Fast Secure Processor for Inhibiting Software Piracy and Tampering

Jun Yang, Youtao Zhang*, and Lan Gao
University of California, Riverside
*University of Texas at Dallas

Outline
- Motivation
- XOM Architecture Overview
- Offloading Crypto-Computation from Critical Path
- Architecture Design
- Experiment Evaluation
- Summary

From Outside Computer
- Software Piracy
- Billions of $ loss each year due to software piracy

From Inside Computer
- Software Tampering
- Debuggers
  - SoftICE, ...
- Disassemblers
  - IDA Pro, ...
- Memory Dump Utilities
  - PEDump, ...
Hardware Support Against Software Piracy

- One solution - eXecution Only Memory (XOM)  
  [David Lie et al. ASPLOS 2000]
- Who is trustworthy?
  - Only the processor itself is trusted
  - Co-processor, operating system, memory, system bus are NOT tamper resistant
- What needs encryption?
  - Software stored in the system storage
  - Data communicated on the system bus
  - Register values on interrupts

Software Distribution Method in XOM

The Lengthened Memory Path in XOM

Performance Degradation in XOM

Assume the following latencies:
- Average Memory Access — 100 cycles
- Encryption/Decryption — 50 cycles (optimistic)
Offloading Crypto-Computation from Critical Path

- The crypto-computation in XOM:
  - Data dependent on memory accesses
  - Carried in serial with the memory accesses
- Our One-Time Pad based scheme:
  - Decouple en/decryption from memory access
  - They can be carried in parallel
- The memory encryption scheme: G. Edward Suh et al
  - Similarity: One-Time-Pad Encryption
  - Major Difference:
    § Timestamp storage: off-chip v.s. on-chip

One-Time Pad (OTP) Encryption

- Sender
  - random number
  - (pad)
  - Plain text -> ciphertext
  - receiver
  - random number
  - (pad)
  - Plain text

- CPU
  - $E_{key}(seed)$
  - Memory
  - $E_{key}(seed)$
  - CPU
  - data
  - encrypted data
  - data

Speed Up XOM

- Let us assume:
  - Memory access latency = 100 cycles
  - Encryption/Decryption latency = 50 cycles
  - XOR needs 1 cycle
- Memory access latency with crypto operations:

  XOM: 150 cycles
  Our Scheme: 101 cycles

Two Issues

- OTP based encryption strength
  - in authentic OTP:
    - strength(ciphertext) $\geq$ strength(random number)
  - in our scheme:
    - strength(encrypted data) $\geq$ strength($E_{key}(seed)$)
- Seed selection
  - independent of data value, known before data is available — address
  - multiple accesses of same data use different seeds — one-time seed
Introducing Sequence Numbers

Write V \rightarrow A

<table>
<thead>
<tr>
<th>time</th>
<th>t_0</th>
<th>t_1</th>
<th>t_2</th>
<th>t_3</th>
</tr>
</thead>
</table>
| values at A | 1   | 2   | 3   | ...

1. XOM: \binom{E_{\text{seed}}(I)}{E_{\text{seed}}(A)} \oplus \binom{E_{\text{seed}}(A+t_1)}{E_{\text{seed}}(A+t_2)} \oplus \binom{E_{\text{seed}}(A+t_3)}{E_{\text{seed}}(A+t_4)} \ldots
2. Use A only: \binom{E_{\text{seed}}(A)}{E_{\text{seed}}(A+t_1)} \oplus \binom{E_{\text{seed}}(A+t_2)}{E_{\text{seed}}(A+t_3)} \ldots
3. Use A and t: \binom{E_{\text{seed}}(A+t_1)}{E_{\text{seed}}(A+t_2)} \oplus \binom{E_{\text{seed}}(A+t_3)}{E_{\text{seed}}(A+t_4)} \ldots

\text{seed} = \text{address} + \text{timestamp}

Comparing XOM and OTP Based XOM

<table>
<thead>
<tr>
<th>XOM</th>
<th>XOM w/ OTP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spatially</td>
<td>A_1 \quad A_2</td>
</tr>
<tr>
<td>\quad E_{\text{seed}}(100)</td>
<td>\quad E_{\text{seed}}(100)</td>
</tr>
<tr>
<td>Temporally</td>
<td>A_1 \quad t_1 \quad t_2</td>
</tr>
<tr>
<td>\quad E_{\text{seed}}(100)</td>
<td>\quad E_{\text{seed}}(A+t_1) \oplus 100</td>
</tr>
<tr>
<td>\quad E_{\text{seed}}(A+t_2) \oplus 100</td>
<td></td>
</tr>
</tbody>
</table>

- Our scheme better randomizes encrypted data in memory

Seed Storage

- Store seeds in memory
  - not beneficial since mem. accesses are doubled
- Use an on-chip cache to remember seeds
  - need only to store the sequence numbers (t) since they can be narrower than the seeds

Architecture Design

- L2 Cache
- Encryption/Decryption Unit
- Main Memory
- Security Boundary
- Write Buffer
- Read Buffer

Write V \rightarrow A
SNC Capacity is Limited

- Stop using OTP once it’s full
  - only partial memory blocks have seeds
  - simple control, good for programs with modest mem. requirement
- Use replacement (LRU) to store all the seeds
  - Three ways to store evicted sequence numbers
    - Encrypt using one-time pad
      - They themselves would need the sequence numbers!
    - Encrypt directly as XOM
      - Increase memory access latency
    - Store plaintext
      - secure since the private key is not revealed

Other Issues

- Context Switching
  - Flush SNC to the memory
  - Tag each entry with XOM ID
- Shared library and program inputs
  - Both should be provided in plaintexts
  - Do not need sequence numbers in SNC

SNC Operation: Best and Worst Cases

- The best case (hit SNC):
  - \[ \text{Lat}_{\text{hit}} = \text{MAX}(T_1, T_2) + T_3 = 101 \text{ cycles} \]
- The worst case (miss SNC):
  - \[ \text{Lat}_{\text{miss}} = T_0 + \text{MAX}(T_1, T_2) + T_3 = 201 \text{ cycles} \]

Other Issues

- Context Switching
  - Flush SNC to the memory
  - Tag each entry with XOM ID
- Shared library and program inputs
  - Both should be provided in plaintexts
  - Do not need sequence numbers in SNC

Experiments

- Tools
  - SimpleScalar V3.0
  - 11 SPEC2000 benchmarks
- Baseline
  - 4-issue out-of-order processor
  - Caches:
    - Separate L1 cache and D-cache: 32KB, 4-way
    - Unified L2 cache: 256KB, 4-way, 128B/line
  - Latencies:
    - Memory access latency: 100 cycles
    - Encryption latency: 50 cycles
- Execution
  - Fast forwarded by 10 billion instructions
  - Then execute 10 billion instructions
Other Experiments

- **SNC Induced Memory Traffic**
  - On average, there is only 0.31% of the L2 memory traffic posed by SNC replacements on to the system bus.

- **Sensitivity to Encryption Latency**
  - XOM degrades greatly from 16.7% to 34.2% slowdown.
  - The performance of our design with LRU replacements is almost unchanged.

Conclusion

- Apply one-time pad (OTP) cryptography to speed up the secure processor.
- Develop the hardware support.
- Reduce the performance overhead from 16.7% for critical path cryptography to 1.28% for OTP cryptography.

*Thanks*

*Questions?*