



Tamper Evident Processor Architecture

Binoy Ketan Dash, Biswajit Jena



Outline

- Introduction
- Problem Statement
- Related Work
- AEGIS Framework
- Simulation
- Conclusion



Physical Security

- Usually security protects computer owner from attack. Sometimes, the computer owner is the potential "enemy".
 - Distributed computation (SETI@home)
 - Peer to peer network
 - Digital Rights Management
- Some plausible attacks:
 - Tampering with binaries/Using debugging tools
 - Bus probing/dual ported RAM
- Objective is to ensure correct execution.



Problem Statement

- To provide a private and tamper-proof execution environment for applications
- In particular in the case of physical attacks on the components located around the processor
- Confidentiality and Integrity of the software Execution



Solution to the problem

- Process Isolation
- Confidentiality and integrity of software
- Confidentiality and integrity of data storage
- Confidentiality and integrity of data during interrupt handling



Tamper-Evident Processors

- The above mentioned solution can be implemented to prevent the leakage of sensitive data under software and physical attacks by:
 - Secure processor
 - Co-processors



Possible Solutions

- Palladium (Microsoft Next-Generation Secure Computing Base)
 - Address only Software Attacks
- XOM(eXecute Only Memory)
 - Vulnerable to Replay attack
- AEGIS
 - Addresses both Software and Hardware attacks



Palladium

- Palladium: Microsoft
 - Stated goal: Protect from software attacks.
 - Combination of hardware and software mechanisms.
 - Adds "curtained" memory to avoid DMA attacks.
 - Uses a security kernel (Nexus).
 - Memory integrity is assumed (only software attacks).



XOM

- XOM (eXecute Only Memory): David Lie et al
 - Stated goal: Protect integrity and privacy of code and data.
 - Operating System is completely untrusted.
 - Memory integrity verification does not prevent replay attacks.
 - Memory encryption only provides partial privacy.
 - Privacy is expensive but not necessary for all applications.



AEGIS Framework



AEGIS Framework

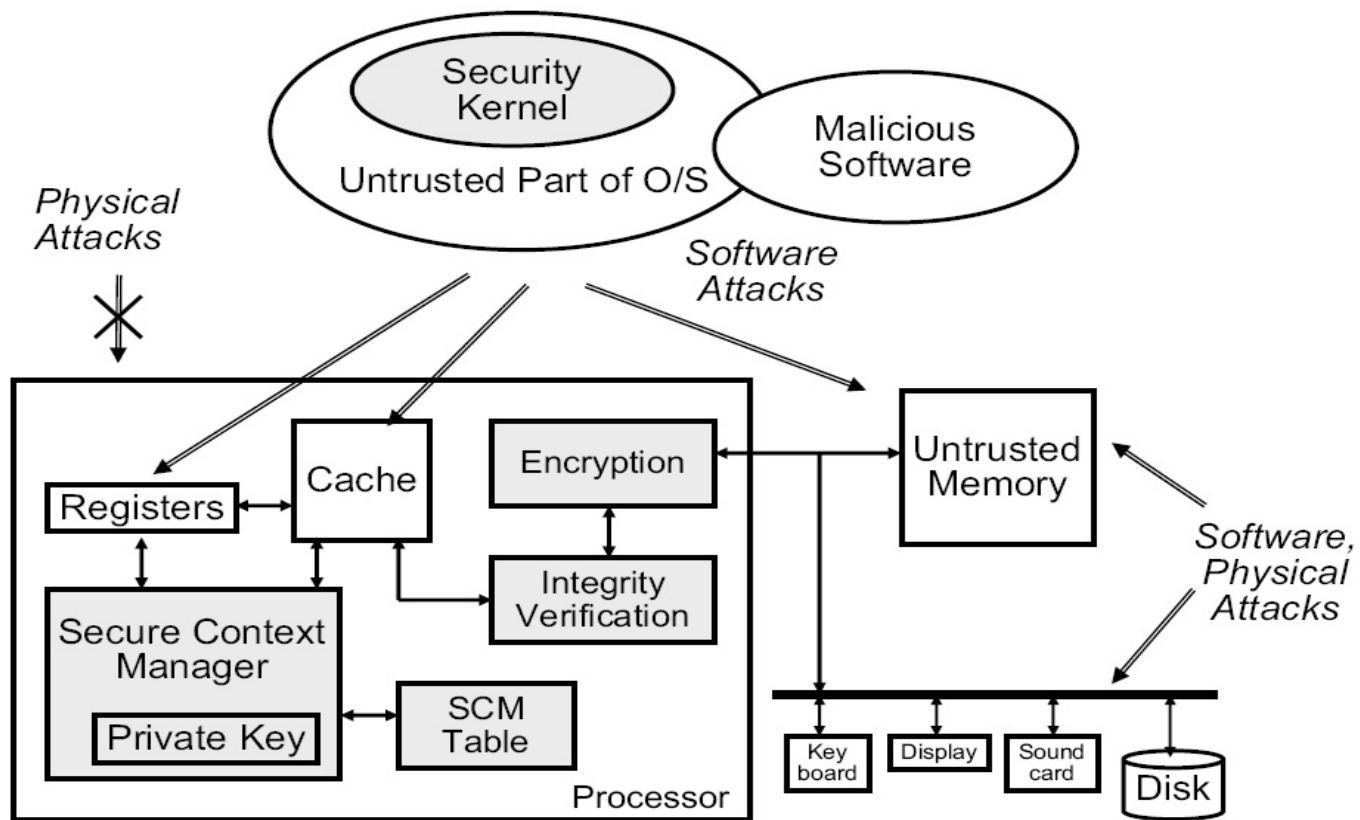
- AEGIS architecture provides:
 - Tamper-evident environment
 - Detects the physical or software tampering
 - Private and authenticated tamper-resistant (PTR) execution environment
 - Supports private and authenticated execution
 - Ensures the privacy of registers, on-chip caches and off-chip memory
- AEGIS framework protects against:
 - Software Attacks
 - Hardware Tampering



What AEGIS provides?

- Secure Computing Model
- System Authentication
- Program Authentication
- Message Authentication

AEGIS Computing Model



AEGIS: Tamper Evident Architecture



- It guarantees the integrity of a program execution.
- A valid execution can be secured by:
 - Initial State
 - State on a context Switch
 - On Chip-Off Chip Memory



Secure Context manager

- A specialized hardware that ensures protection for each process.
 - Computes hash of program and data
 - Assigns a Secure Process ID for on-chip memory access.
 - Performs memory integrity verification for off-chip memory access.



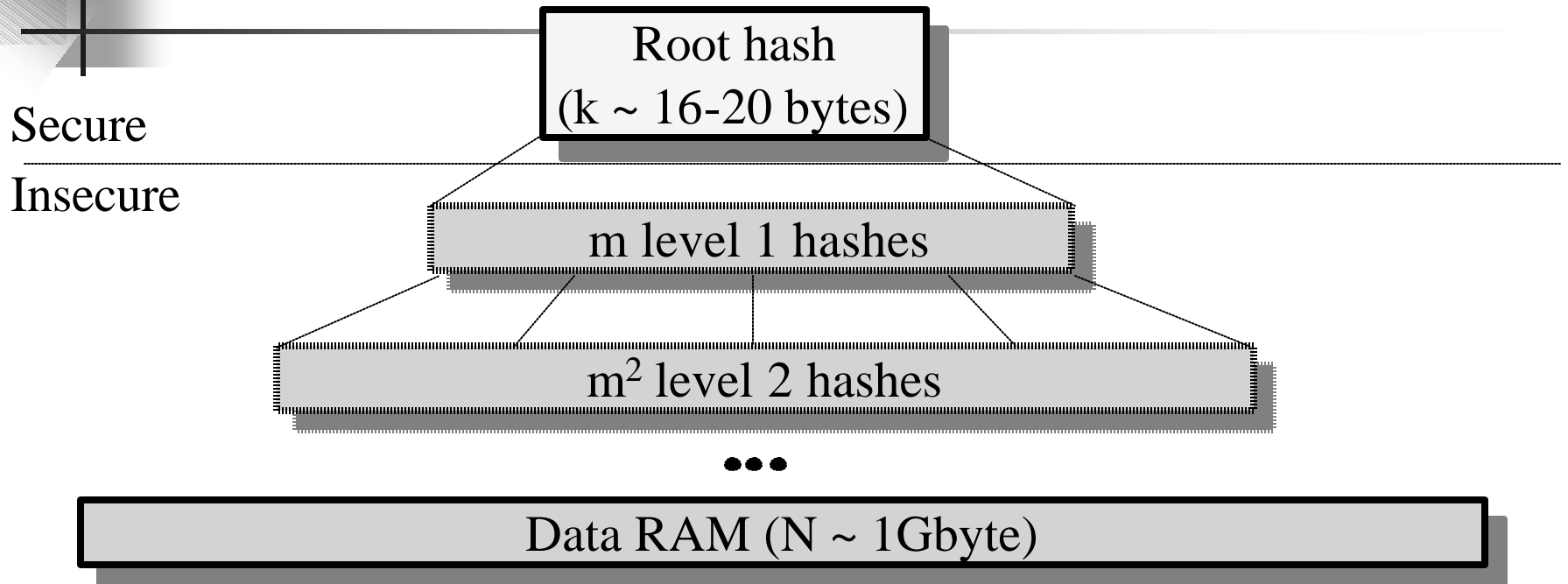
Memory Protection Schemes



Memory protection

- Mechanisms used to protect off-chip memory are:
 - Integrity verification
 - Protects the integrity of off-chip data
 - Encryption
 - Protects the privacy of data

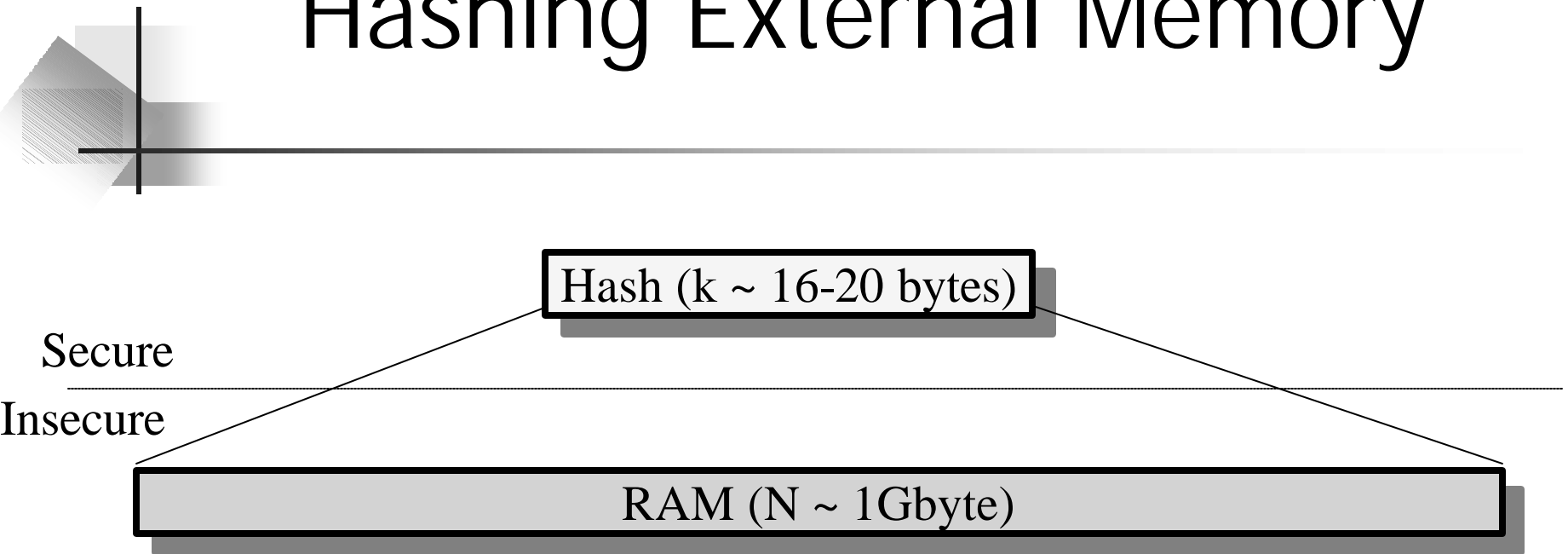
Checking external memory: Hash Tree (or Merkle Tree)



By generalizing we can build a tree of hashes, which gives a better compromise.

- **Secure Storage: k**
- **RAM overhead: $1/(m-1)$**
- **Read/Write cost: $\log_m(N)$**

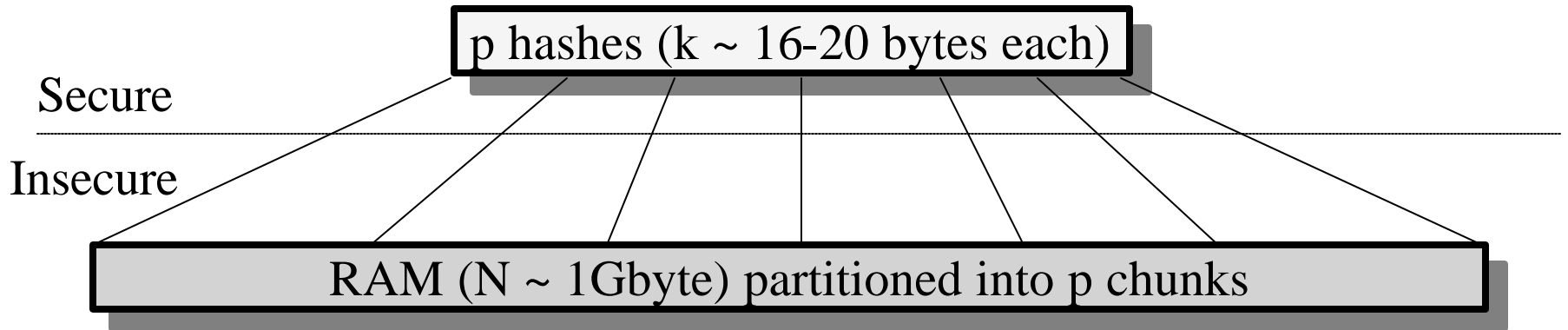
Checking external memory: Hashing External Memory



Single hash can be used to protect the whole memory.

- **Secure Storage: k**
- **RAM overhead: none**
- **Read/Write cost: N**

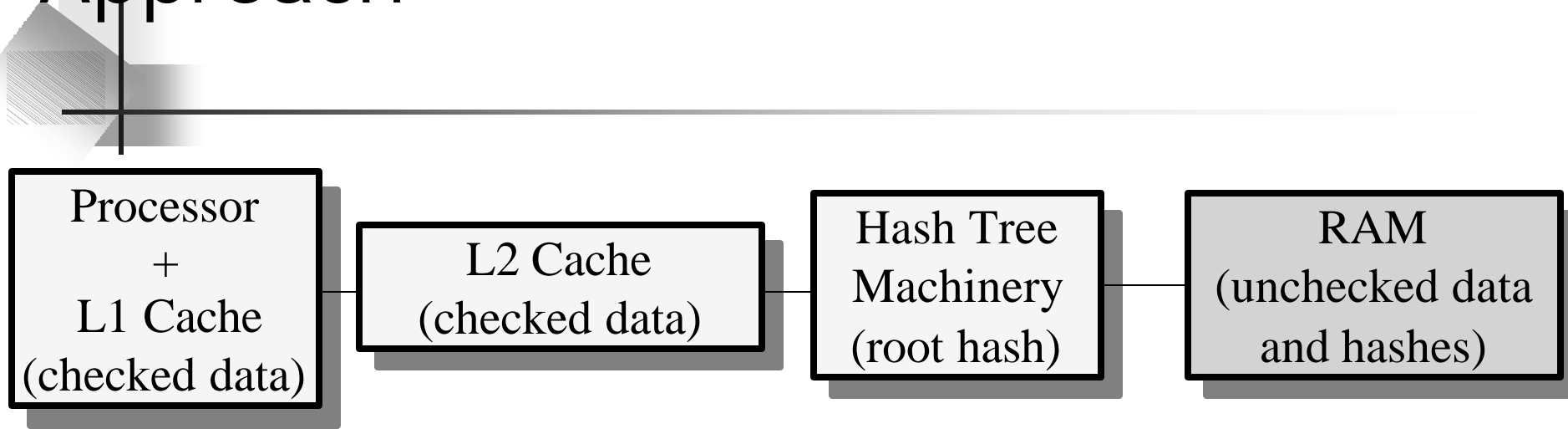
Checking external memory: Multiple Hashes



Read/write cost can be reduced by splitting the memory into chunks that have different hashes.

- **Secure Storage: pk**
- **RAM overhead: none**
- **Read/Write cost: N/p**

Caching and Hash Trees: Naïve Approach



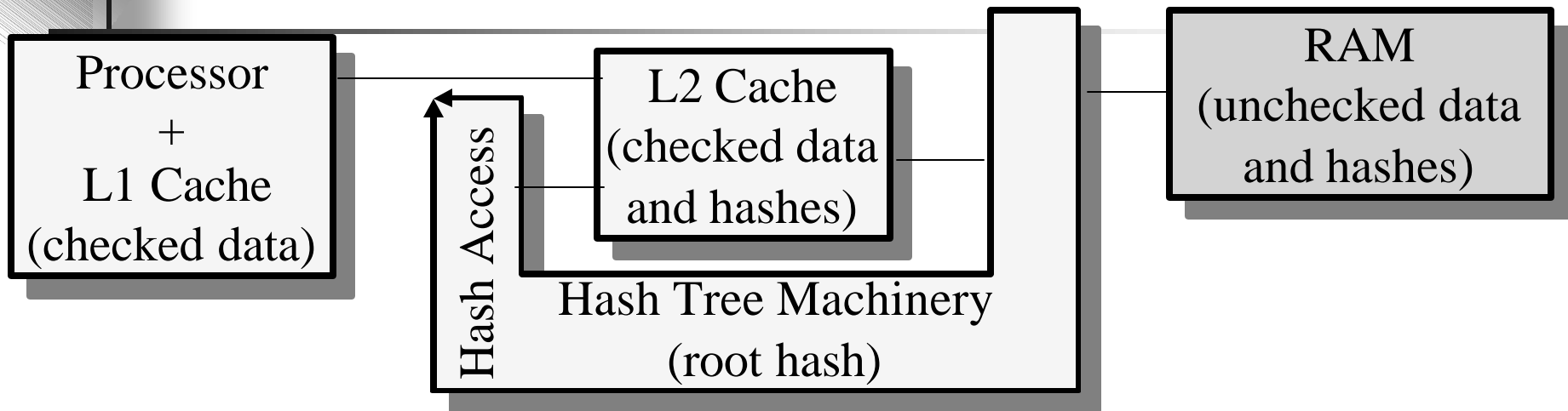
The simplest way of integrating hash trees into the memory hierarchy is to place all the hash tree machinery between two cache levels.

- Trusted side: checked data.
- Untrusted side: unchecked data and hashes.

Main problem:

- Each miss in L2 produces $\log_m(N)$ accesses to RAM.

Caching Hashes: Integrated approach



- Much better performance if the hash tree machinery is integrated with L2 cache:
 - Hash tree machinery intercepts accesses to RAM.
 - Hash tree machinery reads and writes hashes it needs via L2.
- Big performance boost because:
 - Checked hashes are cached in trusted L2, so most accesses to external memory do not have to go to the root of the tree.



Memory Encryption

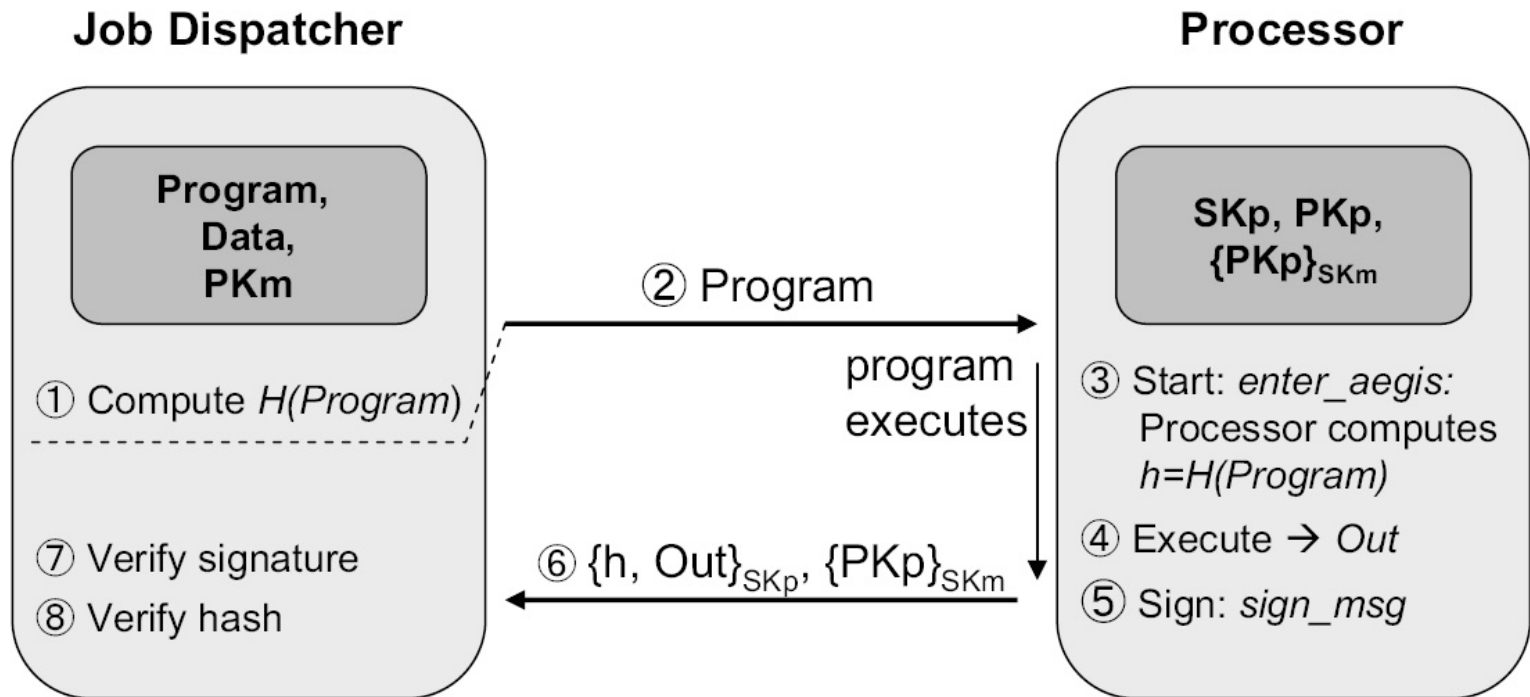
- Symmetric key encryption algorithm is used
- Encryption of off-chip memory is essential for providing privacy to programs.
- The scheme uses modified AES for off-chip encryption.



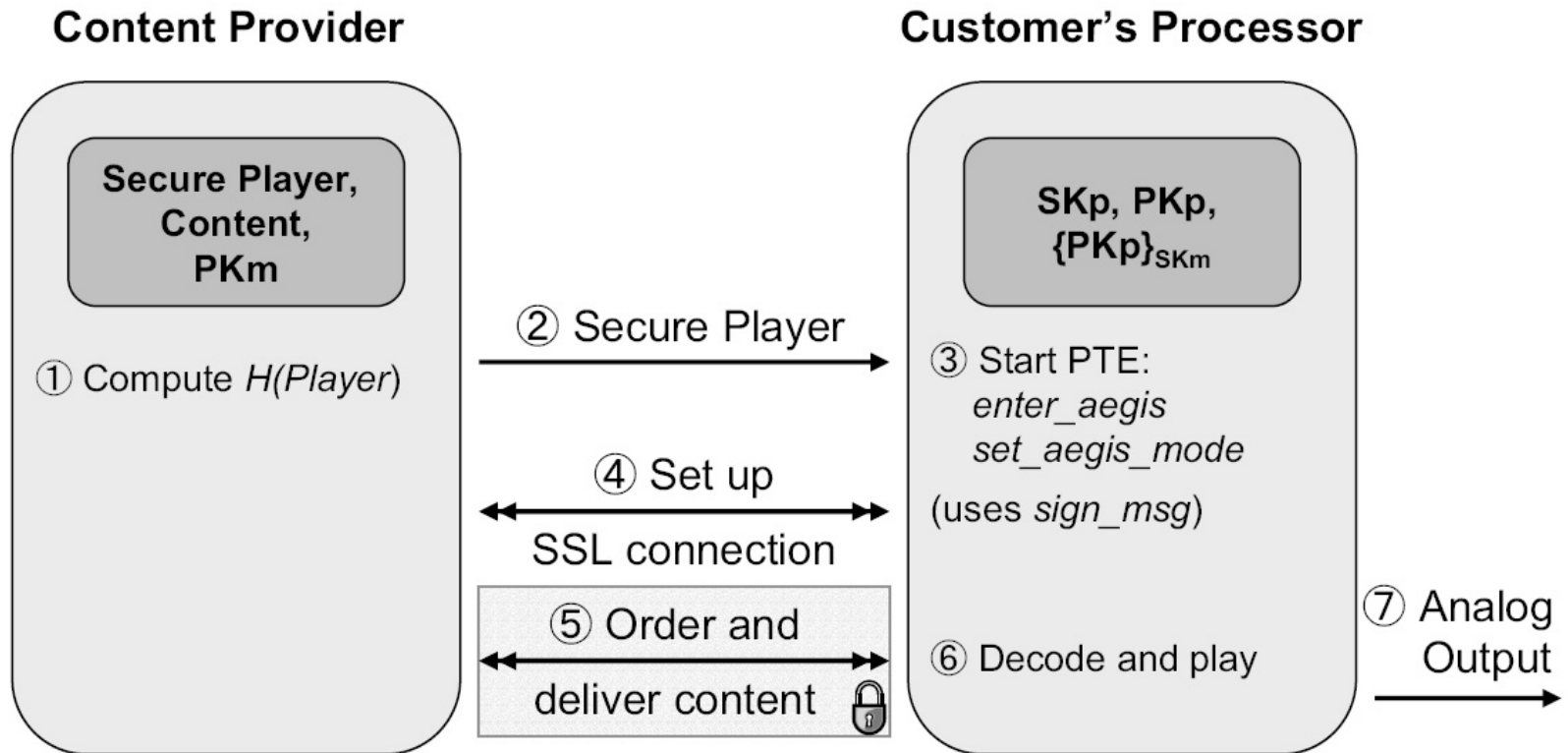
Applications

- Certified Execution
- Digital Rights Management

Certified Execution



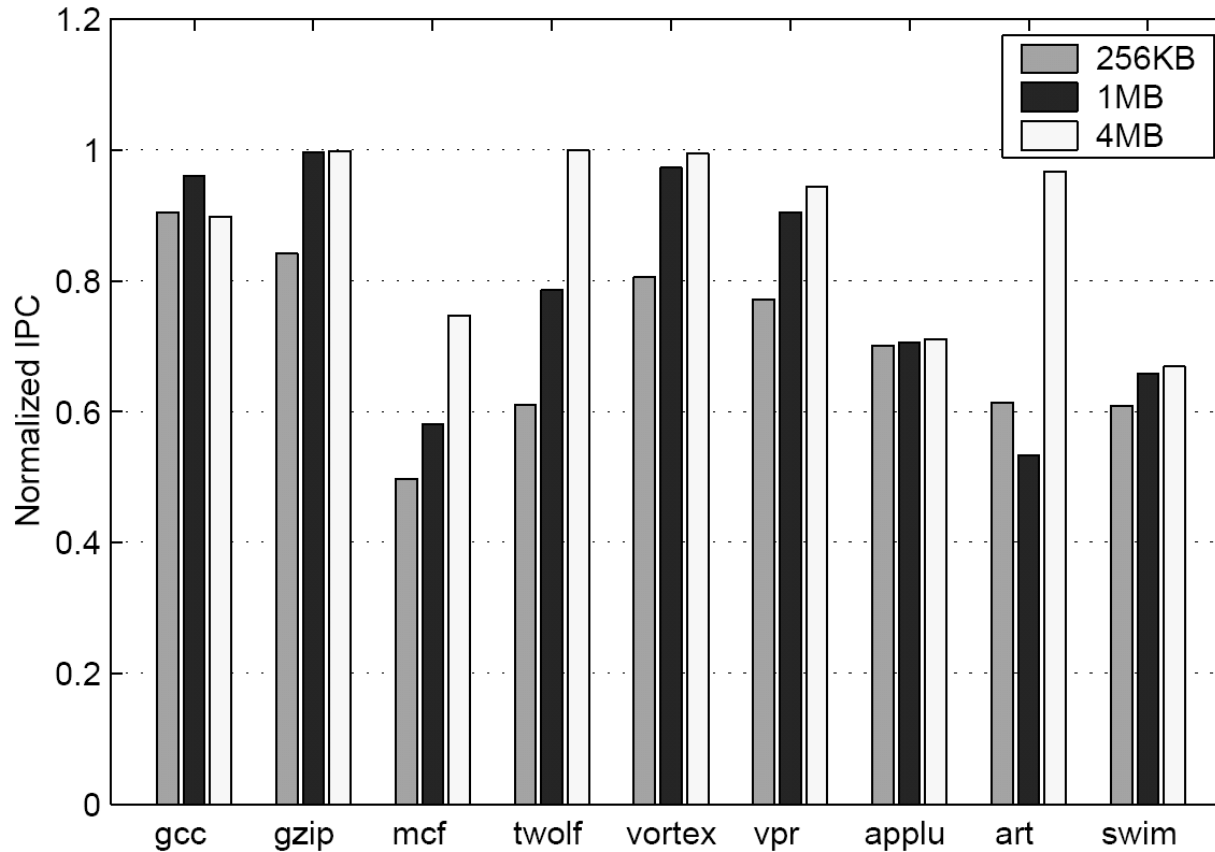
Digital Rights Management





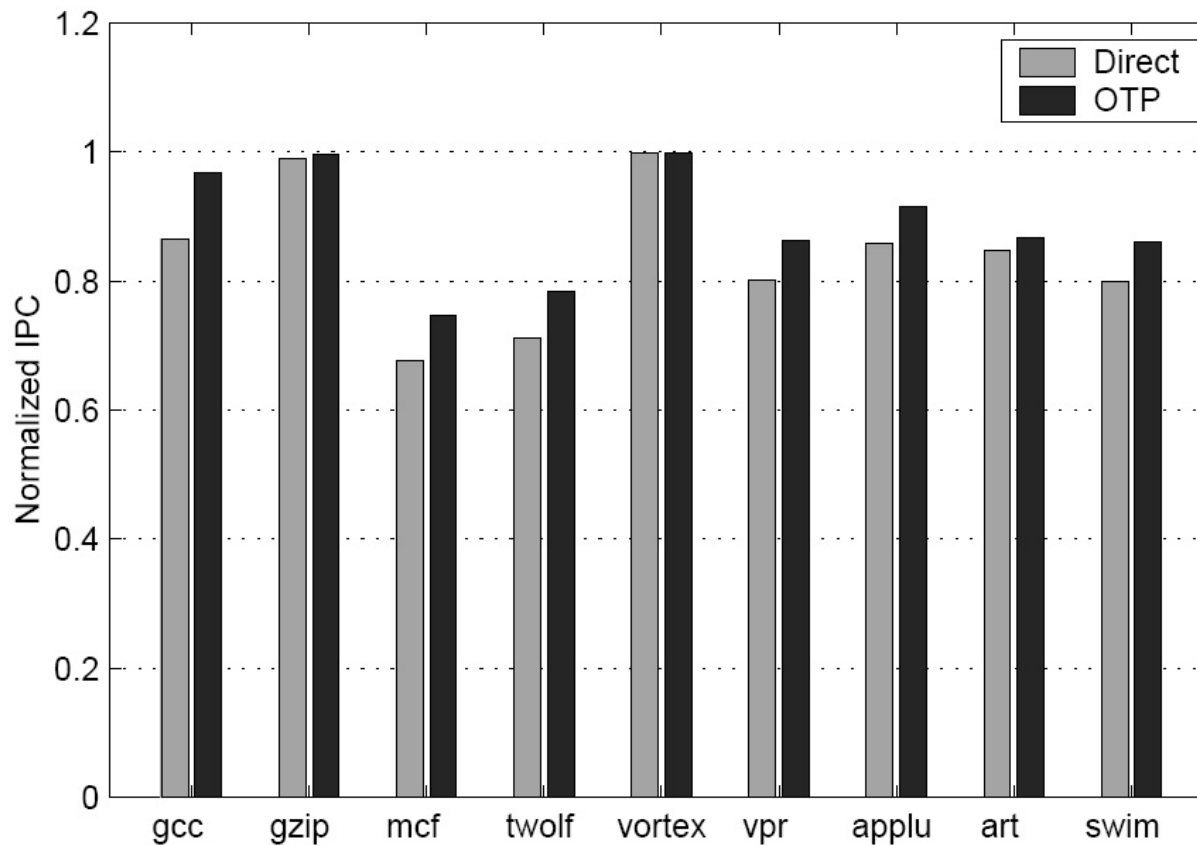
Simulation Results

Performance Overhead



For 64B block size and various L2 cache sizes

Impact of Encryption





Conclusion

- Discusses Secure execution of processes in a single processor
- Secure execution of processes in multiprocessor systems need to be addressed



Acknowledgement

- Thanks to Blaise Gassend for providing his presentation slides for reference.
- Few of slides contents are taken from Blaise Gassend's presentation slides.



Reference

- Suh et al., "AEGIS: Architecture for Tamper-Evident and Tamper-Resistant Processing", ICS, 2003
- Gassend et al., "Caches and Hash Trees for Efficient Memory Integrity Verification", HPCA, 2003.
- Efficient Memory Integrity Verification and Encryption for Secure Processors
- Peter C. S. Kwan, "Secure Computing Architecture: A survey on Recent Proposals and Industry Directions".
- Lie et al., "Architectural support for copy and tamper resistant software", ASPLOS-IX, 2000.
- Suh et al., "Efficient Memory Integrity Verification and Encryption for Secure Processors", MICRO, 2003