MIT PROJECT OXYGEN
PERVASIVE, HUMAN-CENTERED COMPUTING

AEGIS:
Architecture for Tamper-Evident and Tamper-Resistant Processing

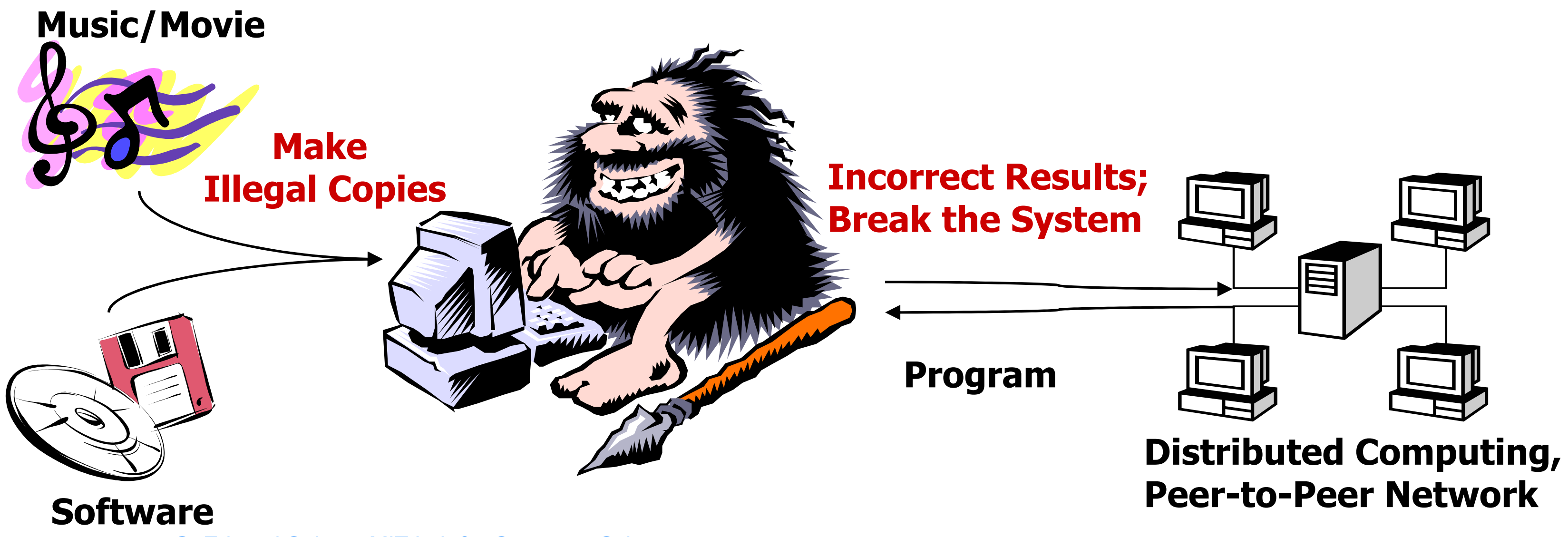
G. Edward Suh, Dwaine Clarke,
Blaise Gassend, Marten van Dijk,
Srinivas Devadas

Massachusetts Institute of Technology

L C S

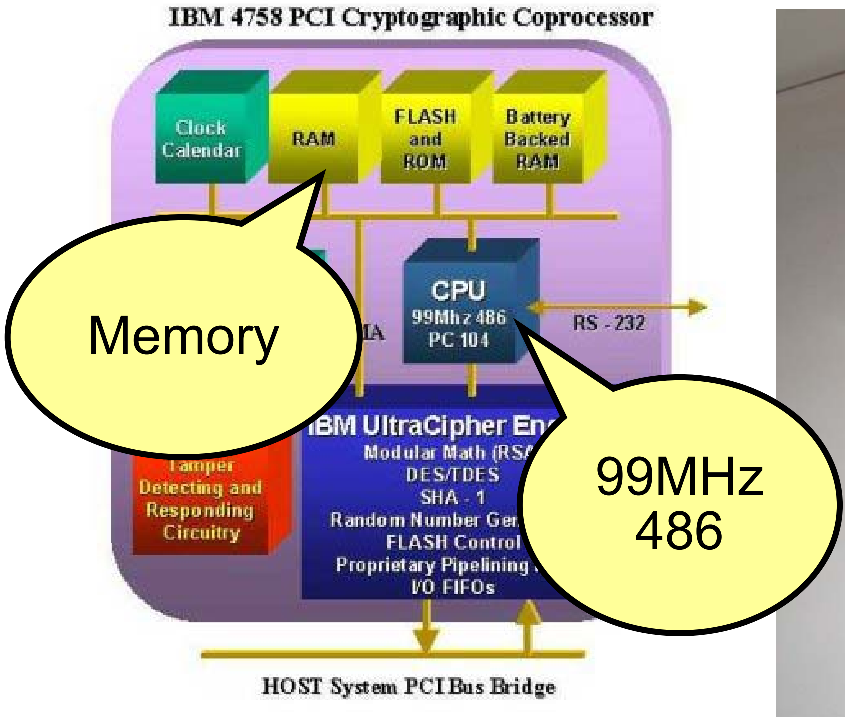
Cases for Physical Security

- Applications on untrusted hosts with untrusted owners
  - Digital Rights Management (DRM), Software licensing
  - Distributed computation on Internet
  - Mobile agents
- New challenges
  - Untrusted OS
  - Physical attacks


Music/Movie
Make Illegal Copies
Software
Incorrect Results; Break the System
Program
Distributed Computing, Peer-to-Peer Network

Conventional Tamper-Proof Packages

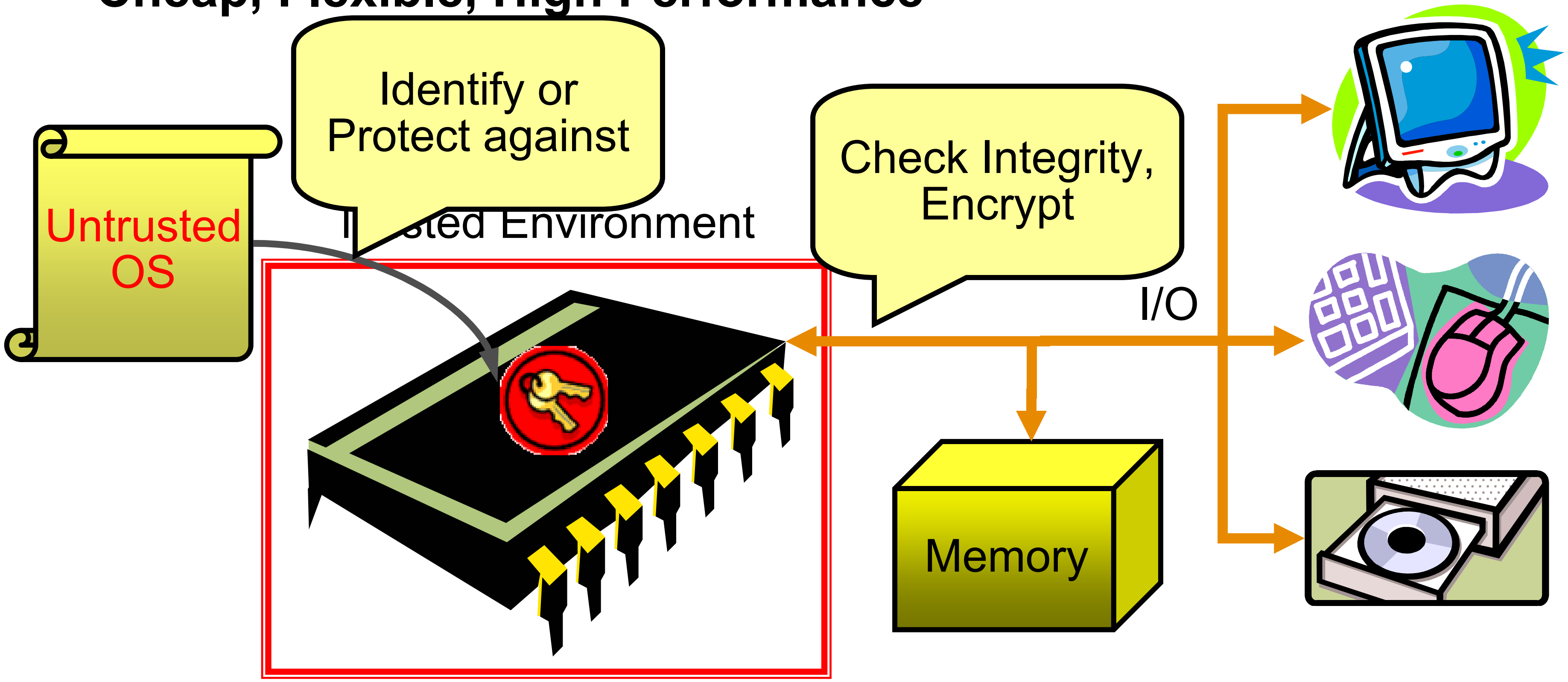
- **Processing system in a tamper-proof package (IBM 4758)**
  - **Expensive: many detecting sensors**
  - **Needs to be continuously powered: battery-backed RAM**



Source: IBM website

Single-Chip Secure Processors

- **Only trust a single chip: tamper-resistant**
  - **Off-chip memory: verify the integrity and encrypt**
  - **Untrusted OS: identify a core part or protect against OS attacks**
- **Cheap, Flexible, High Performance**



Related Research

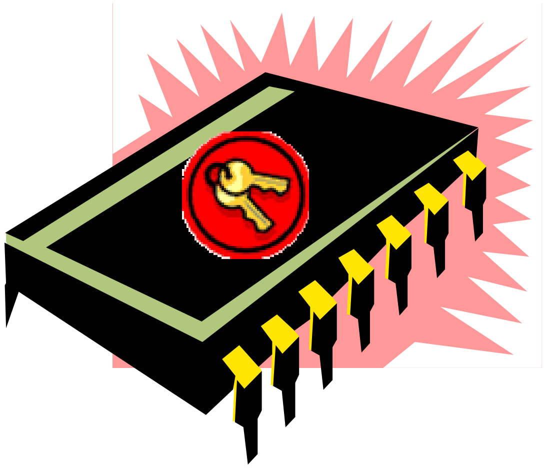
- XOM (eXecution Only Memory): David Lie et al
  - Stated goal: Protect integrity and privacy of code and data
  - Operating system is completed untrusted
  - Memory integrity checking does not prevent replay attacks
  - Privacy is expensive but not necessary for all applications

- Palladium/NGSCB: 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 and privacy are assumed (only software attacks).

MIT PROJECT OXYGEN
PERVASIVE, HUMAN-CENTERED COMPUTING

AEGIS:

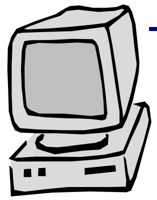
High-Level Architecture



L C S

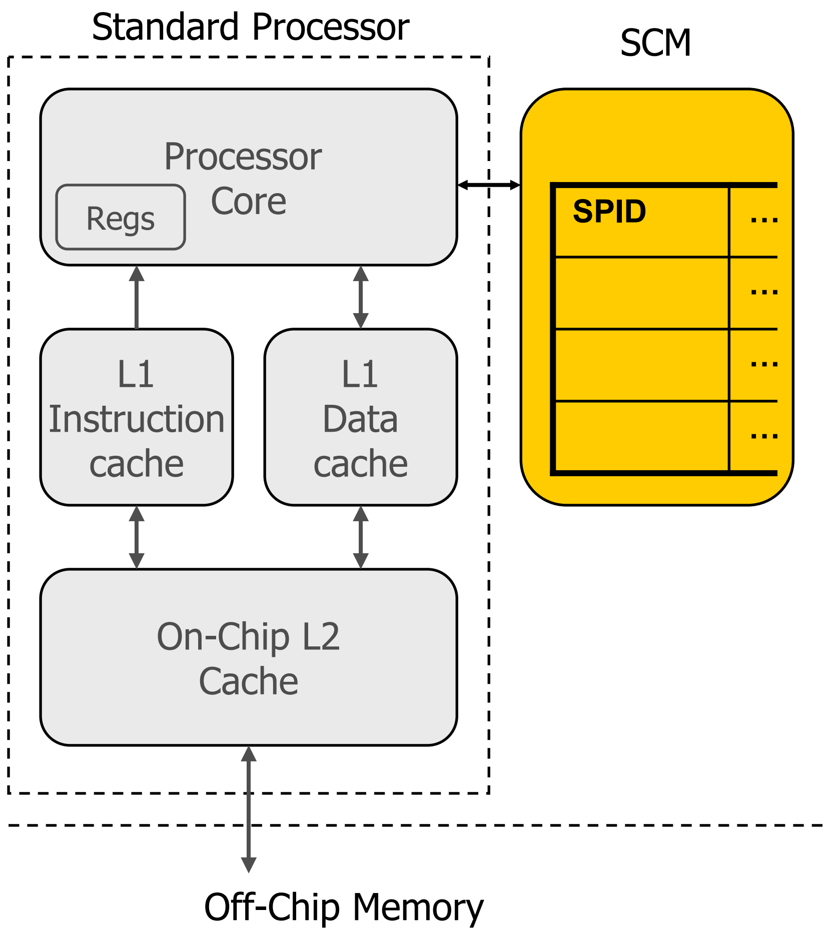
Secure Execution Environments

- Tamper-Evident (TE) environment
  - Guarantees a valid execution and the identity of a program; no privacy
  - Any software or physical tampering to alter the program behavior should be detected
- Private Tamper-Resistant (PTR) environment
  - TE environment + privacy
  - Encrypt instructions and data
  - Assume programs do not leak information via memory access patterns
- Implementation
  - Either have a trusted part of the OS or completely untrust the OS
  - Secure context manager, encryption and integrity verification



Secure Context Manager (SCM)

- **A specialized module in the processor**
- **Assign a secure process ID (SPID) for each secure process**
- **Implements new instructions**
  - ***enter_aegis***
  - ***set_aegis_mode***
  - ***random***
  - ***sign_msg***
- **Maintains a secure table**
  - **Even operating system cannot modify**

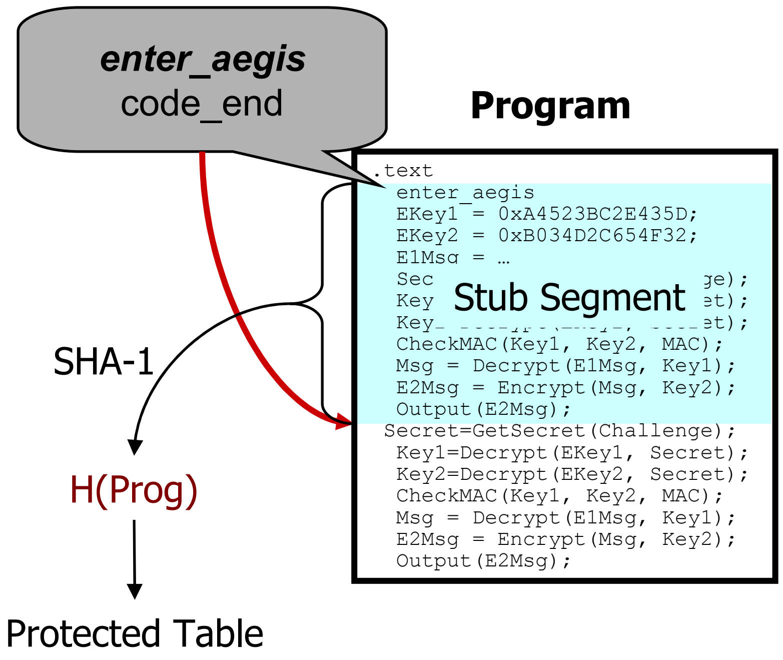


SCM: Program Start-Up

- ‘*enter_aegis*’: TE mode
  - Start protecting the integrity of a program
  - Compute and store the hash of the stub code: H(Prog)

    → Tampering of a program results in a different hash
  - Stub code verifies the rest of the code and data

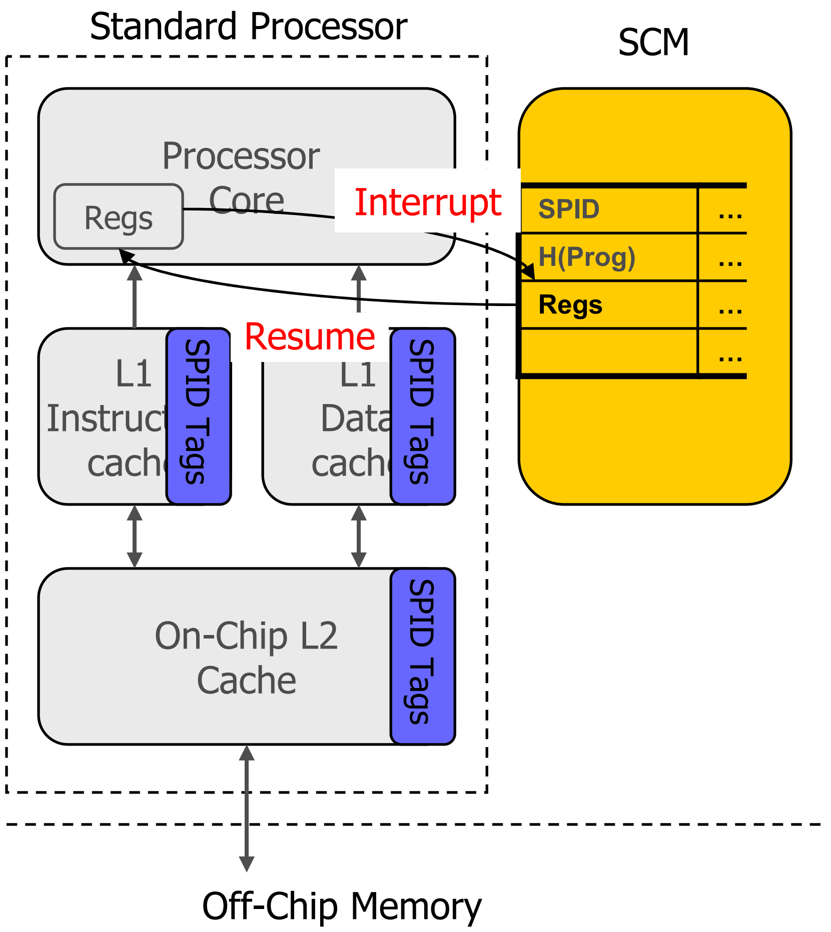
- ‘*set_aegis_mode*’
  - Start PTR mode on top of the TE mode



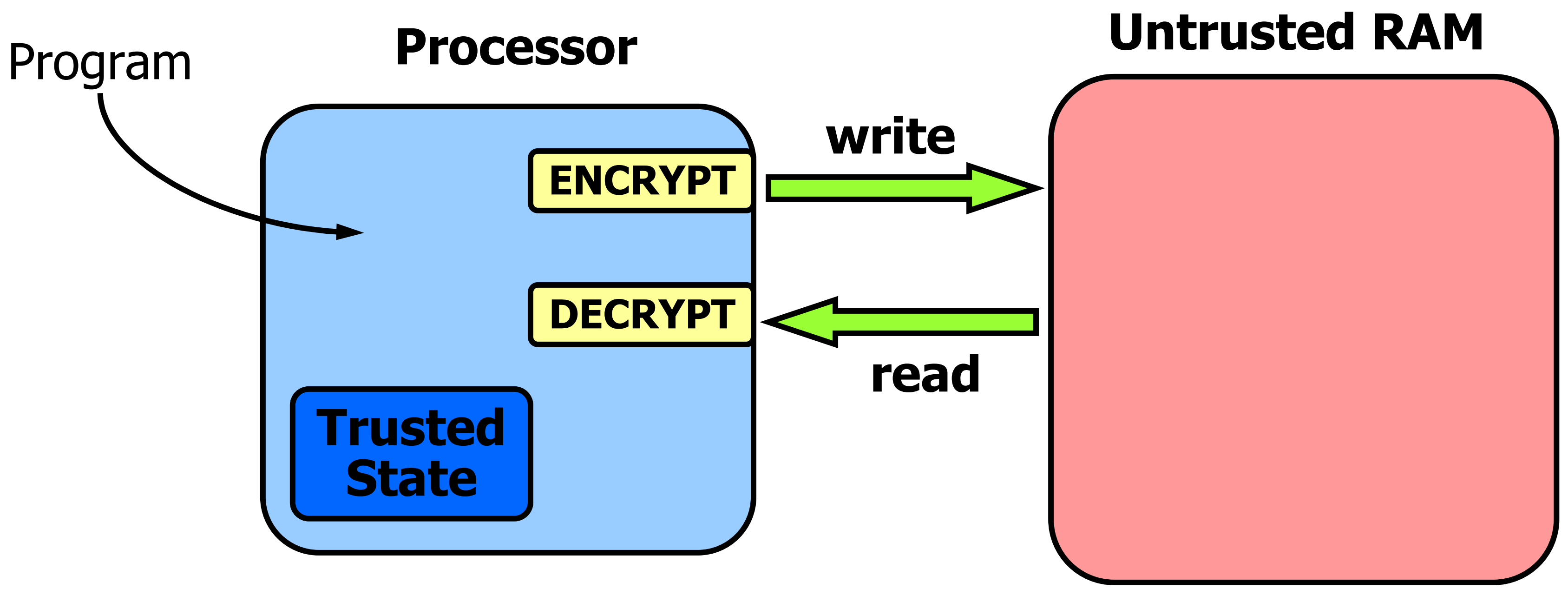
```
.text
  enter_aegis
  EKey1 = 0xA4523BC2E435D;
  EKey2 = 0xB034D2C654F32;
  E1Msg = …
  Sec                         ge);
  Key                         et);
  Key_                        et);
  CheckMAC(Key1, Key2, MAC);
  Msg = Decrypt(E1Msg, Key1);
  E2Msg = Encrypt(Msg, Key2);
  Output(E2Msg);
 Secret=GetSecret(Challenge);
  Key1=Decrypt(EKey1, Secret);
  Key2=Decrypt(EKey2, Secret);
  CheckMAC(Key1, Key2, MAC);
  Msg = Decrypt(E1Msg, Key1);
  E2Msg = Encrypt(Msg, Key2);
  Output(E2Msg);
```

SCM: On-Chip Protection

- **Registers on interrupts**
  - **SCM saves Regs on interrupts, and restore on resume**
- **On-chip caches**
  - **Need to protect against software attacks**
  - **Use SPID tags and virtual memory address**
  - **Allow accesses from the cache only if both SPID and the virtual address match**

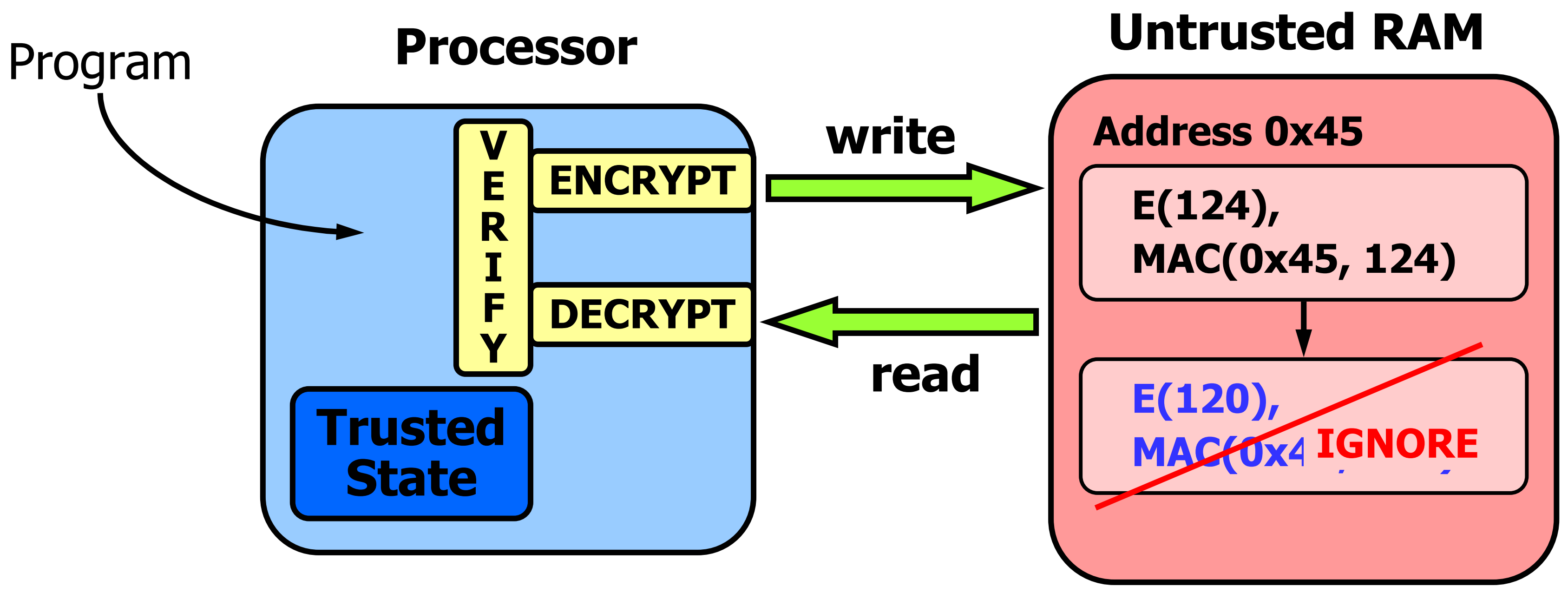


Memory Encryption



- Encrypt on an L2 cache block granularity
  – Use symmetric key algorithms with CBC mode
  – Randomize initial vectors

Integrity Verification

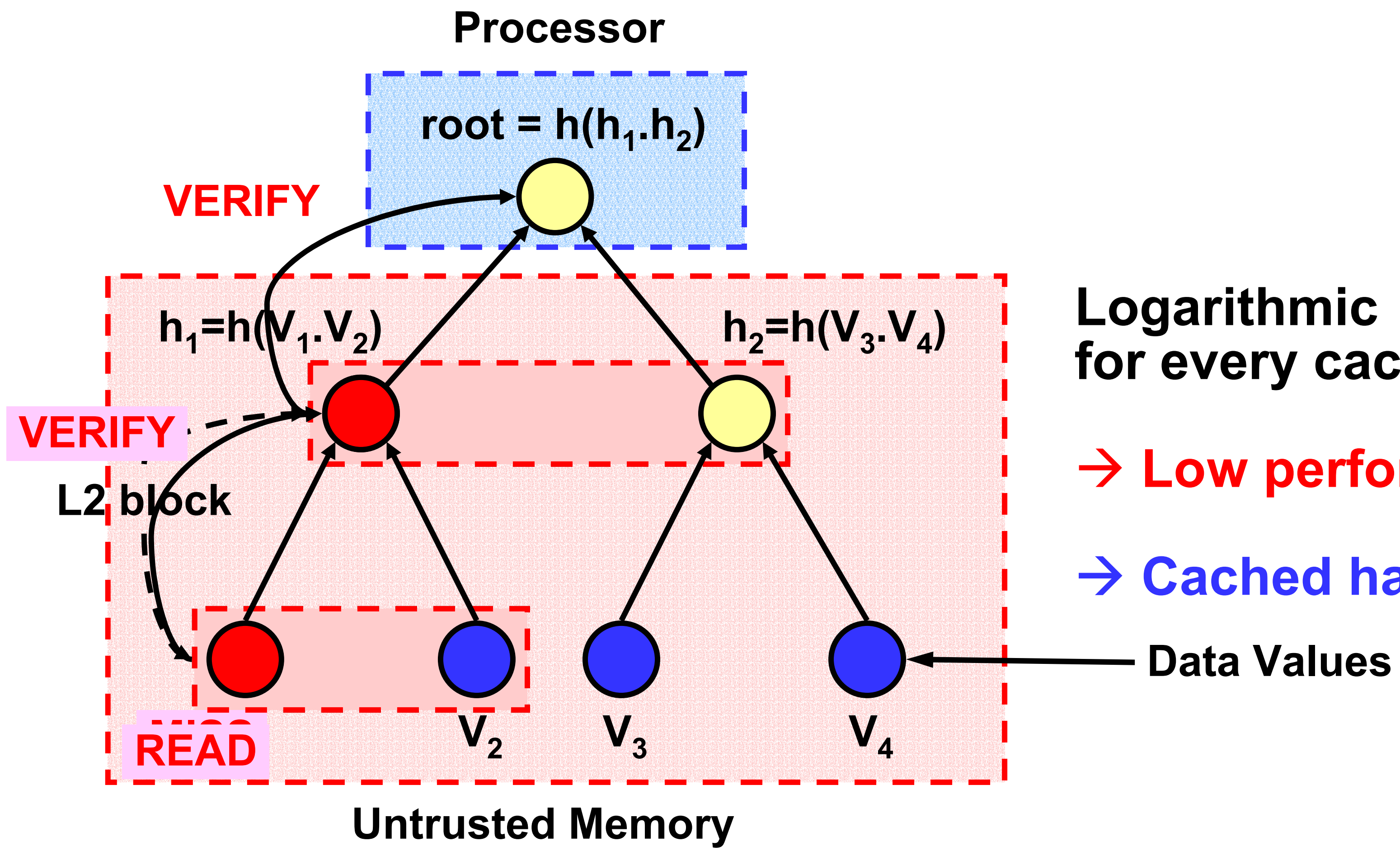


Cannot simply MAC on writes and check the MAC on reads

→ Replay attacks

Hash trees for integrity verification

Hash Trees

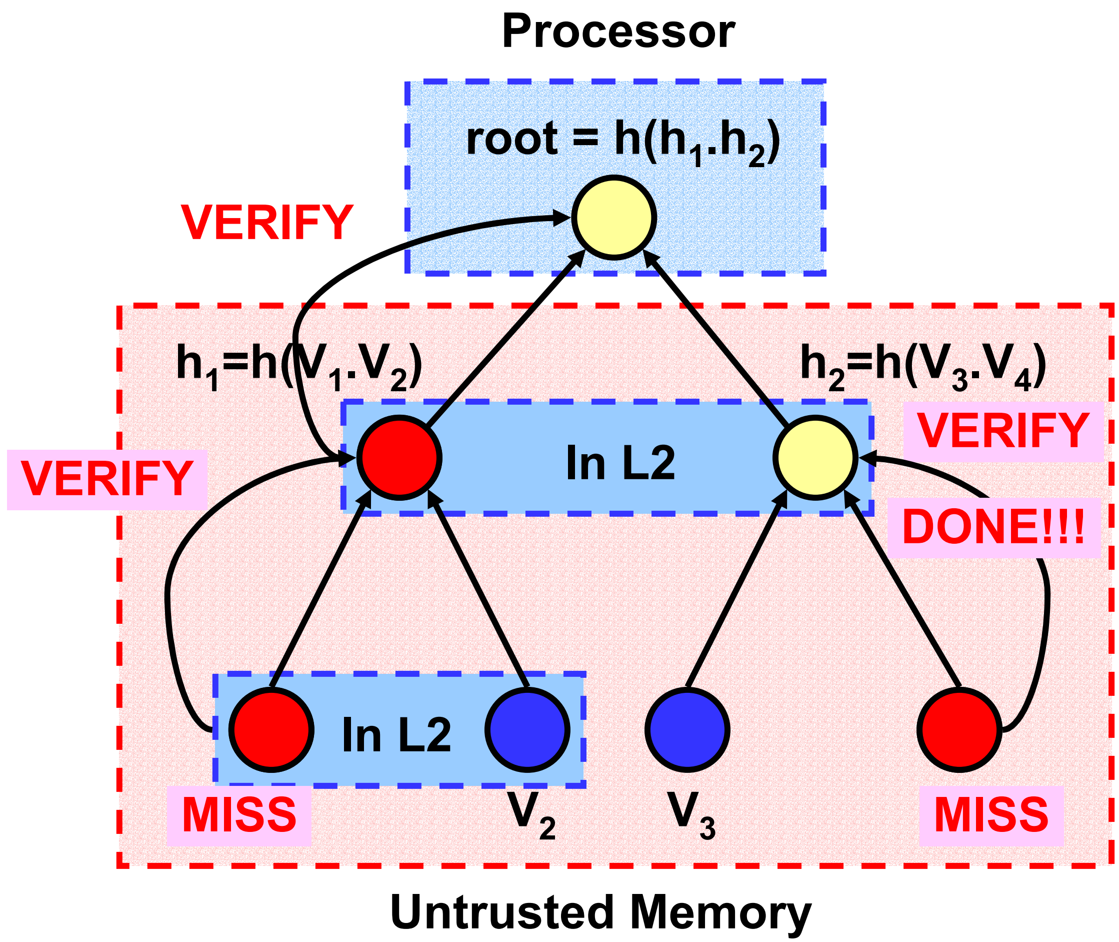


Logarithmic overhead for every cache miss

→ **Low performance**

→ **Cached hash trees**

Cached Hash Trees (HPCA'03)



Cache hashes in L2

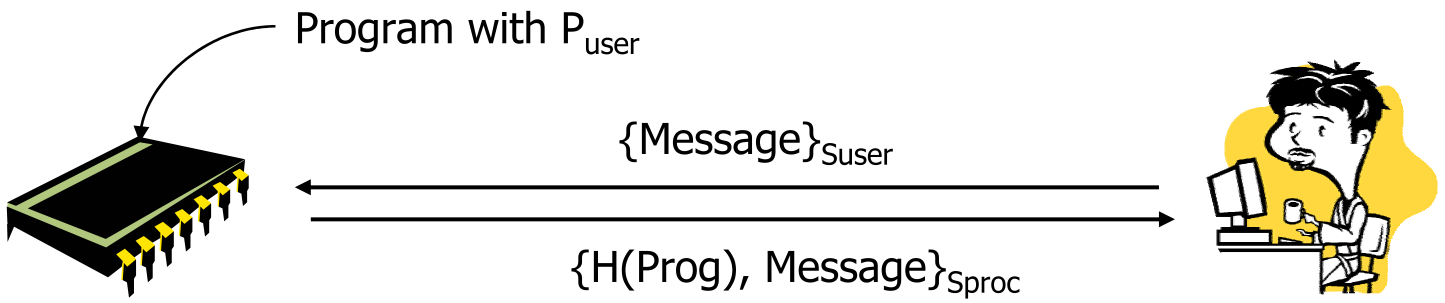
- ✓ L2 is trusted
- ✓ Stop checking earlier

→ Less overhead

Message Authentication

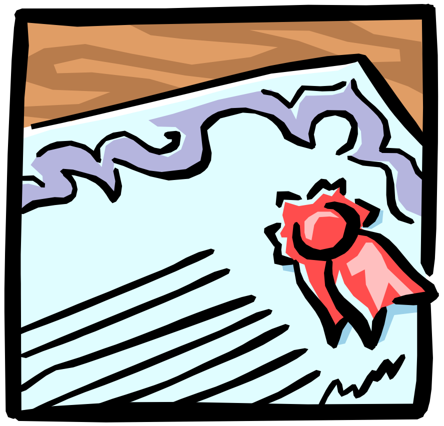
- Processor → Other systems
  - The processor signs a message for a program
    → ***sign_msg M***: $\{H(Prog), M\}_{SKproc}$
  - Unique for each program because H(Prog) is always included

- Other systems → Processor
  - Embed the user's public key in a program
  - Incoming messages are signed with the user's private key



MIT PROJECT OXYGEN
PERVASIVE, HUMAN-CENTERED COMPUTING

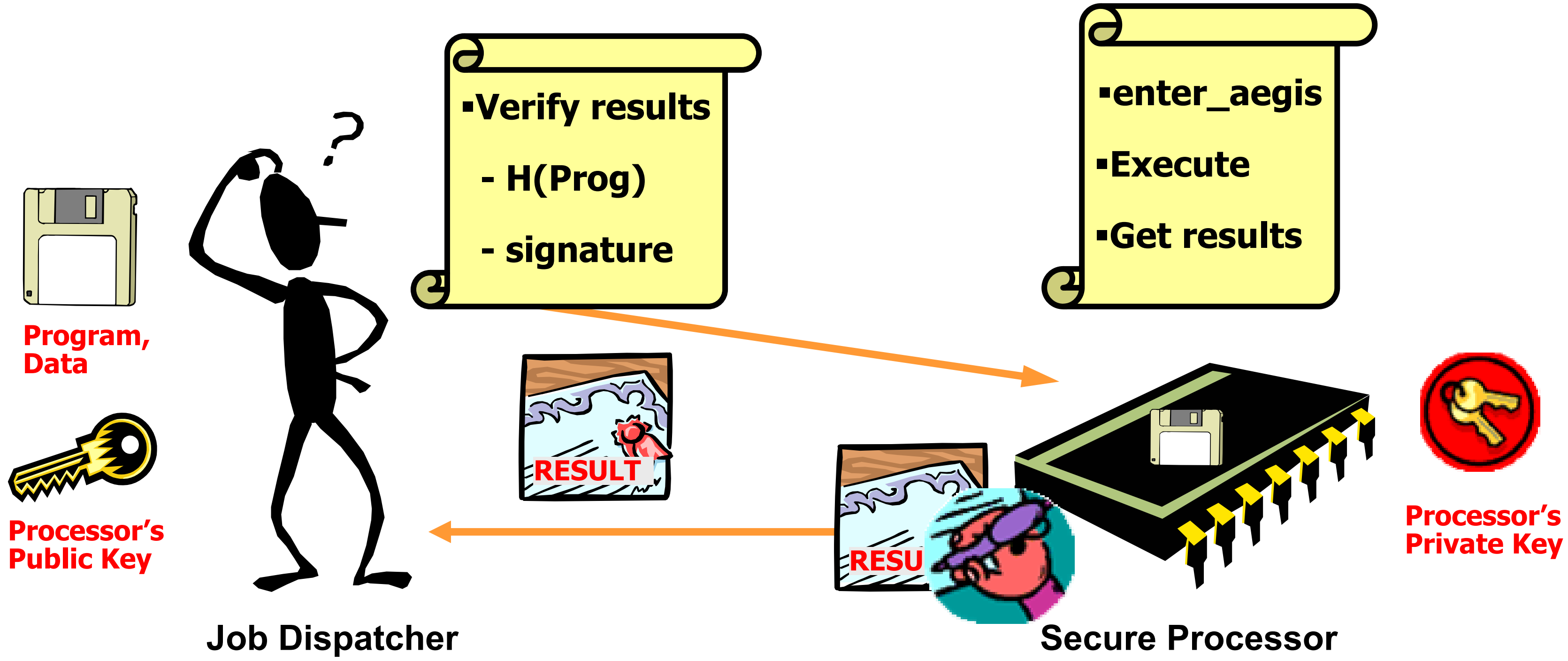
Applications



L C S

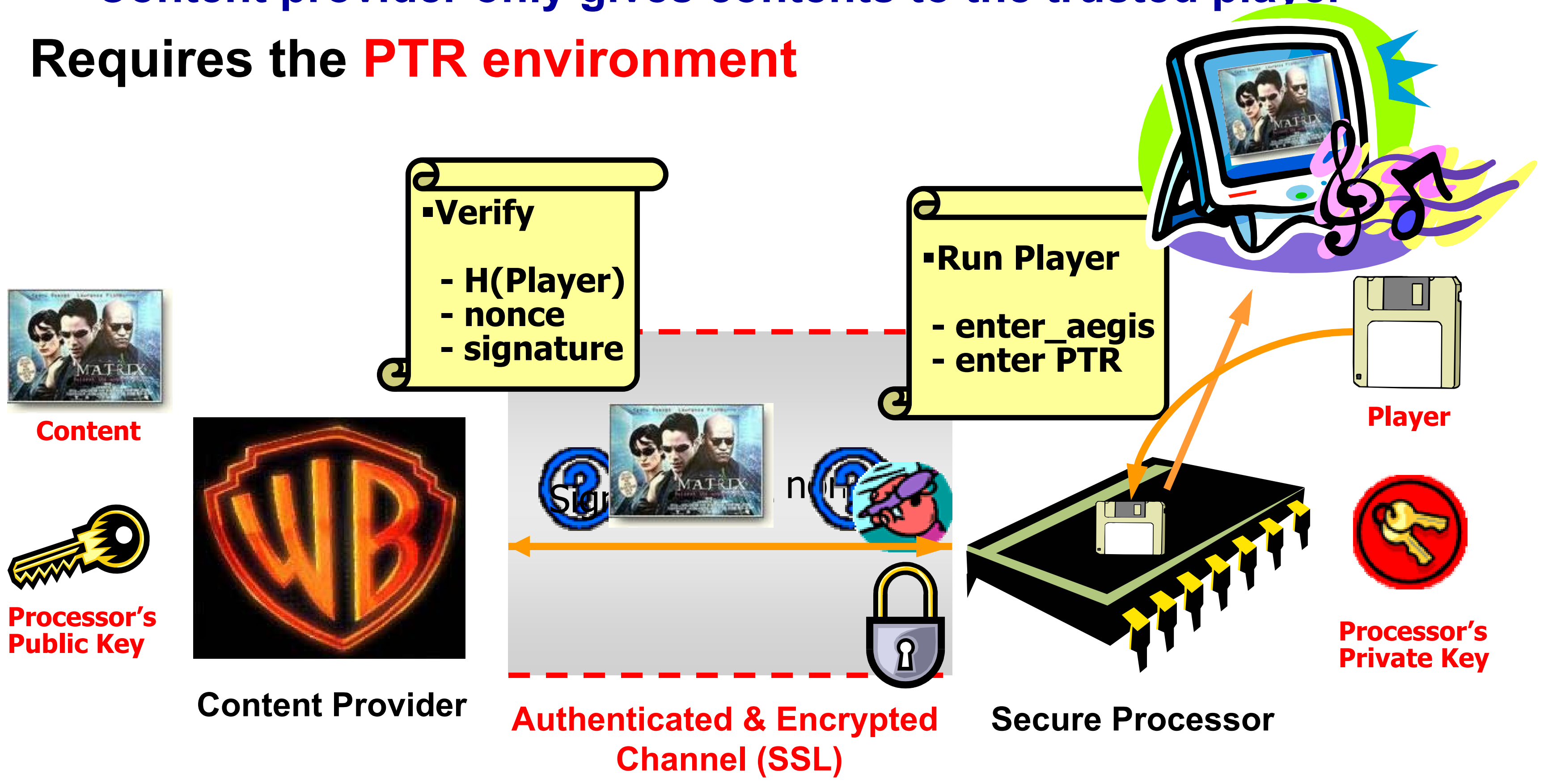
Certified Execution

- Execution certified by the secure processor
  - Dispatcher provides a program and data
  - Processor returns the results with the signature
- Requires the TE environment



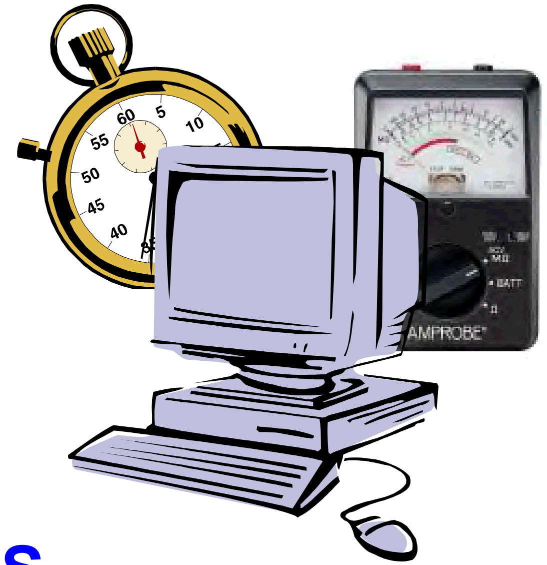
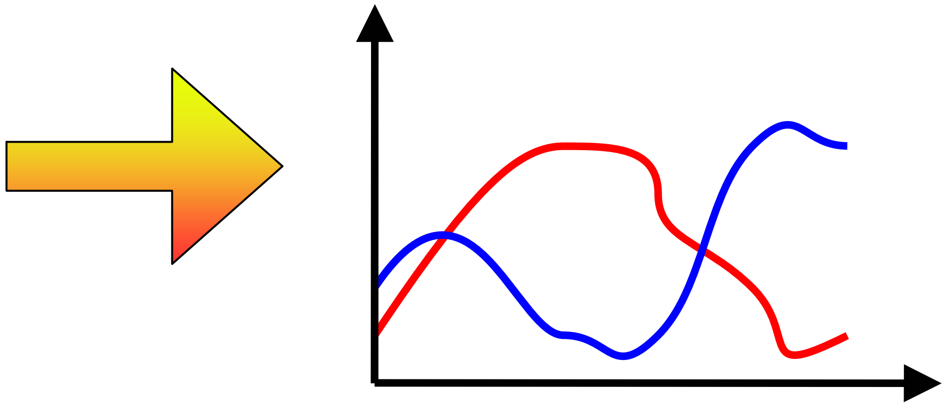
Digital Rights Management

- Protects digital contents from illegal copying
  - Trusted software (player) on untrusted host
  - Content provider only gives contents to the trusted player
- Requires the PTR environment



MIT PROJECT OXYGEN
PERVASIVE, HUMAN-CENTERED COMPUTING

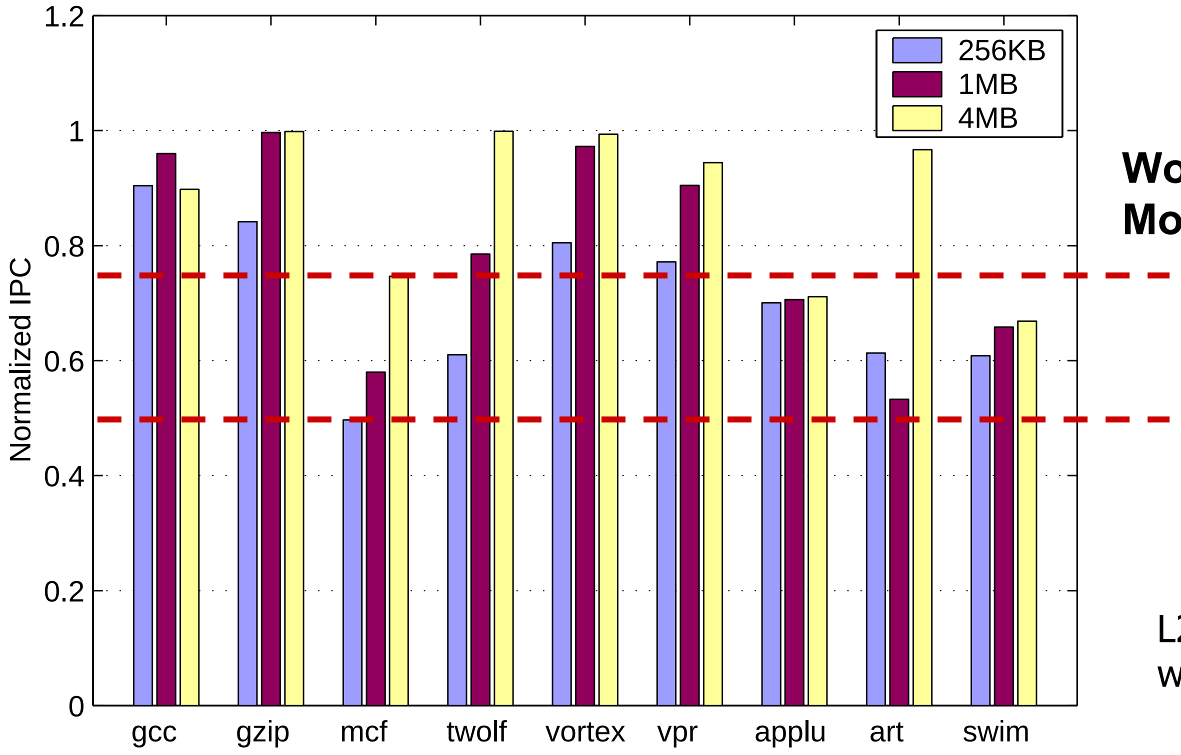
Performance

L C S

Performance Implication: TE processing

- Major performance degradation is from off-chip integrity checking
  - Start-up and context switches are infrequent
  - no performance overhead for on-chip tagging

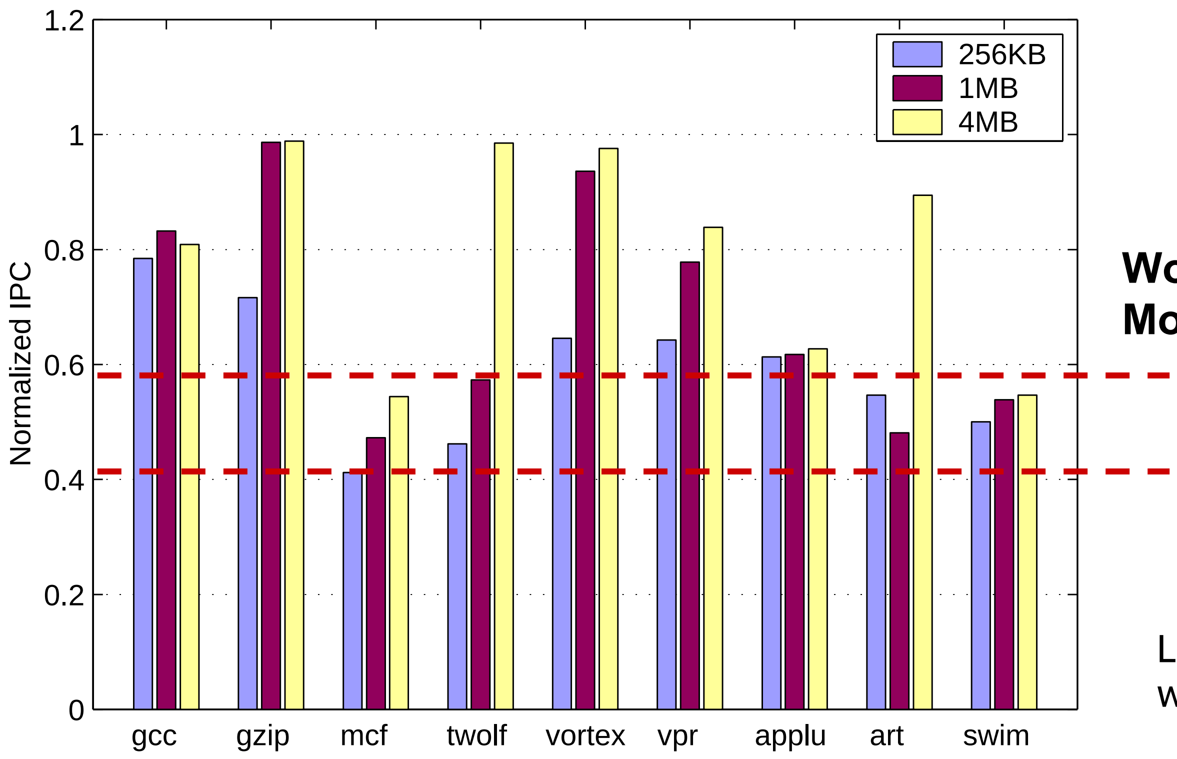


Worst case 50% degradation
Most cases < 25% degradation

L2 Caches
with 64B blocks

Performance Implication: PTR processing

- **Major performance degradation is from off-chip integrity checking and encryption**



Worst case 60% degradation
Most cases < 40% degradation

L2 Caches
with 64B blocks

Summary

- **Physical attacks are becoming more prevalent**
  - **DRM, software licensing, distributed computing, etc.**
- **Single-chip secure processors provide trusted execution environments with acceptable overhead**
  - **Tamper-Evident environment, Private Tamper-Resistant environment**
  - **Simulation results show 25-50% overhead for TE, 40-60% overhead for PTR processing**
  - **New mechanisms can reduce the overhead to 5-15% for TE, and 10-25% for PTR processing (CSG Memo 465)**
- **Significant development effort underway**
  - **FPGA/ASIC implementation of AEGIS processor**

Questions?

More Information at www.csg.lcs.mit.edu