DISKSHIELD: A Data Tamper-Resistant Storage for Intel SGX

Jinwoo Ahn†, Junghee Lee‡, Yungwoo Ko†, Donghyun Min†,
Jihyun Park†, Sungyong Park†, Youngjae Kim†

†Sogang University, Republic of Korea,
‡Korea University, Republic of Korea
Motivation: The dangers of ransomware

Ransomware

User

Storage
Motivation: Two characteristics of powerful ransomware

1. Ring-0 level rootkit malware
   - Disabling the system (OS) using the latest exploits.

2. Backup data attack
   - Deleting all local or remote backup files of infected computer.
Motivation: Two characteristics of powerful ransomware

1. Ring-0 level rootkit malware
   - Disabling the system (OS) using the latest exploits.

Defense against data tampering attack:
Isolation from OS + Safe Disk Partition from Malware

- Deleting all local or remote backup files of infected computer.
Opportunity: TEE and Storage devices

**Trusted Execution Environment (TEE)**

TEE provides code and data areas isolated from the OS.

**Defense on storage devices**

Storage is the last barrier to storing data.
Background: Intel SGX

- Intel SGX guarantees the **confidentiality** and **integrity** of applications even if system components such as OS are damaged.
Background: Solid State Drive

1. In-device computation capability

2. Isolated space from host server
1. TEE-based solution: IPFS

- IPFS guarantees the confidentiality and integrity of files created by SGX application.
- IPFS is a user-level file system dependent on native filesystem (EXT4).

- WRITE: Encrypt & Sign Data
- READ: Decrypt & Verify Data
Related work: Intel Protected File System (IPFS)

1. TEE-based solution: IPFS
   - IPFS guarantees the confidentiality and integrity of files created by SGX

   **WRITE**: Encrypt & Sign Data
   **READ**: Decrypt & Verify Data

   Read verification alone can not prevent data tampering attack.
Related work: limitation of IPFS
Related work: limitation of IPFS

1. Persistent Data Attack
   - Untrusted Part
   - Malicious App
   - ioctl()
pwrite()

2. Metadata Attack
   - Metadata Corruption
   - Native filesystem (EXT4)
Related work: Inuksuk

2. TEE + Disk based solution: 2. Inuksuk
Related work: Inuksuk

2. TEE + Disk based solution: 2. Inuksuk

3. Fresh Data Attack
System proposal: DISKSHIELD

- Robust data-tampering resistant storage system from privileged malware.
  - Protect all kinds of data: persistent data, fresh data, metadata.

- It runs without additional disks in local environment.
  - No additional cost is required for purchasing external disk.
System proposal: Data protection from any attack surface

Persistent data attack

Malicious App

\texttt{pwrite()} \hspace{1cm} \texttt{ioctl()}
System proposal: Data protection from any attack surface

Persistent data attack

Access control in device

Malicious App

\texttt{pwrite()} \quad \texttt{ioctl()}

Secure Path

FTL

Secure I/O

Normal I/O

NAND Flash

Storage

Malicious App

\texttt{pwrite()} \quad \texttt{ioctl()}
System proposal: Data protection from any attack surface

Fresh data attack

- System Call Hijacking
- Malicious Driver Attack

Secure two-way authentication channel

- SGX App
- Secure two-way authentication channel
System proposal: Data protection from any attack surface

Meta Data attack

APP

OS
File System (EXT4, F2FS)

SSD

Inode

Destroy Inode, Indexing Tree

In storage filesystem

SGX APP

Two-way authentication

SSD

Inode

Device File System

Nand Flash
Design & Implementation: system components

**DSFS**:  
- Extension of IPFS in SGX  
- Two-way authentication module  

**DSSSD**:  
- SSD Firmware implementation  
- Device filesystem  
- Device level authentication system  

**DSAE**:  
- Key manager between DSFS and DSSSD
DSAFE enables two-way authentication by sharing a file key between DSFS and DSSSD.

When creating a file, the key generated by DSFS is securely shared with DSSSD.
Design & Implementation: \textit{DSAE}

DSAE enables two-way authentication by sharing a file key between DSFS and DSSSD.

When creating a file, the key generated by DSFS is securely shared with DSSSD.

\textbf{DEAE: File key is shared when the file is created}
Design & Implementation: **DSFS**

- **APP**
- **IPFS**
  - **E** (Write)
  - **RV** (Read)
  - Untrusted FS Lib
  - File System
  - Device Driver

- **DSFS**
  - Two-way Authentication
  - Trusted DS Lib
  - **WV**

- **IPFS**
  - **RV** (Read)
  - **E** (Write)

- **Device Driver**
  - **DS_rdafwr()**
  - call_write_iter()
  - call_read_iter()

- **SUCCESS!**
  - **DS_SSSD**
  - **MAC Verification**
  - **Write Verification**
Design & Implementation: **DSFS**

Integrity of the file is guaranteed from the point when the application flushes the file.
Design & Implementation: **DSSSD**

- **Device Driver**
  - Normal I/O
  - Secure I/O

- **G-FTL**
- **MAC Verification**
- **Device FS**
  - DS-FTL

- **NAND Flash**

---

24
Design & Implementation: **DSSSD**

In Memory Structure:
- **DS Open**
  - Root Directory
  - Per-Key Directory
  - DS structure

- **DS Read/Write**
  - File Table
  - Inode
  - DS-FTL
  - DS Read/Write (DS-FTL)

- **DS Close**

On Disk Structure:
- **Firmware BIN**
- **G-FTL**
  - Super block
    - DS inodes
    - DS FTL
  - Physical Data

Diagram:
- **MAC Verification**
- **Unmarshalling**
- **Inode**
  - name
  - key
  - ...
  - Direct
    - Single Indirect
    - Double Indirect

- **FTL Chunk**
  - Offset
  - PPN

Steps:
1. DS Open
2. DS Close
3. DS Open
4. DS Read/Write
5. DS Read/Write (DS-FTL)
DSSSD: secure file open flow

1. DS open (secure.txt)
   - In Memory Structure

2. Obtain inode number from directory.
   - DS Read/Write

3. Load inode.
   - DS Open

4. Update File table.
   - DS Read/Write

5. Return file descriptor.
   - DS Read/Write

In Memory Structure:
- Root Directory
- Per-Key Directory
- DS-FTL
- File Table
- Inode

On Disk Structure:
- Firmware BIN
- G-FTL
- Super block
- DS inodes
- DS FTL
- Physical Data

General Read/Write

1. DS open (secure.txt)
2. Obtain inode number from directory.
3. Load inode.
4. Update File table.
5. Return file descriptor.
**DSssd**: secure file write flow

1. **DS write** *(fd, offset, size, buf)*
   - Unmarshalling
   - MAC Verification

   ![Diagram](image)

   - **DS Open**
   - Root Directory
   - Per-Key Directory
   - **DS structure**
   - File Table
   - Inode
   - **DS-FTL**

2. **Obtain inode from File table.**

3. **Acquire physical address from DS-FTL.**

4. **Write data to physical address**

   - **Super block**
   - **DS inodes**
   - **DS FTL**

   ![Physical Data](image)
Evaluation: Prototype implementation

**DS_Fs, DS_AE**
Intel SGX

**Two-way Authentication Module**
Linux 4.10.16
SATA Interface

**DS_SSD**
Open SSD Jasmine

SSD: 2661 Loc
Kernel: 243 Loc
SGX SDK: 791 Loc
Evaluation

Baseline, R-zone: Normal I/O
S-zone: Secure I/O

Baseline: IPFS + EXT4 + Normal SSD
R-zone: IPFS + EXT4 + DSSSD (normal)
S-zone: DSFS + DSSSD (secure)

1. Baseline and R-zone throughput are the same
   • There is no normal I/O overhead of DSSSD.

Baseline, R-zone, S-zone throughput comparison

Baseline, R-zone, S-zone throughput comparison
Evaluation

Baseline, R-zone: Normal I/O
S-zone: Secure I/O

Baseline: IPFS + EXT4 + Normal SSD
R-zone: IPFS + EXT4 + DSSSD (normal)
S-zone: DSFS + DSSSD (secure)

1. Baseline and R-zone throughput are the same
   • There is no normal I/O overhead

2. In S-zone, throughput was 17% lower than baseline.
   • Two-way authentication module overhead

3. S-zone is not scalable even if the number of threads increases.
   • Coarse-grained lock of two-way authentication module
Evaluation

1. S-Zone's read throughput drops by more than 50%.
   • Overhead to load/store multiple inodes in DSSSD
     • DSSSD loads only one inode. -> No cache effect!
     • It should prefetch multiple inodes at once like EXT4 filesystem.
Conclusion

- DISKSHIELD guarantees file integrity from privileged data tampering attacks.
  - Defense of all attack surfaces: Persistent data attack, Fresh data attack, metadata attack
- DISKSHILED runs in local environment without additional disk.
  - It can be implemented only by firmware update to existing SSD without hardware cost.
DISKSHIELD: A Data Tamper-Resistant Storage for Intel SGX

Jinwoo Ahn
jinu37@sogang.ac.kr