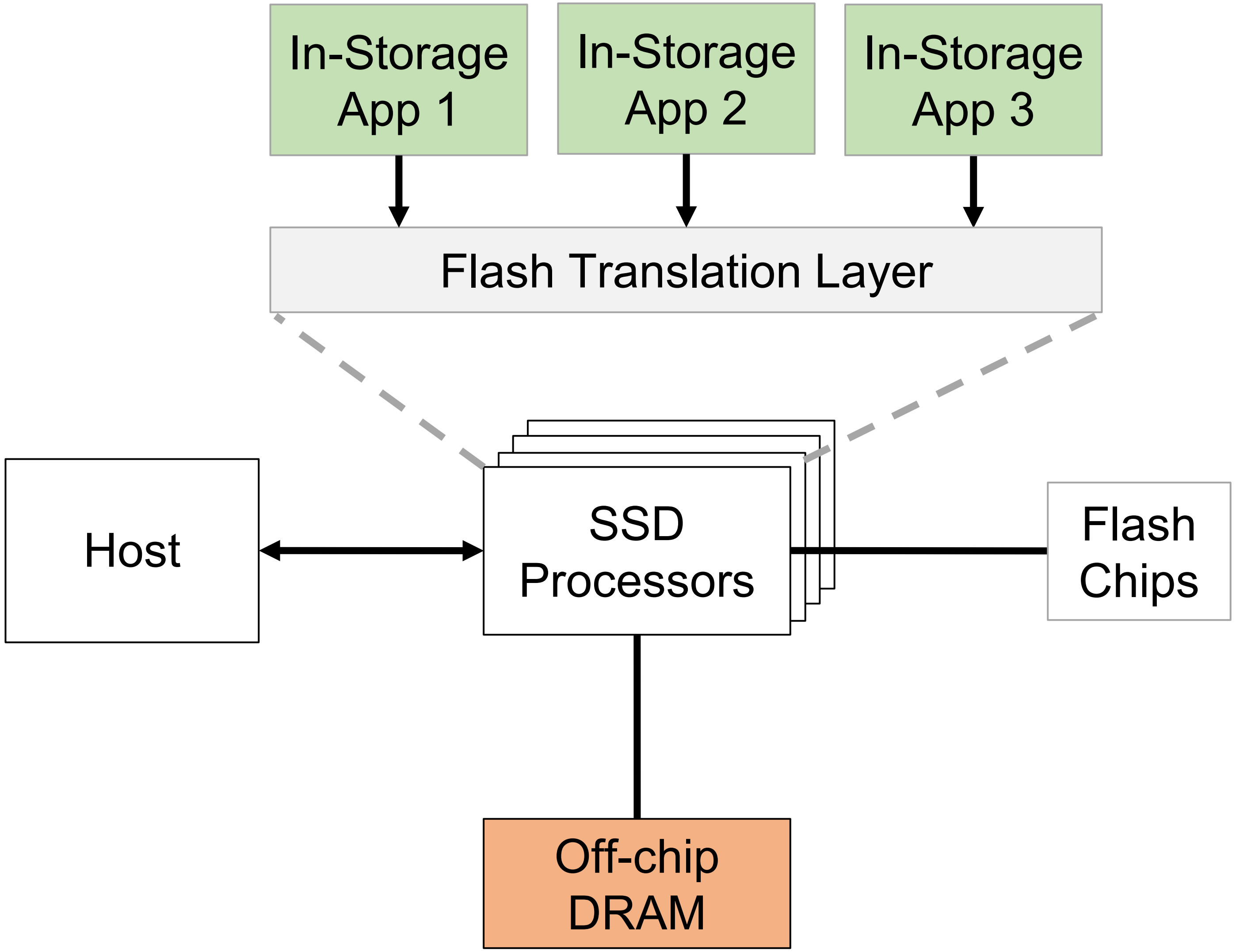


Bare-metal Environment

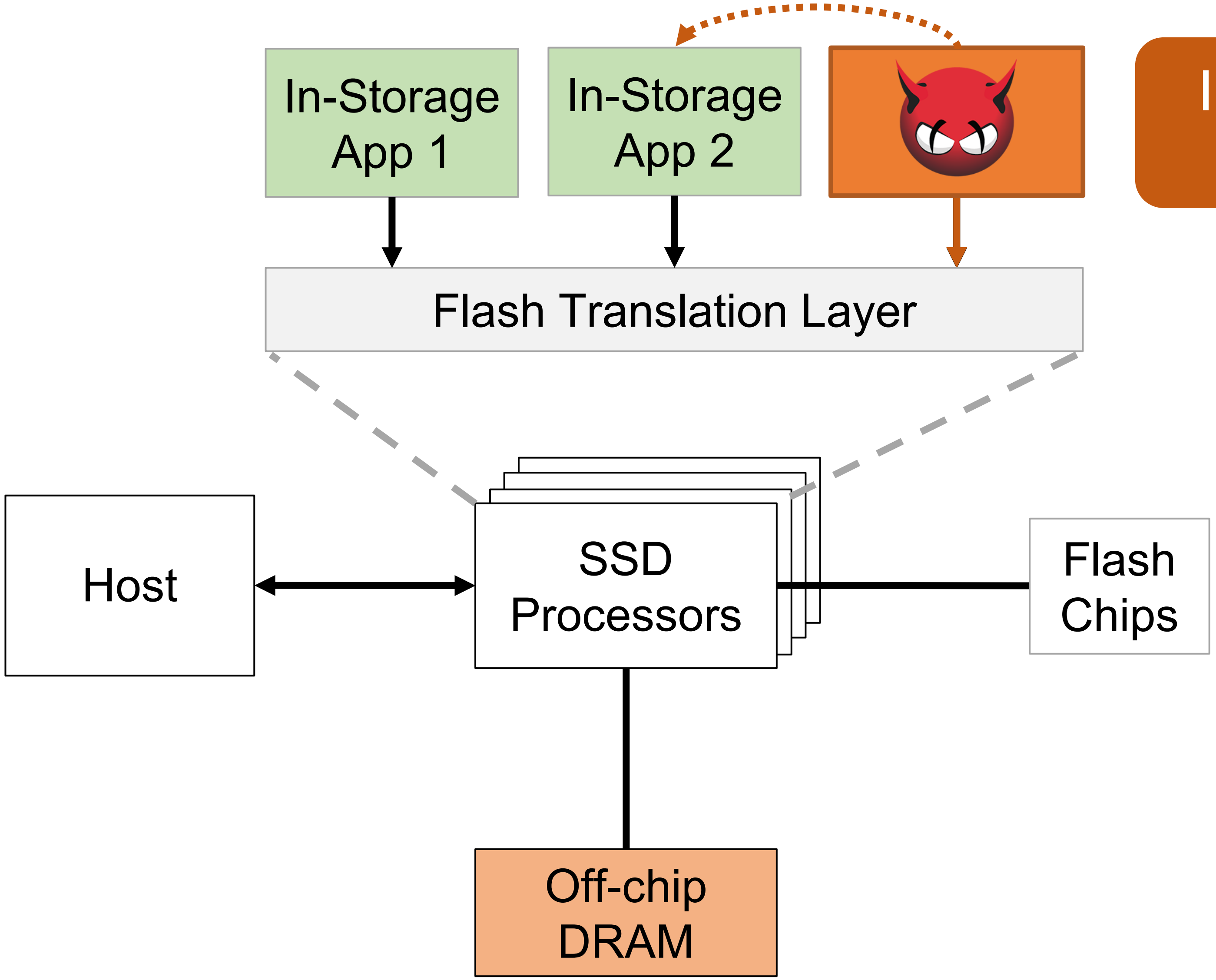
Efficient Flash Access

Limited Resources in SSD Device

Threat Model

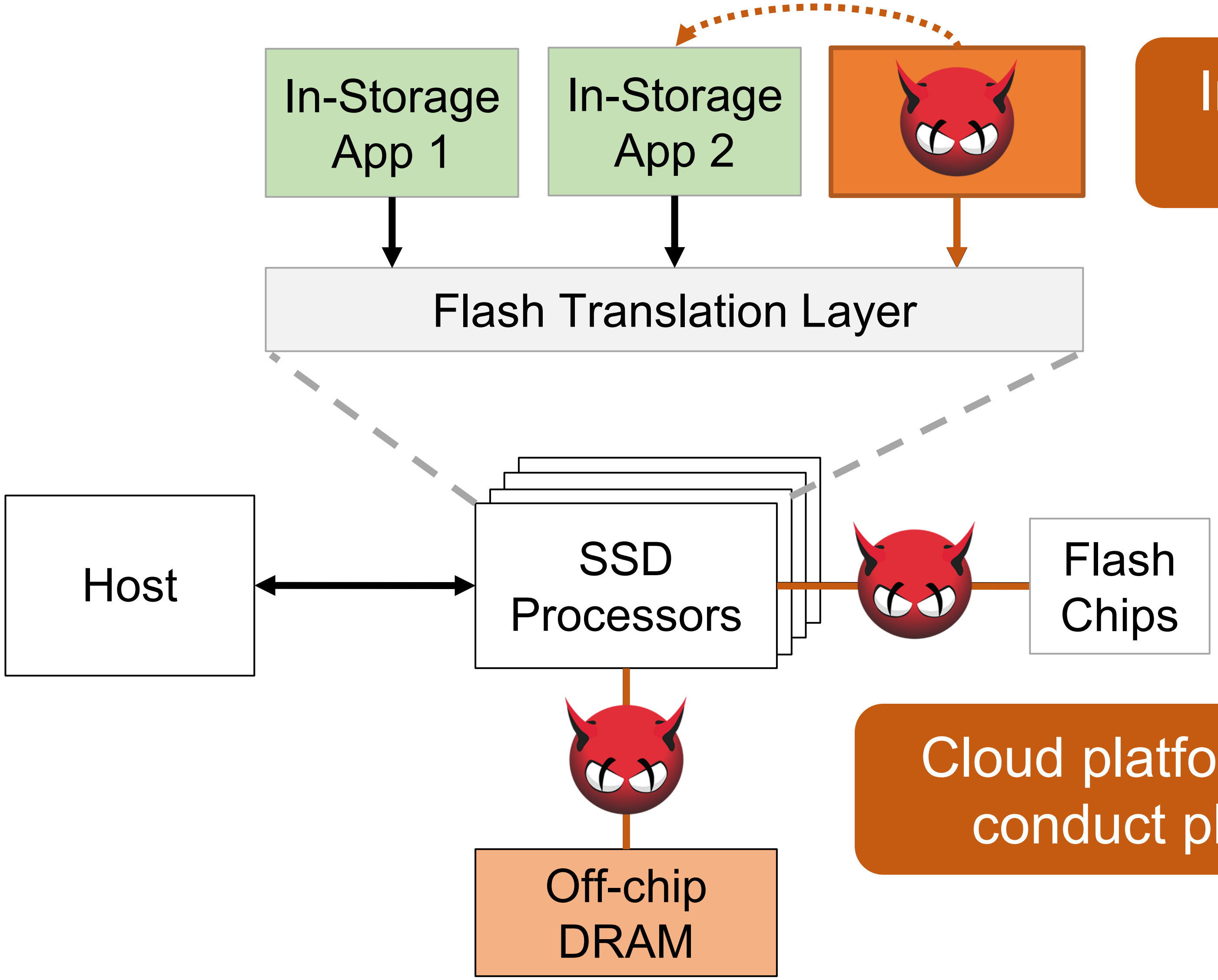


Threat Model



In-storage applications can be malicious

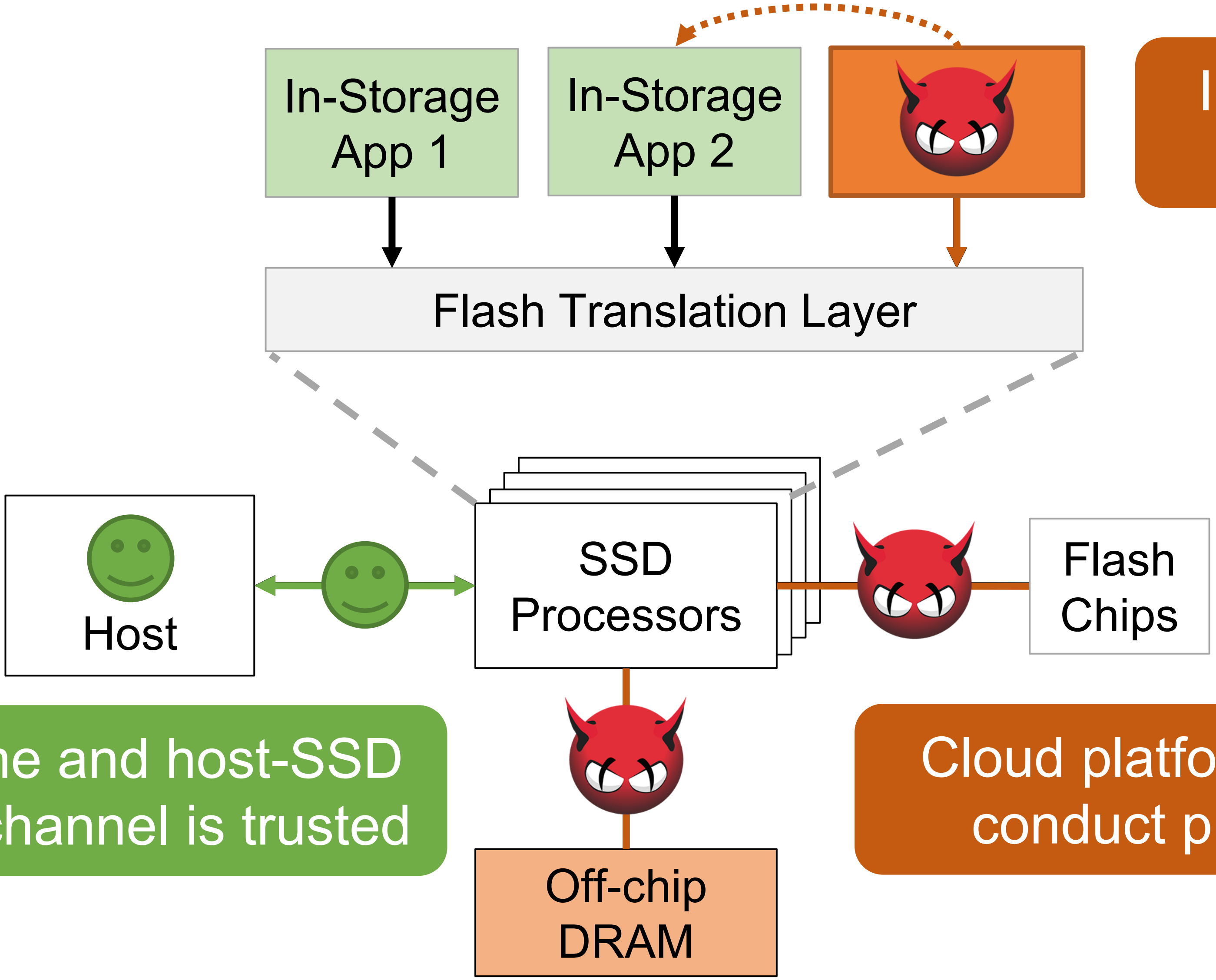
Threat Model



In-storage applications can be malicious

Cloud platform operator may conduct physical attacks

Threat Model

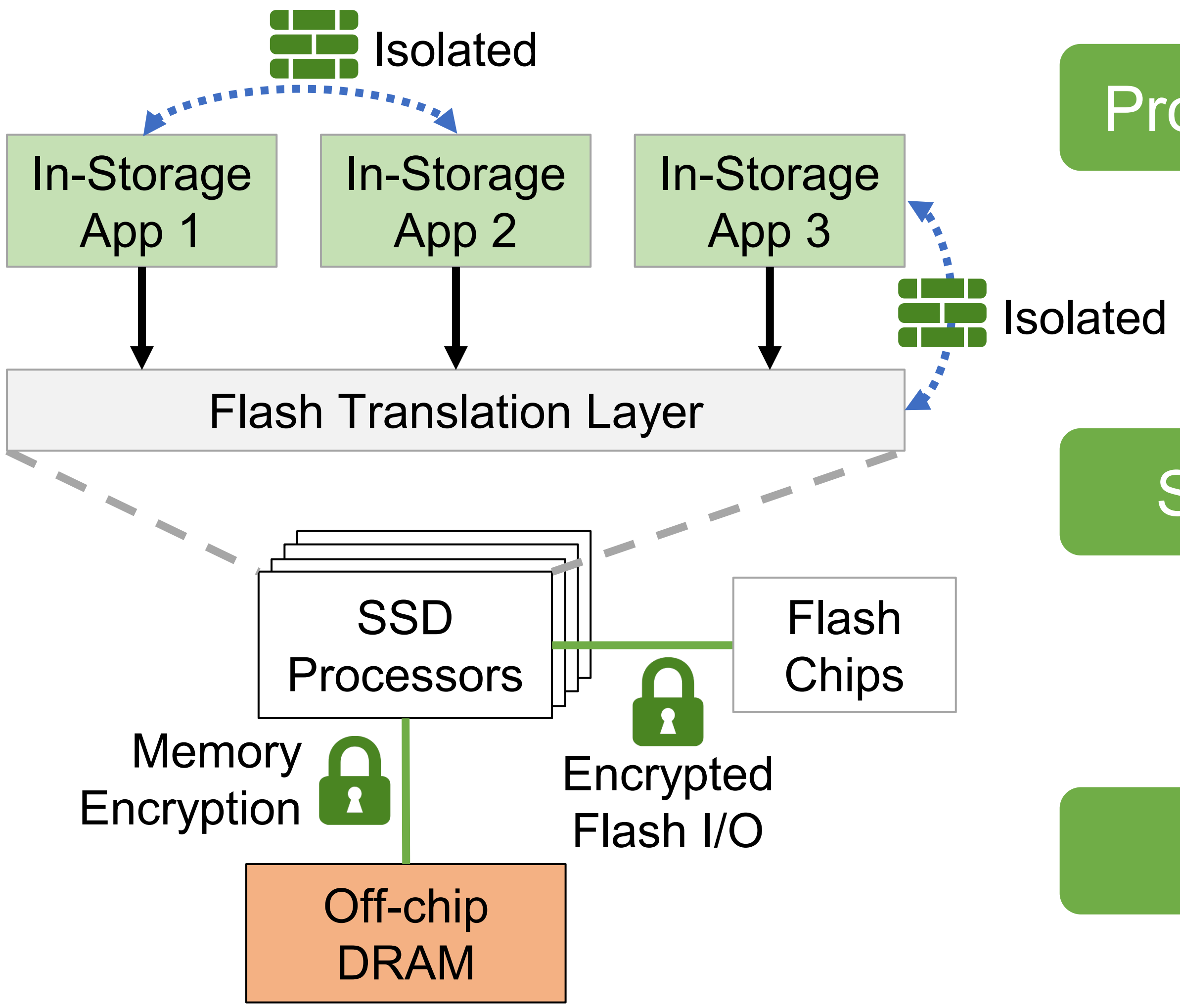


In-storage applications can be malicious

Host runtime and host-SSD offloading channel is trusted

Cloud platform operator may conduct physical attacks

Protecting Flash Translation Layer



Protecting FTL from malicious in-storage apps

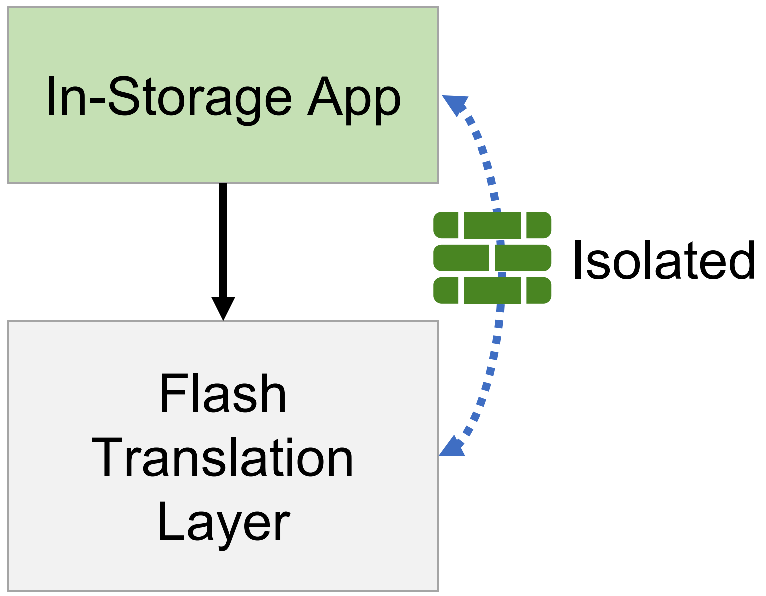
+

Security isolation between in-storage apps

+

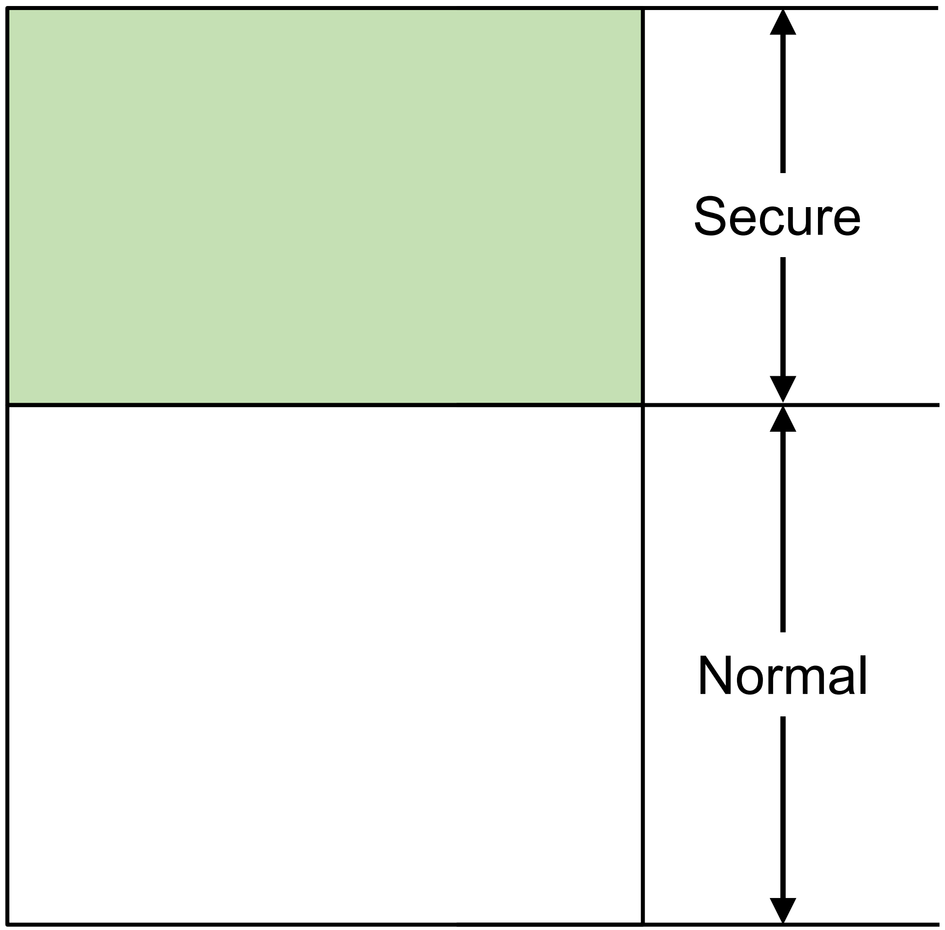
Securing data against physical attacks

Protecting Flash Translation Layer

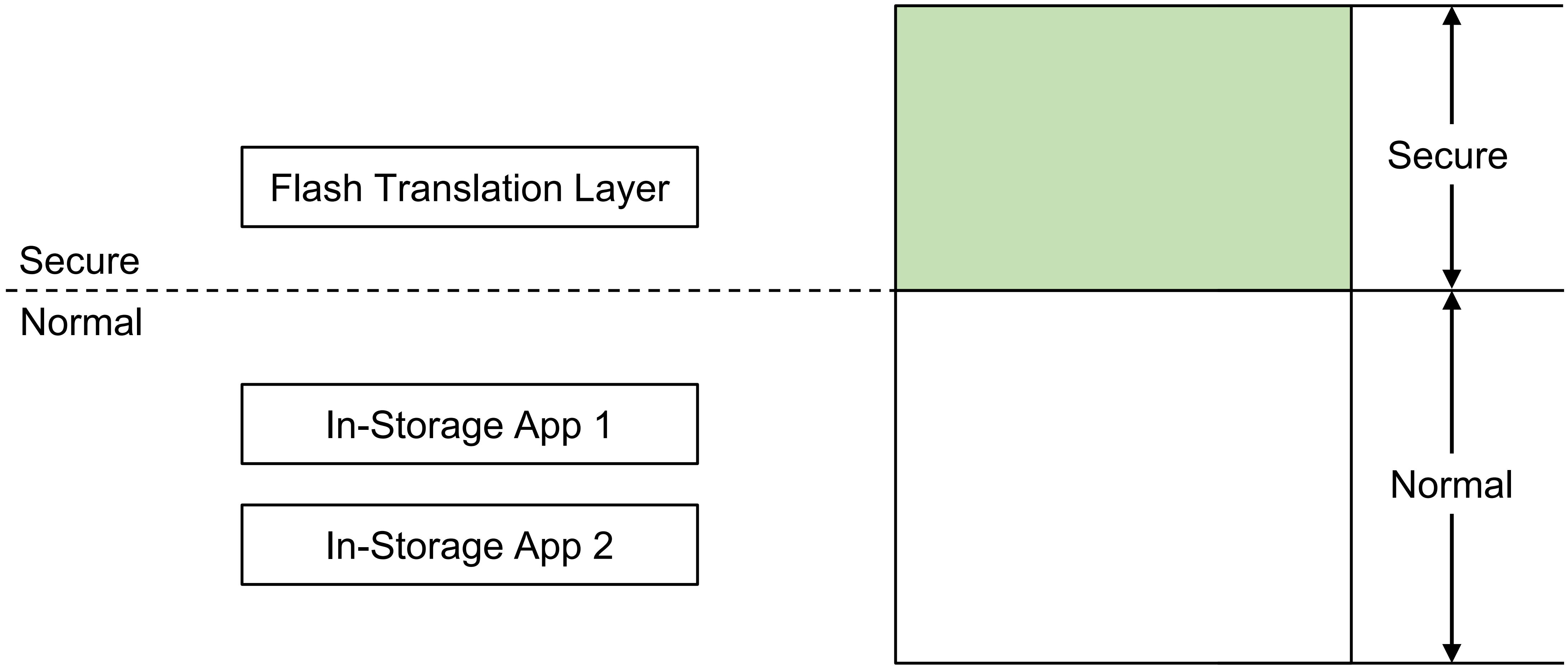


Protecting FTL from malicious in-storage apps

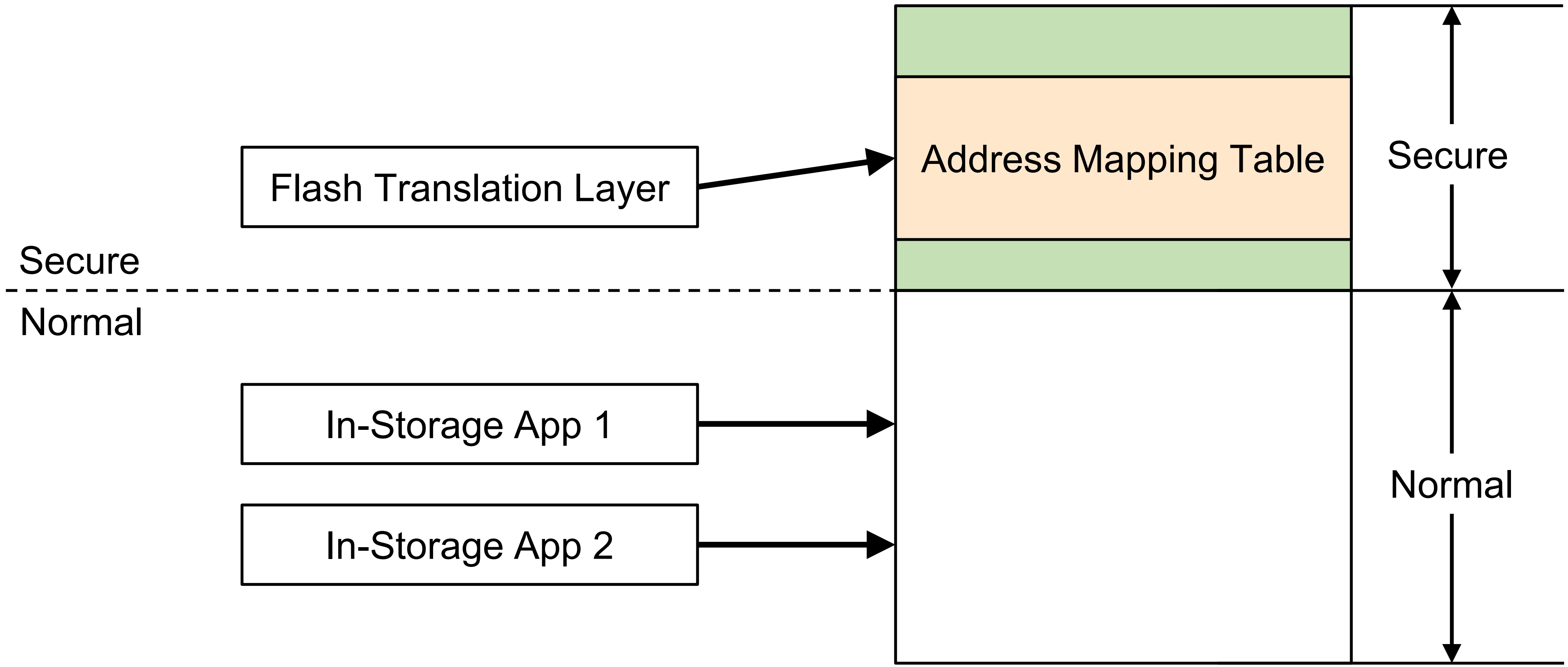
Protecting Flash Translation Layer



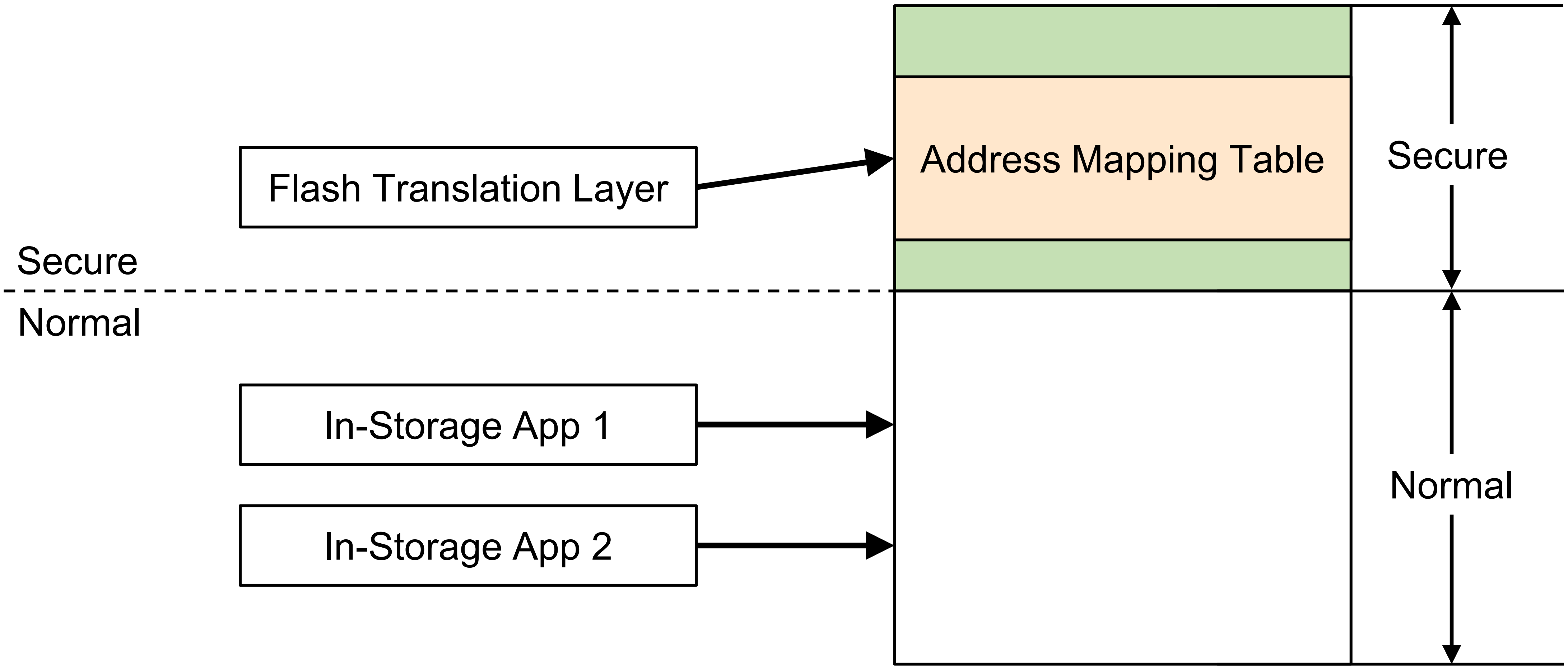
Protecting Flash Translation Layer



Protecting Flash Translation Layer

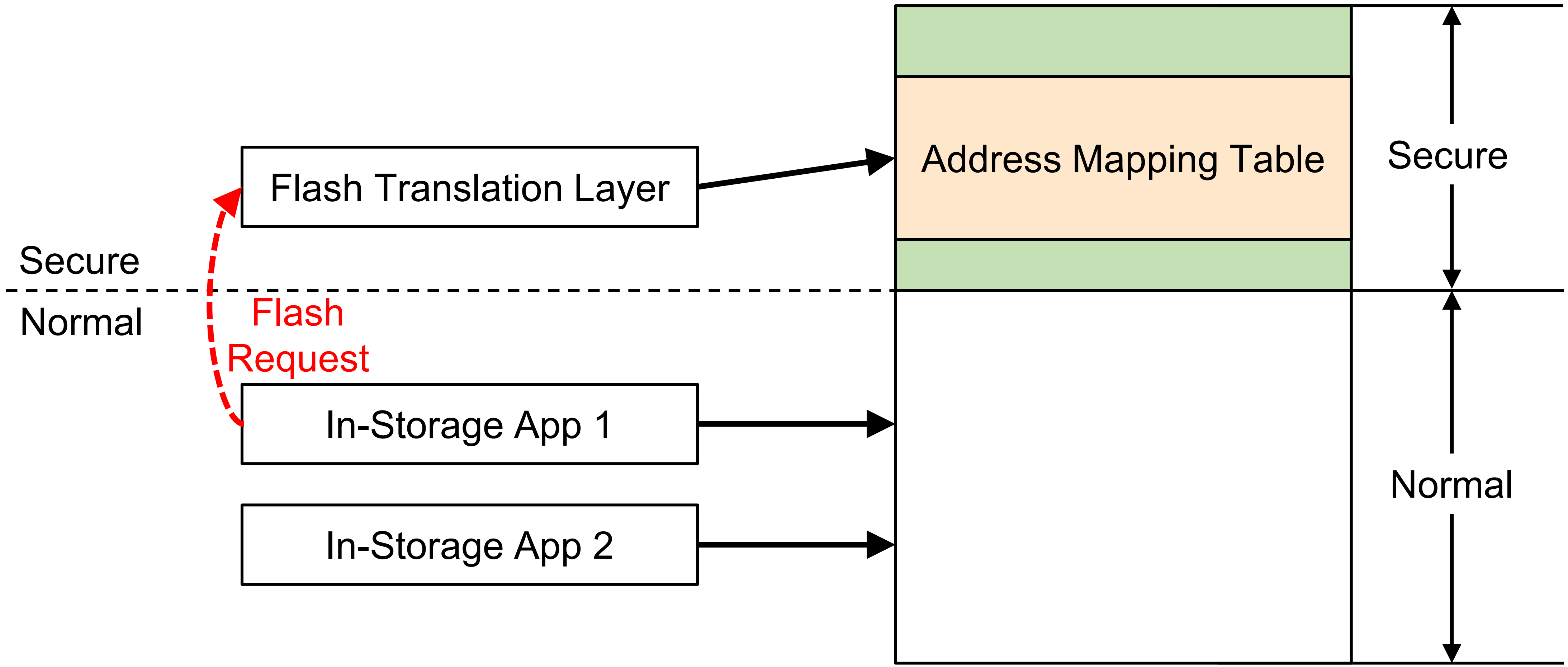


Protecting Flash Translation Layer



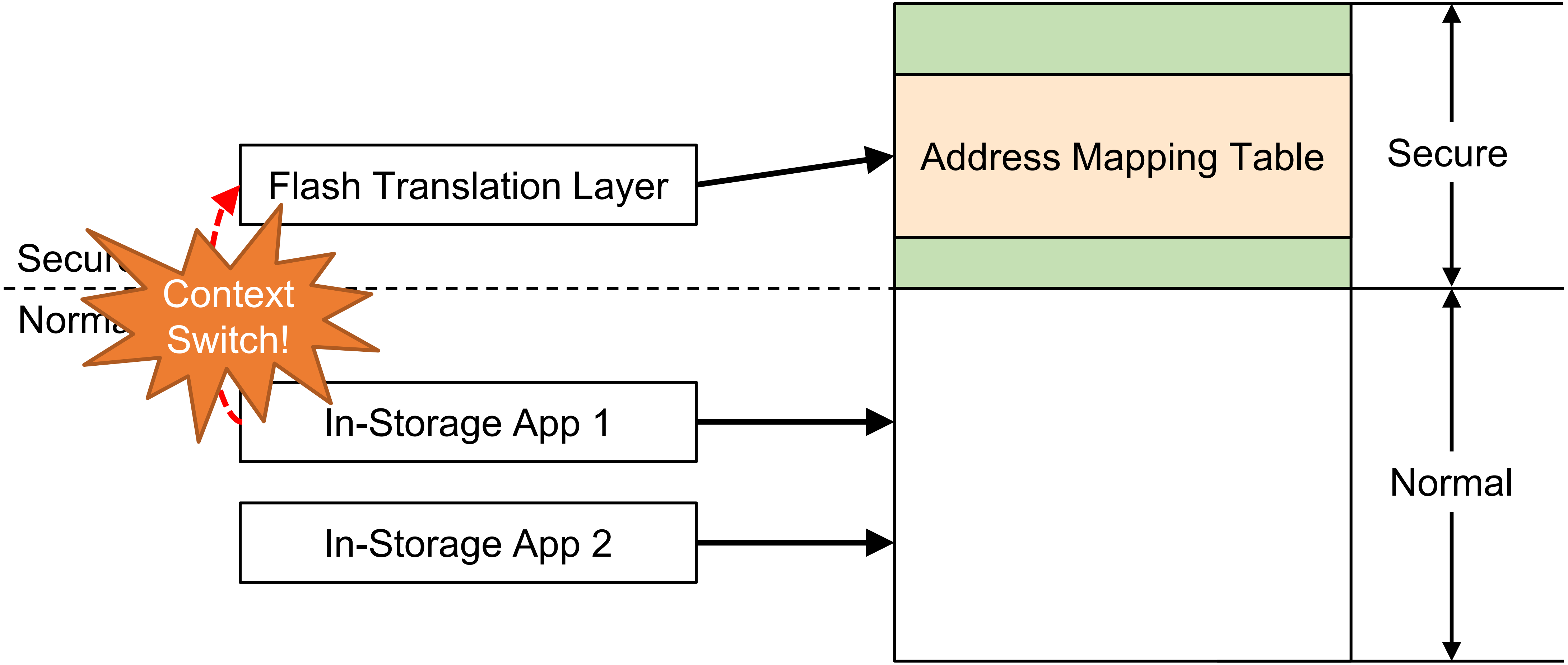
Naively applying TrustZone partitioning incurs significant performance penalty!

Protecting Flash Translation Layer



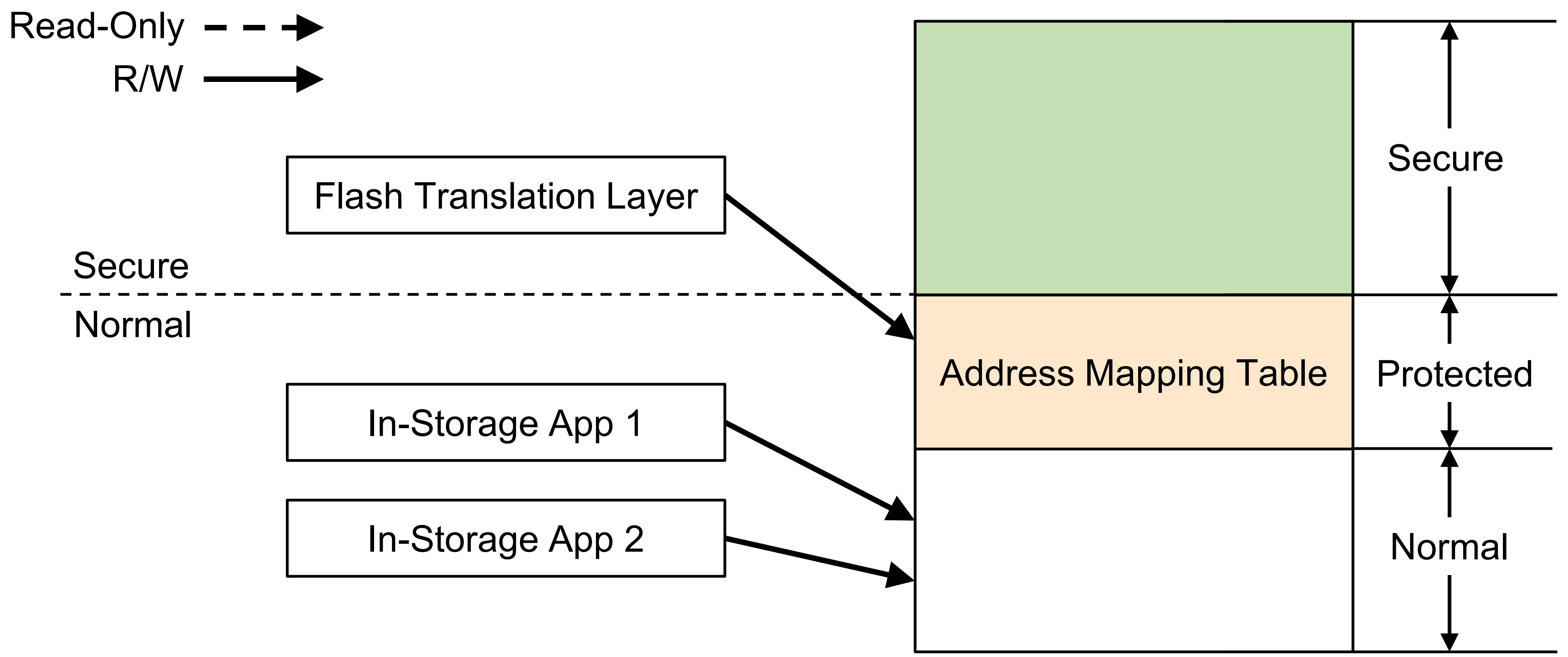
Naively applying TrustZone partitioning incurs significant performance penalty!

Protecting Flash Translation Layer

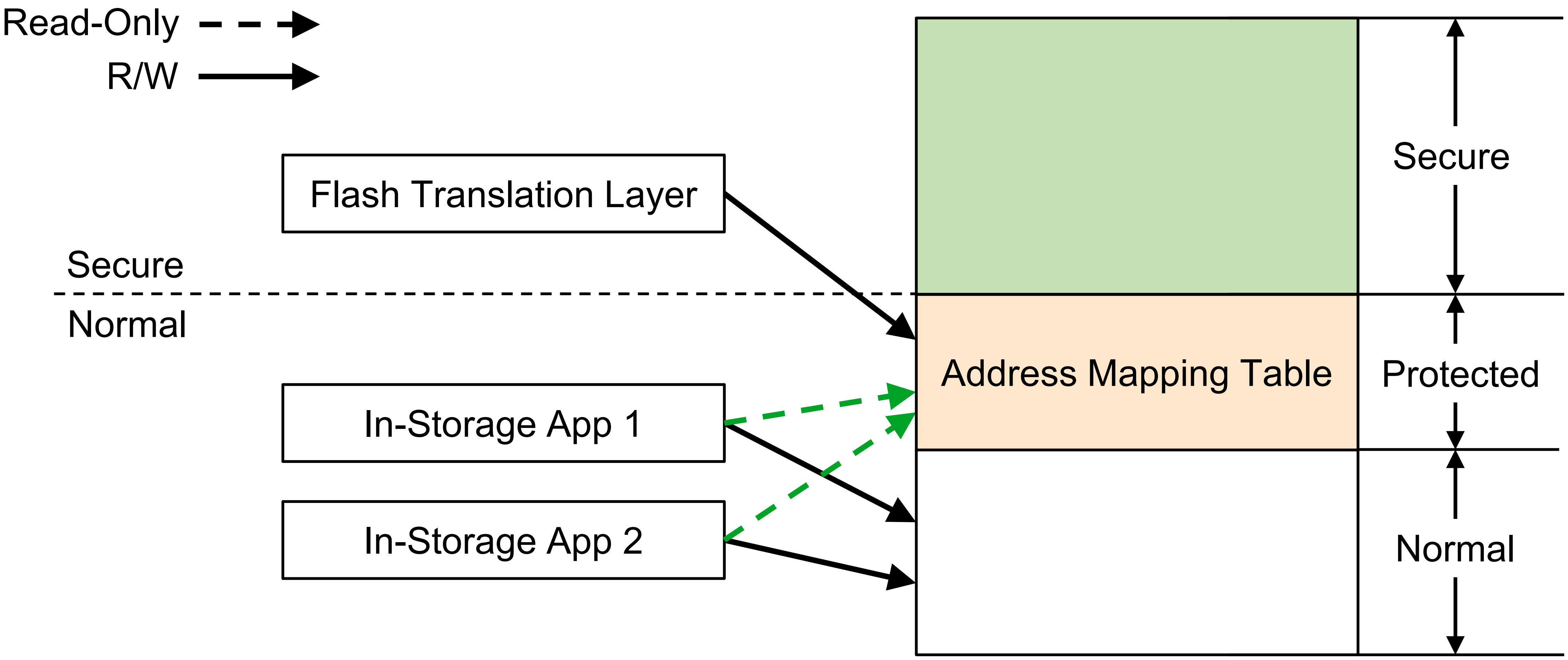


Naively applying TrustZone partitioning incurs significant performance penalty!

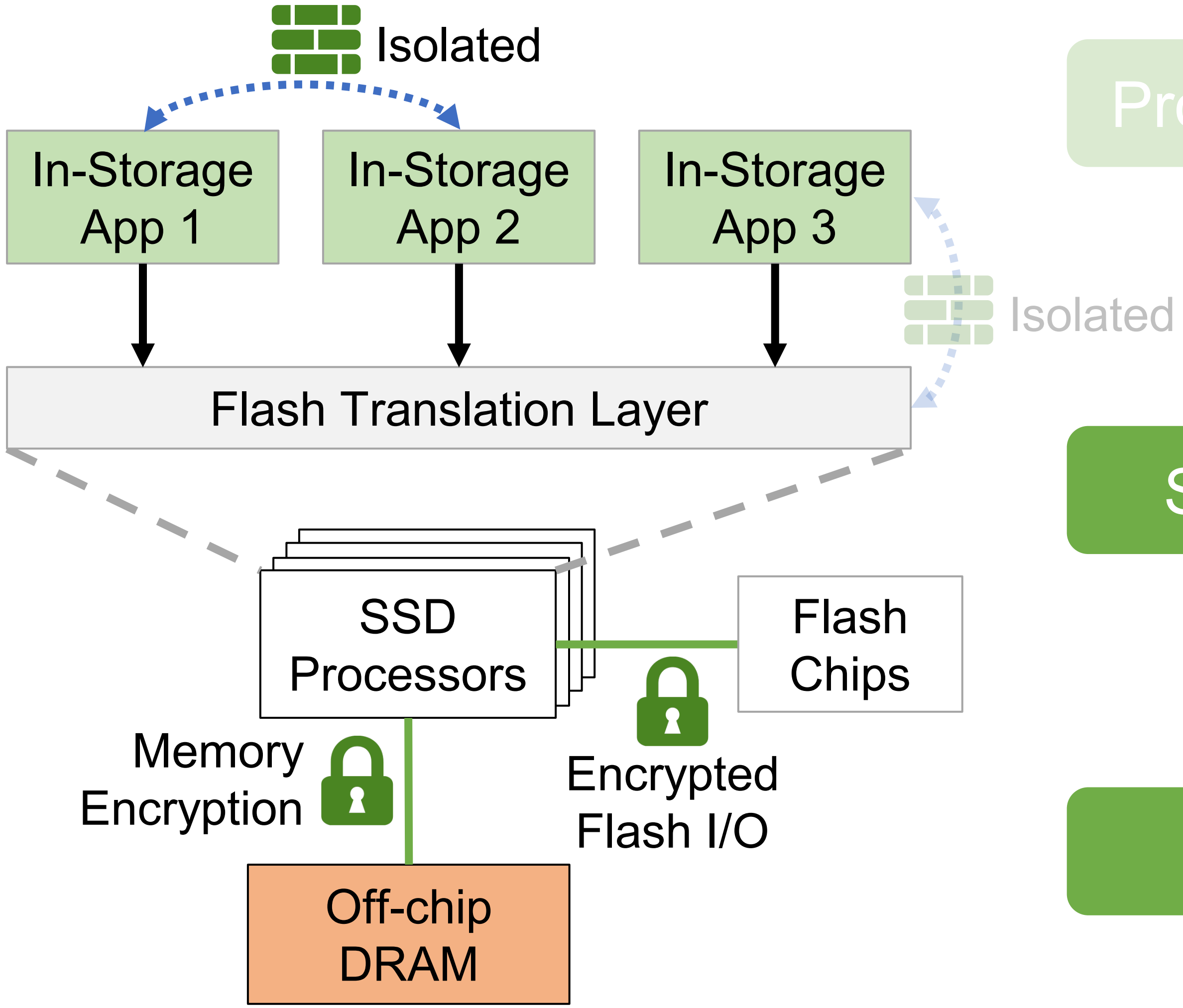
Protecting Flash Translation Layer



Protecting Flash Translation Layer



Isolating In-Storage Applications



Protecting FTL from malicious in-storage apps

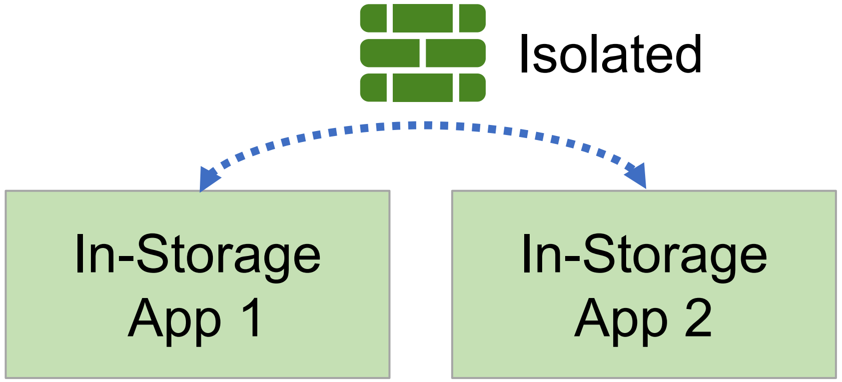
+

Security isolation between in-storage apps

+

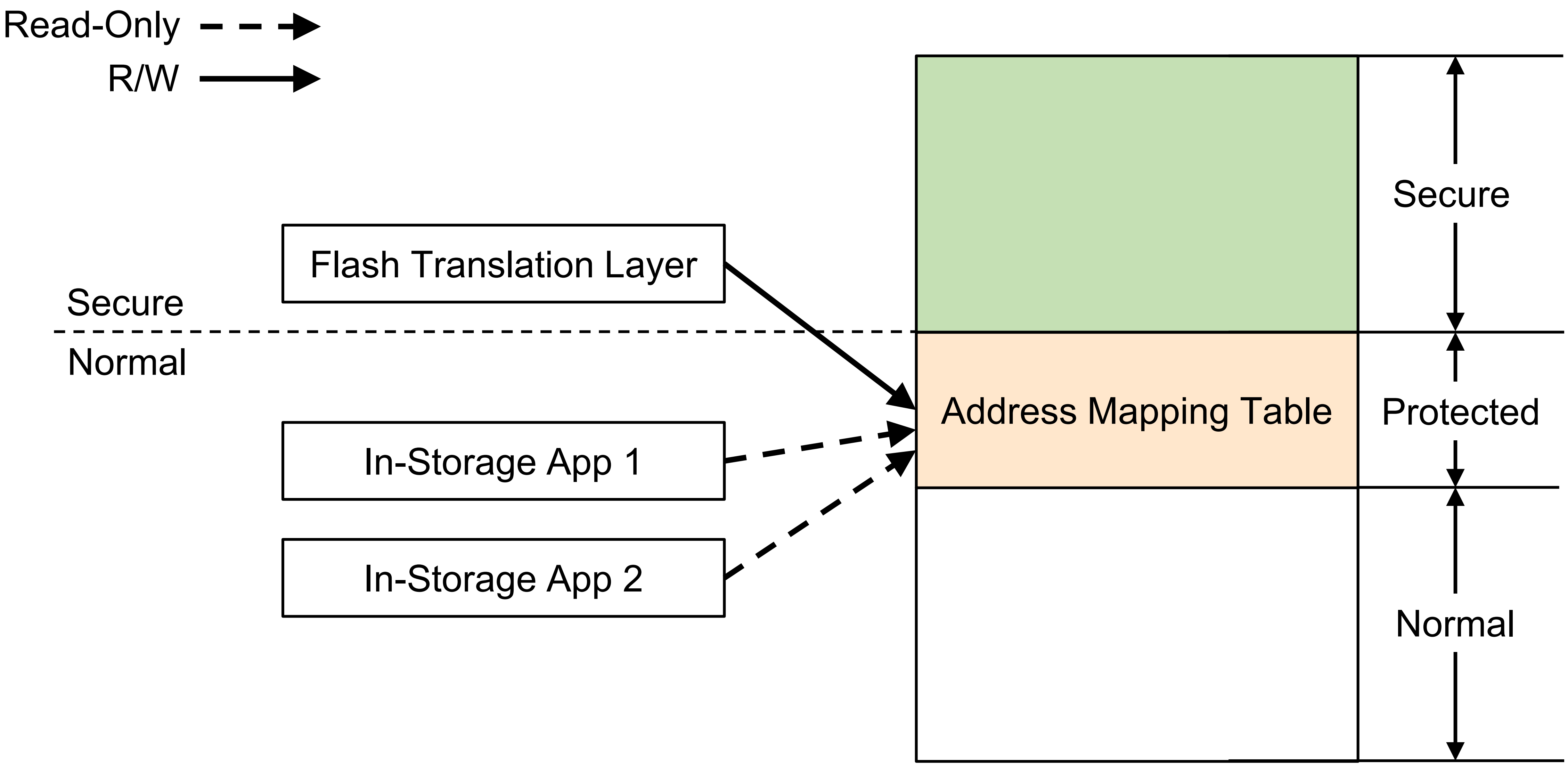
Securing data against physical attacks

Isolating In-Storage Applications

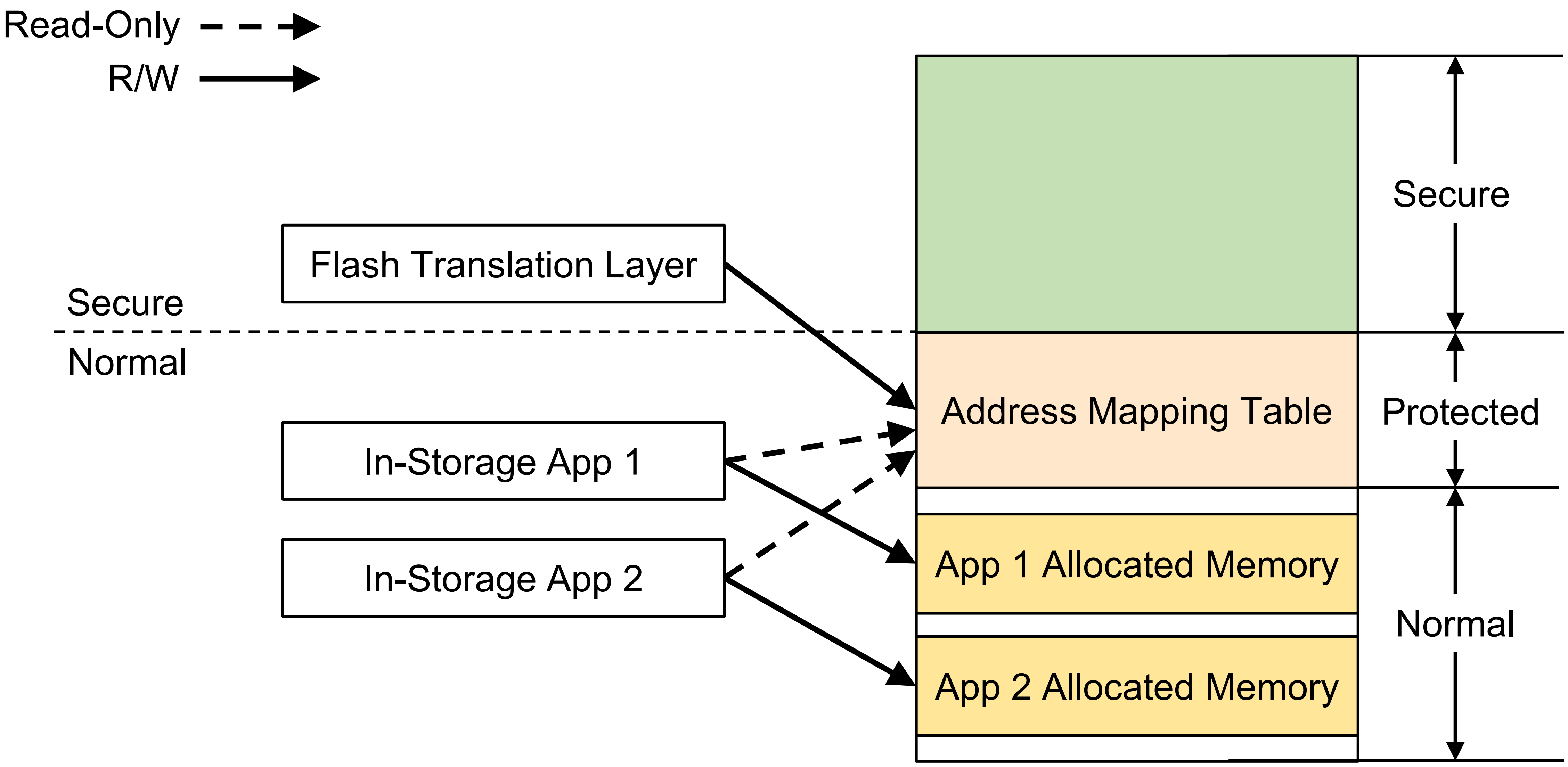


Security isolation between in-storage apps

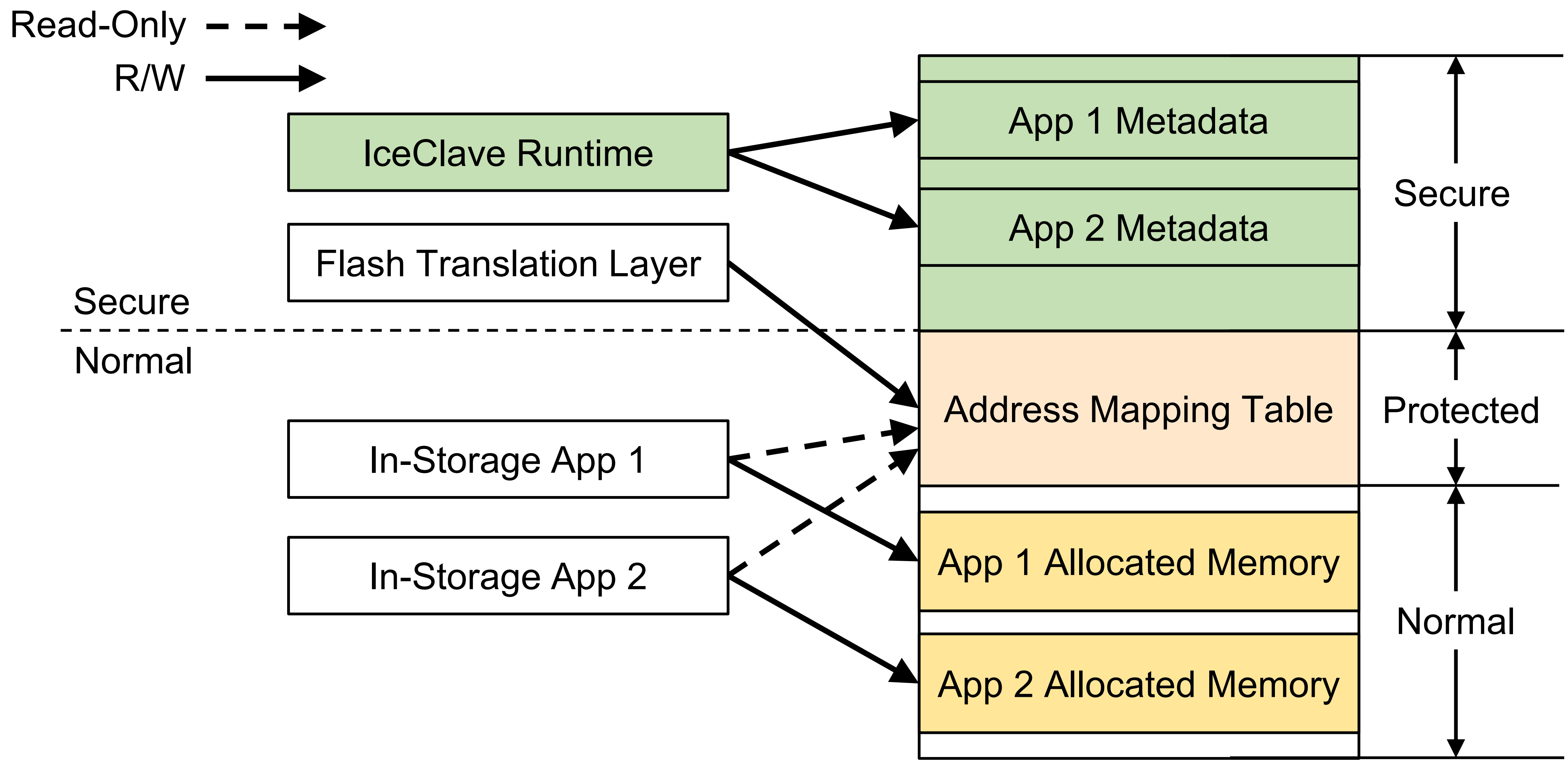
Isolating In-Storage Applications



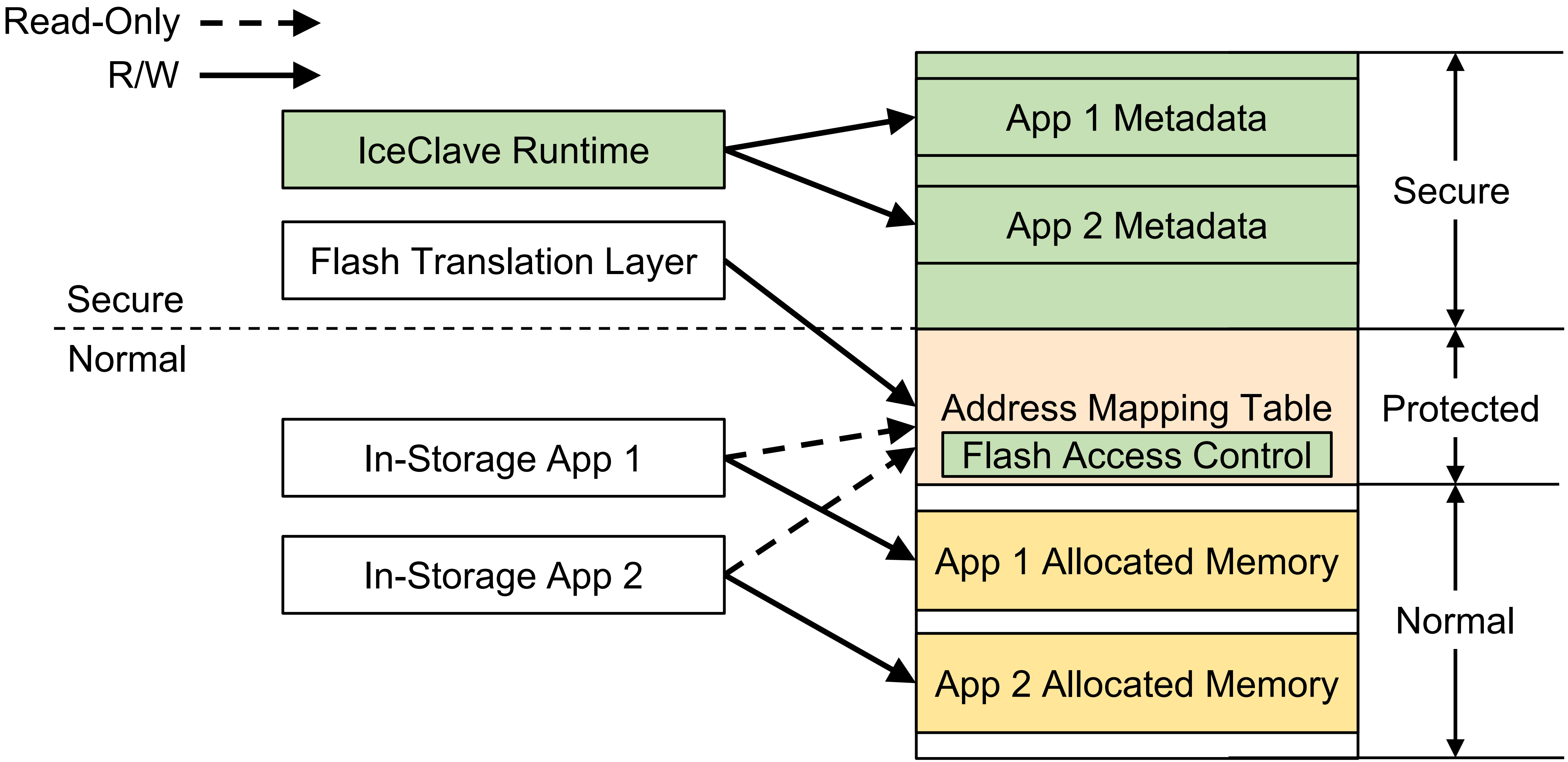
Isolating In-Storage Applications



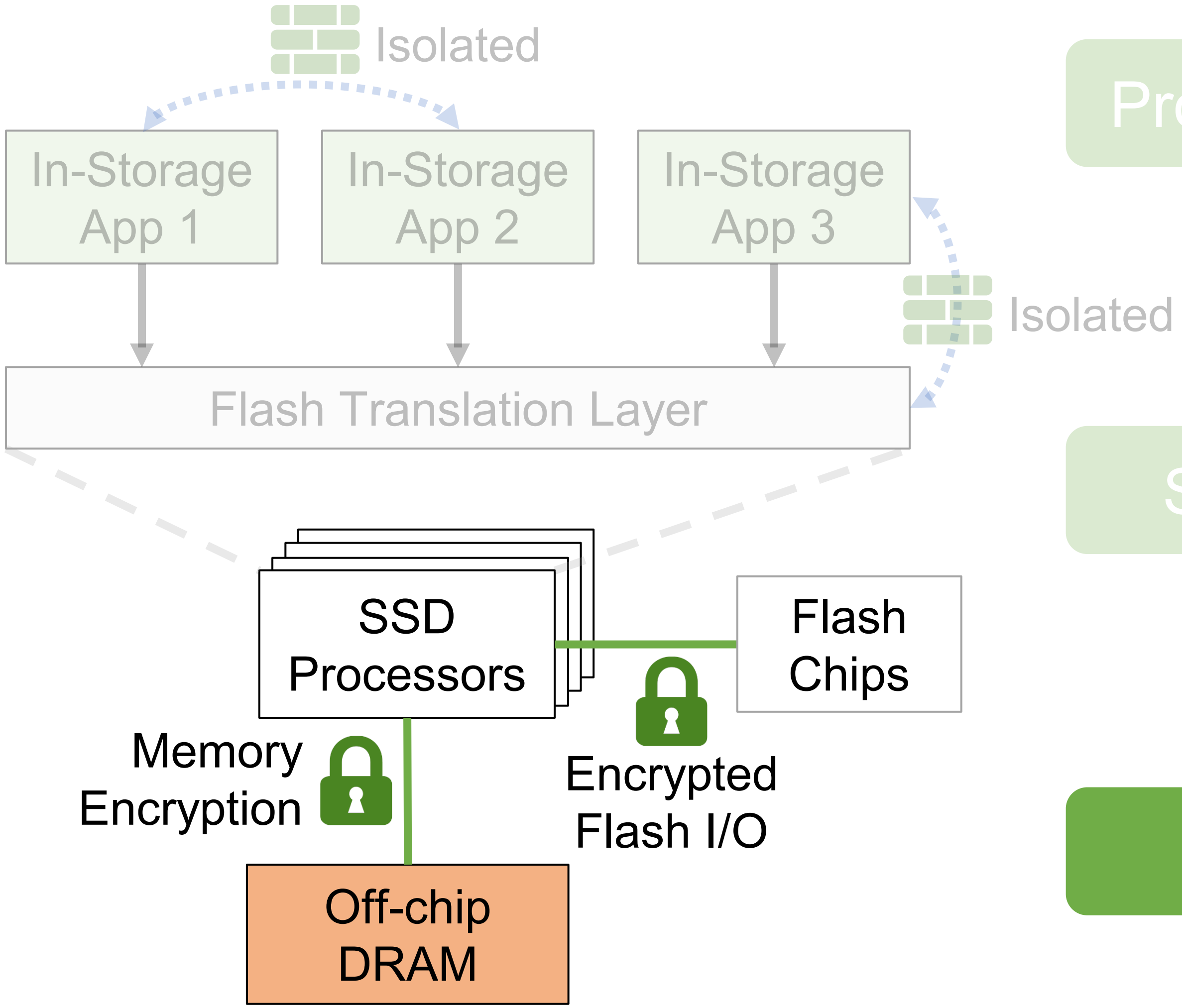
Isolating In-Storage Applications



Isolating In-Storage Applications



Protecting Against Physical Attacks



Protecting FTL from malicious in-storage apps

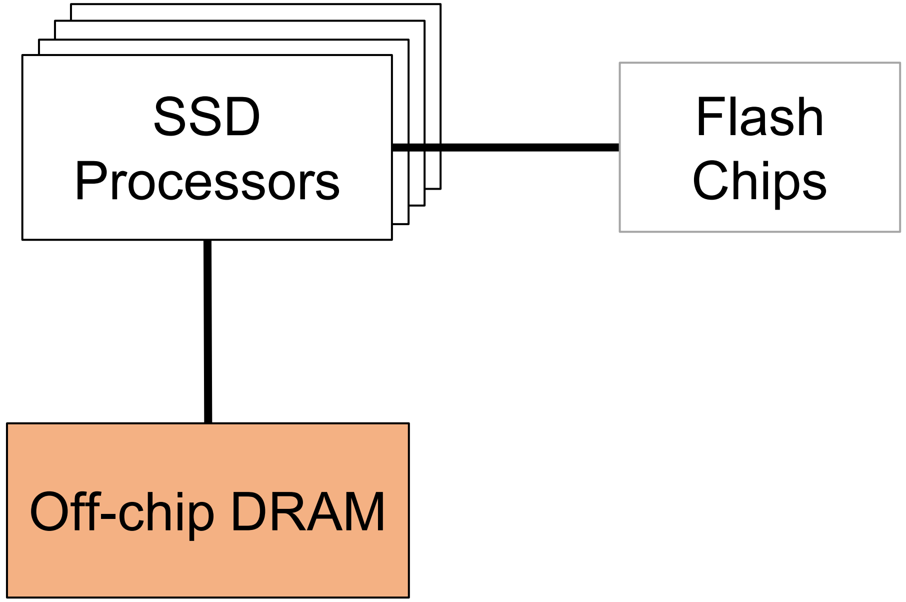
+

Security isolation between in-storage apps

+

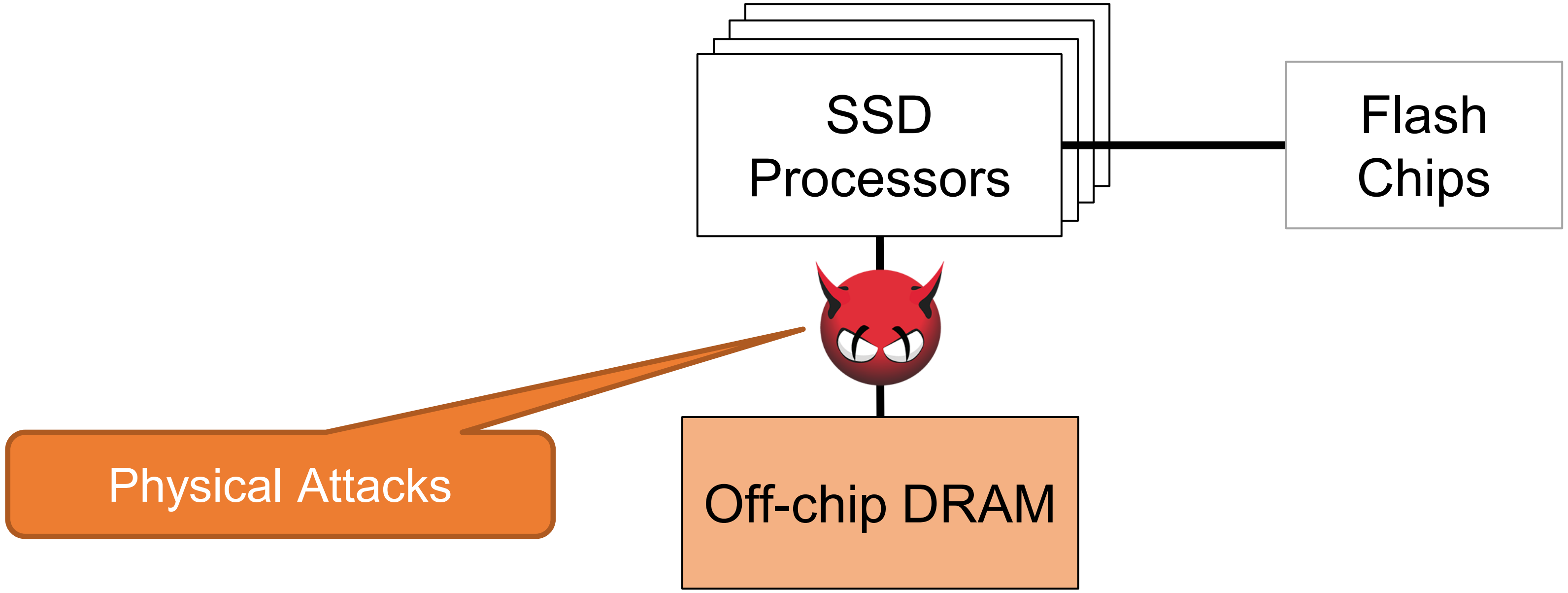
Securing data against physical attacks

Protecting Against Physical Attacks



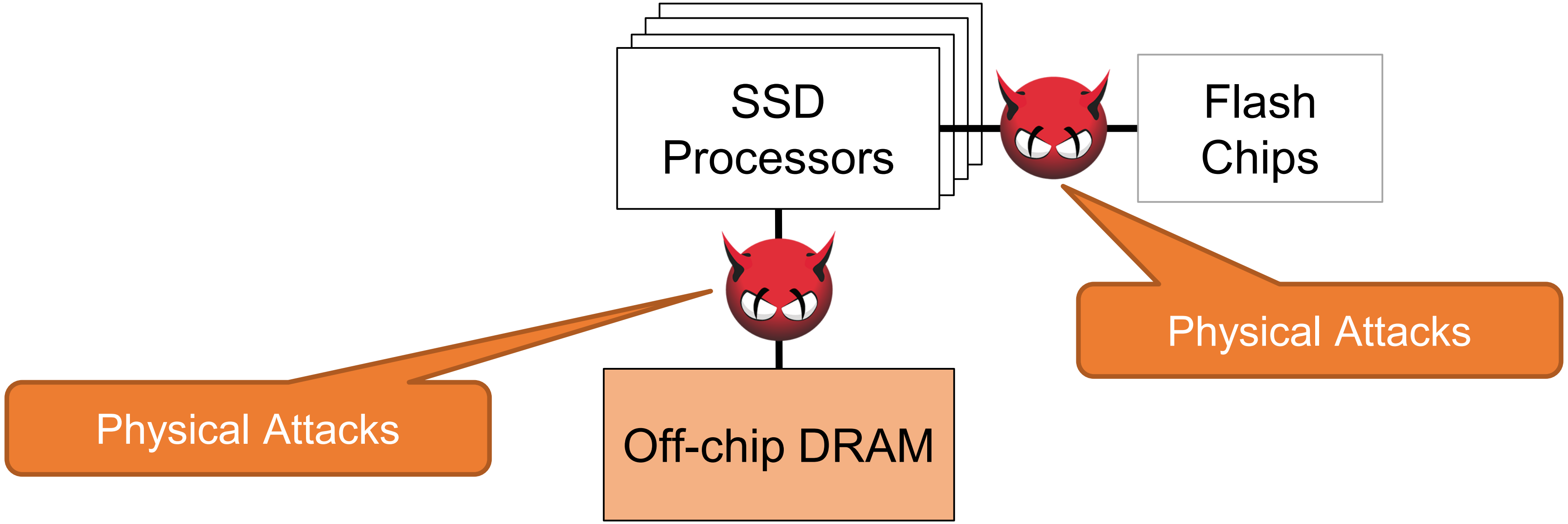
Securing data against physical attacks

Protecting Against Physical Attacks



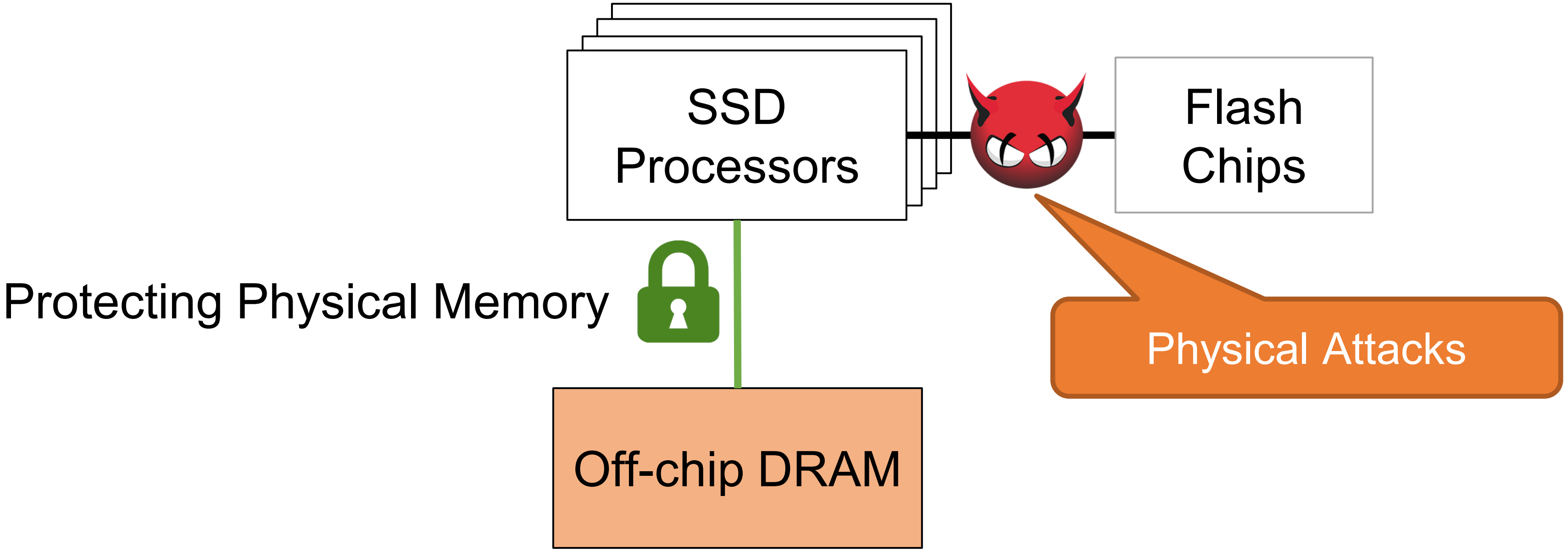
Securing data against physical attacks

Protecting Against Physical Attacks



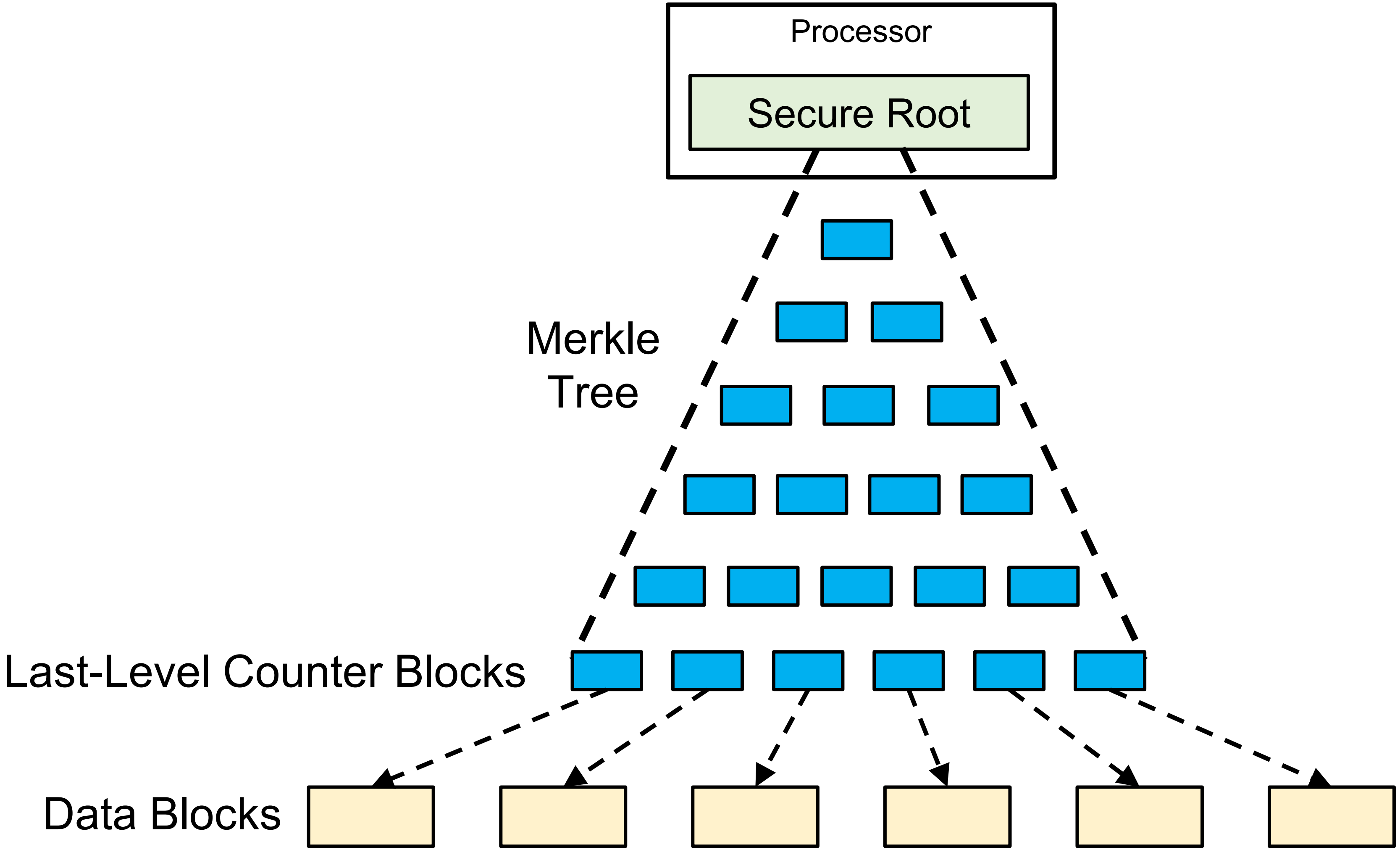
Securing data against physical attacks

Protecting Against Physical Attacks

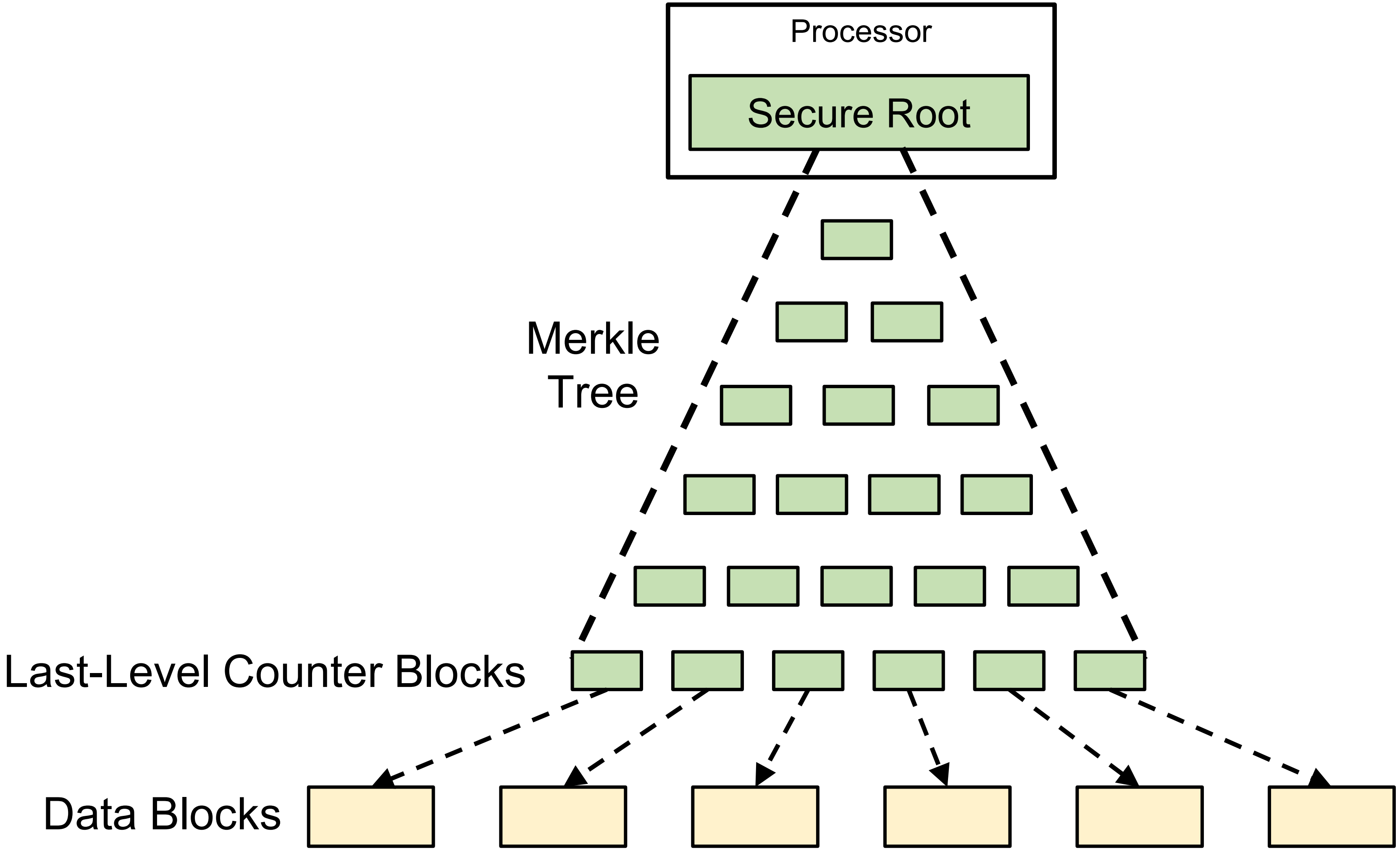


Securing data against physical attacks

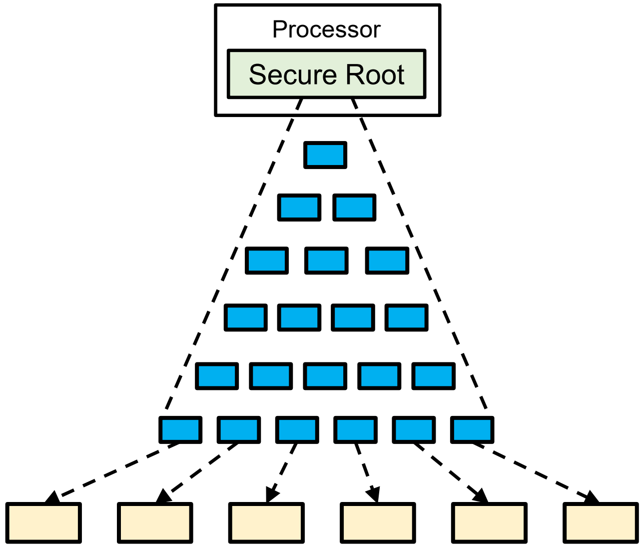
Protecting Physical Memory



Protecting Physical Memory

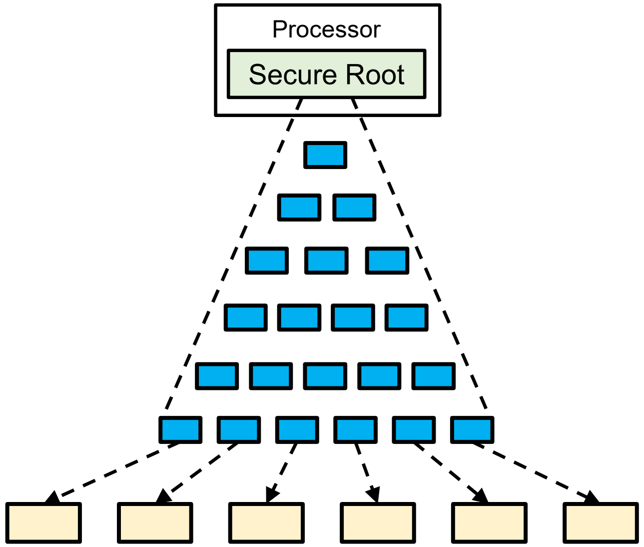


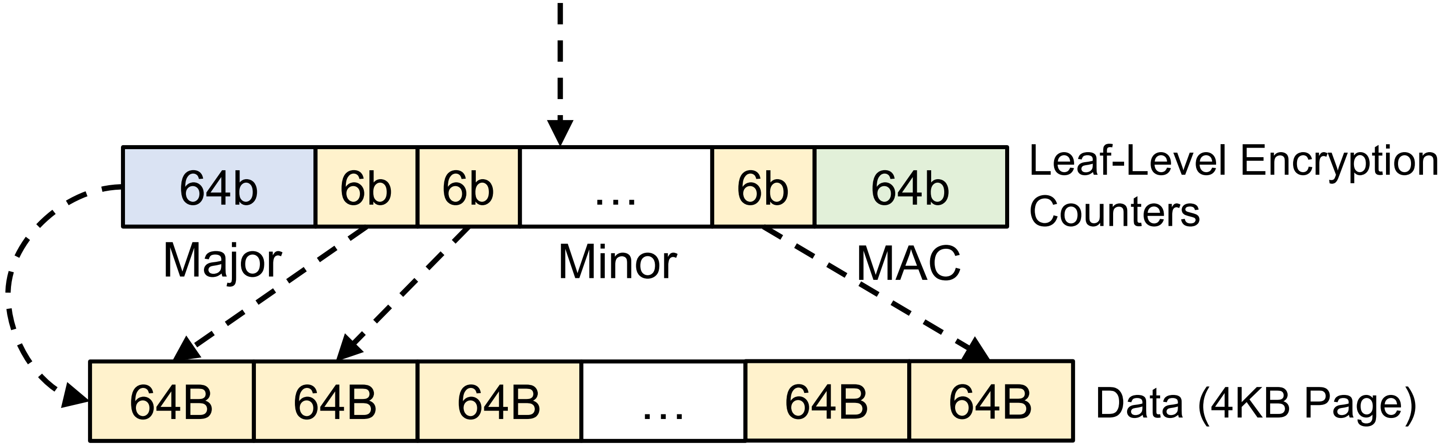
Protecting Physical Memory



Split Counter Mode (ISCA'06)

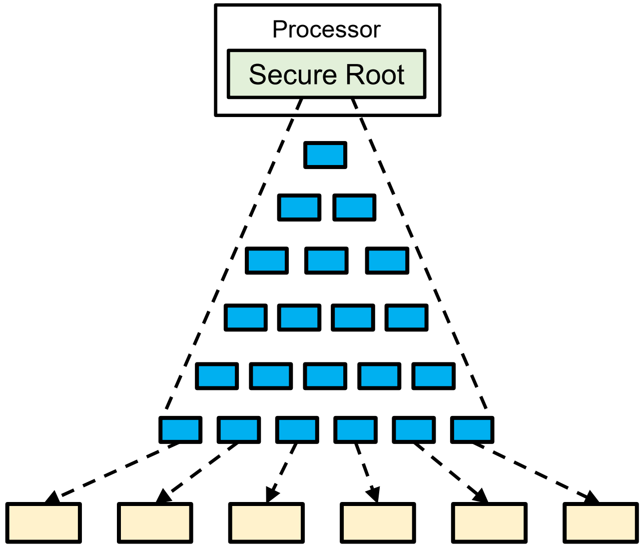
Protecting Physical Memory

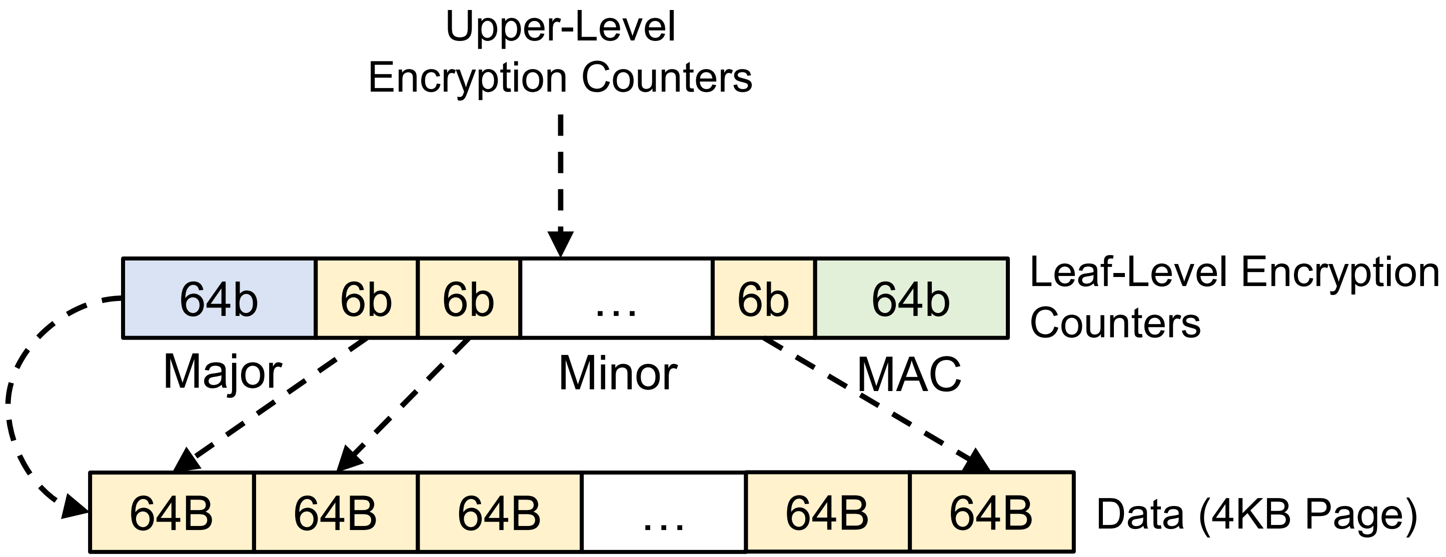




Split Counter Mode (ISCA’06)

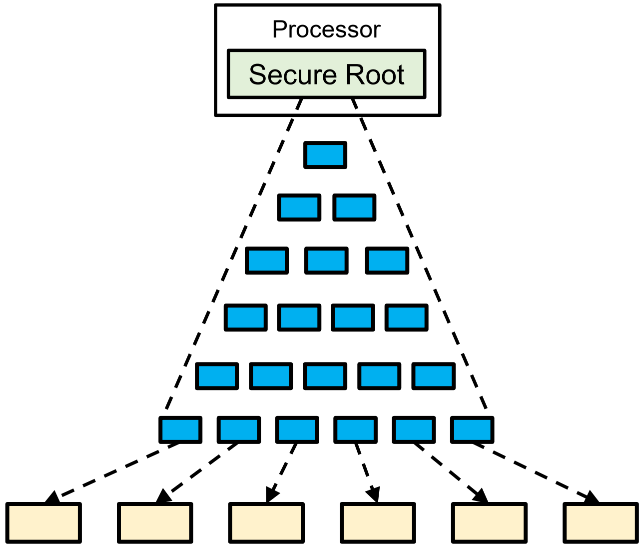
Protecting Physical Memory

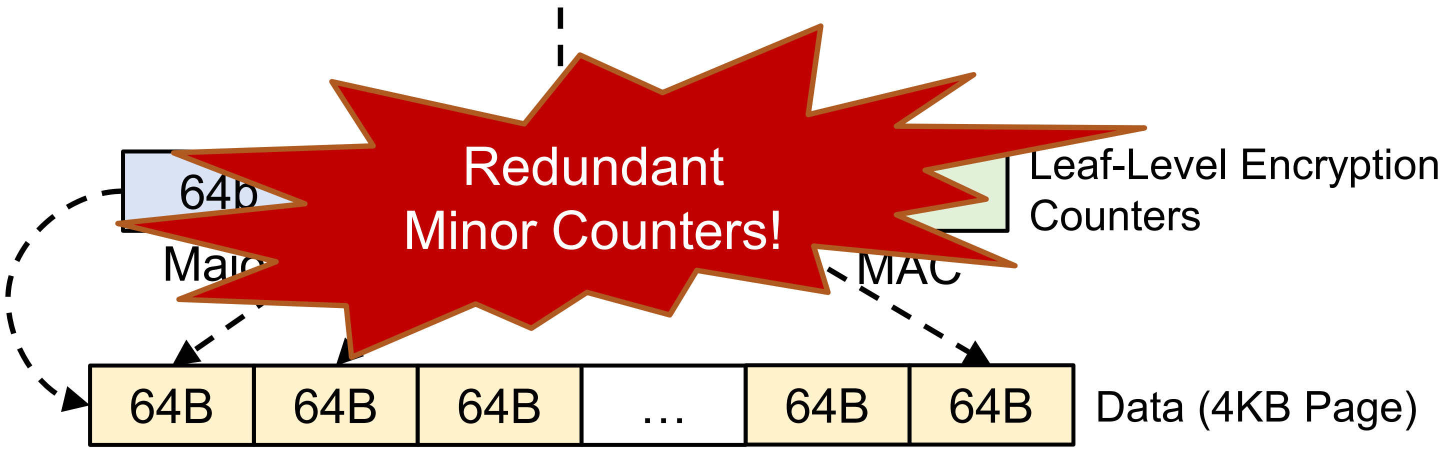




In-storage programs are read-intensive

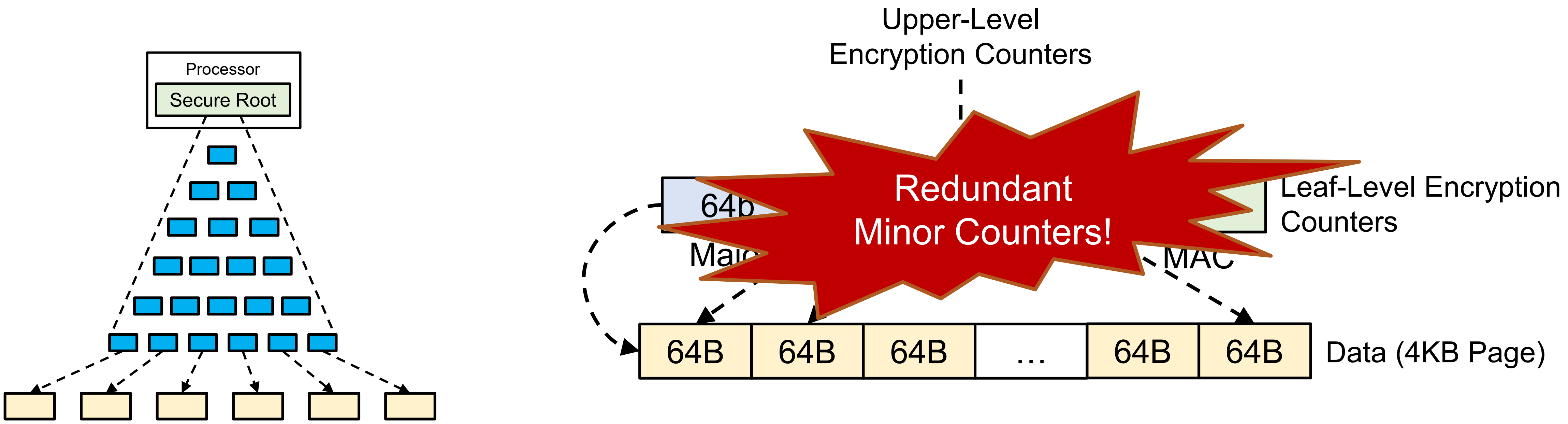
Protecting Physical Memory





In-storage programs are read-intensive

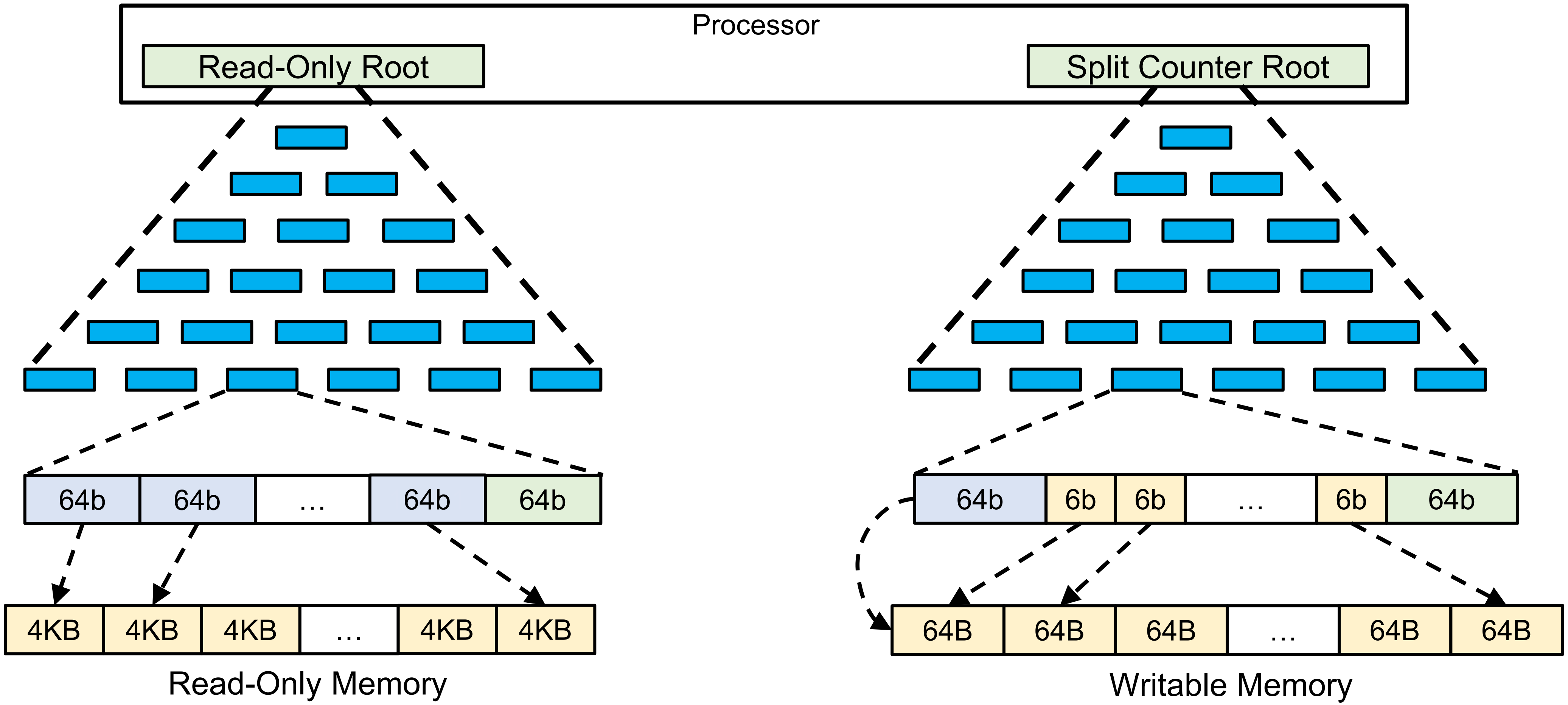
Protecting Physical Memory



In-storage programs are read-intensive

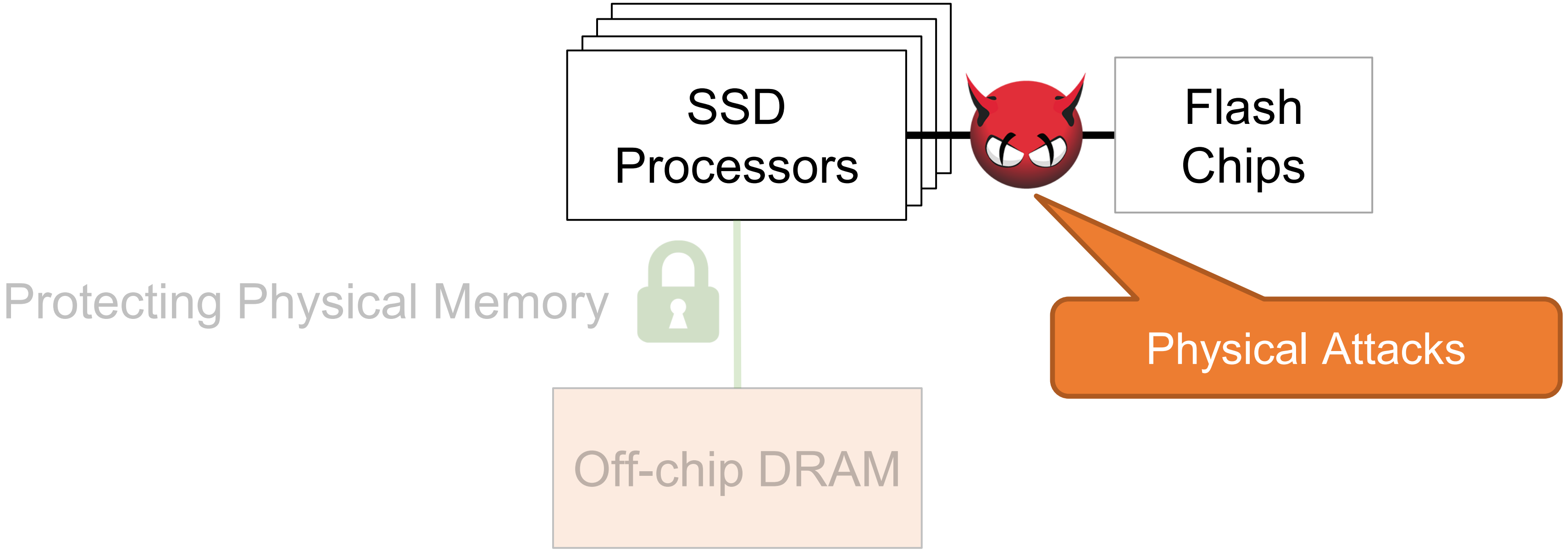
State-of-the-art Split Counter Mode is not optimal for in-storage computing

Protecting Physical Memory



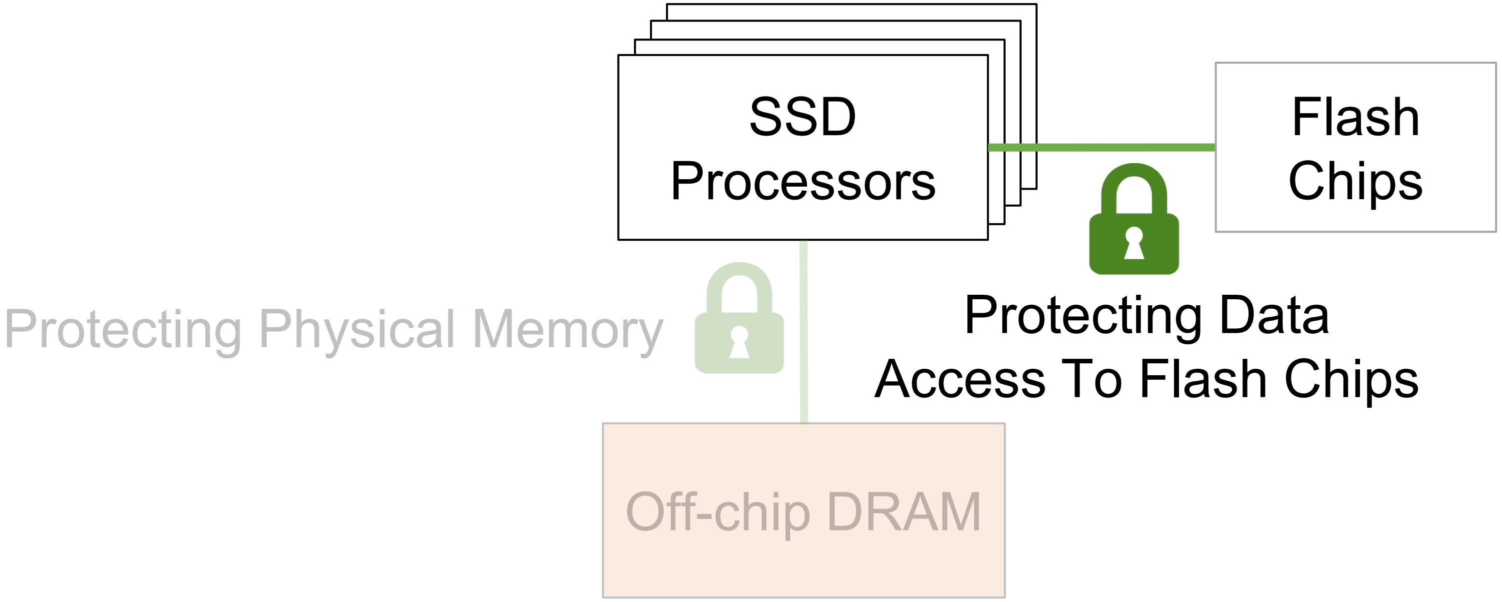
IceClave Hybrid Counter

Protecting Against Physical Attacks



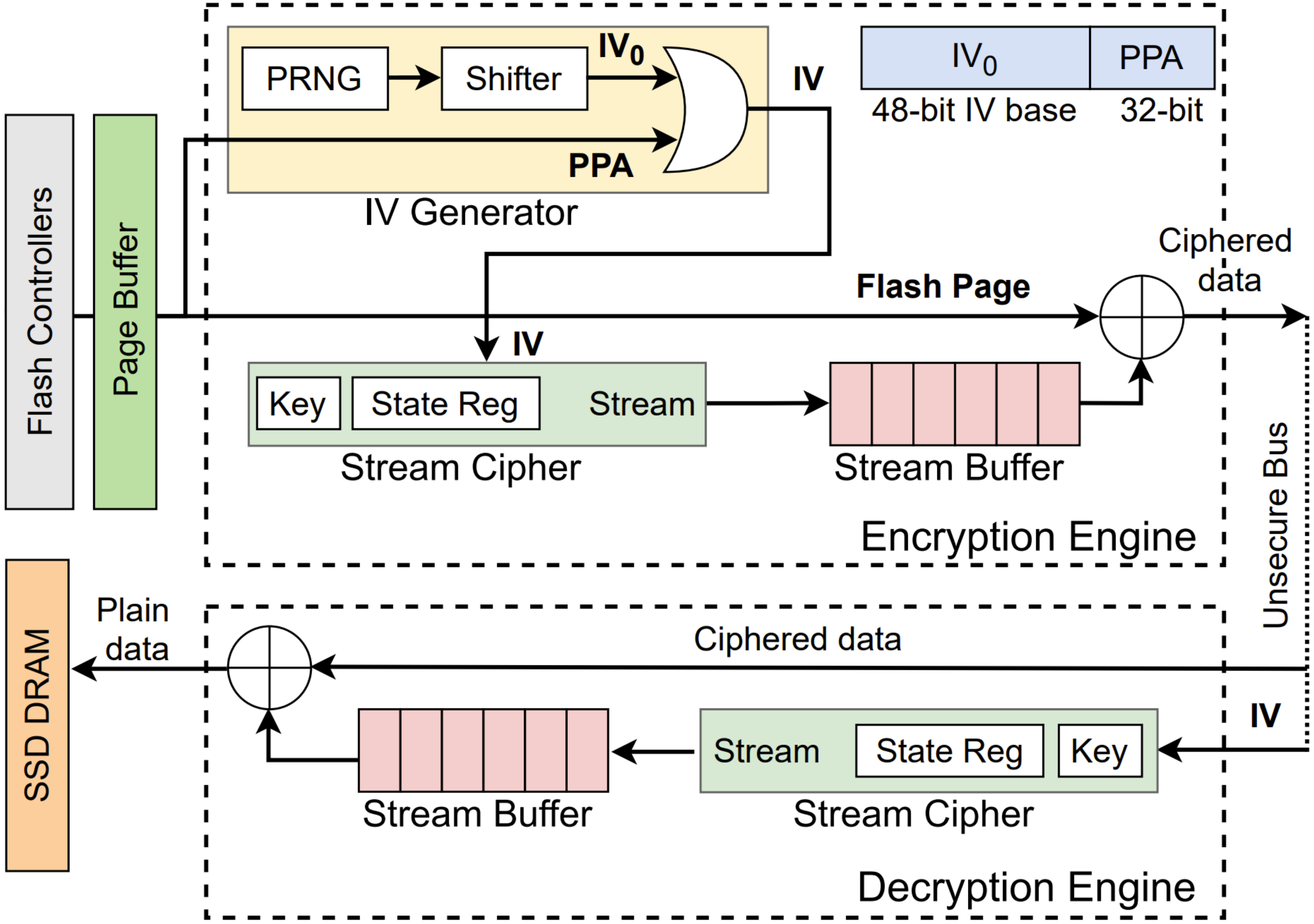
Securing data against physical attacks

Protecting Against Physical Attacks

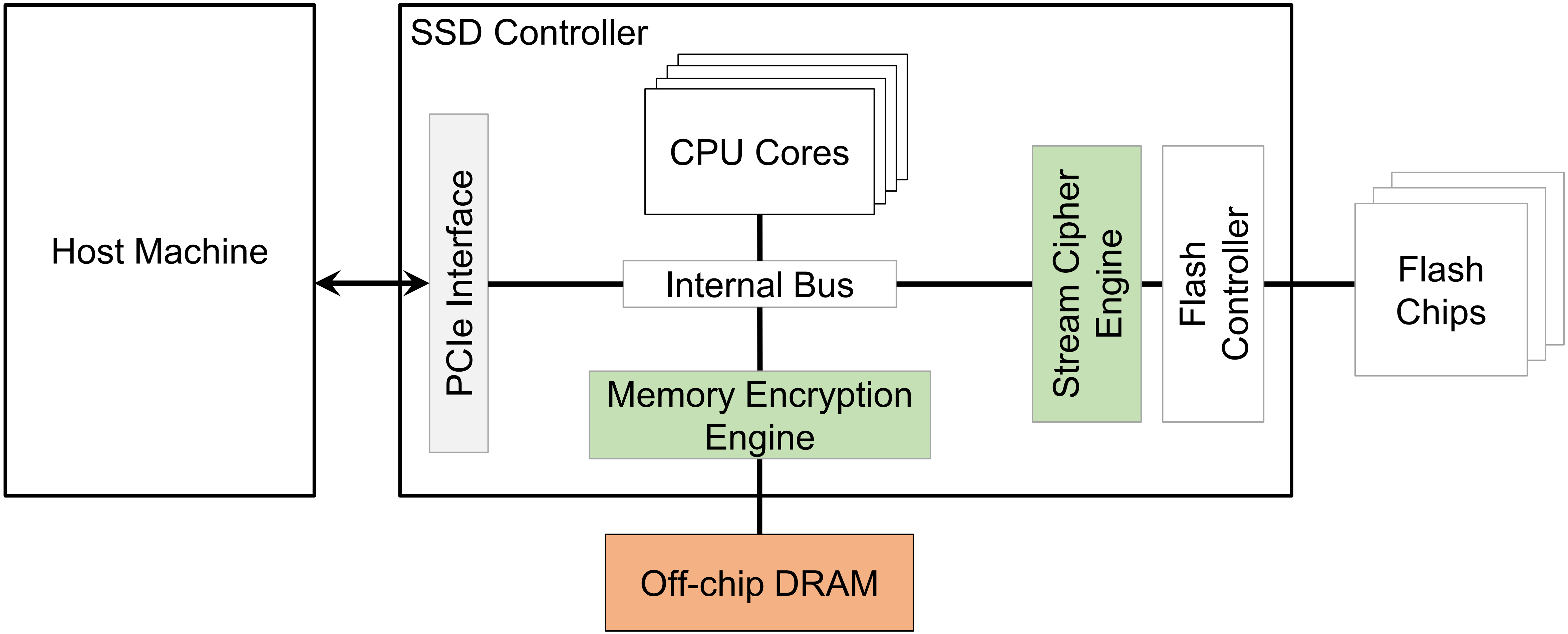


Securing data against physical attacks

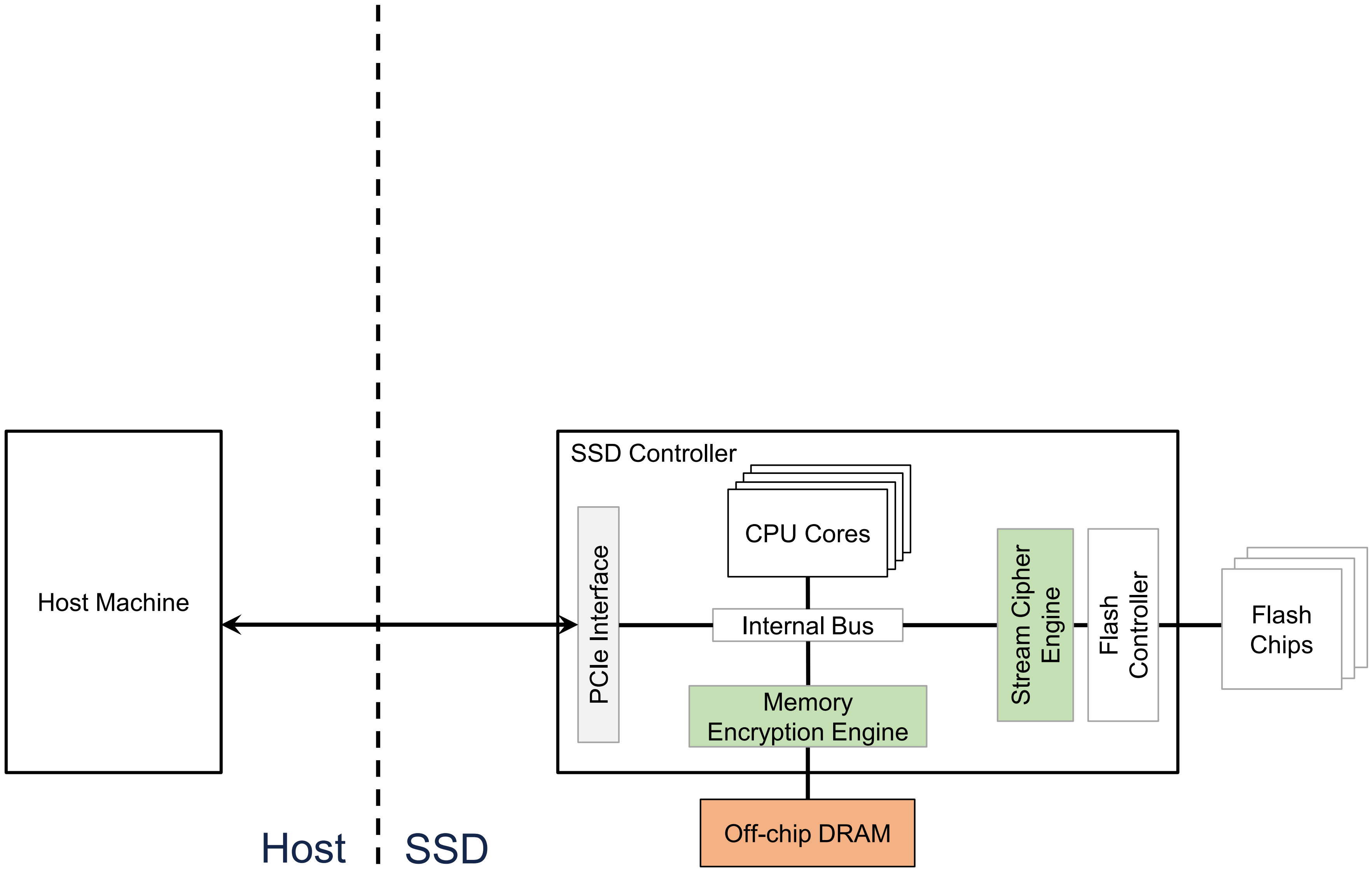
Protecting Data Access To Flash Chips



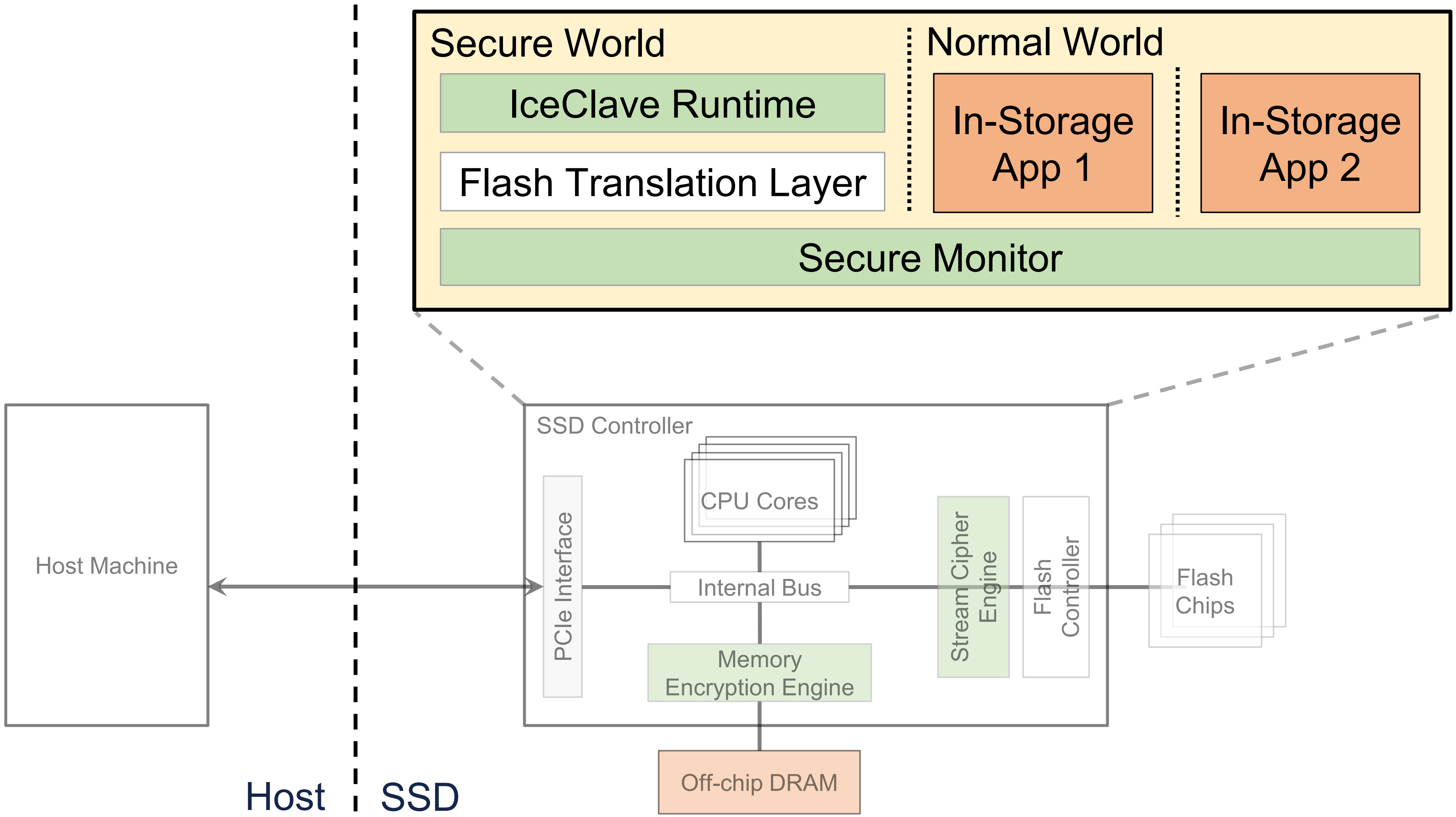
Put It All Together



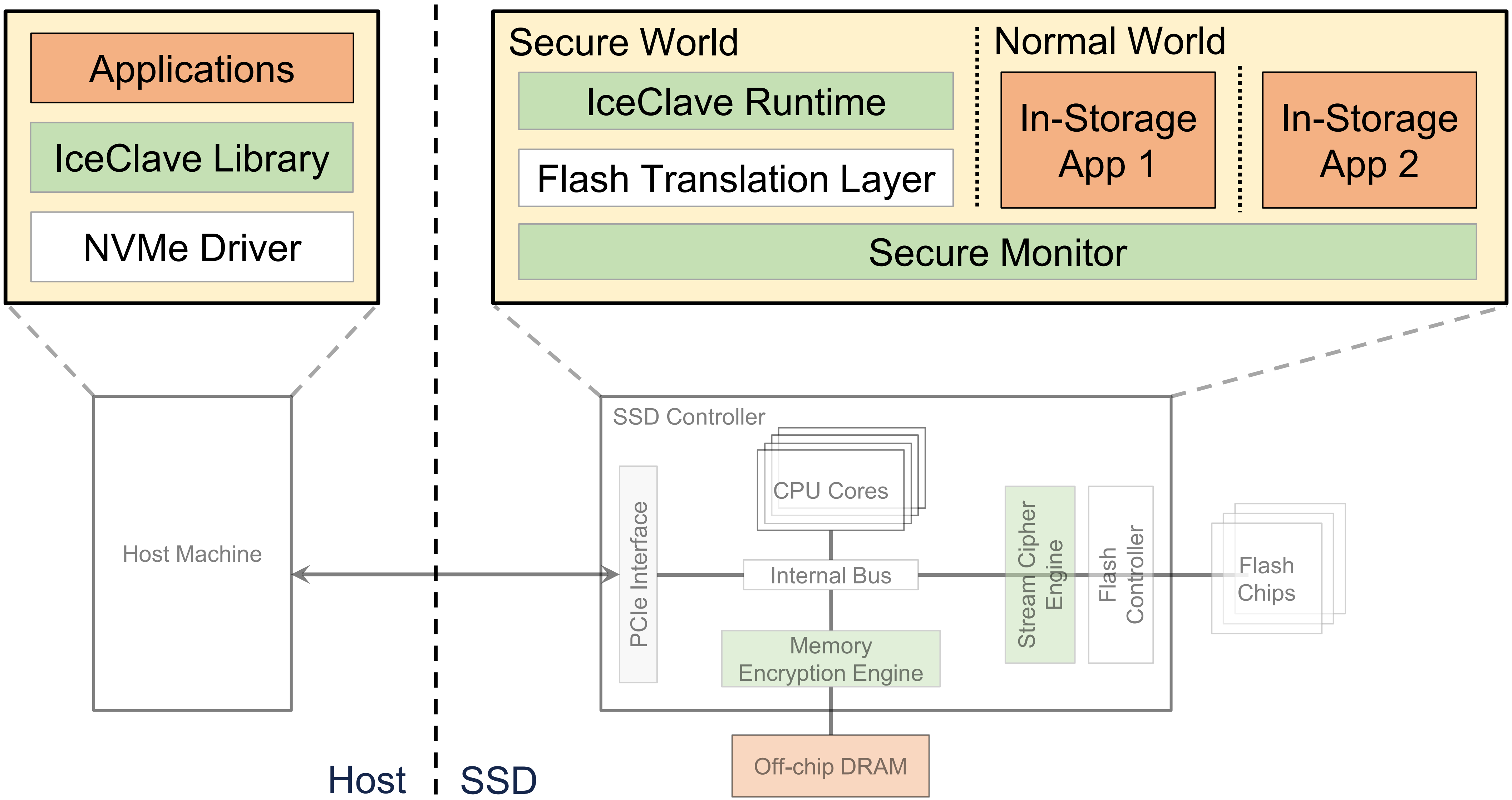
Put It All Together



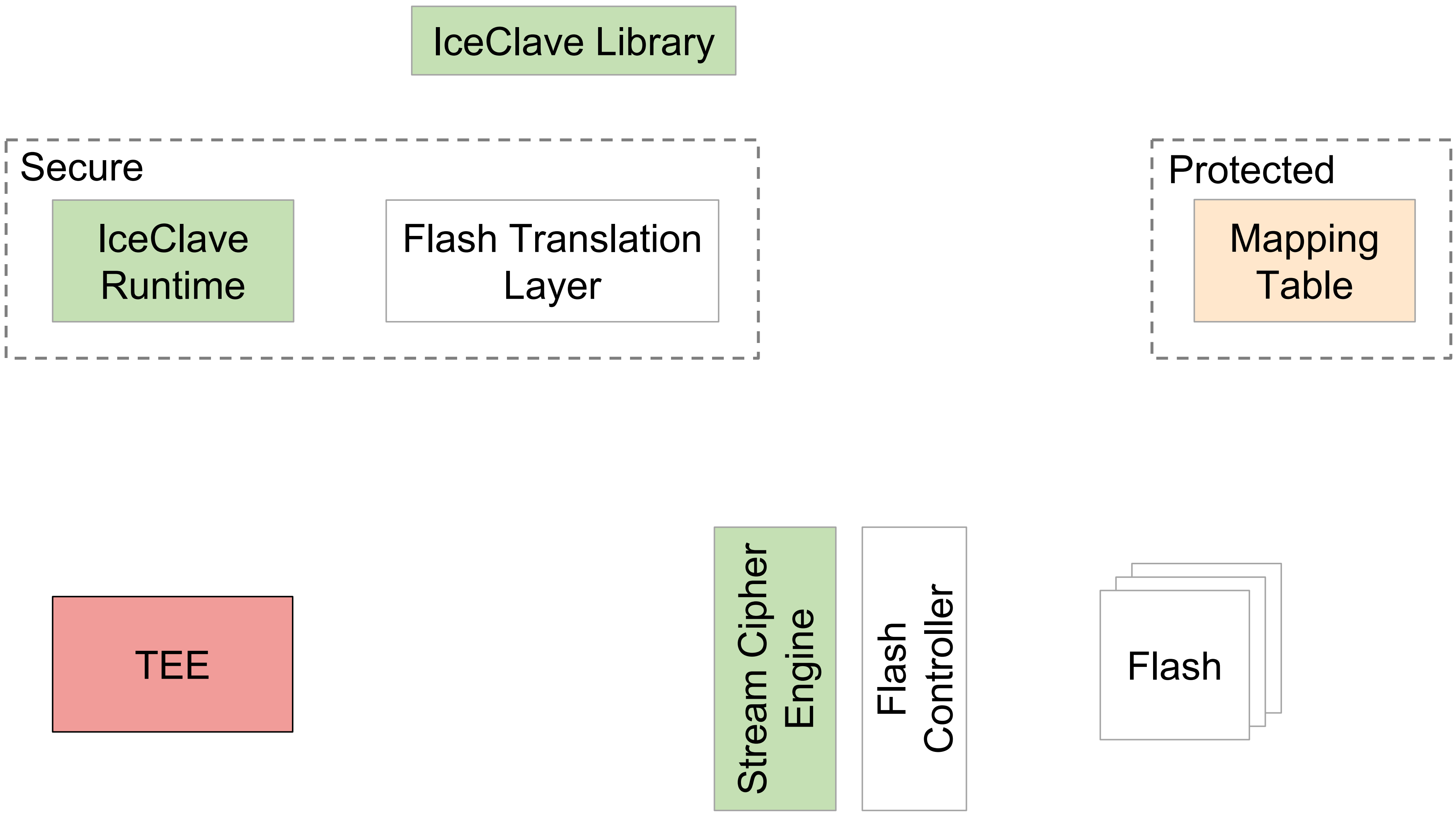
Put It All Together



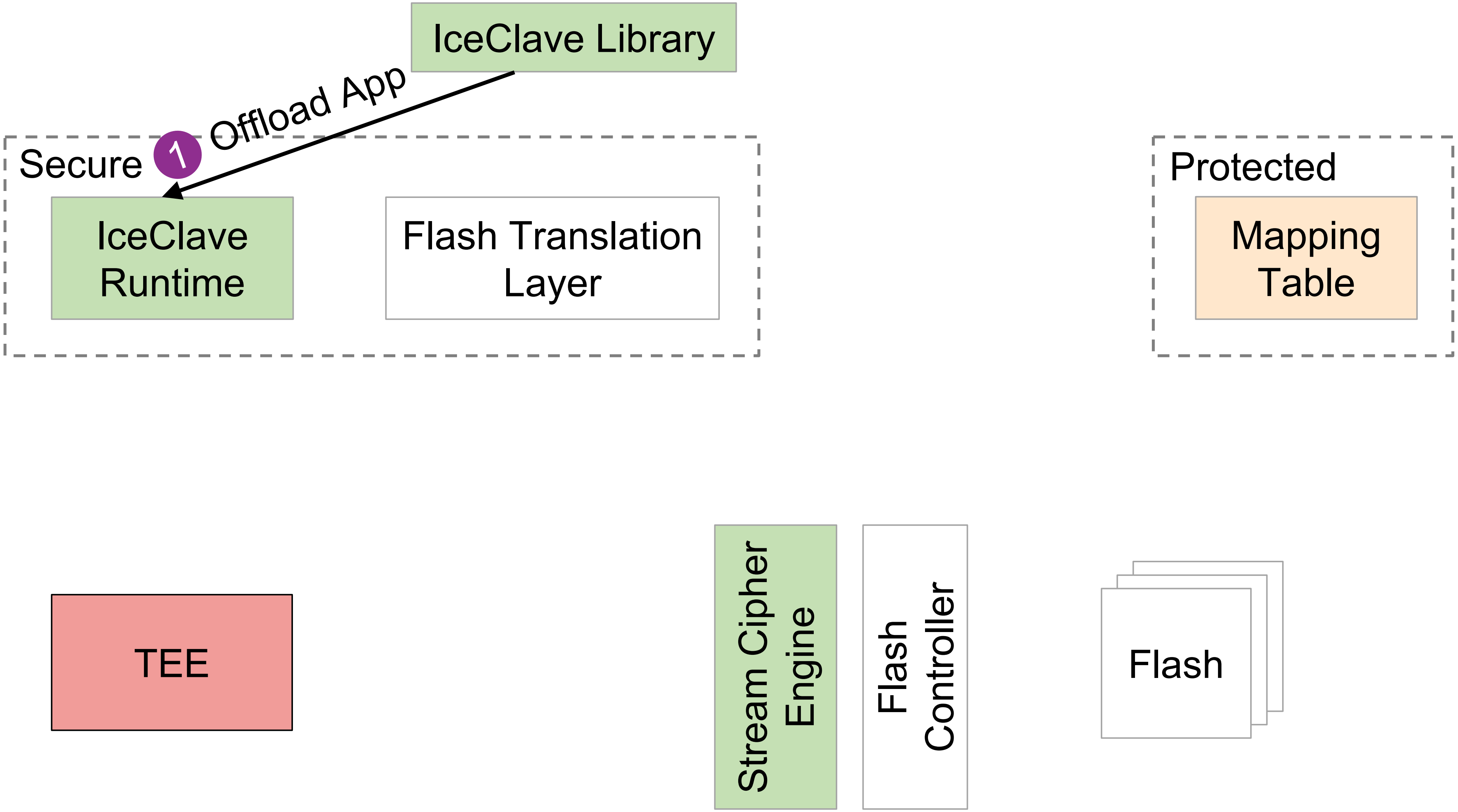
Put It All Together



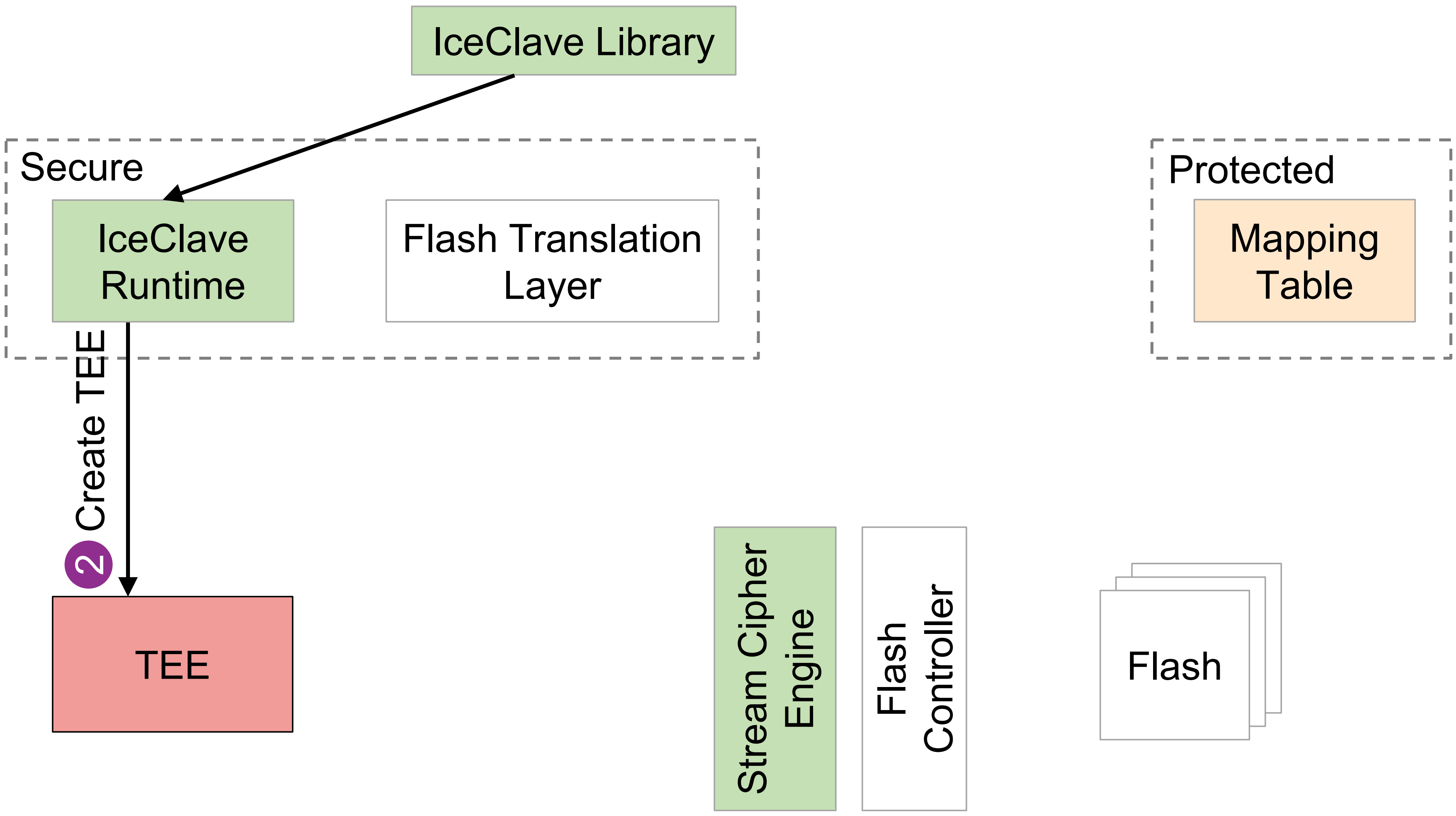
IceClave Workflow



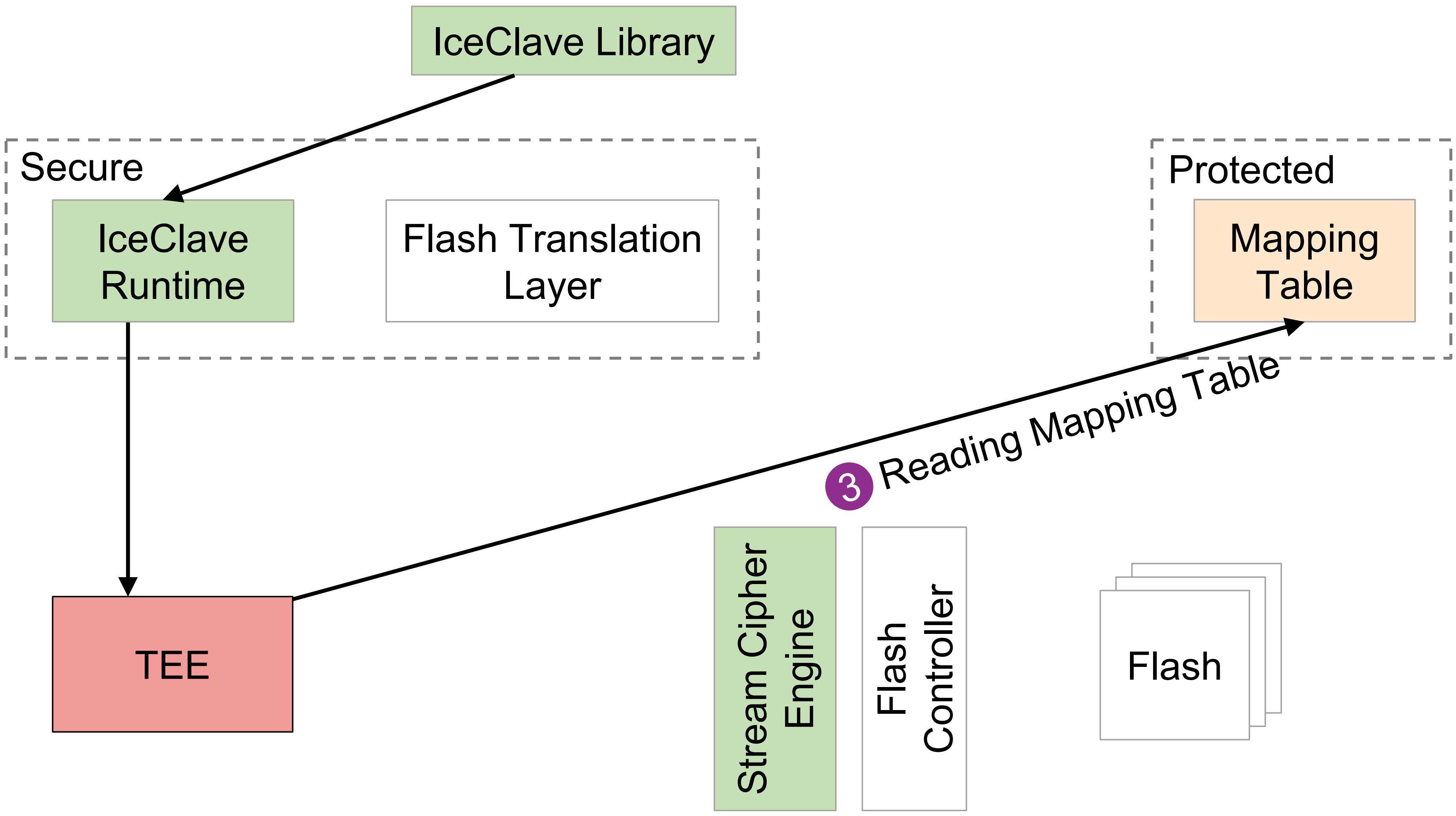
IceClave Workflow



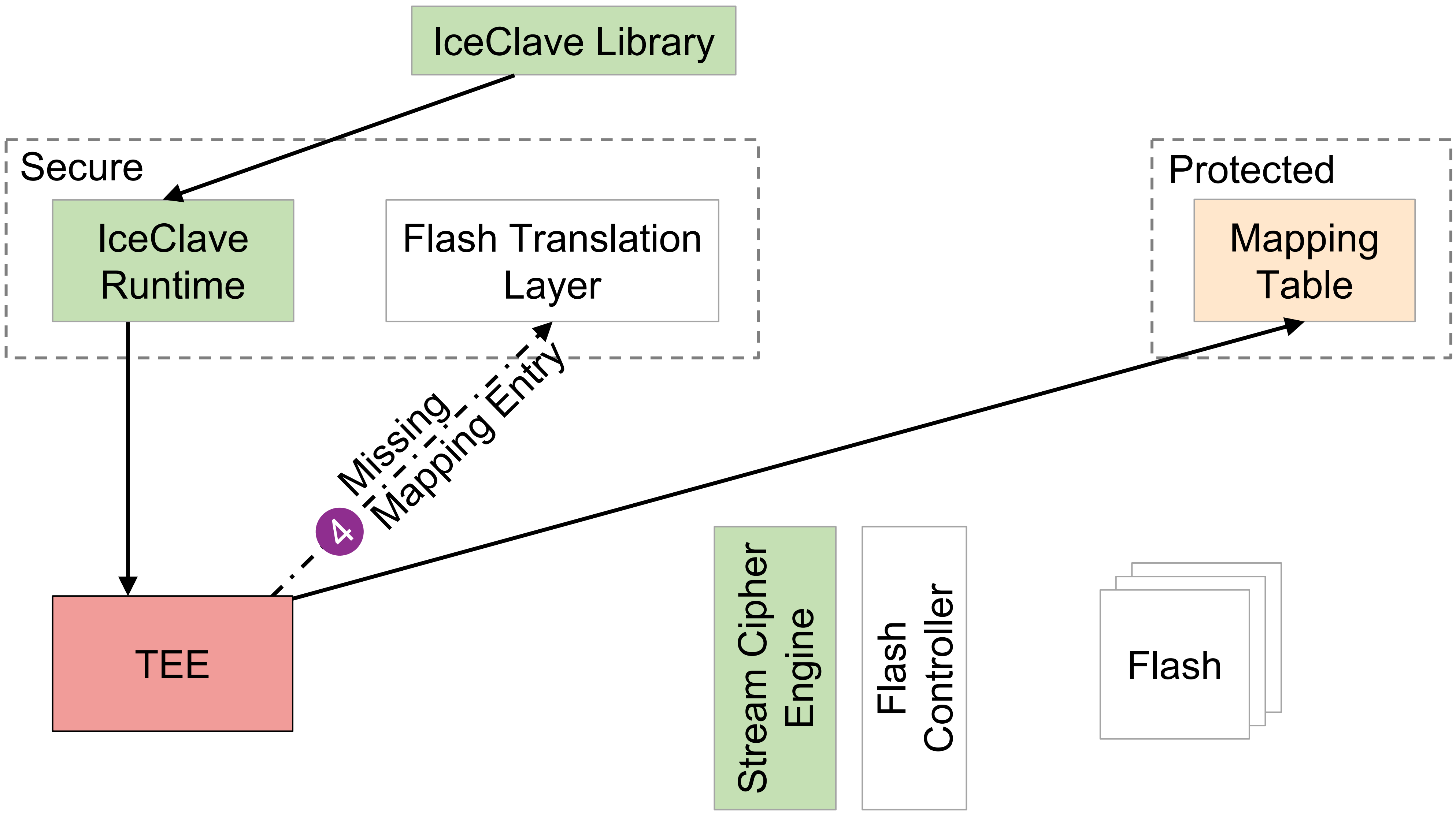
IceClave Workflow



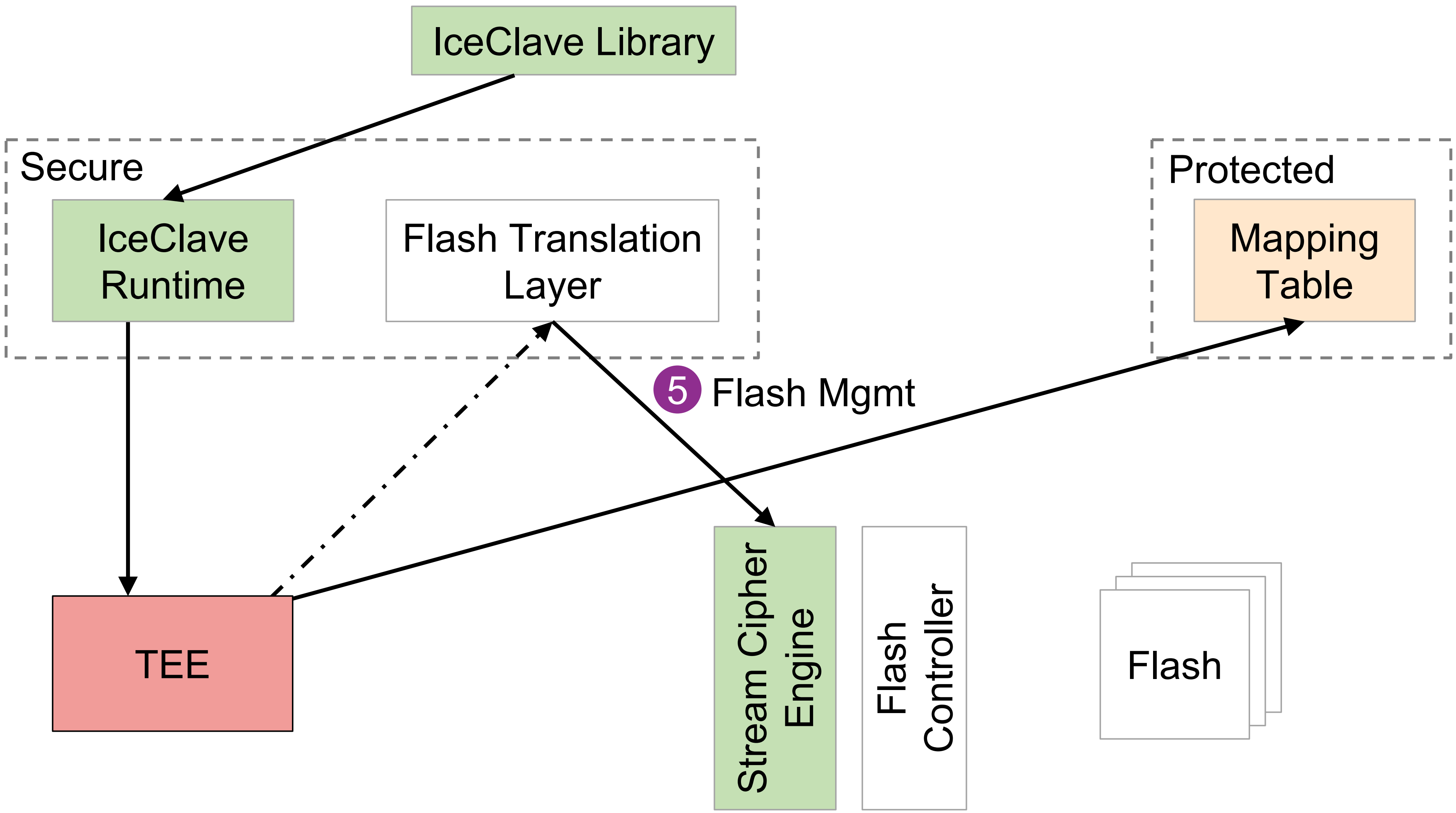
IceClave Workflow



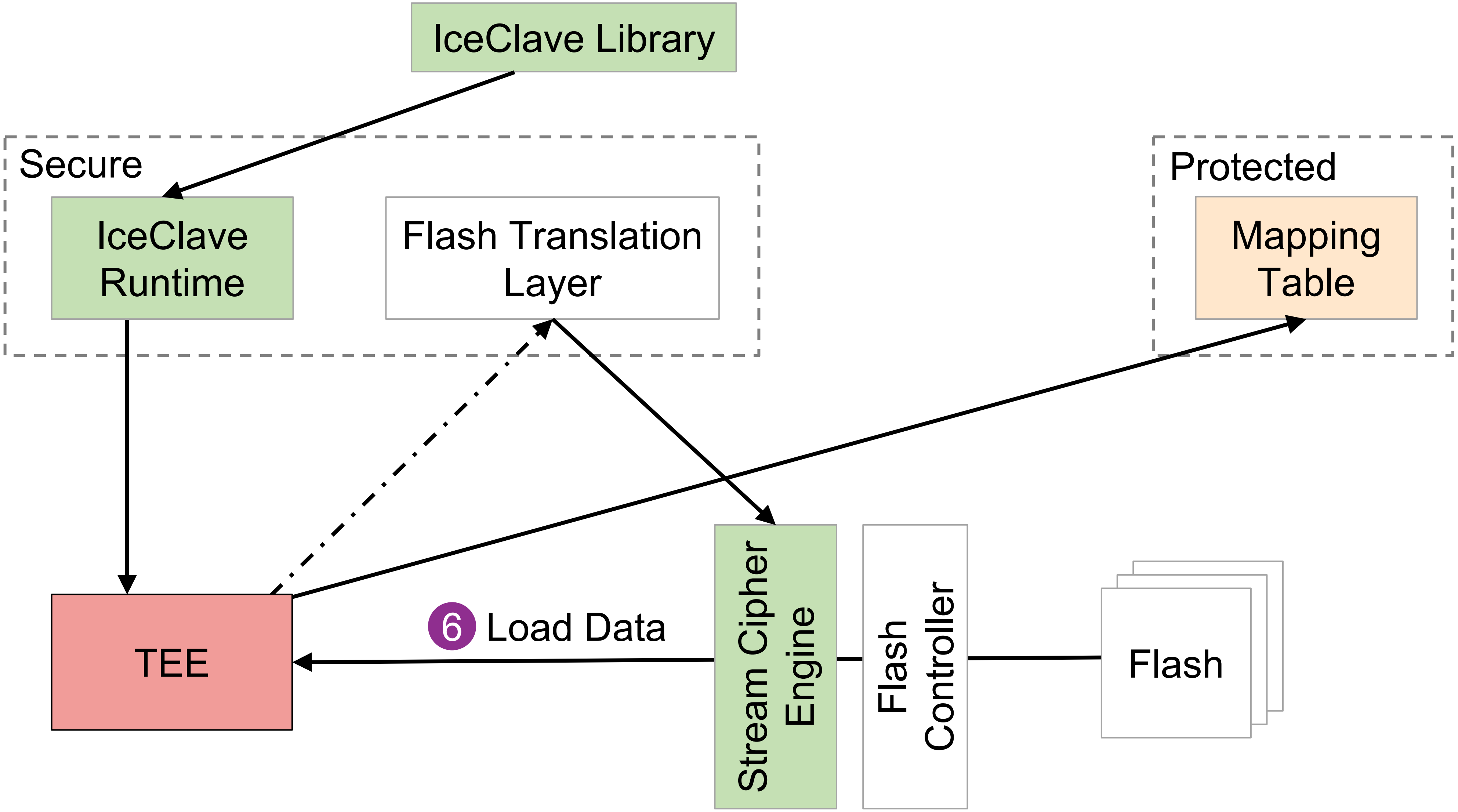
IceClave Workflow



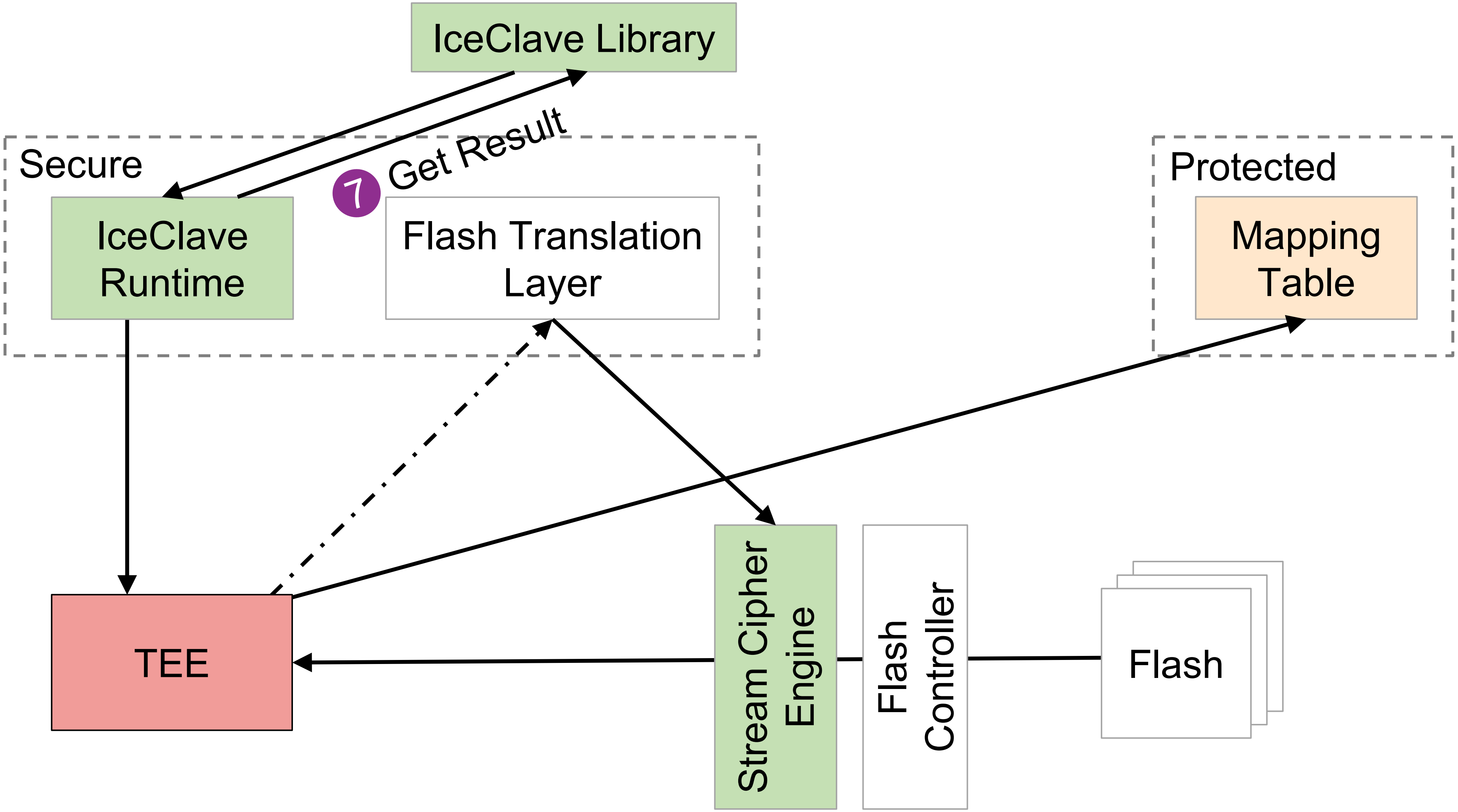
IceClave Workflow



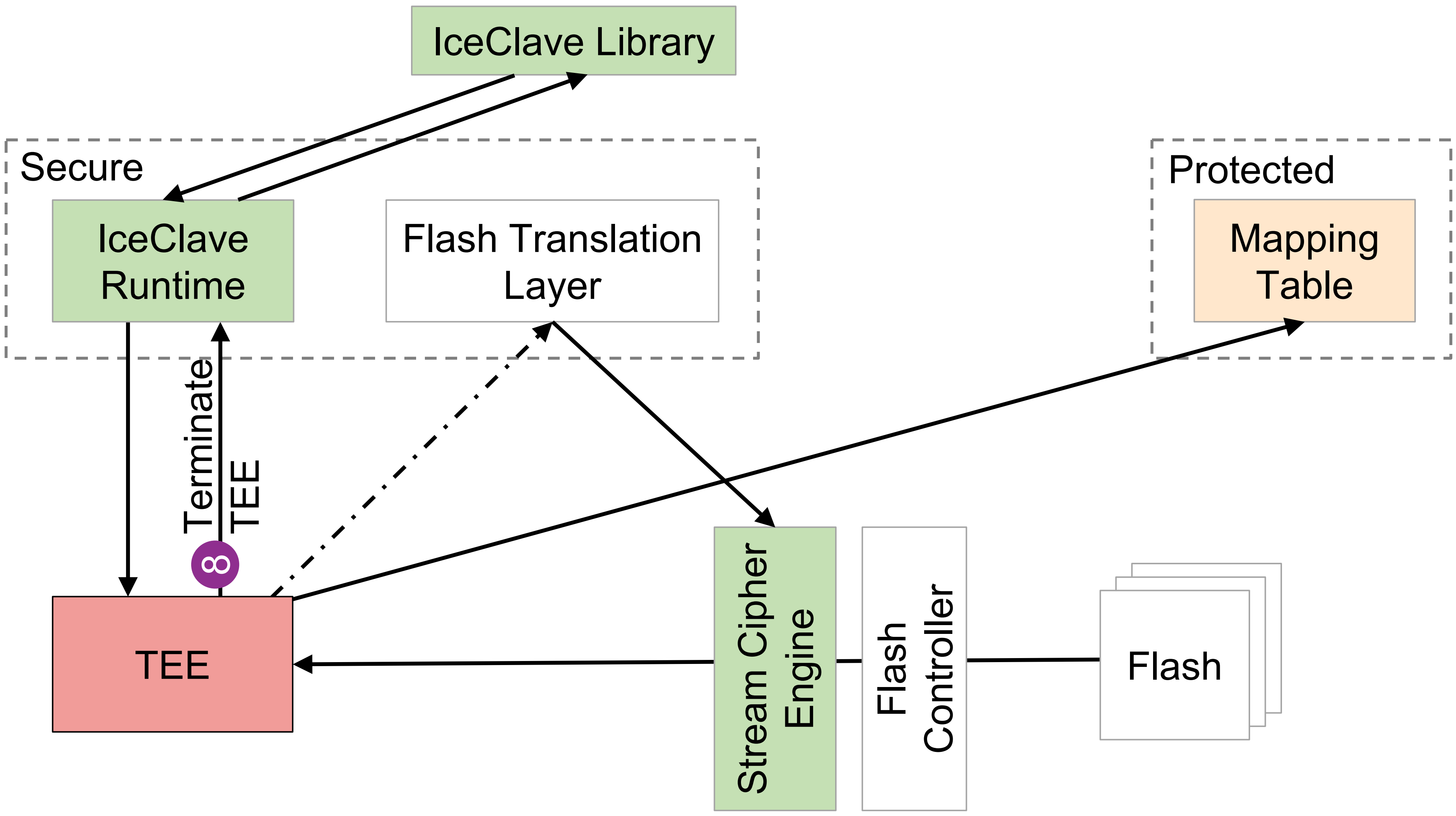
IceClave Workflow



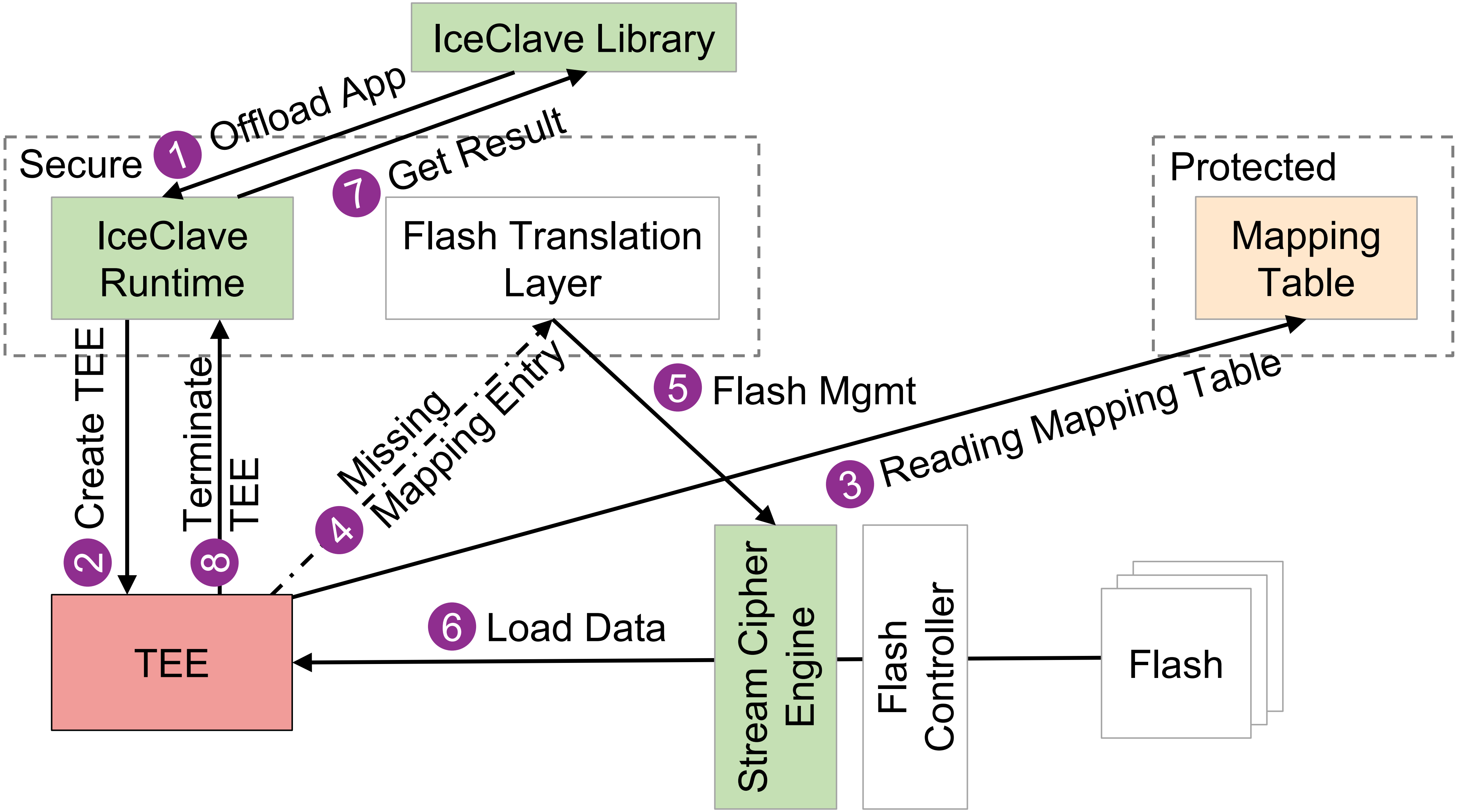
IceClave Workflow



IceClave Workflow



IceClave Workflow



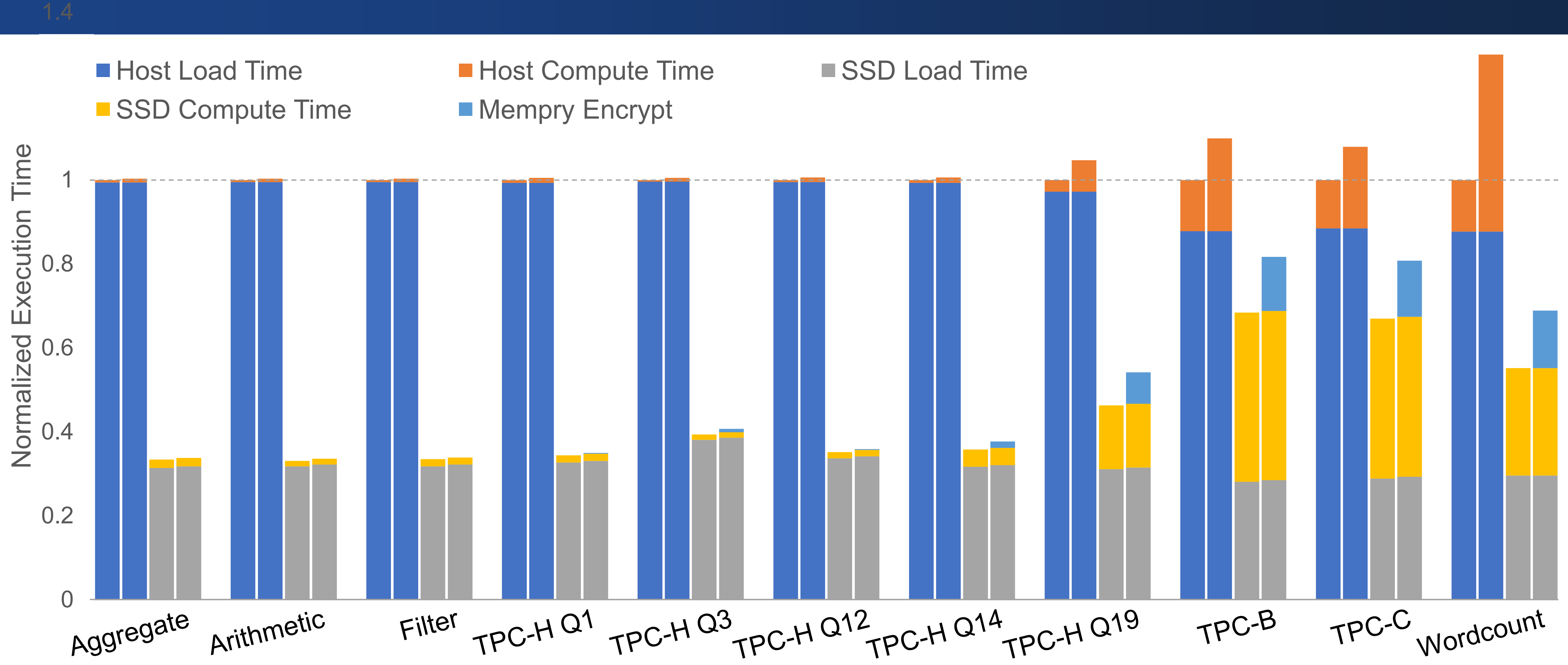
IceClave Implementation

Simulator	gem5 + USIMM + SimpleSSD
Prototype	OpenSSD Cosmos+ FPGA

Experimental Setup

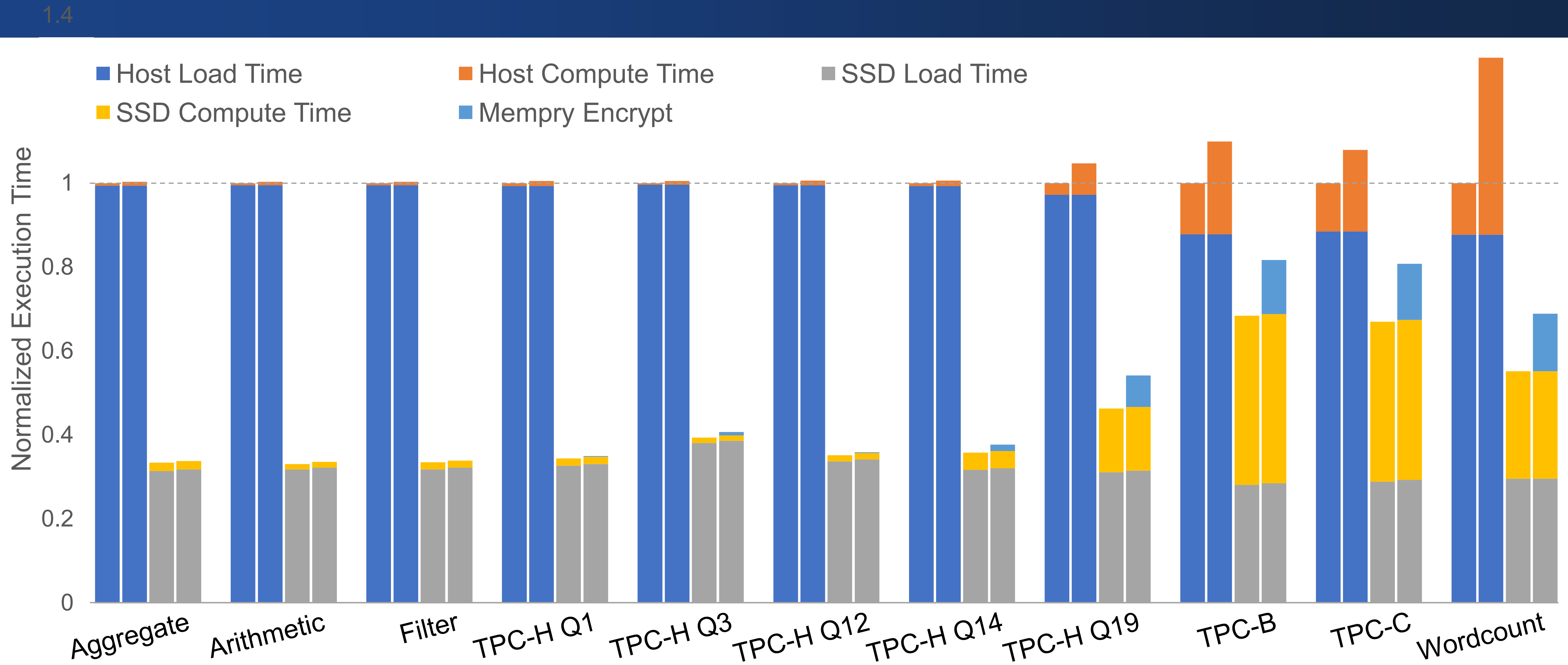
Synthetic Workloads	Arithmetic, Aggregate, Filter, Wordcount
Real-world Workloads	TPC-H, TPC-B, TPC-C

IceClave Overall Performance



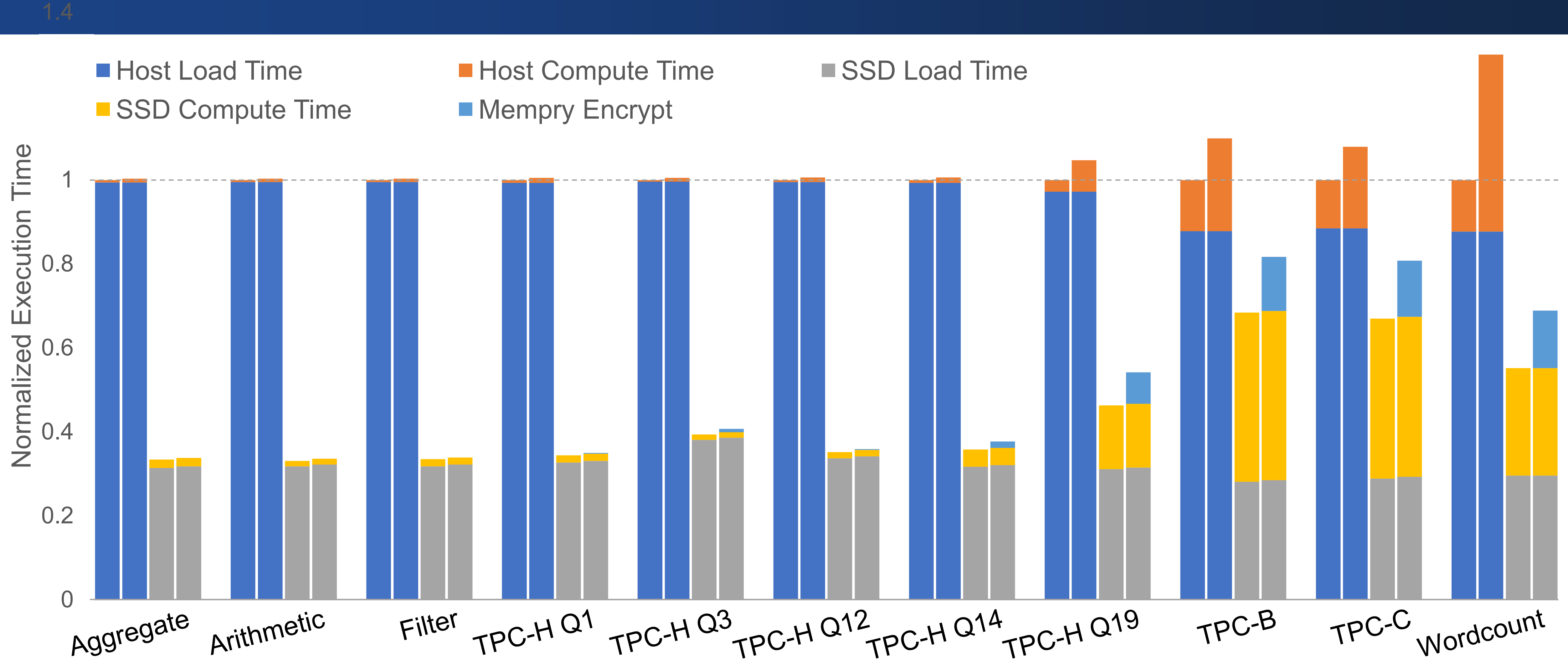
Left to Right: **Host** **Host+SGX** **ISC** **IceClave**

IceClave Overall Performance



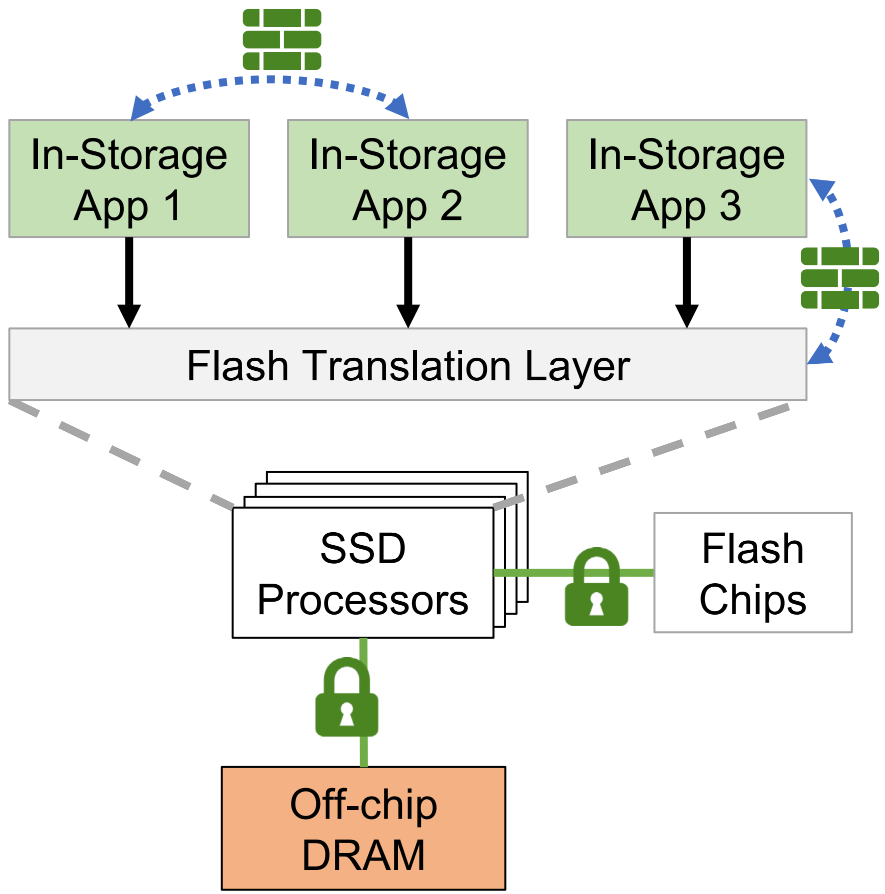
IceClave introduces minimal overhead while providing strong security

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

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN