Secure IVSHMEM: End-to-End Shared-Memory Protocol with Hypervisor-CA Handshake and In-Kernel Access Control

Hyunwoo Kim, Jaeseong Lee, Sunpyo Hong, Changmin Han

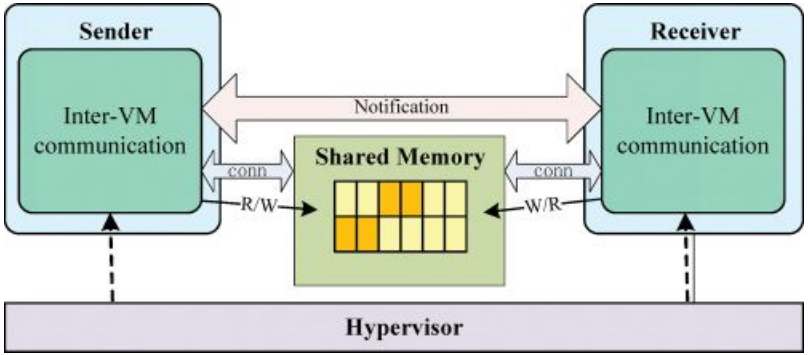
Intel, Seoul, South Korea

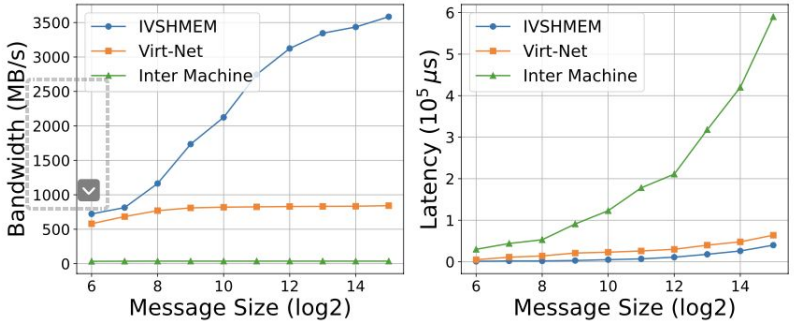
MemSys '25 - International Symposium on Memory Systems

October 7–8, 2025, Washington, VA, USA

Problem & Motivation

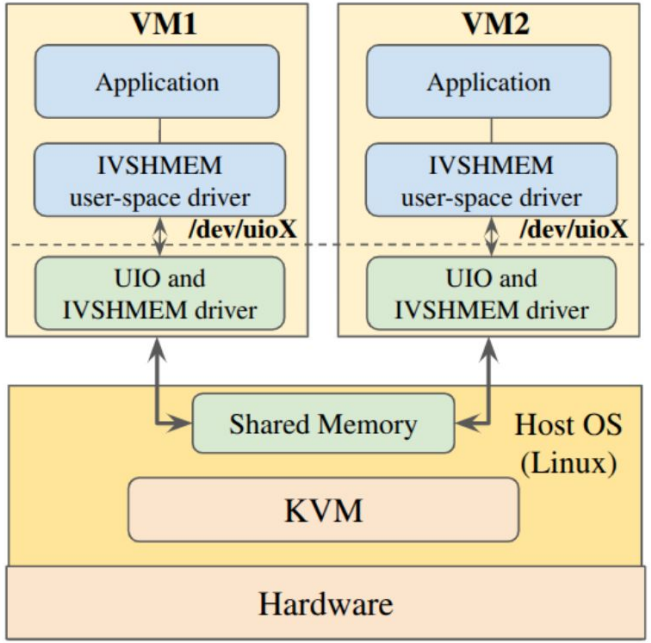
- Modern automotive and embedded systems use virtualization for flexibility and isolation.
- IVSHMEM enables high-speed zero-copy communication between VMs, but offers no security controls.
- Security is critical: Eavesdropping or tampering in mixed-criticality environments (e.g., RTOS with Android VMs) can have severe consequences.





Background: IVSHMEM and Security Concerns

- IVSHMEM: Inter-VM Shared Memory protocol, implemented as a PCI device, mapping regions into each VM.
- Delivers superior performance compared to TCP/virtio for inter-VM comms.
- Lacks any inherent access control: All VMs with IVSHMEM device can access all regions.



Threat Model

Assets:

- Shared-memory contents (e.g., control commands, sensor data)
- VM Identities (certificates, keys)

Adversary:

- Malicious or compromised VM/process
- Can attempt to map IVSHMEM and access others' data

Security Goals:

- Confidentiality, Integrity, Mutual Authentication

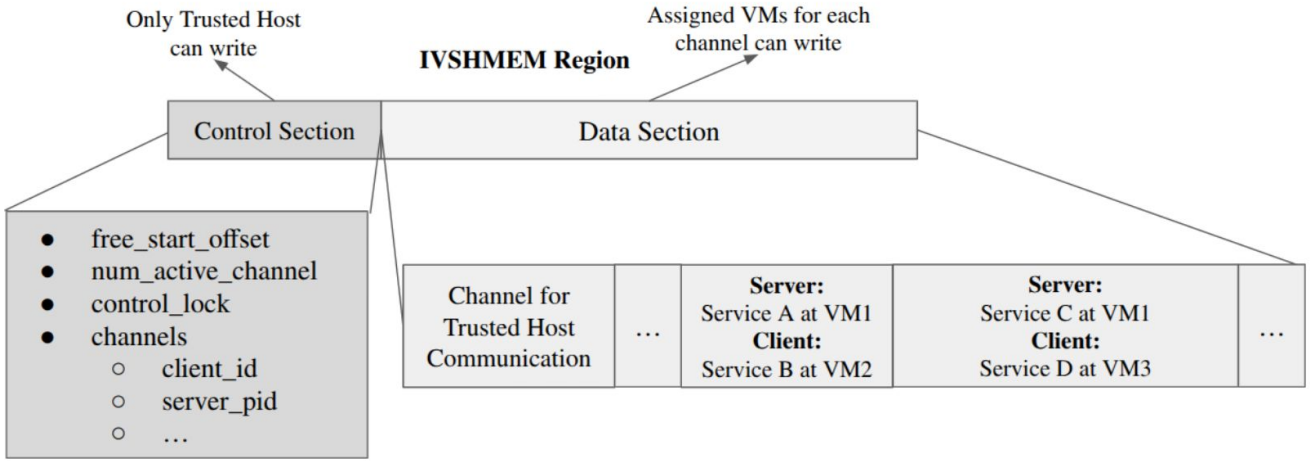
Solution Overview

Secure IVSHMEM combines 3 techniques:

1. Channel separation – Per service/channel allocation in the shared memory region
2. Kernel module enforcement – Prevent unauthorized access at the OS level through system call hooks
3. Hypervisor-mediated handshake – Mutual authentication with hypervisor as certificate authority

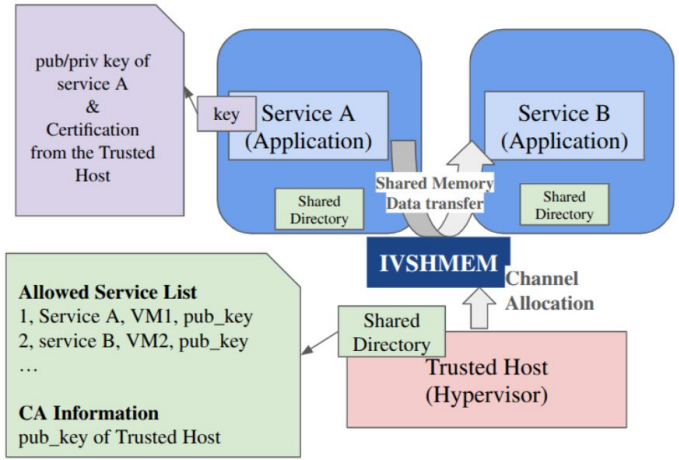
Design Details: Channel Separation & Access Control

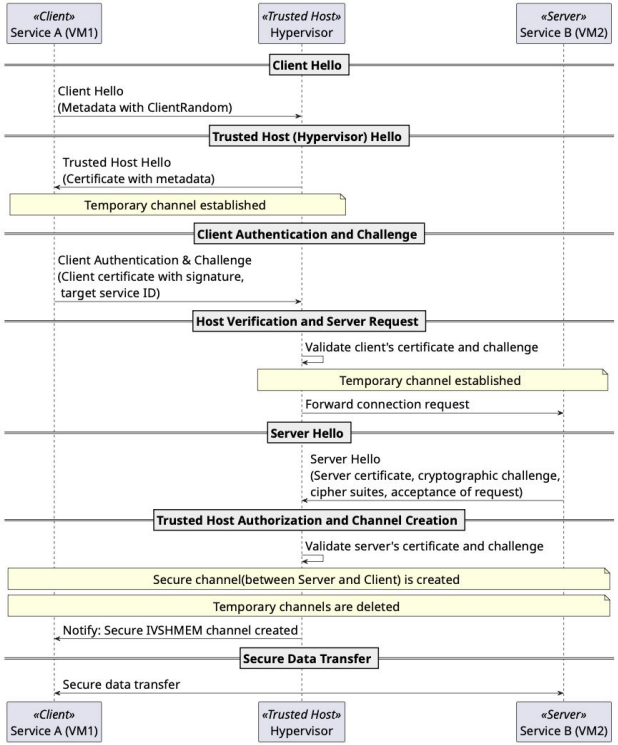
- Channel separation: Each service pair gets a distinct buffer in shared memory; host controls allocation.
- Strict kernel enforcement: System call hooks block unauthorized access before handshake completes.



Design Details: Hypervisor-mediated Handshake

- Handshake: Mutual authentication, certificates/no-replay, host allocates secure channel only on success.
- Hypervisor act as trusted CA, Certificates and private keys are provisioned to each VM by the hypervisor.



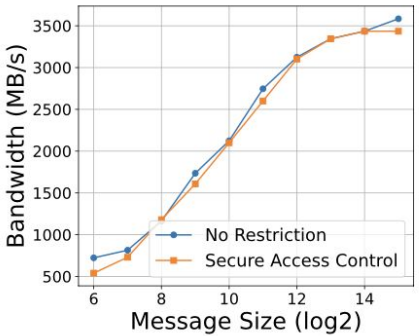


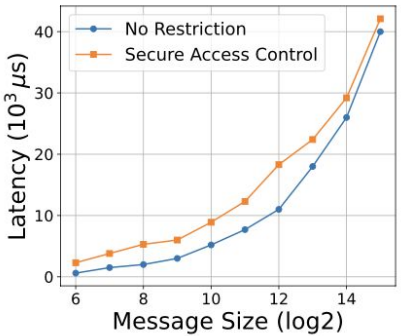
Implementation

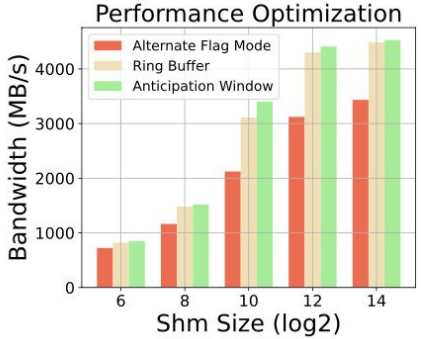
- **Kernel module:** Dynamically hooks IVSHMEM driver syscalls (open, mmap, etc.) to enforce per-channel/ID access.
- **User-space:** OpenSSL-based handshake; hosts provision certificates and exchange signed handshake messages.
- **Library:** BSD socket–like C API for easy adoption (ivshmem_connect, ivshmem_send, …).
- **Channel mapping:** Zero-copy ring buffers with doorbell interrupts for performance optimization.

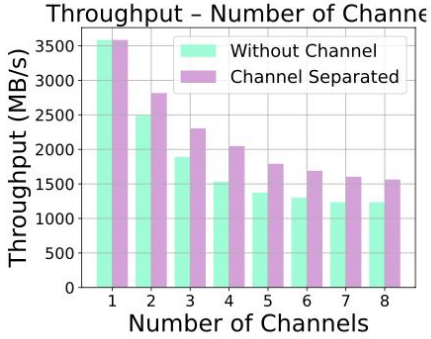
Performance Results

- Handshake latency: <100μs (one-time)
- Runtime overhead: <5% latency, negligible bandwidth loss compared to vanilla IVSHMEM
- Security validation: 100% blocking of unauthorized access scenarios









Conclusion & Future Work

- Secure IVSHMEM provides high-speed, secure shared memory for virtualized systems.
- Achieves mutual authentication and fine-grained access control with minimal overhead.
- Well suited for safety/life-critical domains (automotive, embedded, etc.)

Future Work

- Full transparency for legacy applications
- Key management enhancements