Alohomora: Protecting Files from Ransomware Attacks Using Fine-Grained I/O Whitelisting

Sanggu Lee, Yoona Kim, Dusol Lee, Inhyuk Choi , and Jihong Kim

Dept. of Computer Science & Engineering

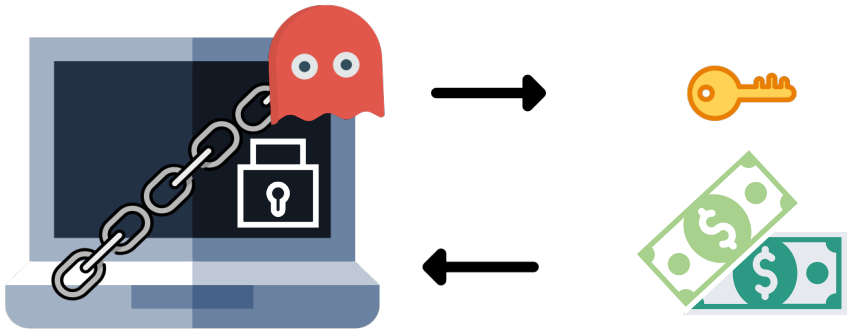
Seoul National University

June 28. 2022 (Virtual Event)

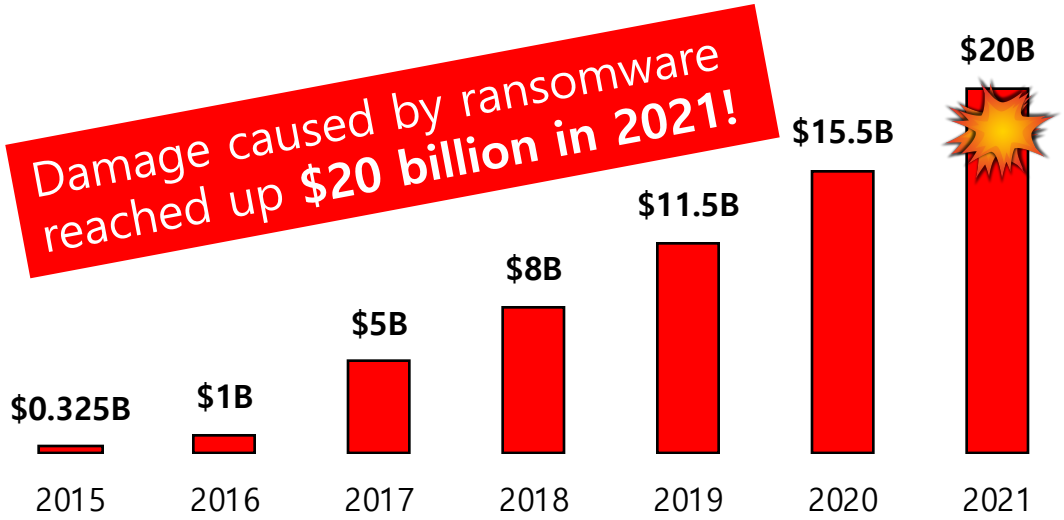
The 14th ACM Workshop on Hot Topics in Storage and File Systems (HotStorage '22)

Ransomware is Becoming a Major Cyber Threat

- Potential high financial payoff made ransomware one of the most serious threats in cyber security.
- Recent ransomware attack cases:
  - *Colonial Pipeline* paid $4.4 million for the ransom (2021)
  - *CNA Financial* paid $4 million for the ransom (2021)



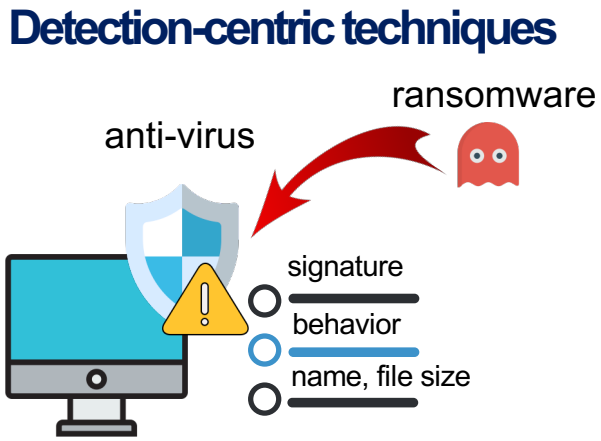
Encrypting ransomware



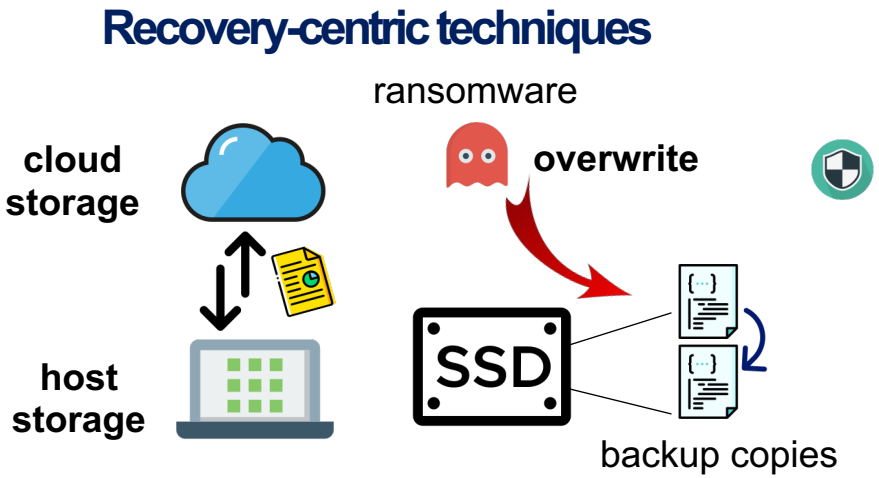
Global damage costs by ransomware

Existing Anti-ransomware Solutions

- Detection-centric techniques (e.g., anti-virus software)
- Recovery-centric techniques
  - Host-level data recovery (e.g., cloud backup)
  - SSD-level data recovery (e.g., FlashGuard (CCS'17), SSD-Insider (ICDCS'18))
- Permission-centric techniques (e.g., whitelisting solutions)

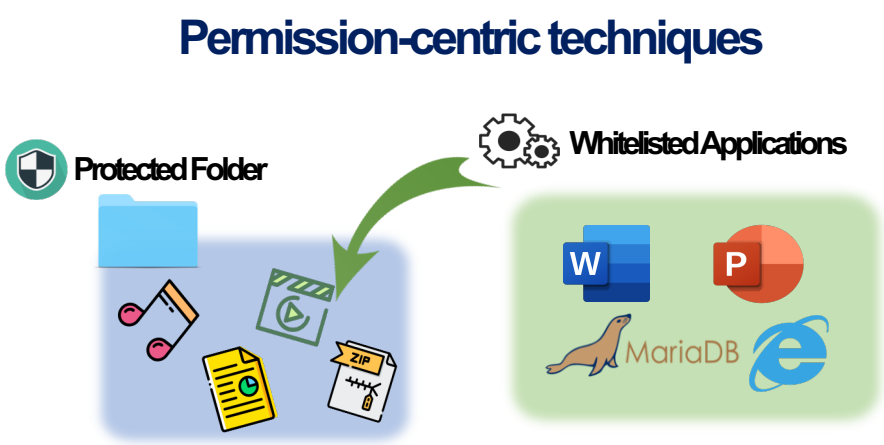


Anti-virus software



Host-level data recovery

SSD-level data recovery

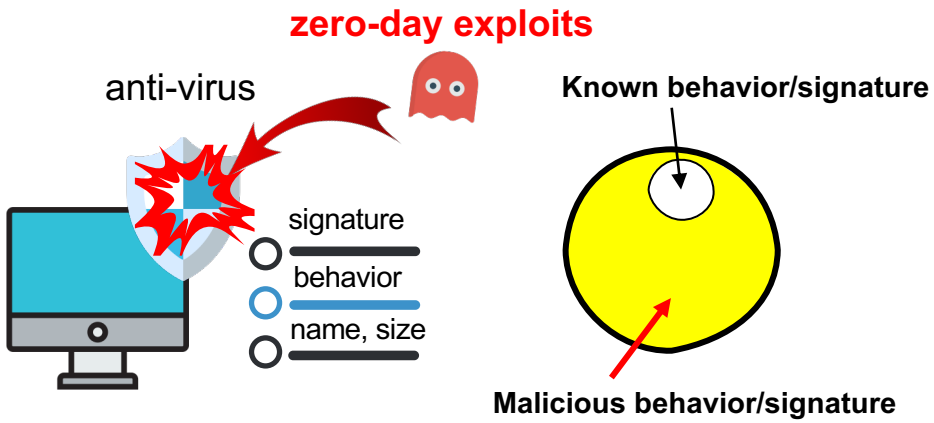


Whitelisting-based file access control

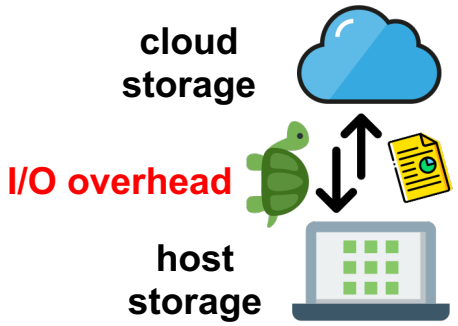
Limitations of Existing Anti-ransomware Solutions

- Detection-centric techniques:
  - Cannot prevent unknown ransomware behavior/signatures.
- Recovery-centric techniques
  - Backup operations incur excessive I/Os.
  - Performance degradation due to many backup pages.

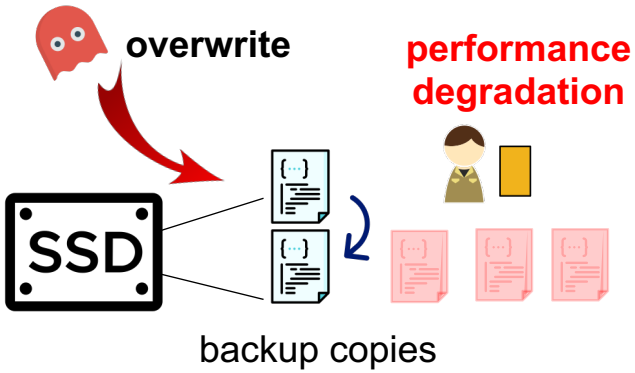
Limitations of detection-centric techniques



Limitations of recovery-centric techniques



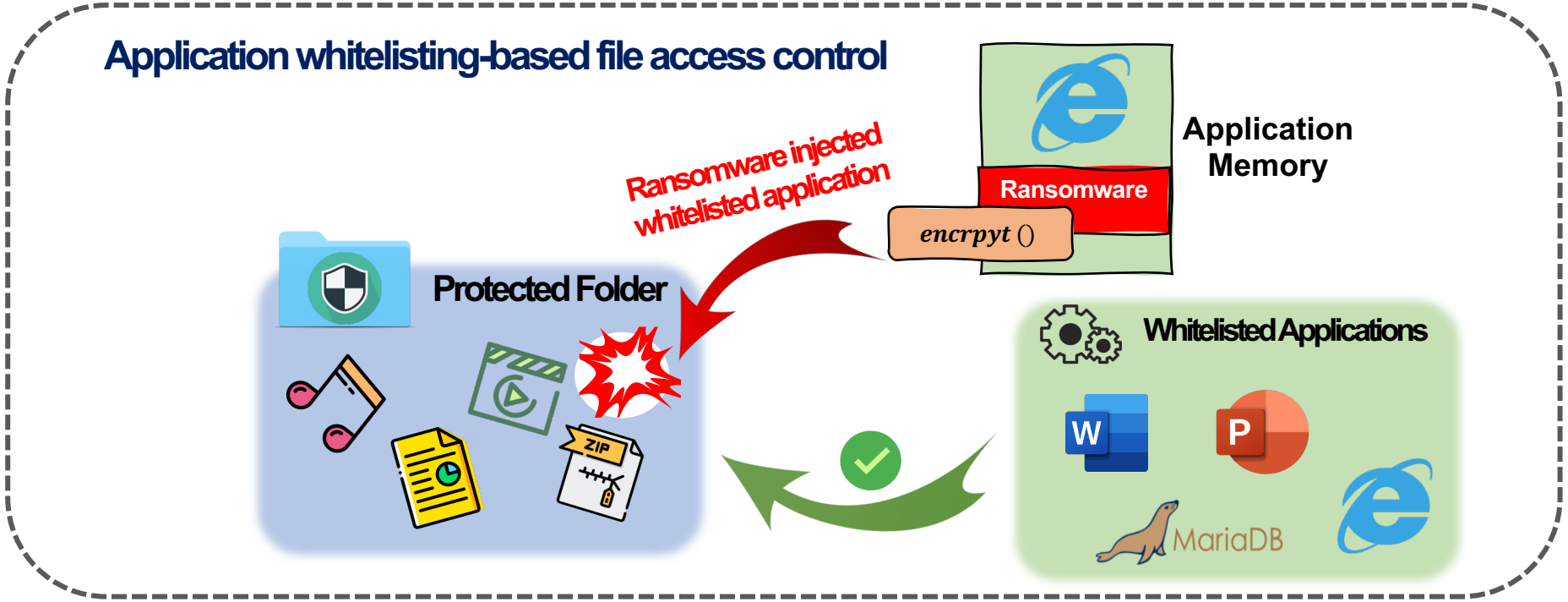
Host-level recovery



SSD-level recovery

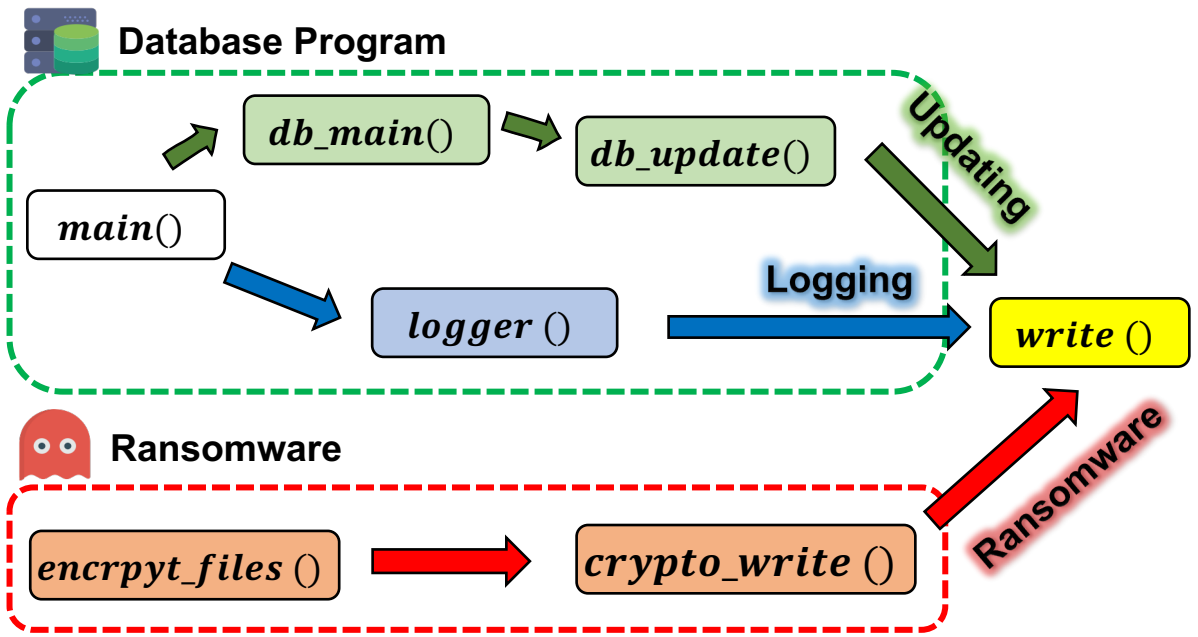
Limitations of Existing Anti-ransomware Solutions (cont'd)

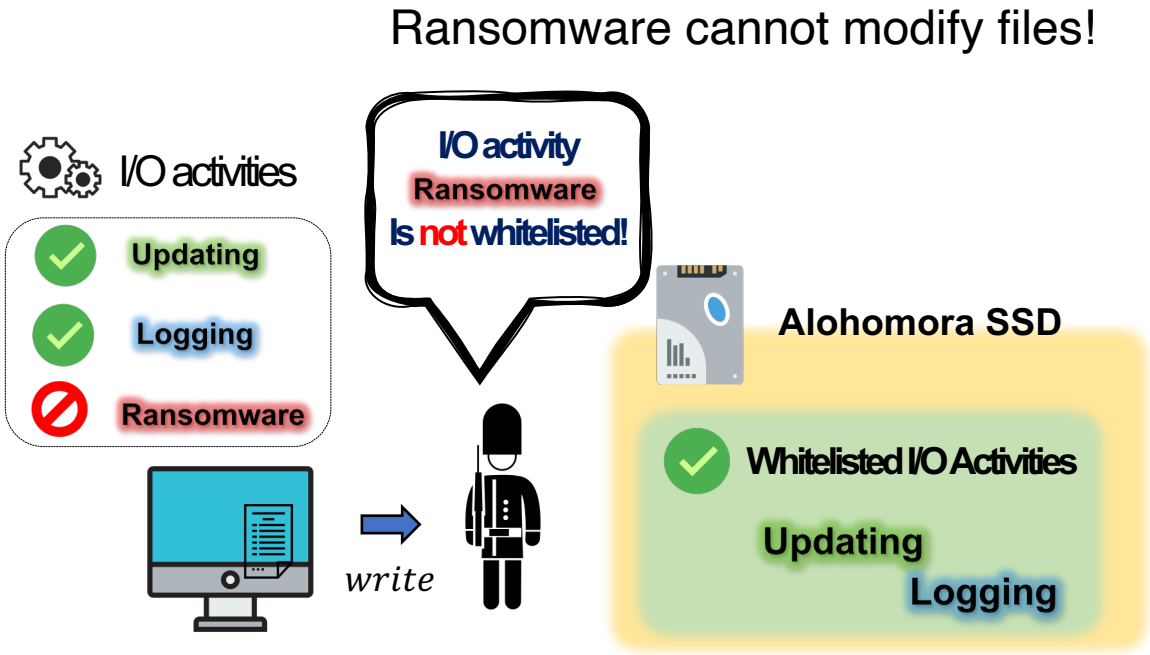
- Permission-Centric Techniques
  - Most solutions grant permissions at the application granularity.
  - Cannot prevent if ransomware is injected into a pre-approved (whitelisted) program.



Our Work: Whitelisting based on I/O Activity

- An **I/O activity** represents a particular I/O execution semantic context, an execution path of a program up to an I/O system call.
- **Alohomora**: whitelisting granularity is based on **I/O activity**.





Key Claim: I/O activities are considered **unique** over different apps!

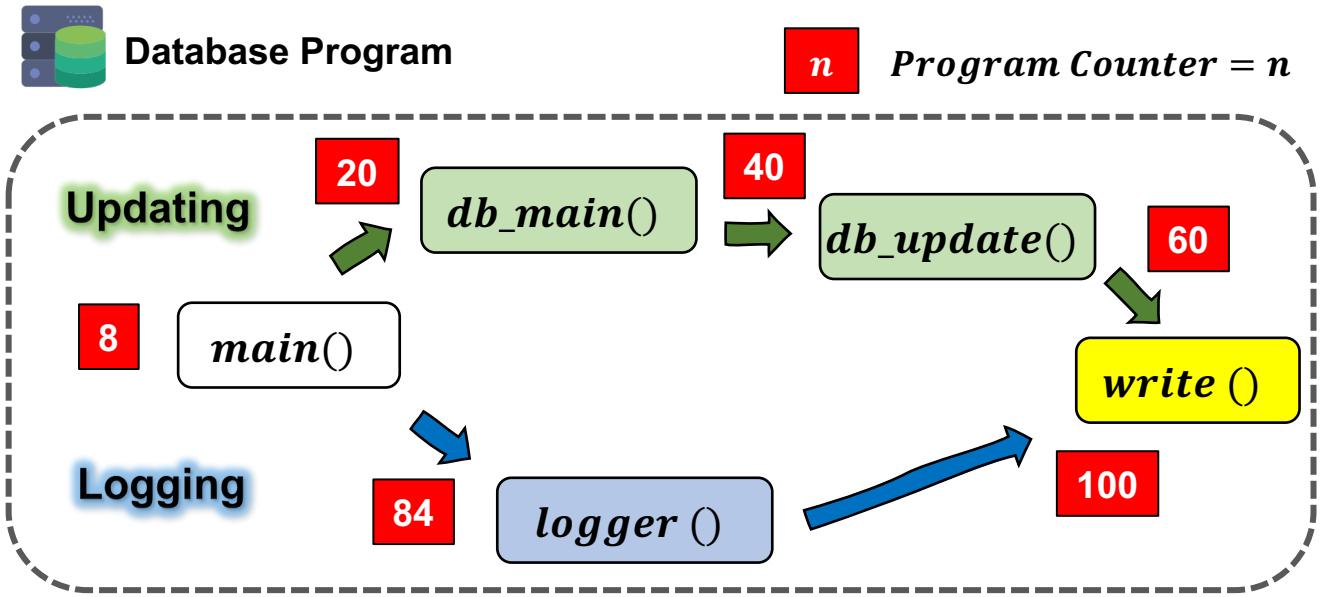
The Concept of I/O Whitelisting

Outline

- **Design of Alohomora**
- Experimental Results

I/O Activity Identification Using Program Contexts (PrCs)

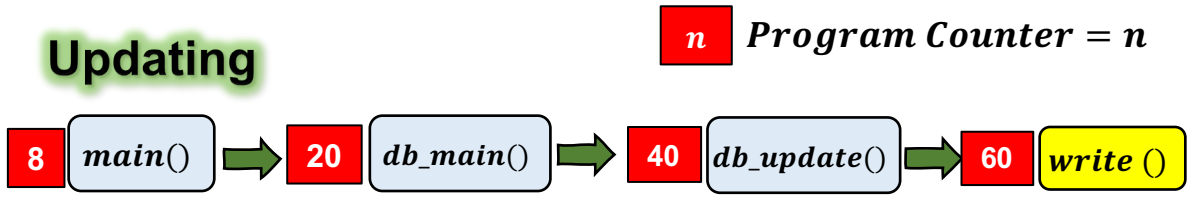
- I/O activity is represented using a **PrC (Program Context) value**.
  - A PrC value is specific to the execution path.
  - A PrC value is computed by **summing program counter (PC) values** of function calls along the execution path up to a write-related system call.

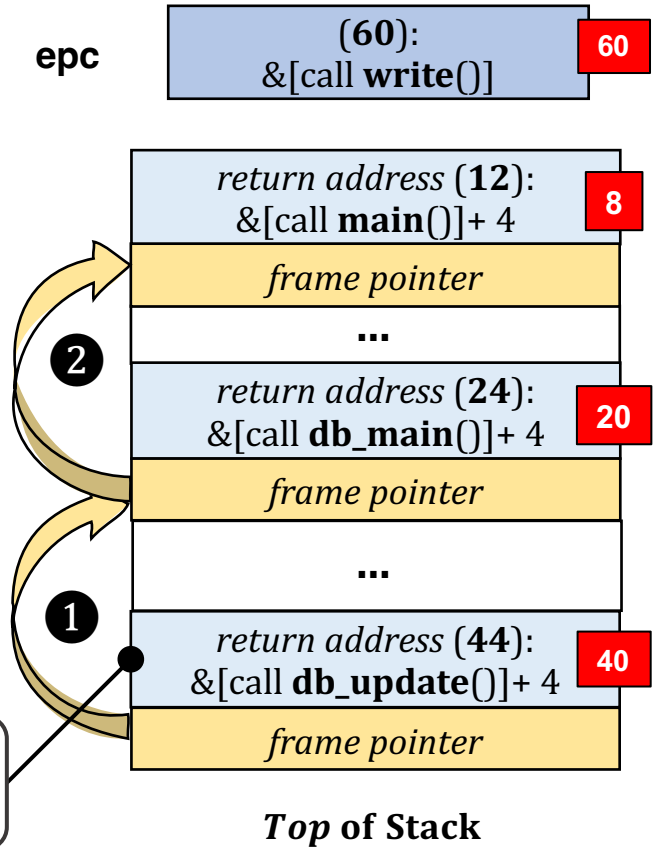


IO Activity	PrC Value
Updating	128 (8 + 20 + 40 + 60)
Logging	192 (8 + 84 + 100)

How to Extract PrC?

- A frame pointer-based SW method
  - The execution **call addresses** are acquired by backtracking stack frames using a frame-pointer.
  - Difficult to use in practice because many modern C/C++ compilers omit frame pointers.



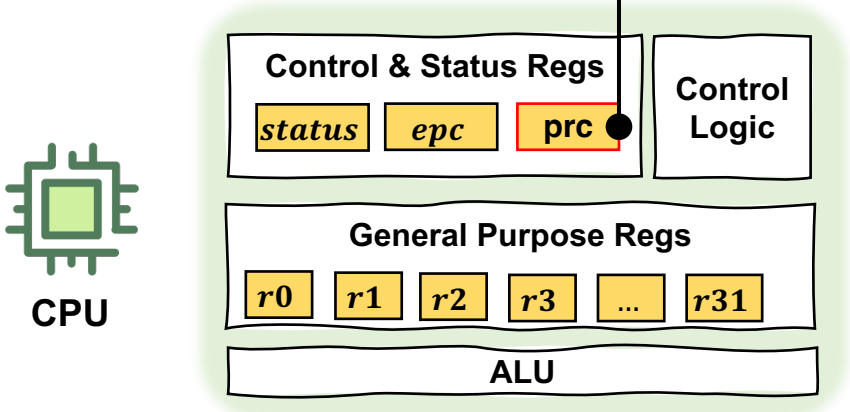


- The ***call address*** (***return address*** $-$ **4**) is saved in stack memory

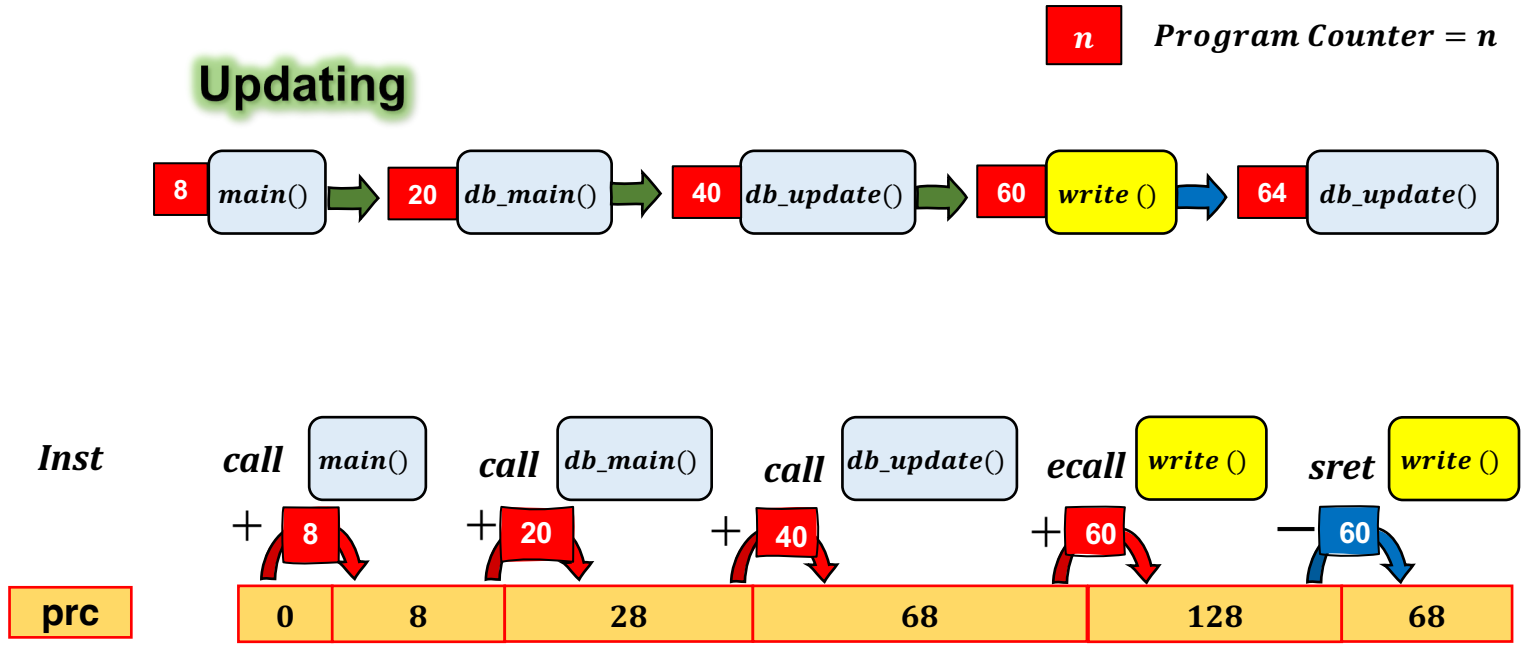
HW-based Automatic PrC Calculation

- **Alohomora** computes PrC value **fully by hardware** with an extension of a privileged register **prc.**

- A **prc** register is added to the Control & Status register.
- Upon ***Call*** / ***Ret*** instruction, the call address is **added/subtracted**.



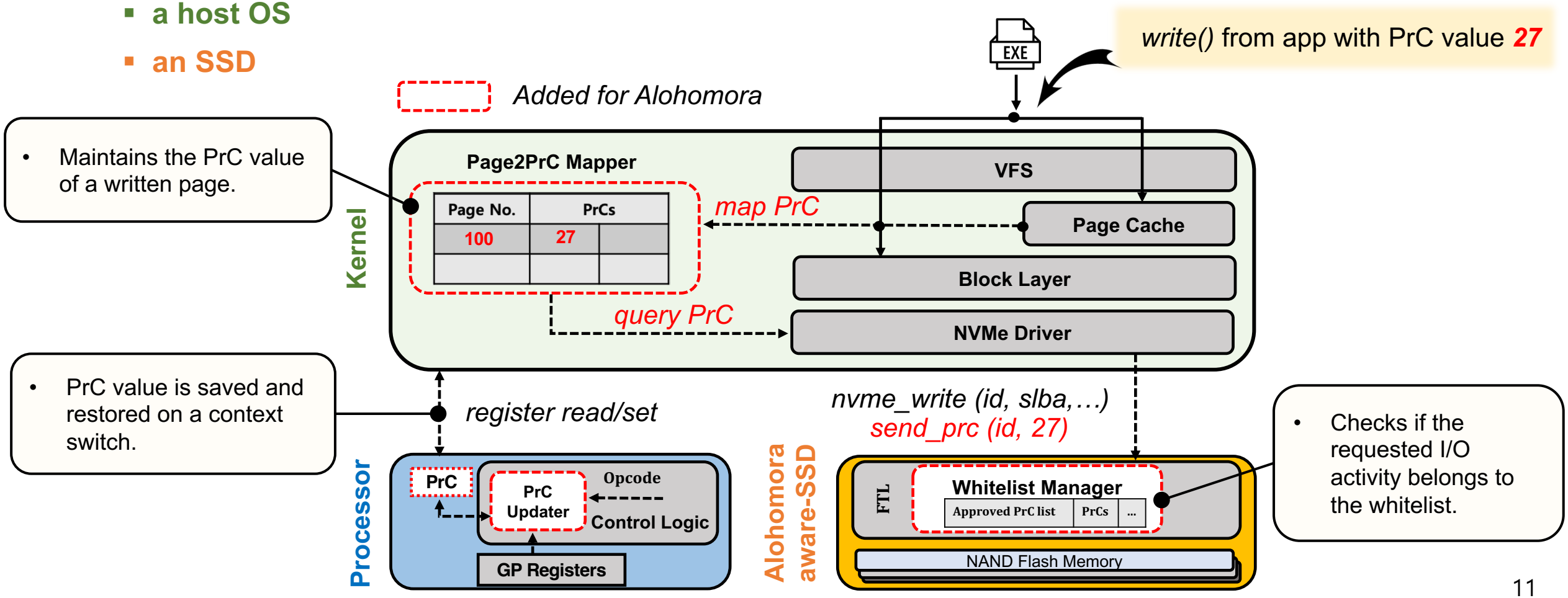
Extended RISC-V CPU with PrC Support



A hardware-based PrC Calculation method

Overview of Alohomora

- Alohomora requires modifications in:
  - a host CPU
  - a host OS
  - an SSD

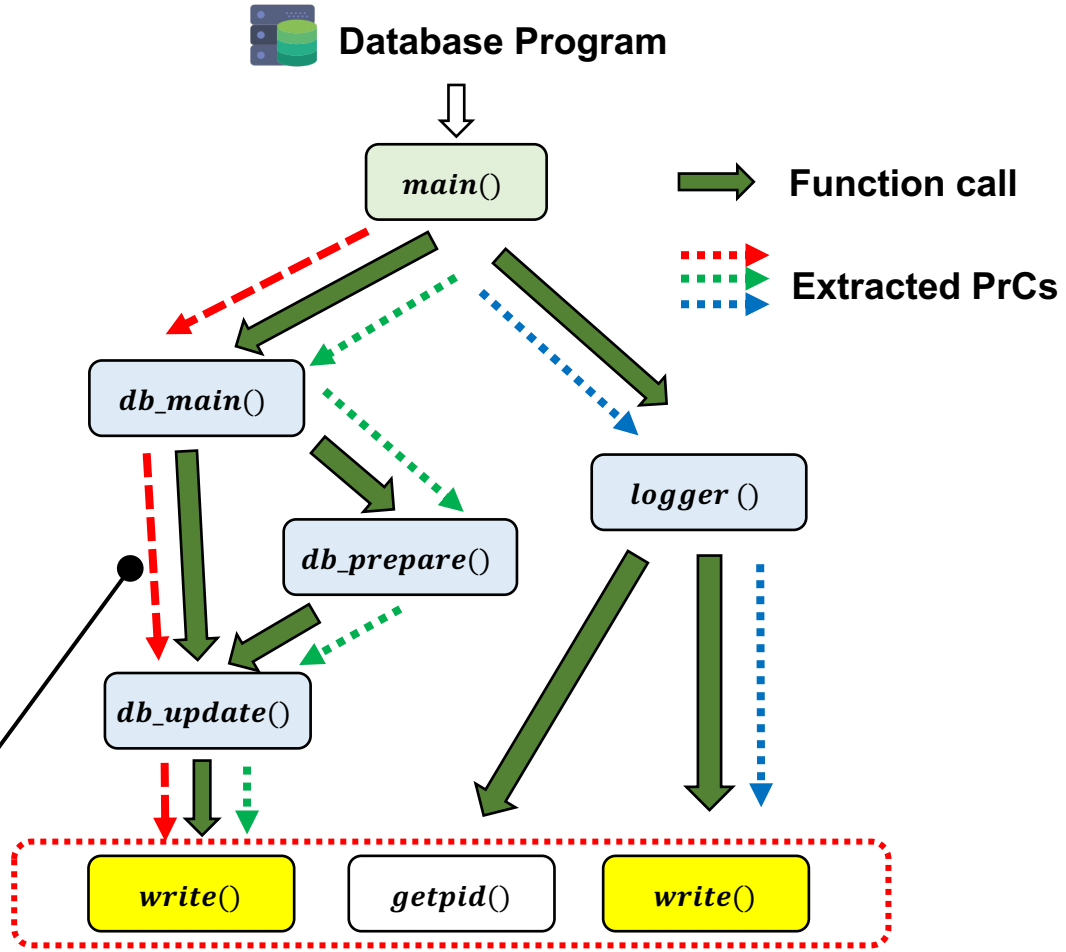


Whitelist Management

- A PrC value of a pre-approved I/O activity **should be known in advance** to create a whitelist.

- **Alohomora** employs both:
  - Static PrC extraction method
  - Dynamic PrC extraction method

- Using a call graph, potential write activates are identified with their PrC values.



Call graph of a Database Program

Outline

- Design of Alohomora
- **Experimental Results**

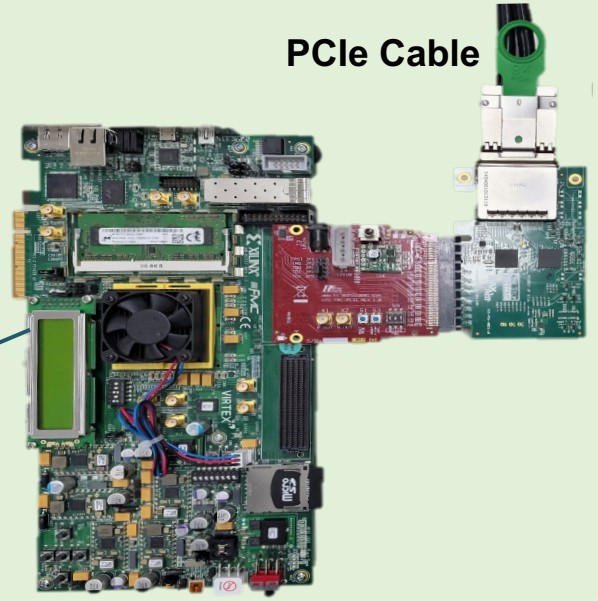
Prototype Alohomora Implementation

- **Alohomora Host**
  - Extended RISC-V (PrC Enabled) CPU synthesized on VC707 FPGA board
- **Alohomora-aware SSD**
  - PrC Whitelist Manger Implemented in OpenSSD Greedy-FTL

Alohomora Host

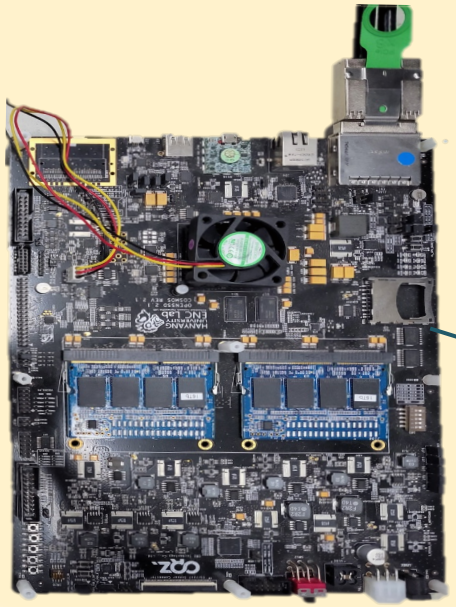
VC707 FPGA Board (Extended RISC-V)

PCIe Cable



CPU: Synthesized RISC-V X 4 Cores
ISA: RV64GC
DRAM Size: 4GB
OS: Linux Kernel 5.10

Alohomora-aware SSD



Cosmos+ OpenSSD Board

CPU: Arm Cortex A9
DRAM Size: 1GB
FTL: Greedy-FTL
Capacity: 512GB

Result 1: Ransomware Defense Capability

- **Alohomora** successfully defended against 37 public ransomware programs (e.g., GonnaCry, RAASNet, Ransom0, Hidden-tear, etc).
- **Alohomora** successfully defended against sophisticated ransomware programs.

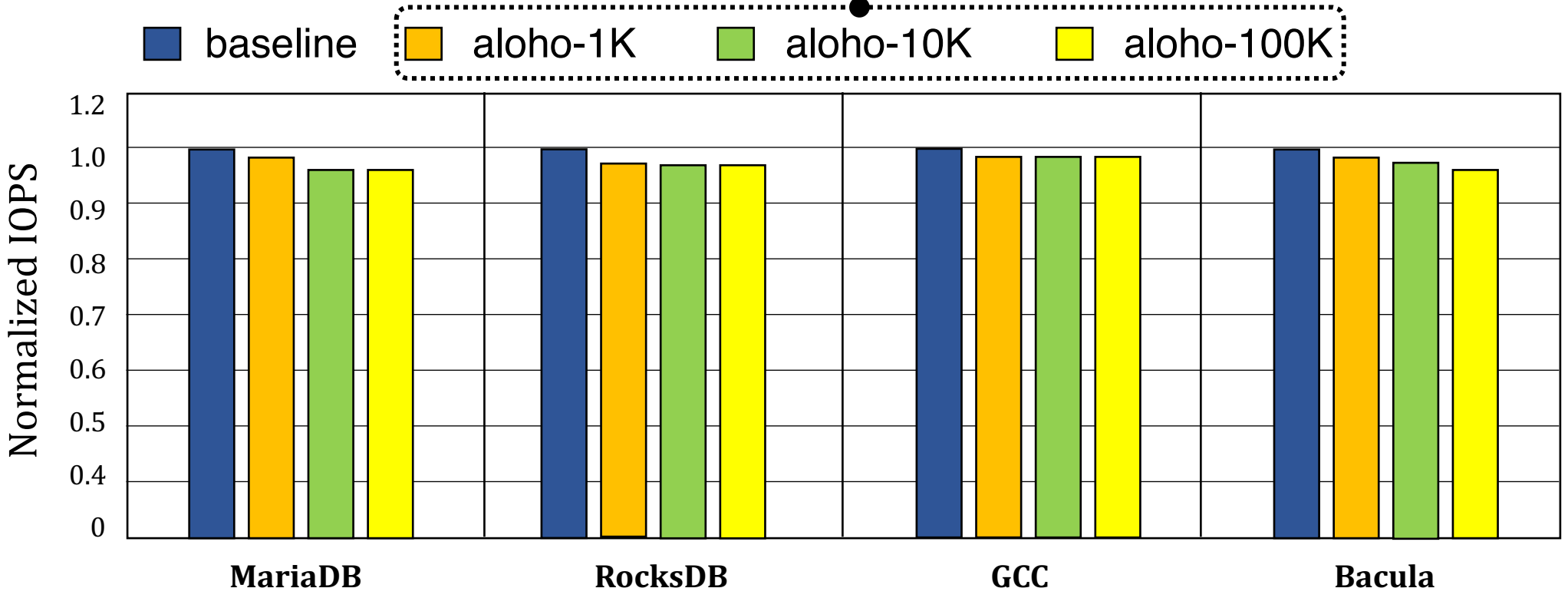
Ransomware		Application	
Name	# of PrCs	Name	# of PrCs
GonnaCry	7	MariaDB	152
RAASNet	5	RocksDB	51
Ransom0	3	GCC	131
Hidden-tear	4	Bacula	35
FSociety	4	MariaDB	152
CustomRS	60	RocksDB	51

A summary of synthetic attack cases.

- **Intense attack scenario:**
  Ransomware code is injected to the pre-approved applications.

Result 2: Performance Overhead

Tested on different number of whitelist entries (aloho-1K, 10K, 100K)



For aloho-1K, only 1.9% of average IOPS drop occurs.

For aloho-100K, the average IOPS penalty is merely 3.7%.

Conclusions

- Presented **Alohomora**, a whitelisting-based anti-ransomware solution.
  - The whitelisting of **Alohomora** is based on the **I/O activity** of an application.
  - The I/O activity of the application is represented by a **hardware-supported** PrC value.
- Demonstrated that **Alohomora** provides near-perfect protection with almost no I/O performance degradation.

Thank You