Authenticated Storage Using Small Trusted Hardware

Hsin-Jung Yang, Victor Costan, Nickolai Zeldovich, and Srini Devadas

Massachusetts Institute of Technology

November 8th, CCSW 2013
Cloud Storage Model

cloud users

storage

cloud servers
Cloud Storage Requirements

• Privacy
  – Sol: encryption at the client side

• Availability
  – Sol: appropriate data replication

• Integrity
  – Sol: digital signatures & message authentication codes

• Freshness
  – Hard to guarantee due to replay attacks
Cloud Storage: Replay Attack

User A

Cloud Server

User B
Cloud Storage: Replay Attack

User A ➔ Cloud Server ➔ User B
Cloud Storage: Replay Attack

User A  Cloud Server  User B
Cloud Storage: Replay Attack

User A

Cloud Server

User B
Cloud Storage: Replay Attack

User A Cloud Server User B
Cloud Storage: Replay Attack

User A Cloud Server User B
Cloud Storage: Replay Attack
Cloud Storage: Replay Attack

User A  Cloud Server  User B
Cloud Storage: Replay Attack

User A

Cloud Server

User B
Cloud Storage: Replay Attack

Software solution:
Two users contact with each other directly

User A
Cloud Server
User B
Solution: Adding Trusted Hardware
Solution: Adding Trusted Hardware
Solution: Adding Trusted Hardware

Internet

server
disk
trusted hardware

client_1
client_2
client_3

single chip
Solution: Adding Trusted Hardware

- **Secure NVRAM**
- **trusted hardware**
- **single chip**
Solution: Adding Trusted Hardware

- Secure NVRAM
- Computational Engines

Single chip
Solution: Adding Trusted Hardware

Secure NVRAM

Computational Engines

Slow under NVRAM process!
Solution: Adding Trusted Hardware

- Secure NVRAM
- Computational Engines
- State chip (S chip)
- Processing chip (P chip)
Solution: Adding Trusted Hardware

Smart Card

state chip (S chip)

Secure NVRAM

Computational Engines

processing chip (P chip)

Fast!

FPGA / ASIC

Internet

client₁

client₂

client₃

server

disk

generic

trusted hardware

Fast!
Solution: Adding Trusted Hardware

- Secure NVRAM
- Computational Engines
- Processing chip (P chip)
- FPGA / ASIC
- Fast!
Outline

• Motivation: Cloud Storage and Security Challenges

• System Design
  – Threat Model & System Overview
  – Security Protocols
  – Crash Recovery Mechanism

• Implementation

• Evaluation

• Conclusion
Threat Model

Internet -----> server -----> disk

client\(_1\) -----> Internet

client\(_2\) -----> Internet

client\(_3\) -----> Internet

S chip --------> trusted

P chip --------> trusted
Threat Model

- Untrusted connections
- Disk attacks and hardware failures
- Untrusted server that may
  1. send wrong response
  2. pretend to be a client
  3. maliciously crash
  4. disrupt P chip’s power
- Clients may try to modify other’s data
Threat Model

- Untrusted connections
- Disk attacks and hardware failures
- Untrusted server that may
  (1) send wrong response
  (2) pretend to be a client
  (3) maliciously crash
  (4) disrupt P chip’s power
- Clients may try to modify other’s data
Threat Model

- Untrusted connections
- Disk attacks and hardware failures
- Untrusted server that may
  1. send wrong response
  2. pretend to be a client
  3. maliciously crash
  4. disrupt P chip’s power
- Clients may try to modify other’s data
Threat Model

- Untrusted connections
- Disk attacks and hardware failures
- Untrusted server that may:
  1. Send wrong response
  2. Pretend to be a client
  3. Maliciously crash
  4. Disrupt P chip’s power
- Clients may try to modify other’s data
System Overview

- **Client <-> S-P chip:** HMAC key
- **S-P chip:** integrity/freshness checks, system state storage & updates, sign responses
- **Server:** communication, scheduling, disk IO
Security Protocols

- Message Authentication
- Memory Authentication
- Write Access Control
- System State Protection against Power Loss
Design: Message Authentication

- Untrusted network between client and server
  - Sol: HMAC Technique
- Session-based protocol (HMAC key)
Security Protocols

• Message Authentication
• Memory Authentication
• Write Access Control
• System State Protection against Power Loss
Design: Memory Authentication

- Data protection against untrusted disk
- Block-based cloud storage API
  - Fixed block size (1MB)
  - Write (block number, block)
  - Read (block number) \( \rightarrow \) block
  - Easy to reason about the security

![Diagram of disk with blocks B1 to B4]
Design: Memory Authentication

- **Solution:** Merkle tree

Disk is divided into many blocks
Design: Memory Authentication

- **Solution:** Merkle tree

```
  h1..4
 /    /
/      /
 h12   h34
 |      |
 |      |
 B1     B2
```

```
  h5..8
 /    /
/      /
 h56   h78
 |      |
 |      |
 B5     B8
```

- $h_{12} = H(h_1 || h_2)$
- $h_1 = H(B_1)$

Root Hash (securely stored)

Disk is divided into many blocks
Design: Memory Authentication

- **Solution:** Merkle tree

\[ h_{12} = H(h_1 || h_2) \]

\[ h_1 = H(B_1) \]

Disk is divided into many blocks
Design: Memory Authentication

- **Solution:** Merkle tree

| B₁ | B₂ | B₃ | B₄ | B₅ | B₆ | B₇ | B₈ |

Disk is divided into many blocks

- $h_{12} = H(h₁ \| h₂)$
- $h₁ = H(B₁)$
- $h₁ = H(B₁)$

Root Hash (securely stored)
Merkle Tree Caching

- Caching policy is controlled by the server

<table>
<thead>
<tr>
<th>Node #</th>
<th>Hash</th>
<th>Verified</th>
<th>Left child</th>
<th>Right child</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>fabe3c05d8ba995af93e</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>2</td>
<td>e6fc9bc13d624ace2394</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>4</td>
<td>53a81fc2dccc53e4da819</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>5</td>
<td>b2ce548dfaf91d83ec6</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
</tbody>
</table>

Cache management commands: LOAD, VERIFY, UPDATE
Security Protocols

- Message Authentication
- Memory Authentication
- **Write Access Control**
- System State Protection against Power Loss
Design: Write Access Control

• **Goal:** to ensure all writes are authorized and fresh

• **Coherence model assumption:**
  – Clients should be aware of the latest update

• **Unique write access key (Wkey)**
  – Share between authorized writers and the S-P chip

• **Revision number (V_{id})**
  – Increase during each write operation
Design: Write Access Control

- **Protect Wkey and V_{id}**
  - Add another layer at the bottom of Merkle tree
Security Protocols

- Message Authentication
- Memory Authentication
- Write Access Control
- **System State Protection against Power Loss**
Design: System State Protection

- **Goal:** to avoid losing the latest system state
  - Server may interrupt the P chip’s supply power
- **Solution:** root hash storage protocol
Design: Crash Recovery Mechanism

• **Goal:** to recover the system from crashes
  
  – Even if the server crashes, the disk can be recovered to be consistent with the root hash stored on the S chip

• **Solution:**

![Diagram of the crash recovery mechanism showing S Chip, NV-RAM, and RAM with nodes and hashes for recovery storage and request log.](image-url)
Implementation

- ABS (authenticated block storage) server architecture
Implementation

- ABS client model
Performance Evaluation

• **Experiment configuration**
  – **Disk size:** 1TB
  – **Block size:** 1MB
  – **Server:** Intel Core i7-980X 3.33GHz 6-core processor with 12GB of DDR3-1333 RAM
  – **FPGA:** Xilinx Virtex-5 XC5VLX110T
  – **Client:** Intel Core i7-920X 2.67GHz 4-core processor
  – **FPGA-server connection:** Gigabit Ethernet
  – **Client-server connection:** Gigabit Ethernet
File System Benchmarks (Mathematica)

- **Fast network:**
  - Latency: 0.2ms
  - Bandwidth: 1Gbit/s
File System Benchmarks (Mathematica)

- **Slow network:**
  - Latency: 30.2ms
  - Bandwidth: 100Mbit/s
File System Benchmarks (Modified Andrew Benchmark)

- Slow network:
  - Latency: 30.2ms
  - Bandwidth: 100Mbit/s
Customized Solutions

• Hardware requirements

<table>
<thead>
<tr>
<th>Demand Focused</th>
<th>Performance</th>
<th>Budget</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection</td>
<td>PCIe x16 (P) / USB (S)</td>
<td>USB</td>
</tr>
<tr>
<td>Hash Engine</td>
<td>8 + 1 (Merkle)</td>
<td>0 + 1 (Merkle)</td>
</tr>
<tr>
<td>Tree Cache</td>
<td>large</td>
<td>none</td>
</tr>
<tr>
<td>Response Buffer</td>
<td>2 KB</td>
<td>300 B</td>
</tr>
</tbody>
</table>

• Estimated performance

<table>
<thead>
<tr>
<th>Demand Focused</th>
<th>Performance</th>
<th>Budget</th>
</tr>
</thead>
<tbody>
<tr>
<td>Randomly Write</td>
<td>Throughput</td>
<td>2.4 GB/s</td>
</tr>
<tr>
<td></td>
<td>Latency</td>
<td>12.3 ms + 32 ms</td>
</tr>
<tr>
<td>Randomly Read</td>
<td>Throughput</td>
<td>2.4 GB/s</td>
</tr>
<tr>
<td></td>
<td>Latency</td>
<td>0.4 ms</td>
</tr>
<tr>
<td># HDDs supported</td>
<td></td>
<td>24</td>
</tr>
</tbody>
</table>
## Customized Solutions

### Hardware requirements

<table>
<thead>
<tr>
<th>Demand Focused</th>
<th>Performance</th>
<th>Budget</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection</td>
<td>PCIe x16 (P) / USB (S)</td>
<td>USB</td>
</tr>
<tr>
<td>Hash Engine</td>
<td>8 + 1 (Merkle)</td>
<td>0 + 1 (Merkle)</td>
</tr>
<tr>
<td>Tree Cache</td>
<td>large</td>
<td>none</td>
</tr>
<tr>
<td>Response Buffer</td>
<td>2 KB</td>
<td>300 B</td>
</tr>
</tbody>
</table>

### Estimated performance

<table>
<thead>
<tr>
<th>Demand Focused</th>
<th>Performance</th>
<th>Budget</th>
</tr>
</thead>
<tbody>
<tr>
<td>Randomly Write</td>
<td>Throughput</td>
<td>2.4 GB/s</td>
</tr>
<tr>
<td></td>
<td>Latency</td>
<td>12.3 ms + 32 ms</td>
</tr>
<tr>
<td>Randomly Read</td>
<td>Throughput</td>
<td>2.4 GB/s</td>
</tr>
<tr>
<td></td>
<td>Latency</td>
<td>0.4 ms</td>
</tr>
<tr>
<td># HDDs supported</td>
<td>24</td>
<td>4</td>
</tr>
</tbody>
</table>

Single chip!
Conclusion

• We build an authenticated storage system
  – Efficiently ensure data integrity and freshness
  – Prevent unauthorized/replayed writes
  – Can be recovered from accidentally/malicious crashes
• The system has 10% performance overhead on the network with 30 ms latency and 100 Mbit/s bandwidth
• We provide customized solutions
  – With limited resources: single-chip solution
  – With more hardware resources: two-chip solution
Thank You!