Reliably Erasing Data from Flash-Based Solid State Drives

Michael Wei*
Laura Grupp*, Fredrick E. Spada†, Steven Swanson*

*Non-Volatile Systems Laboratory
Department of Computer Science and Engineering
University of California, San Diego

†Center for Magnetic Recording Research
University of California, San Diego
Confidential Data

Sensitive information which...

- Limited to people with need
- Destroyed at end of life
YOU...

have confidential data on your computer right now!
CORPORATIONS...

must protect their own data as well as client’s data.
GOVERNMENTS...

must protect information
to protect the state and lives of its citizens
Confidential Data

Sensitive information which...

• Limited to people with need
• Destroyed at end of life

How?
What we know comes from years of research on hard drives.
Solid State Disks (SSDs)

next generation storage...

- Flash-based
- No moving parts
- Uses a complex controller (Flash Translation Layer)
SSDs are becoming quite popular... 

Source: DRAMeXchange
You might have left confidential data and not even realized it.
Why is it hard to erase SSDs?

Current sanitization tools are designed for hard drives. But SSDs are very different!
SSD Differences

• Recovery process is cheap
• Wide space of manufacturers for poor implementation
• Easy Disassembly / Reassembly

Let’s see what’s on this SSD...

• Low cost compared to hard drives
• Someone could steal your data overnight!
Overview

• Motivation
• **Sanitization Background**
• Validating Sanitization and Results
• Single-File Sanitization Enhancement
Sanitization

Erasing data so that it is difficult or impossible to recover
For this talk, we’ll talk about the chip level.

- There’s leftover data
  - It’s cheap
- The next level is much more complex
Physical Level

- **Destroying Flash Memory-Based Storage Devices**, Steven Swanson, University of California, San Diego Computer Science & Engineering technical report cs2011-0968.

- 0.2mm particles
- Good until 2022 (8nm technology node)
Writing Data

Operating System’s View

<table>
<thead>
<tr>
<th>Hard Drive</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

Stale Data

Operating System’s View

<table>
<thead>
<tr>
<th>Solid State Disk</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

17
Writing more data...

Operating System’s View

![Diagram showing the difference between a Hard Drive and a Solid State Disk in terms of data write process.]

Hard Drive

Stale Data

Solid State Disk
Lots of stale data can be left over on the drive...
Overview

• Motivation
• Sanitization Background
• Validating Sanitization and Results
• Single-File Sanitization Enhancement
We now want to measure the stale data left over.
First, we constructed a “fingerprint” that was easily identifiable.
Second, We needed a way to see more than what the operating system sees.
Second, we needed a way to see more than what the operating system sees.
We built a custom hardware platform to extract data off the chips.
The drive is successfully sanitized if no stale data is left over.
Whole-disk sanitization

Erase the whole disk so that no old data remains.

- **Built-in Commands**
  - ATA Security “Erase Unit” (ATA-3), 1995
  - Cryptographic techniques

- **Software Overwrite**
  - Various Standards
Built-in commands

- ATA Security “Erase Unit”
ATA Security Erase Unit (1995)

- Normal: Replace the contents of LBA 0 to MAX LBA with binary zeroes or ones.

- Enhanced: All previously written user data shall be overwritten.

*Predates SSDs: doesn’t distinguish overwritten from erase.*
### ATA Security Erase Enhanced

Some drives tested supported and passed

<table>
<thead>
<tr>
<th>SSD Name</th>
<th>Controller</th>
<th>SECURITY ERASE UNIT (ATA-3)</th>
<th>SECURITY ERASE UNIT ENHANCED (ATA-3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>No (Reports yes)</td>
<td>No</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>Partial (Bugged)</td>
<td>No</td>
</tr>
<tr>
<td>D</td>
<td>3</td>
<td>Partial (Bugged)</td>
<td>No</td>
</tr>
<tr>
<td>E</td>
<td>4</td>
<td>Crypto Scrambles</td>
<td>Crypto Scrambles</td>
</tr>
<tr>
<td>F</td>
<td>5</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>G</td>
<td>6</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>H</td>
<td>7</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>I</td>
<td>8</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
ATA Security Erase Unit

One drive reported success, even though all data remained.

<table>
<thead>
<tr>
<th>SSD Name</th>
<th>Controller</th>
<th>SECURITY ERASE UNIT (ATA-3)</th>
<th>SECURITY ERASE UNIT ENHANCED (ATA-3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>No (Reports yes)</td>
<td>No</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>Partial (Bugged)</td>
<td>No</td>
</tr>
<tr>
<td>D</td>
<td>3</td>
<td>Partial (Bugged)</td>
<td>No</td>
</tr>
<tr>
<td>E</td>
<td>4</td>
<td>Crypto Scrambles</td>
<td>Crypto Scrambles</td>
</tr>
<tr>
<td>F</td>
<td>5</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>G</td>
<td>6</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>H</td>
<td>7</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>I</td>
<td>8</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
**ATA Security Erase Unit**

- Others only worked after the drive was reset

<table>
<thead>
<tr>
<th>SSD Name</th>
<th>Controller</th>
<th>SECURITY ERASE UNIT (ATA-3)</th>
<th>SECURITY ERASE UNIT ENHANCED (ATA-3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>No (Reports yes)</td>
<td>No</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>Partial (Bugged)</td>
<td>No</td>
</tr>
<tr>
<td>D</td>
<td>3</td>
<td>Partial (Bugged)</td>
<td>No</td>
</tr>
<tr>
<td>E</td>
<td>4</td>
<td>Crypto Scrambles</td>
<td>Crypto Scrambles</td>
</tr>
<tr>
<td>F</td>
<td>5</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>G</td>
<td>6</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>H</td>
<td>7</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>I</td>
<td>8</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Vendor Dependent

ATA SECURITY ERASE UNIT ENHANCED

ATA SECURITY ERASE UNIT

Software Overwrite
ATA Security Erase Unit

- Some drives crypto-scrambled, so we could not verify them

<table>
<thead>
<tr>
<th>SSD Name</th>
<th>Controller</th>
<th>SECURITY ERASE UNIT (ATA-3)</th>
<th>SECURITY ERASE UNIT ENHANCED (ATA-3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>No (Reports yes)</td>
<td>No</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>Partial (Bugged)</td>
<td>No</td>
</tr>
<tr>
<td>D</td>
<td>3</td>
<td>Partial (Bugged)</td>
<td>No</td>
</tr>
<tr>
<td>E</td>
<td>4</td>
<td>Crypto Scrambles</td>
<td>Crypto Scrambles</td>
</tr>
<tr>
<td>F</td>
<td>5</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>G</td>
<td>6</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>H</td>
<td>7</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>I</td>
<td>8</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Crypto-Scramble

Works by deleting key

• Fast, but…

• Encrypted data remains

• Data isn’t erased

• Crypto scramble makes drives unverifiable
Hardware Commands

• Wide variation in results
  – Not supported
  – Success
  – Crypto-scramble
  – Buggy implementation (works sometimes)
  – Failure (all data leftover)

• Result is implementation-dependent

• Will not know what happens until it is tested
SAFE: Scramble and Finally Erase

• UCSD Technical Report cs2011-0963
• Cryptography is desirable
• However, it is hard to verify
• A sanitized disk is easy to verify
• Why not crypto-scramble AND erase?
SAFE: Scramble and Finally Erase

- Traditional Sanitization Process
  - Sanitize and Initialize in a single step
  - Drive is \textit{INITIALIZED} after a sanitize
SAFE: Scramble and Finally Erase

- Crypto-Erase “Sanitization” Process
  - Delete keys
  - Drive is *INITIALIZED* after a sanitize
SAFE: Scramble and Finally Erase

SAFE breaks this up and adds two new states: KEYLESS and VERIFIABLE
SAFE: Scramble and Finally Erase

Scramble: Drive is actively being encrypted
– On sanitize, delete the keys *(KEYLESS)*
– This step takes milliseconds
SAFE: Scramble and Finally Erase

- Erase: Perform a block erase after scramble
  - We can easily verify the drive (VERIFIABLE)
  - This step takes minutes
SAFE: Scramble and Finally Erase

• We can now **verify** if the drive is erased
  – Via pulling off the chips
  – Possibly via hardware commands that don’t exist yet
  – External connector

• Best of both worlds
  – Fast cryptographic scramble
  – Slower, more secure erase
Myth: Flash takes a long time to erase

- 13 seconds to erase 4 Gbit
- 2.1 minutes to program 4 Gbit
- Can work on multiple chips in parallel
- #of channels scales with drive size (in general)

- Average disk (250GB) may take ~20s to fully erase

- With simple optimizations, a very fast erase is possible
SAFE: Scramble and Finally Erase

• *Problem*: We still have to trust the firmware designer to do it right!

• *Challenge*: How do we avoid the need to trust the firmware?
Software overwrite

• Various Government Standards
• According to NIST 800-88 (2006) “Studies today have shown that most of today’s media can be effectively cleared by one overwrite.”

Operating System’s View

Stale Data

Solid State Disk (Flash Chips)
Software overwrite

Operating System’s View

Stale Data
Solid State Disk (Flash Chips)

Operating System’s View

Stale Data
Solid State Disk (Flash Chips)
Software overwrite
How many times?

Our experiments show 2 passes are *typically* necessary

But even on the same drive, the number of required passes varied between 2 to more than 20.

Unreliable - hardware commands are best, if they are correctly implemented.
Single-File Sanitization

Erasing single files while leaving other parts of the drive intact
We want to sanitize only part of the disk.
Let’s try overwriting it…

Operating System’s View

Hard Drive

Solid State Disk (Flash Chips)
And again...
We tested with a 1000MB file, and got pretty bad results…
We tried to augment the existing procedures to do better…

- Wipe the free space
- Defragment and wipe

…but that didn’t help at all.
We’d like a hardware command that would tell the controller to delete stale data.
Overview

• Motivation
• Sanitization Background
• Validating Sanitization and Results
• Single-File Sanitization Enhancement
Scrubbing

An enhancement to the FTL to sanitize single files
Unfortunately, it’s not that easy.
First, flash is arranged into areas we can write to called \textit{pages}.
And pages are arranged into larger sections we can erase called **blocks**.
Erasing one piece of data would erase everything else in that block.
One method to get around the limitation is to copy.. But that’s slow!

Operating System’s View

Stale Data

Solid State Disk (Flash Chips)
We can overwrite individual pages

Operating System’s View

Stale Data

Overwrite

Solid State Disk (Flash Chips)
We can overwrite individual pages

Operating System’s View

Stale Data

Overwrite

Solid State Disk (Flash Chips)
We can overwrite individual pages

Operating System’s View

☐ Stale Data

Overwrite

Solid State Disk (Flash Chips)
We can overwrite individual pages

Operating System’s View

- Stale Data

Solid State Disk (Flash Chips)

Overwrite
The datasheet says we have to program pages in order though…
Our research has shown that it’s okay, with specific restrictions. We call this a “scrub”.

We call this a “scrub”.
Low density, high reliability SLC memory: No caveat.

MLC:
High Density MLC: We are limited by a “scrub budget”
Sanitizing single files with scrub

• When do we do it?
  – *Immediate*: Right away
  – *Background*: When we’re free
  – *Scan*: When we’re told to
Immediate & Background

- Automatically scrubs stale data from SSD
- Immediate
  - Maximum Security
  - Writes don’t complete until scrub is done
- Background
  - Good Security
  - Better performance, writes finish immediately
Harm. Mean of Financial, Software Devel., Patch, OLTP, Berkeley–DB, BTreeSwap
Harm. Mean of Financial, Software Devel., Patch, OLTP, Berkeley–DB, BTreeSwap
Harm. Mean of Financial, Software Devel., Patch, OLTP, Berkeley–DB, BTtreeSwap
Scan is what we wanted earlier: A built-in command to sanitize individual files.
In MLC, we still have to manage the scrub budget with copies.

Operating System’s View

Stale Data

Solid State Disk (Flash Chips)
Scrubbing

- The solution for single-file sanitization
- Sanitization level is selectable
- On-demand with scan mode
Conclusion

• Sanitizing storage media is essential for data security
• Need to verify sanitization effectiveness
  – Built-in mechanisms are reliable when implemented correctly
  – Hard-drive techniques don’t necessarily work
  – SAFE allows us to verify encrypted drives
• Sanitizing single files (in place) is difficult
  – Software overwrite cannot reliably sanitize
  – Scrubbing allows us to sanitize files by modifying the FTL