# Cabin: Confining Untrusted Programs within Confidential VMs

**Benshan Mei**, Saisai Xia, Wenhao Wang, Dongdai Lin

Key Laboratory of Cyberspace Security Defense, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China;

School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China

**The 26th International Conference on Information and Communications Security**

26-28 August 2024 | Mytilene, Greece.

中国科学院信息工程研究所

INSTITUTE OF INFORMATION ENGINEERING,CAS

中国科学院大学

University of Chinese Academy of Sciences

# Outline

- Introduction
- System Design
- Implementation
- Performance Evaluation
- Discussion
- Conclusion

# Introduction

# Introduction

- Confidential Computing
  - Safeguards sensitive computations from untrusted clouds
- Confidential Virtual Machines (CVMs)
  - Provide a secure environment for guest OS

# Confidential Virtual Machines

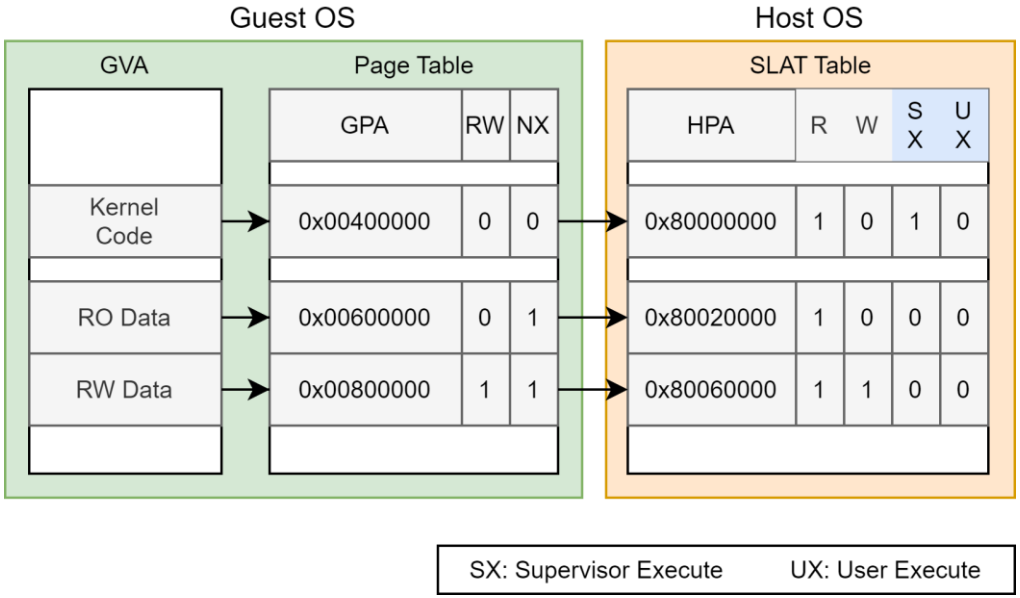
- AMD SEV (Secure Encrypted Virtualization)
- Intel TDX (Trust Domain eXtension)
- ARM CCA (Confidential Computing Architecture)

# Challenges with existing CVMs

- Large and vulnerable guest OS
- Imprecise control over page table on x86 processors
  - There are R/W, NX, and U/S bits currently.
  - Read/write access rights are non-orthogonal
  - Hinder effective implementation of execute-only memory (XOM)
- Lack of security hierarchy
  - Direct threat from untrusted programs

# Mode-Based Execution Control

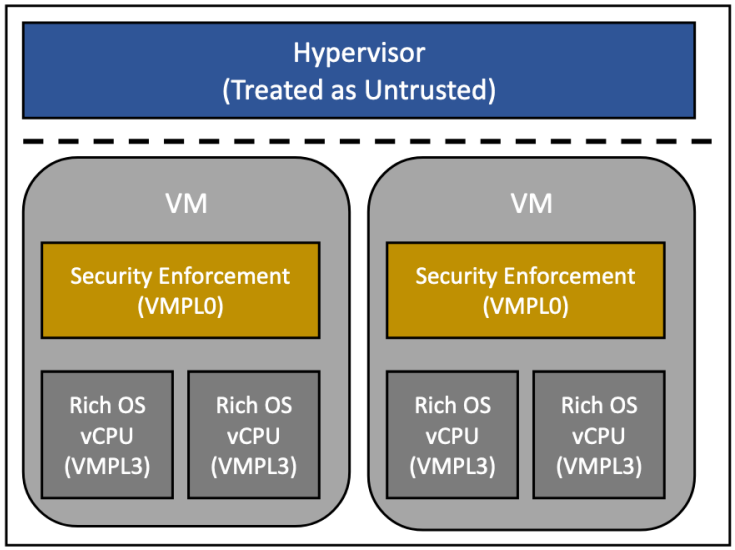
- Mode-Based Execution Control
  - Read/Write privilege separation
  - User/Supervisor execute privilege separation




- Intel Extended Page Table (EPT) and AMD Rapid Virtualization Indexing (RVI) support Second-Level Address Translation (SLAT)
- Intel Mode-Based Execution Control (MBEC) and AMD Guest Mode Execution Trap (GMET) support the mode-based execution

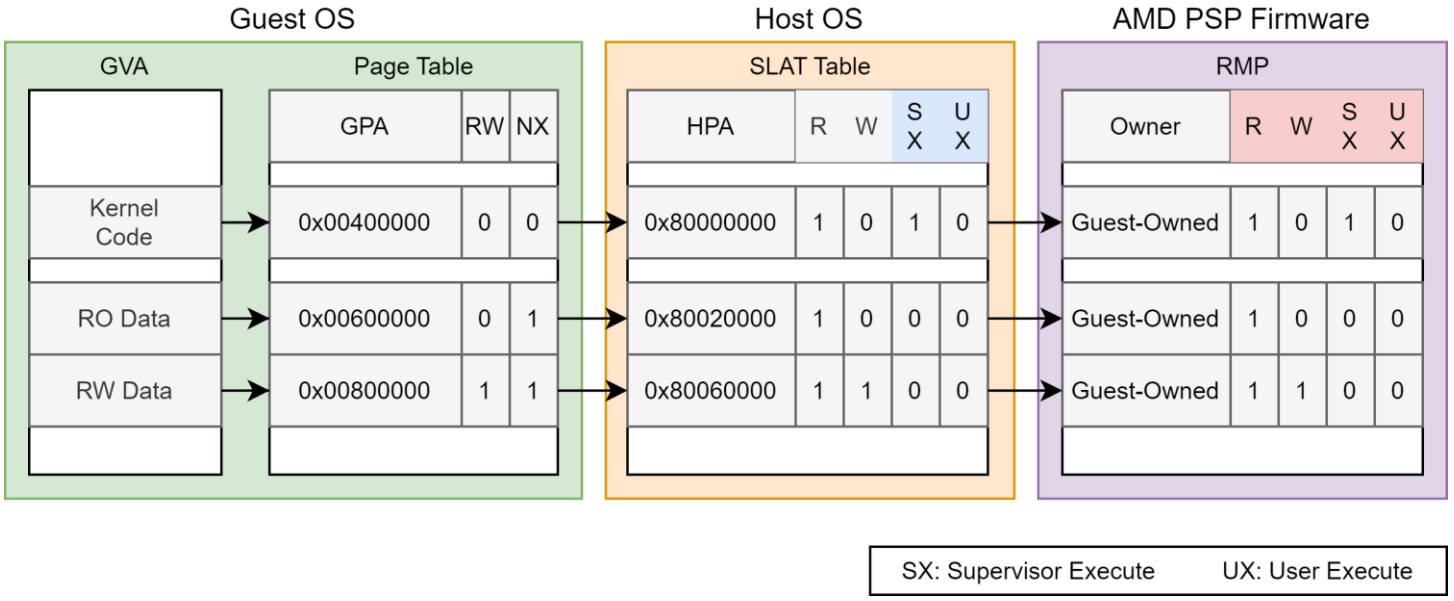
# AMD SEV-SNP (Secure Nested Paging)

- The 3rd generation of AMD SEV technology
  - Memory encryption and isolation
  - Reverse Mapping Table (RMP)
  - Virtual Machine Privilege Levels (VMPLs)



# Virtual Machine Privilege Levels

- Read/write privilege separation
- User/supervisor execute privilege separation
- Supports up to four VMPLs (VMPL0~VMPL3)

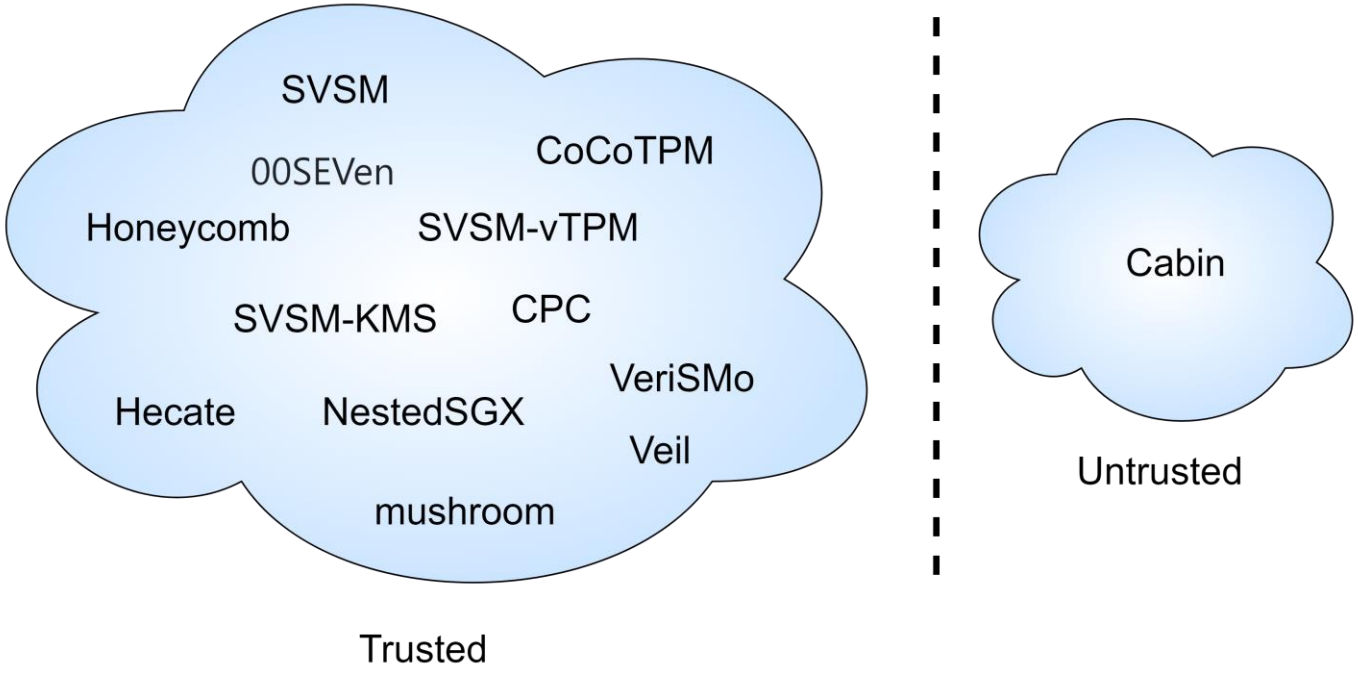


# Guest-Hypervisor Communication Block

- GHCB (Guest-Hypervisor Communication Block) protocol
  - Shared memory pages between guest OS and hypervisor
- Critical service requests
  - AP_CREATE: Create vCPU for lower VMPLs
  - RUN_VMPL: Request the hypervisor to run in specific VMPL.

# Motivations

- Existing works focus on Trusted Applications
- We focus on Untrusted Applications



# Threat Model

- Confidential Computing
  - Untrusted: Hypervisor
  - Trusted: Guest OS, Firmware, Hardware
- This work
  - Aligns with Confidential Computing
  - Focus on Untrusted Applications in CVM
  - Exclusion of ~~side channels~~ and ~~hardware attacks~~

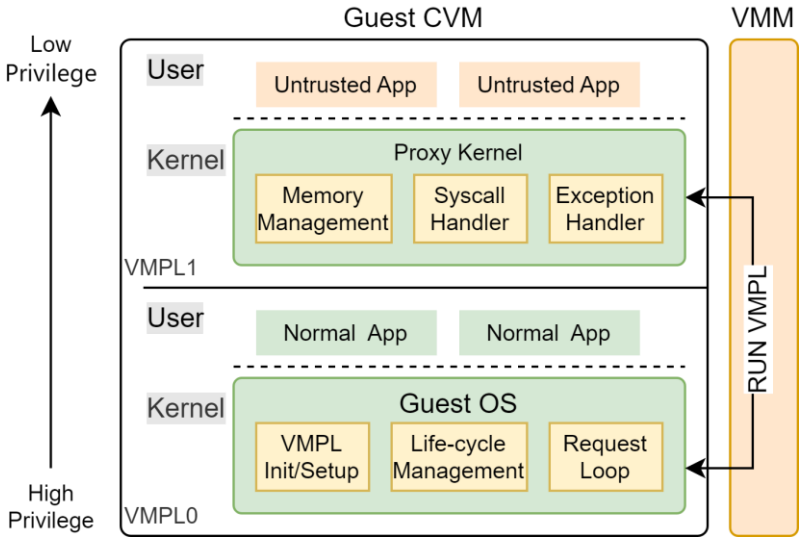
# Introduction to Cabin

- Objective
  - Shields untrusted applications from guest OS
- Approach
  - Utilizes AMD SEV-SNP technology
  - Fine-grained control over VMPL permissions
  - Introduction of a proxy-kernel

# System Design

# System Design

- Overview of the Cabin framework
  - Proxy-kernel
- Key components:
  - Life-cycle management
  - Context switch
  - Syscall routing
  - Exception model

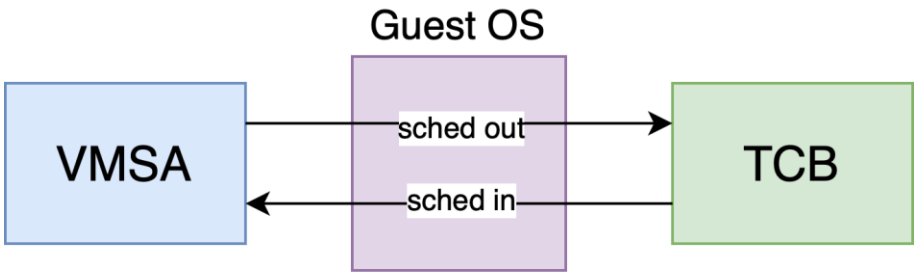


# Life-Cycle Management

- Setup
  - Prepare environment for all lower VMPLs
- Entering
  - Initialize runtime environment for current thread
- Request Loop
  - Forward syscall/exception events from lower VMPL
- Finalize
  - Release resource for confined process

# Context-Switch

- VM Saved Area (VMSA)
  - Contains hardware state of each VMPL
  - Managed by guest OS
- Task Control Block (TCB)



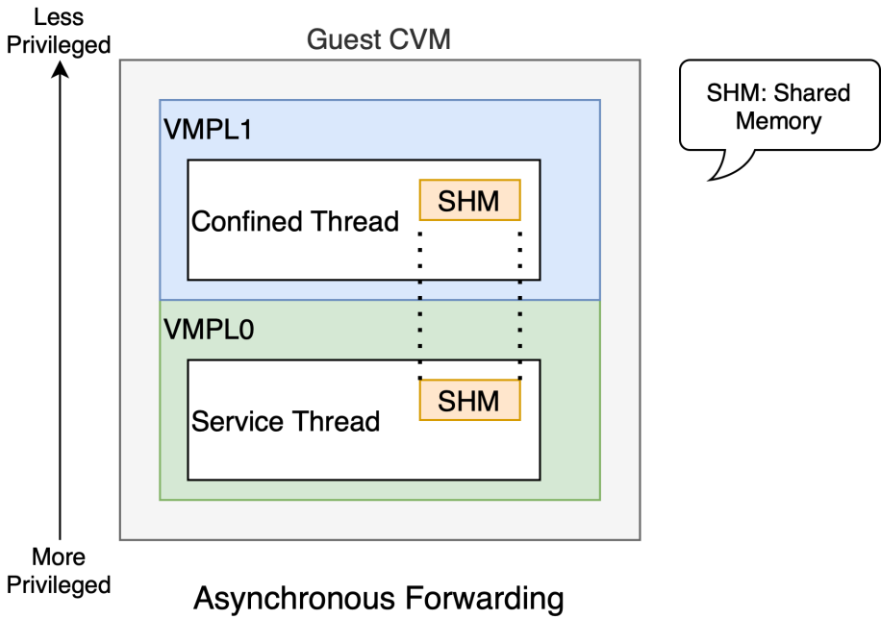

Bi-directional Synchronization

# Syscall and Exception

- Forwarded to guest OS
  - GHCB Protocol (RUN_VMPL0)
- Directly Handled by proxy-kernel

# Performance optimization

- Asynchronous forwarding
  - Cross-thread communication
- Anonymous memory management
  - Handle anonymous memory allocation (i.e., MAP_ANON)



# Case Study

- Execute-only protection
  - VMPL-based XOM (Execute-only Memory)
  - VMPL-enhanced SMEP (Supervisor Mode Execution Prevention)
- Process monitoring
  - Syscalls filtering
  - Process tracing
  - Malware analysis

# Implementation

# Implementation

- Line of Codes
  - kernel module (6700 LoCs)
  - proxy kernel (11000 LoCs)
  - musl-libc (500 LoCs)
- Application Programming Interface (API)
  - vmpl_init
  - vmpl_enter_user
- Preload Library

# Syscall and exception handling

- Asynchronous forwarding
  - Adapted from SGX-HotCalls [ISCA'17]
- vDSO (virtual Dynamic Shared Object)
  - Support syscalls like *clock_gettime*, *getcpu*
- Transparent debugging
  - Support debugging the confined process

# Dynamic VMPL management

- Syscalls interposition
  - Virtual memory related syscalls (mmap, mremap, …)
- Page fault interception
  - Non-present pages
  - Lazy-allocated pages
  - Pre-fault pages

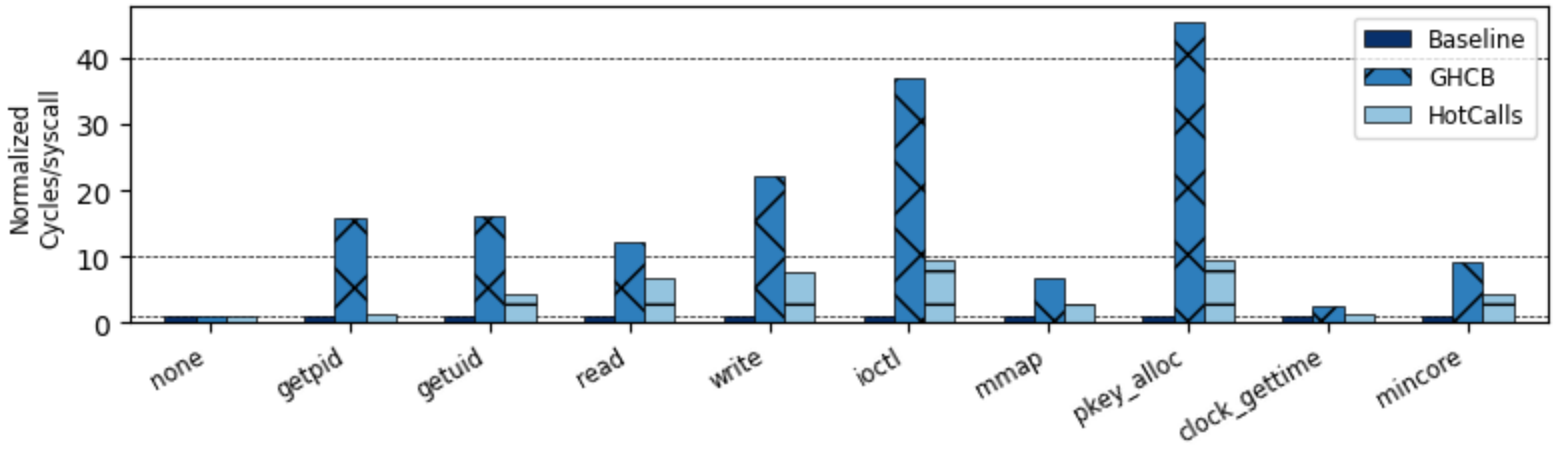
# Performance Evaluation

# Environment Setup

- Hardware
  - Dual-socket 3rd Gen AMD EPYC 7543 (Milan) with 128 logical cores and 64GB RAM
- Host System
  - QEMU 6.1.50 on Ubuntu 22.04 (Kernel 6.5.0-rc2-snp-host)
- Virtual Machine (VM)
  - Ubuntu 22.04 (Kernel 6.5.0-snp-guest) with 64 vCPUs, 16GB RAM

# Evaluation

- The evaluation of syscall overhead in different scenarios
  - Baseline (original user-space)
  - GHCB protocol-based forwarding
  - HotCalls based forwarding



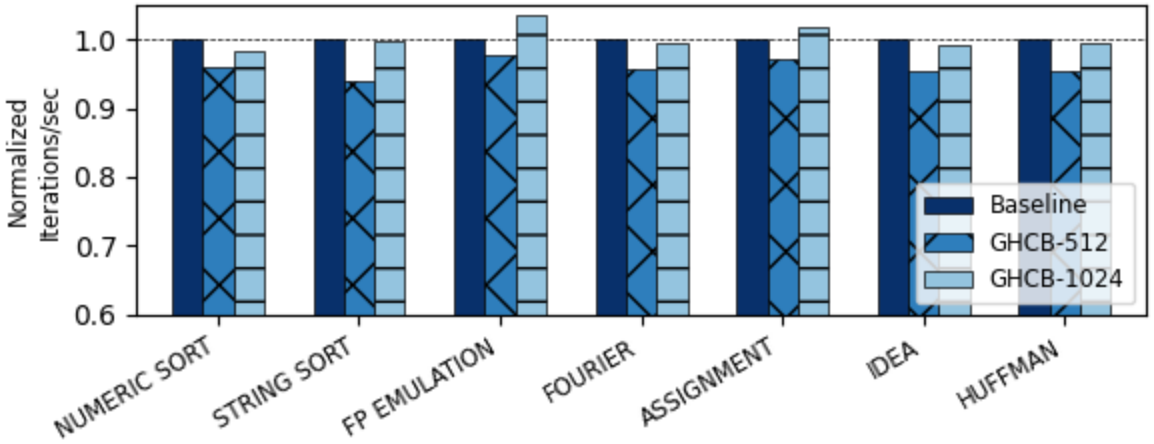
# Evaluation

- Delay in handling page fault in different scenarios
  - Baseline (original user-space)
  - VMPL-CPL0 (kernel-space of the lower VMPL)
  - VMPL-CPL3 (user-space of the lower VMPL)

| | Baseline | VMPL-CPL0 | VMPL-CPL3 |
|---|---|---|---|
| page fault | 13026 | 29627 | 29936 |

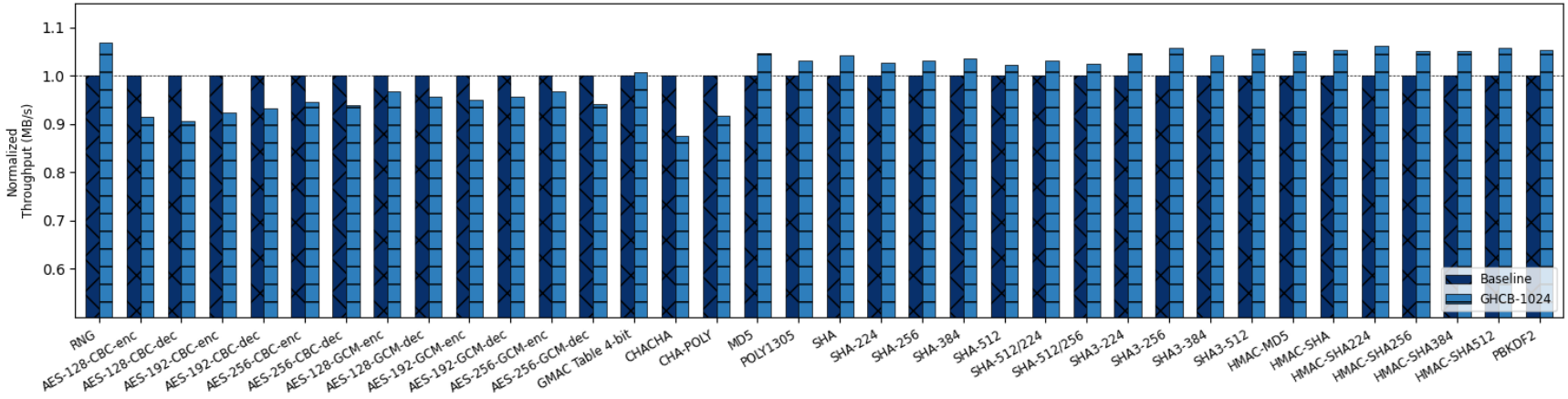
# Evaluation

- The performance evaluation on Nbench
  - Baseline (without memory management)
  - GHCB-512 (manage 512 pages at once)
  - GHCB-1024 (manage 1024 pages at once)



# Evaluation

- The performance evaluation on WolfSSL benchmark
  - Baseline (without memory management)
  - GHCB-1024 (manage 1024 pages at once)



# Summary of evaluation results

- Performance impact on benchmarks
  - 5%~10% overhead
- Efficiency of the Cabin framework

# Discussion

# Discussion

- Advantages
  - Defense-in-depth
  - Compatibility
- Limitations
  - Multi-thread support
    - Lacks support for fork, clone syscalls
  - Overhead from VMPL switching

# Extending to Other CVM Platforms

- Intel TDX (TD-Partitioning)
  - Most privileged L1-VM
  - Less privileged L2-VM
- ARM CCA (ARM Planes)
  - Most privileged Plane 0
  - Less privileged other Planes

# Conclusion

# Conclusion

- Recap of the Cabin framework
  - An isolated execute framework in CVM
  - Asynchronous forwarding mechanisms
  - Anonymous memory management
  - VMPL-based execute-only protection
- Benefits and potential applications
- Future work and improvements

# Q&A

中国科学院信息工程研究所
INSTITUTE OF INFORMATION ENGINEERING,CAS

E-mail: meibenshan@iie.ac.cn