Self-Encrypting Deception:
Weaknesses in the Encryption of Solid State Drives (SSDs)

Carlo Meijer
Radboud University Nijmegen
Midnight Blue Labs

Bernard van Gastel
Radboud University Nijmegen
Open University of the Netherlands
whoami

Carlo Meijer

- PhD student at Radboud University Nijmegen
- Focused on analysis of crypto systems deployed in the wild
- Independent security researcher at Midnight Blue Labs

✉ c.meijer@cs.ru.nl
🌐 https://cs.ru.nl/~cmeijer/
🌐 https://midnightbluelabs.com/
Bernard van Gastel

• Assistant professor at Open University of the Netherlands
• Focused on analysis and correctness of systems

✉️ b.vangastel@cs.ru.nl
🌐 https://sustailblesoftware.info/
What is a Self-Encrypting Drive?
What is a Self-Encrypting Drive?

Traditional encryption (in software)
What is a Self-Encrypting Drive?

Traditional encryption (in software)

Self-Encrypting Drive
What is a Self-Encrypting Drive? (2)

Samsung 840 EVO mSATA SSD Specifications:

- Max capacity: 1TB
- Memory: 1GB LPDDR2 DRAM
- Controller: Samsung MEX (3x ARM Cortex R4 cores @400MHz)
- NAND: 19nm Samsung TLC
- Interface: SATA
- Form Factor: mSATA
- Power Consumption
  - Start-up: 2.01W
  - Idle: 0.44W
- Dimensions Height x length x Thickness: 3cm x 5cm x 3.85mm
- Weight: 8.5 grams
- Warranty: 3 year limited
What is a Self-Encrypting Drive? (2)

Samsung 840 EVO mSATA SSD Specifications:

- Max capacity: 1TB
- Memory: 1GB LPDDR2 DRAM
- Controller: Samsung MEX (3x ARM Cortex)
- NAND: 19nm Samsung TLC
- Interface: SATA
- Form Factor: mSATA
- Power Consumption
  - Start-up: 2.01W
  - Idle: 0.44W
- Dimensions Height x length x Thickness:
- Weight: 8.5 grams
- Warranty: 3 year limited

The Marvell 88SS9189 controller supports high-speed NAND flash interfaces up to 200MBps, channel and integrates a dual-core Marvell 88FR102 V5 CPU with shared DTCM and ITCM, can support up to eight NAND flash channels, ~500MBps sequential write performance, EPP and T10 CRC Checks.
What is a Self-Encrypting Drive? (2)

https://www.storagereview.com/samsung_840_evo_msata_ssd_review
https://www.custompcreview.com/reviews/crucial-mx100-512gb-ssd-review/
What is a Self-Encrypting Drive? (2)

https://www.storagereview.com/samsung_840_evo_msata_ssd_review
https://www.custompcreview.com/reviews/crucial-mx100-512gb-ssd-review/
Democratically proven

The best way to enhance data security:
Swap out vulnerable hard drives for self-encrypting SSDs

The best way to protect data stored on servers, desktops, or laptops is to encrypt it at the hardware level on a device’s storage drive. This is just one of many standard data security steps, but it’s critical — and often overlooked. The reason: New systems often come with low-grade, preinstalled hard drives, which often lack encryption technology. Or, if the hard drive offers encryption, it’s typically software-based, which is one of the weakest forms of encryption and may severely slow system performance, plus it’s also easier for hackers to attack. Here’s why.

The best way to enhance data security:

Swap out vulnerable hard drives for self-encrypting SSDs

The best way to protect data stored on servers, desktops, or laptops is to encrypt it at the hardware level on a device’s storage drive. This is just one of many standard data security steps, but it’s critical – and often overlooked. The reason: New systems often come with low-grade, preinstalled hard drives, which often lack encryption technology. Or, if the hard drive offers encryption, it’s typically software-based, which is one of the worst system performance, plus it’s also slow.

A study released a few months ago by TCG and the Ponemon Institute found that most IT professionals agree that hardware-based encryption is superior to software varieties at protecting data-at-rest. In fact, 70 percent of the respondents said that self-encrypting drives would have an enormous and positive impact on the protection of sensitive and confidential information in the event that a data breach should occur.


Democratically proven

Hardware based encryption is very secure; far more secure than any software-based offering. Software can be corrupted or negated, while hardware cannot.

Software runs under an operating system that is vulnerable to viruses and other attacks. An operating system, by definition, provides open access to applications and thus exposes these access points to improper use.

Hardware based security can more effectively restrict access from the outside, especially to unauthorized use. Additionally, dedicated hardware can have superior performance compared to software.

At the hardware level on a device's storage drive. This is just one of many elements data security steps, but it's critical— and often overlooked. The reason: New systems often come with low-grade, preinstalled hard drives, which often lack encryption technology. Or, if the hard drive offers encryption, it's typically software-based, which is one of the well-known slow system performance, plus it's also

A study released a few months ago by TCG and the Ponemon Institute found that most IT professionals agree that hardware based encryption is superior to software varieties at protecting data-at-rest. In fact, 70 percent of the respondents said that self encrypting drives would have an enormous and positive impact on the protection of sensitive and confidential information in the event that a data breach should occur.

https://trustedcomputinggroup.org/resource/self-encrypting-drives-sed-overview/
Democratically proven

Hardware based encryption is very secure; far more secure than any software-based offering. Software can be corrupted or negated, while hardware cannot.

Software runs under an operating system that is vulnerable to viruses and other attacks. An operating system, by definition, provides open access to applications and thus exposes these access points to improper use.

Self-encryption is superior to Software-based Solutions.

- **Transparency**: No system or application modifications required; encryption key generated in the factory by on-drive random number process; drive is always encrypting
- **Ease of management**: No encryption key to manage; software vendors exploit standardized interface to manage SEDs, including remote management, pre-boot authentication, and password recovery
- **Disposal or re-purposing cost**: With an SED, erase on-board encryption key
- **Re-encryption**: With SED, there is no need to ever re-encrypt the data
- **Performance**: No degradation in SED performance; hardware-based
- **Standardization**: Whole drive industry is building to the TCG/SED Specifications
- **Simplified**: No interference with upstream processes

Impact on the protection of sensitive and confidential information in the event that a data breach should occur.
Hardware based encryption is very secure; far more secure than any software-based offering. Software can be corrupted or negated, while hardware cannot. Software runs under an operating system that is vulnerable to viruses and other attacks. An operating system, by definition, provides open access to applications and thus exposes these access points to improper use. Self-encryption is superior to Software-based Solutions.

- Transparency: No system or application modifications required; encryption key generated in the factory by on-drive random number process; drive is always encrypting
- Ease of management: No encryption key to manage; software vendors exploit standardized interface to manage SEDs, including remote management, pre-boot authentication, and password recovery
- Disposal or re-purposing cost: With an SED, erase on-board encryption key
- Re-encryption: With SED, there is no need to ever re-encrypt the data
- Performance: No degradation in SED performance; hardware-based
- Standardization: Whole drive industry is building to the TCG/SED Specifications
- Simplified: No interference with upstream processes

Impact on the protection of sensitive and confidential information in the event that a data breach should occur.

BitLocker (built into Windows) opts for hardware encryption by default if available, software as a fall-back
Security guarantees of Self-Encrypting Drives
Security guarantees of Self-Encrypting Drives

Typical three attacker models for Full-Disk Encryption. All involve physical access by the attacker.
Security guarantees of Self-Encrypting Drives

Typical three attacker models for Full-Disk Encryption. All involve physical access by the attacker.

(i) **PC on**
Security guarantees of Self-Encrypting Drives

Typical three attacker models for Full-Disk Encryption. All involve physical access by the attacker.

(i) PC on
(ii) PC off, victim unaware:
    Physical encounter is not noticed by the victim
Security guarantees of Self-Encrypting Drives

Typical three attacker models for Full-Disk Encryption. All involve physical access by the attacker.

(i) **PC on**
(ii) **PC off, victim unaware:**
    Physical encounter is not noticed by the victim
(iii) **PC off, victim aware:**
    Drive is lost or stolen, machine considered “tainted”.
Security guarantees of Self-Encrypting Drives

Typical three attacker models for Full-Disk Encryption. All involve physical access.

(i) **PC on**
(ii) **PC off, victim unaware:**
    Physical encounter is not noticed by the victim
(iii) **PC off, victim aware:**
    Drive is lost or stolen, machine considered “tainted”.
Software encryption: secret key kept in RAM, which has weaknesses.

(i) Coldboot attack
Reboot, load custom OS, extract key from RAM

(ii) DMA attack
Extract key through DMA interface (PCI-e, Firewire, Thunderbolt, etc.)

Hardware encryption: immune in theory, however
• Key is kept in RAM for virtually all implementations to support Suspend-to-RAM (S3)
• Key is kept in storage controller (not secure hardware by any standard)
• Many have debugging interfaces exposed on PCB
• Adversary has physical access: can hot-plug the device

Overall: Attack opportunities are more or less equivalent
PC on

Software encryption: secret key kept in RAM, which has weaknesses.

(i) **Cold boot** attack

   Reboot, load custom OS, extract key from RAM
Software encryption: secret key kept in RAM, which has weaknesses.

(i) **Cold boot** attack
    Reboot, load custom OS, extract key from RAM

(ii) **DMA** attack
    Extract key through DMA interface (PCI-e, Firewire, Thunderbolt, etc.)
Software encryption: secret key kept in RAM, which has weaknesses.

(i) **Cold boot** attack
    Reboot, load custom OS, extract key from RAM

(ii) **DMA** attack
    Extract key through DMA interface (PCI-e, Firewire, Thunderbolt, etc.)

Hardware encryption: immune *in theory*, however
PC on

Software encryption: secret key kept in RAM, which has weaknesses.

(i) **Cold boot** attack
Reboot, load custom OS, extract key from RAM

(ii) **DMA** attack
Extract key through DMA interface (PCI-e, Firewire, Thunderbolt, etc.)

Hardware encryption: immune *in theory*, however
• Key **is** kept in RAM for virtually all implementations
  To support Suspend-to-RAM (S3)
Software encryption: secret key kept in RAM, which has weaknesses.

(i) **Cold boot** attack
Reboot, load custom OS, extract key from RAM

(ii) **DMA** attack
Extract key through DMA interface (PCI-e, Firewire, Thunderbolt, etc.)

Hardware encryption: immune *in theory*, however

- Key is kept in RAM for virtually all implementations
  To support Suspend-to-RAM (S3)
- Key is kept in storage controller (Not secure hardware by any standard)
PC on

Software encryption: secret key kept in RAM, which has weaknesses.

(i) Cold boot attack
    Reboot, load custom OS, extract key from RAM

(ii) DMA attack
    Extract key through DMA interface (PCI-e, Firewire, Thunderbolt, etc.)

Hardware encryption: immune in theory, however
- Key is kept in RAM for virtually all implementations
  To support Suspend-to-RAM (S3)
- Key is kept in storage controller (Not secure hardware by any standard)
  Many have debugging interfaces exposed on PCB
Software encryption: secret key kept in RAM, which has weaknesses.

(i) **Cold boot** attack

Reboot, load custom OS, extract key from RAM

(ii) **DMA** attack

Extract key through DMA interface (PCI-e, Firewire, Thunderbolt, etc.)

Hardware encryption: immune *in theory*, however

- Key **is** kept in RAM for virtually all implementations
  
  To support Suspend-to-RAM (S3)

- Key is kept in storage controller *(Not secure hardware by any standard)*
  
  Many have debugging interfaces exposed on PCB

- Adversary has physical access: can **hot-plug** the device
PC on

Software encryption: secret key kept in RAM, which has weaknesses.

(i) **Cold boot** attack
    Reboot, load custom OS, extract key from RAM

(ii) **DMA** attack
    Extract key through DMA interface (PCI-e, Firewire, Thunderbolt, etc.)

Hardware encryption: immune *in theory*, however

- Key **is** kept in RAM for virtually all implementations
  To support Suspend-to-RAM (S3)
- Key is kept in storage controller (Not secure hardware by any standard)
  Many have debugging interfaces exposed on PCB
- Adversary has physical access: can **hot-plug** the device

Overall: Attack opportunities are more or less equivalent
Security guarantees of Self-Encrypting Drives

Typical three attacker models for Full-Disk Encryption. All involve physical access.

(i) **PC on**

(ii) **PC off, victim unaware:**

   Physical encounter is not noticed by the victim

(iii) **PC off, victim aware:**

   Drive is lost or stolen, machine considered “tainted”.
PC off, victim unaware

Evil maid attack
PC off, victim unaware

Evil maid attack

(1) Install backdoor functionality
PC off, victim unaware

**Evil maid** attack

1. Install backdoor functionality
2. Wait for victim to enter secret key in the machine
PC off, victim unaware

**Evil maid** attack

1. Install backdoor functionality
2. Wait for victim to enter secret key in the machine
3. Exfiltrate data
PC off, victim unaware

**Evil maid** attack

1. Install backdoor functionality
2. Wait for victim to enter secret key in the machine
3. Exfiltrate data

Examples:
- Hardware keylogger

Overall: SEDs don’t offer added protection → equivalent
PC off, victim unaware

**Evil maid** attack

(1) Install backdoor functionality
(2) Wait for victim to enter secret key in the machine
(3) Exfiltrate data

Examples:
- Hardware keylogger
- Backdoor in boot loader

Overall: SED don't offer added protection → equivalent
PC off, victim unaware

**Evil maid** attack

1. Install backdoor functionality
2. Wait for victim to enter secret key in the machine
3. Exfiltrate data

Examples:
- Hardware keylogger
- Backdoor in boot loader

Overall: SEDs don’t offer added protection → equivalent
Security guarantees of Self-Encrypting Drives

Typical three attacker models for Full-Disk Encryption. All involve physical access.

(i) **PC on**
(ii) **PC off, victim unaware:** Physical encounter is not noticed by the victim
(iii) **PC off, victim aware:** Drive is lost or stolen, machine considered “tainted”.
PC off, victim aware

Software encryption provides full confidentiality of the data
(given that the implementation is sound)
PC off, victim aware

Software encryption provides full confidentiality of the data
(given that the implementation is sound)

Options:
• Open source (audited) software
PC off, victim aware

Software encryption provides full confidentiality of the data (given that the implementation is sound)

Options:
• Open source (audited) software
• Proprietary software with public implementation details

• Extremely hard to audit
• Additional pitfalls that apply particularly to hardware (later)
PC off, victim aware

Software encryption provides full confidentiality of the data (given that the implementation is sound)

Options:
• Open source (audited) software
• Proprietary software with public implementation details
• Proprietary (black-box) implementation

With hardware encryption, no other option than the black-box
• Extremely hard to audit
• Additional pitfalls that apply particularly to hardware (later)
PC off, victim aware

Software encryption provides full confidentiality of the data
(given that the implementation is sound)

Options:
• Open source (audited) software
• Proprietary software with public implementation details
• Proprietary (black-box) implementation

With hardware encryption, no other option than the black-box
PC off, victim aware

Software encryption provides full confidentiality of the data (given that the implementation is sound)

Options:
  • Open source (audited) software
  • Proprietary software with public implementation details
  • Proprietary (black-box) implementation

With hardware encryption, no other option than the black-box
  • Extremely hard to audit
PC off, victim aware

Software encryption provides full confidentiality of the data (given that the implementation is sound)

Options:
• Open source (audited) software
• Proprietary software with public implementation details
• Proprietary (black-box) implementation

With hardware encryption, no other option than the black-box
• Extremely hard to audit
• Additional pitfalls that apply particularly to hardware (later)
Security guarantees of Self-Encrypting Drives

Typical three attacker models for Full-Disk Encryption. All involve physical access.

(i) PC on
(ii) PC off, victim unaware:
    Physical encounter is not noticed by the victim
(iii) PC off, victim aware:
    Drive is lost or stolen, machine considered “tainted”
Security guarantees of Self-Encrypting Drives

Typical three attacker models for Full-Disk Encryption. All involve physical access.

(i) PC on

(ii) PC off, victim unaware:
Physical encounter is not noticed by the victim

(iii) PC off, victim aware:
Drive is lost or stolen, machine considered “tainted”

Thus, security guarantees are equivalent. At best.
Standards for Self-Encrypting Drives
Standards for Self-Encrypting Drives

Two widely used standards exist

(i) **ATA Security Feature Set**
   Originally designed for access control only

https://medium.com/@andrewpgsweeny/beyond-the-red-pill-and-the-blue-pill-9ef953d6e133
Standards for Self-Encrypting Drives

Two widely used standards exist

(i) **ATA Security Feature Set**
    Originally designed for access control only

(ii) **TCG Opal**
    Modern standard designed specifically for SEDs

https://medium.com/@andrewpgsweeny/
beyond-the-red-pill-and-the-blue-pill-9ef953d6e133
Suppose you would implement this yourself

It would probably look something like this

**Stored data**

- User-supplied password
- Keyed hash
- Salt\#1
- Salt\#2
- Hash output
- Hash result
- Compare
- Match/no match

Keyed hash

Sofar, easy
Suppose you would implement this yourself

It would probably look something like this:

- **User-supplied password**
- **Keyed hash**
- **Salt#1** → **Hash result** → **Compare** → **Match/no match**
- **Salt#2**
- **Hash output**
- **DEK**
Suppose you would implement this yourself

It would probably look something like this

So far, easy
Standards for Self-Encrypting Drives

Two widely used standards exist:

(i) **ATA Security Feature Set**
    Originally designed for access control only

(ii) **TCG Opal**
    Modern standard designed specifically for SEDs

https://medium.com/@andrewpgsweeny/
beyond-the-red-pill-and-the-blue-pill-9ef953d6e133
ATA Security feature set

- Originated in the pre-SED era
  
  Thus, “encryption” is not even mentioned in the spec

- Two password types: User, Master
  
  Both are user-settable; initial master password factory set

- Master: High (0), Maximum (1)
  
  - High: both User and Master password unlock drive
  - Maximum: Only User unlocks drive, Master may erase

- Bottomline: Always change the Master password or set to Maximum
  
  In practice, even this is almost always insufficient (later)
ATA Security feature set

• Originated in the pre-SED era
  Thus, “encryption” is not even mentioned in the spec
• Two password types: User, Master

Bottomline: Always change the Master password or set to Maximum
In practice, eventhisis almost always insufficient (later)
ATA Security feature set

- Originated in the pre-SED era
  
  Thus, “encryption” is not even mentioned in the spec

- Two password types: *User, Master*

- Both are user-settable, initial master password factory set
ATA Security feature set

• Originated in the pre-SED era
  Thus, “encryption” is not even mentioned in the spec
• Two password types: User, Master
• Both are user-settable, initial master password factory set
• MASTER PASSWORD CAPABILITY: High (0), Maximum (1)

Bottomline: Always change the Master password or set to Maximum.
In practice, even this is almost always insufficient (later).
ATA Security feature set

• Originated in the pre-SED era
  Thus, “encryption” is not even mentioned in the spec
• Two password types: User, Master
• Both are user-settable, initial master password factory set
• MASTER PASSWORD CAPABILITY: High (0), Maximum (1)
  • High: both User and Master password unlock drive

Bottomline: Always change the Master password or set to Maximum
In practice, eventhis is almost always insufficient (later)
ATA Security feature set

• Originated in the pre-SED era
  Thus, “encryption” is not even mentioned in the spec
• Two password types: User, Master
• Both are user-settable, initial master password factory set
• MASTER PASSWORD CAPABILITY: High (0), Maximum (1)
  • High: both User and Master password unlock drive
  • Maximum: Only User unlocks drive, Master may erase

Bottomline: Always change the Master password or set to Maximum
In practice, even this is almost always insufficient (later)
ATA Security feature set

- Originated in the pre-SED era
  Thus, “encryption” is not even mentioned in the spec
- Two password types: *User, Master*
- Both are user-settable, initial master password factory set
- MASTER PASSWORD CAPABILITY: *High* (0), *Maximum* (1)
  - **High**: both User and Master password unlock drive
  - **Maximum**: Only User unlocks drive, Master may erase
- Bottom line: *Always* change the Master password or set to Maximum
ATA Security feature set

- Originated in the pre-SED era
  
  Thus, “encryption” is not even mentioned in the spec

- Two password types: *User, Master*

- Both are user-settable, initial master password factory set

- MASTER PASSWORD CAPABILITY: *High* (0), *Maximum* (1)
  
  - **High**: both User and Master password unlock drive
  - **Maximum**: Only User unlocks drive, Master may erase

- Bottom line: **Always** change the Master password or set to Maximum

In practice, even this is almost always insufficient (later)
ATA security feature set

Stored data

Master password:
- Salt\#1
- Salt\#2
- Hash output
- KEK

User password:
- Salt\#1
- Salt\#2
- Hash output
- KEK

User-supplied User password

Keyed hash

Hash result

Compare

Match/no match

Key

Decrypt

Shared key

Decrypt

DEK
ATA security feature set

Stored data

Master password: Salt\#1 Salt\#2 Hash output KEK

User password: Salt\#1 Salt\#2 Hash output KEK

User-supplied User password

Keyed hash → Hash result → Compare → Match/no match

Keyed hash → Key → Decrypt → Shared key → Decrypt → DEK
Standards for Self-Encrypting Drives

Two widely used standards exist

(i) **ATA Security Feature Set**
    Originally designed for access control only

(ii) **TCG Opal**
    Modern standard designed specifically for SEDs

https://medium.com/@andrewpgsweeny/beyond-the-red-pill-and-the-blue-pill-9ef953d6e133
TCG Opal

- De facto standard for hardware full-disk encryption
TCG Opal

- De facto standard for hardware full-disk encryption
- Multiple partitions (locking ranges)

<table>
<thead>
<tr>
<th>Range1</th>
<th>Range2</th>
<th>Range3</th>
<th>Range4</th>
<th>Drivespace</th>
<th>Passwords</th>
<th>Password1</th>
<th>Password2</th>
<th>Password3</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>
TCG Opal

- De facto standard for hardware full-disk encryption
- Multiple partitions (*locking ranges*)
- Multiple passwords (*credentials*)
TCG Opal

- De facto standard for hardware full-disk encryption
- Multiple partitions (locking ranges)
- Multiple passwords (credentials)
- Single credential can unlock multiple ranges
TCG Opal

- De facto standard for hardware full-disk encryption
- Multiple partitions (locking ranges)
- Multiple passwords (credentials)
- Single credential can unlock multiple ranges
- Single range can be unlocked by multiple credentials
TCG Opal

- De facto standard for hardware full-disk encryption
- Multiple partitions (locking ranges)
- Multiple passwords (credentials)
- Single credential can unlock multiple ranges
- Single range can be unlocked by multiple credentials
- i.e. many-to-many

![Diagram of TCG Opal](image.png)
TCG Opal

- De facto standard for hardware full-disk encryption
- Multiple partitions (locking ranges)
- Multiple passwords (credentials)
- Single credential can unlock multiple ranges
- Single range can be unlocked by multiple credentials
- i.e. many-to-many
- “Scramble” (i.e. re-generate key) range independently of others
TCG Opal

- De facto standard for hardware full-disk encryption
- Multiple partitions (*locking ranges*)
- Multiple passwords (*credentials*)
- Single credential can unlock multiple ranges
- Single range can be unlocked by multiple credentials
- i.e. **many-to-many**
- “Scramble” (i.e. re-generate key) range independently of others
- Fully trusted by BitLocker
Pitfalls
Pitfall 1: DEK not derived from password

Password unlocks drive and DEK is used to encrypt data. How they are related is unknown. They might not be related at all.
Pitfall 1: DEK not derived from password

- Password unlocks drive and DEK is used to encrypt data
Pitfall 1: DEK not derived from password

- Password unlocks drive and DEK is used to encrypt data
- How they are related is unknown
Pitfall 1: DEK not derived from password

- Password unlocks drive and DEK is used to encrypt data
- How they are related is unknown
- They might not be related at all
Pitfall 2: Single DEK for entire drive

- Strong Password 1
  - Encrypted DEK 1
    - Decrypt
- Strong Password 2
  - Encrypted DEK 2
    - Decrypt
- Weak Password 3
  - Encrypted DEK 3
    - Decrypt

Weakest password will grant access to all ranges event or ranges for which no permission is granted.

- No cryptographic enforcement, but if-statements.
- BitLocker leaves an Opal range unprotected (partition table).
  → Thus, in this case, DEK is recoverable without a password.
Pitfall 2: Single DEK for entire drive

- **Weakest** password will grant access to all ranges
  
  Even to ranges for which no permission is granted
Pitfall 2: Single DEK for entire drive

- **Weakest** password will grant access to all ranges
  Even to ranges for which no permission is granted
- No cryptographic enforcement, but **if-statements**
Pitfall 2: Single DEK for entire drive

- **Weakest** password will grant access to all ranges
  - Even to ranges for which no permission is granted
- No cryptographic enforcement, but **if-statements**
- BitLocker leaves an Opal range unprotected (partition table)
Pitfall 2: Single DEK for entire drive

- **Weakest** password will grant access to all ranges
  Even to ranges for which no permission is granted
- No cryptographic enforcement, but **if-statements**
- BitLocker leaves an Opal range unprotected (partition table)
  → Thus, in this case, DEK is recoverable **without** a password
Pitfall 3: ATA Master password re-enable

Stored data

Master password:
- Salt#1
- Salt#2
- Hash output
- KEK

User password:
- Salt#1
- Salt#2
- Hash output
- KEK

Recall: You should set the MSTRAPASSWORD to Max.

Ideally, this erases key material.

However, the standard allows resetting it to High, using only the user password.

In practice, key material remains stored. If unchanged, factory default master password allows data to be recovered.
Pitfall 3: ATA Master password re-enable

Stored data

Master password:
- Salt#1
- Salt#2
- Hash output
- KEK

User password:
- Salt#1
- Salt#2
- Hash output
- KEK

- Recall: You should set the MASTER PASSWORD CAPABILITY to Max
Pitfall 3: ATA Master password re-enable

- Recall: You should set the MASTER PASSWORD CAPABILITY to Max
- Ideally, this erases key material
Pitfall 3: ATA Master password re-enable

Recall: You should set the MASTER PASSWORD CAPABILITY to Max
Ideally, this erases key material
However, the standard allows resetting it to High, using only the user password
Pitfall 3: ATA Master password re-enable

- Recall: You should set the master password capability to Max.
- Ideally, this erases key material.
- However, the standard allows resetting it to High, using only the user password.
- In practice, key material remains stored. If unchanged, the factory default master password allows data to be recovered.
Pitfall 4: Wear Leveling

Multiple writes to the same logical sector trigger writes to different physical sectors.

Usersetspassword

• Setpassword → overwrite of unprotected DEK with encrypted variant
• Unprotected DEK may still be present in physical flash
Pitfall 4: Wear Leveling

Multiple writes to the \textit{same} logical sector trigger writes to \textit{different} physical sectors

- Plaintext DEK before
- User sets password
- Plaintext DEK and Encrypted DEK after

Unprotected DEK may still be present in physical flash.
Pitfall 4: Wear Leveling

Multiple writes to the *same* logical sector trigger writes to *different* physical sectors

- Set password → overwrite of unprotected DEK with encrypted variant
Pitfall 4: Wear Leveling

Multiple writes to the *same* logical sector trigger writes to *different* physical sectors

- Set password $\rightarrow$ overwrite of unprotected DEK with encrypted variant
- Unprotected DEK may still be present in physical flash
Other pitfalls

• Random entropy generation
Other pitfalls

- Random entropy generation
- Power-saving mode: DEVSLP
Other pitfalls

- Random entropy generation
- Power-saving mode: DEVSLP

Drive may dump its RAM incl. crypto keys to non-volatile memory, and shut off the RAM.
Other pitfalls

• Random entropy generation
• Power-saving mode: DEVSLP

   Drive may dump its RAM incl. crypto keys to non-volatile memory, and shut off the RAM.

• General implementation issues
Other pitfalls

- Random entropy generation
- Power-saving mode: DEVSLP
  Drive may dump its RAM incl. crypto keys to non-volatile memory, and shut off the RAM.
- General implementation issues
  Mode of operation (ECB, CBC, CTR, XTS), Side channels, Key derivation, etc.
Methodology
Methodology

General approach
Methodology

General approach

(i) Obtain a firmware image
Methodology

General approach

(i) Obtain a firmware image
(ii) Gain low level control over the device
Methodology

General approach

(i) Obtain a firmware image
(ii) Gain low level control over the device
(iii) Analyze the firmware
Methodology

General approach

(i) Obtain a firmware image
(ii) Gain low level control over the device
(iii) Analyze the firmware
Obtain a firmware image

(i) Download it  (harder than it seems)
Obtain a firmware image

(i) Download it  (harder than it seems)
    · There’s usually obfuscation applied
Obtain a firmware image

(i) Download it (harder than it seems)
   - There’s usually obfuscation applied
   - Capture SSL traffic, reverse engineer, etc.

```
dword_10222A58 = sub_1003E390();
v131 = 0;
v130 = 1;
v129 = 0;
*(BYTE *)sub_10020920(v1, v0, &v129) = 77; // M
v129 = 1;
*(BYTE *)sub_10020920(v2, v2, &v129) = 54;
v129 = 2;
*(BYTE *)sub_10020920(v5, v4, &v129) = 97; // a
v129 = 3;
*(BYTE *)sub_10020920(v7, v6, &v129) = 56;
v129 = 4;
*(BYTE *)sub_10020920(v9, v8, &v129) = 103; // g
v129 = 5;
*(BYTE *)sub_10020920(v11, v10, &v129) = 51;
v129 = 6;
*(BYTE *)sub_10020920(v13, v12, &v129) = 105; // i
v129 = 7;
*(BYTE *)sub_10020920(v15, v14, &v129) = 37;
v129 = 8;
*(BYTE *)sub_10020920(v17, v16, &v129) = 99; // c
v129 = 9;
*(BYTE *)sub_10020920(v19, v18, &v129) = 50;
v129 = 10;
*(BYTE *)sub_10020920(v21, v20, &v129) = 105; // i
v129 = 11;
*(BYTE *)sub_10020920(v23, v22, &v129) = 33;
v129 = 12;
*(BYTE *)sub_10020920(v25, v24, &v129) = 97; // a
v129 = 13;
*(BYTE *)sub_10020920(v27, v26, &v129) = 122;
v129 = 14;
*(BYTE *)sub_10020920(v29, v28, &v129) = 110; // n
v129 = 15;
```

Decomposition of Samsung Magician tool
Decompile Samsung Magician tool

(i) Download it  (harder than it seems)
   · There’s usually obfuscation applied
   · Capture SSL traffic, reverse engineer, etc.
   · Image may be encrypted, decryption by the unit itself → dead end
Obtain a firmware image

(i) Download it (harder than it seems)
   - There’s usually obfuscation applied
   - Capture SSL traffic, reverse engineer, etc.
   - Image may be encrypted, decryption by the unit itself → dead end

(ii) Pull the firmware from RAM through JTAG (next)

Decompilation of Samsung Magician tool

dword_10222A58 = sub_1003E390();
  v131 = 0;
  v130 = 1;
  v129 = 0;
  *(BYTE *)sub_10020920(v1, v0, &v129) = 77; // M
  v129 = 1;
  *(BYTE *)sub_10020920(v3, v2, &v129) = 54;
  v129 = 2;
  *(BYTE *)sub_10020920(v5, v4, &v129) = 97; // a
  v129 = 3;
  *(BYTE *)sub_10020920(v7, v6, &v129) = 56;
  v129 = 4;
  *(BYTE *)sub_10020920(v9, v8, &v129) = 103; // g
  v129 = 5;
  *(BYTE *)sub_10020920(v11, v10, &v129) = 51;
  v129 = 6;
  *(BYTE *)sub_10020920(v13, v12, &v129) = 105; // i
  v129 = 7;
  *(BYTE *)sub_10020920(v15, v14, &v129) = 37;
  v129 = 8;
  *(BYTE *)sub_10020920(v17, v16, &v129) = 99; // c
  v129 = 9;
  *(BYTE *)sub_10020920(v19, v18, &v129) = 50;
  v129 = 10;
  *(BYTE *)sub_10020920(v21, v20, &v129) = 105; // i
  v129 = 11;
  *(BYTE *)sub_10020920(v23, v22, &v129) = 33;
  v129 = 12;
  *(BYTE *)sub_10020920(v25, v24, &v129) = 97; // a
  v129 = 13;
  *(BYTE *)sub_10020920(v27, v26, &v129) = 122;
  v129 = 14;
  *(BYTE *)sub_10020920(v29, v28, &v129) = 110; // n
  v129 = 15;
Methodology

General approach

(i) Obtain a firmware image
(ii) Gain low level control over the device
(iii) Analyze the firmware
Gaining low level control

More or less equal capabilities:

(i) **JTAG** (allows you to halt the CPU, get/set registers, read/write in the address space, etc.)
Gaining low level control

More or less equal capabilities:

(i) JTAG (allows you to halt the CPU, get/set registers, read/write in the address space, etc.)
   - Some models have it in plain sight

JTAG pins on the Crucial MX100.
Gaining low level control

More or less equal capabilities:

(i) JTAG (allows you to halt the CPU, get/set registers, read/write in the address space, etc.)
   - Some models have it in plain sight
   - Others need some figuring out
Gaining low level control

More or less equal capabilities:

(i) JTAG (allows you to halt the CPU, get/set registers, read/write in the address space, etc.)
   · Some models have it in plain sight
   · Others need some figuring out

(ii) Obtain unsigned code execution

JTAGulator

JTAG pins on the Crucial MX100.
Gaining low level control

More or less equal capabilities:

(i) JTAG (allows you to halt the CPU, get/set registers, read/write in the address space, etc.)
   - Some models have it in plain sight
   - Others need some figuring out

(ii) Obtain unsigned code execution
   - Find an undocumented command that allows this
Gaining low level control

More or less equal capabilities:

(i) **JTAG** (allows you to halt the CPU, get/set registers, read/write in the address space, etc.)
   - Some models have it in plain sight
   - Others need some figuring out

(ii) Obtain unsigned code execution
    - Find an undocumented command that allows this
    - Exploit a vulnerability

JTAG pins on the Crucial MX100.
Gaining low level control

More or less equal capabilities:

(i) **JTAG** (allows you to halt the CPU, get/set registers, read/write in the address space, etc.)
   - Some models have it in plain sight
   - Others need some figuring out

(ii) Obtain unsigned code execution
   - Find an undocumented command that allows this
   - Exploit a vulnerability
   - Modify code stored on memory chips

ARM14 JTAG
JTAG pins on the Crucial MX100.

JTAGulator
Gaining low level control

More or less equal capabilities:

(i) **JTAG** (allows you to halt the CPU, get/set registers, read/write in the address space, etc.)
   - Some models have it in plain sight
   - Others need some figuring out

(ii) Obtain unsigned code execution
   - Find an undocumented command that allows this
   - Exploit a vulnerability
   - Modify code stored on memory chips
   - Bypass cryptographic signatures with fault injection

JTAGulator

JTAG pins on the Crucial MX100.
Methodology

General approach

(i) Obtain a firmware image
(ii) Gain low level control over the device
(iii) Analyze the firmware
Analyze the firmware

(i) Figure out the section information
Analyze the firmware

Parsed header of MX300 FW image

(i) Figure out the section information
   · From image header
Analyze the firmware

**Parsed header of MX300 FW image**

(i) Figure out the section information
   - From image header

(ii) Load the image into a disassembler
   (We used IDA Pro for this purpose)
Analyse the firmware

Parsed header of MX300 FW image

(i) Figure out the section information
   · From image header

(ii) Load the image into a disassembler
    (We used IDA Pro for this purpose)

(iii) Figure out what the firmware does
Analyze the firmware

Parsed header of MX300 FW image

(i) Figure out the section information
   - From image header

(ii) Load the image into a disassembler
    (We used IDA Pro for this purpose)

(iii) Figure out what the firmware does
    - Try to find the ATA dispatch table

ATA Command Dispatch table in firmware

ATA specification
Analyze the firmware

Parsed header of MX300 FW image

(i) Figure out the section information
   · From image header

(ii) Load the image into a disassembler
    (We used IDA Pro for this purpose)

(iii) Figure out what the firmware does
    · Try to find the ATA dispatch table
    · Look through functions with interesting opcodes

ATA Command dispatch table in firmware

ATA specification
Results
## Results

- Models studied released in 2014-2018

<table>
<thead>
<tr>
<th>Drive</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>Impact</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crucial MX100 (all)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Compromised</td>
</tr>
<tr>
<td>Crucial MX200 (all)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Compromised</td>
</tr>
<tr>
<td>Crucial MX300 (all)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Compromised</td>
</tr>
<tr>
<td>Sandisk X600 (SATA)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>Probably compromised</td>
</tr>
<tr>
<td>Samsung 840 EVO (SATA)</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>Depends</td>
</tr>
<tr>
<td>Samsung 850 EVO (SATA)</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>Depends</td>
</tr>
<tr>
<td>Samsung 950 PRO (NVMe)</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>Probably safe</td>
</tr>
<tr>
<td>Samsung T3 (USB)</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>Compromised</td>
</tr>
<tr>
<td>Samsung T5 (USB)</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>Compromised</td>
</tr>
</tbody>
</table>

1. Derivation of the DEK from the password in ATA Security (High mode)
2. Derivation of the DEK from the password in ATA Security (Max mode)
3. Derivation of the DEK from the password in TCG Opal
4. Derivation of the DEK from the password in proprietary standard
5. No single key for entire disk
6. Not vulnerable to ATA Master password re-enabling (only if derivation is present)
7. Randomized DEK on sanitize and sufficient random entropy
8. No wear leveling related issues
9. No DEVSLP related issues
# Results

- Models studied released in 2014-2018
- Different form factors
  - SATA, NVMe, USB

<table>
<thead>
<tr>
<th>Drive</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>Impact</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crucial MX100 (all)</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>Compromised</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Crucial MX200 (all)</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>Compromised</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Crucial MX300 (all)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>Compromised</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sandisk X600 (SATA)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td>X</td>
<td>✓</td>
<td>X</td>
<td>Probably compromised</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Samsung 840 EVO (SATA)</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>Depends</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Samsung 850 EVO (SATA)</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>Depends</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Samsung 950 PRO (NVMe)</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>Probably safe</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Samsung T3 (USB)</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>Compromised</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Samsung T5 (USB)</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>Compromised</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. Derivation of the DEK from the password in ATA Security (High mode)
2. Derivation of the DEK from the password in ATA Security (Max mode)
3. Derivation of the DEK from the password in TCG Opal
4. Derivation of the DEK from the password in proprietary standard
5. No single key for entire disk
6. Not vulnerable to ATA Master password re-enabling (only if derivation is present)
7. Randomized DEK on sanitize and sufficient random entropy
8. No wear leveling related issues
9. No DEVSLP related issues
Results

- Models studied released in 2014-2018
- Different form factors SATA, NVMe, USB
- Most have severe weaknesses

| Drive                  | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  | Impact        |
|------------------------|----|----|----|----|----|----|----|----|----|              |
| Crucial MX100 (all)    | ✓  | ✓  | ✓  | ✓  | ✓  | ✓  | ✓  | ✓  |   |   | Compromised  |
| Crucial MX200 (all)    | ✓  | ✓  | ✓  | ✓  | ✓  | ✓  | ✓  | ✓  |   |   | Compromised  |
| Crucial MX300 (all)    | ✓  | ✓  | ✓  | ✓  | ✓  | ✓  | ✓  | ✓  |   |   | Compromised  |
| Sandisk X600 (SATA)    | ✓  | ✓  | ✓  | ✓  | ✓  | ✓  |   |   |   | Probably compromised |
| Samsung 840 EVO (SATA) | ✓  | ✓  | ✓  | ✓  | ✓  | ✓  |   |   |   | Depends       |
| Samsung 850 EVO (SATA) | ✓  | ✓  | ✓  | ✓  | ✓  | ✓  |   |   |   | Depends       |
| Samsung 950 EVO (NVMe) | ✓  | ✓  | ✓  | ✓  | ✓  | ✓  |   |   |   | Probably safe |
| Samsung T3 (USB)       | ✓  | ✓  | ✓  | ✓  | ✓  | ✓  |   |   |   | Compromised   |
| Samsung T5 (USB)       | ✓  | ✓  | ✓  | ✓  | ✓  | ✓  |   |   |   | Compromised   |

1. Derivation of the DEK from the password in ATA Security (High mode)
2. Derivation of the DEK from the password in ATA Security (Max mode)
3. Derivation of the DEK from the password in TCG Opal
4. Derivation of the DEK from the password in proprietary standard
5. No single key for entire disk
6. Not vulnerable to ATA Master password re-enabling (only if derivation is present)
7. Randomized DEK on sanitize and sufficient random entropy
8. No wear leveling related issues
9. No DEVSLP related issues
Results

- Models studied released in 2014-2018
- Different form factors: SATA, NVMe, USB
- Most have severe weaknesses
- Best case scenario: security guarantees are equivalent to software FDE

<table>
<thead>
<tr>
<th>Drive</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>Impact</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crucial MX100 (all)</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Compromised</td>
</tr>
<tr>
<td>Crucial MX200 (all)</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Compromised</td>
</tr>
<tr>
<td>Crucial MX300 (all)</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td>X</td>
<td>X</td>
<td>✓</td>
<td></td>
<td></td>
<td>Compromised</td>
</tr>
<tr>
<td>Sandisk X600 (SATA)</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td>X</td>
<td>X</td>
<td>✓</td>
<td></td>
<td></td>
<td>Probably compromised</td>
</tr>
<tr>
<td>Samsung 840 EVO (SATA)</td>
<td>X</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td>Depends</td>
</tr>
<tr>
<td>Samsung 850 EVO (SATA)</td>
<td>X</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td>Depends</td>
</tr>
<tr>
<td>Samsung 950 PRO (NVMe)</td>
<td>X</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td>Probably safe</td>
</tr>
<tr>
<td>Samsung T3 (USB)</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Compromised</td>
</tr>
<tr>
<td>Samsung T5 (USB)</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Compromised</td>
</tr>
</tbody>
</table>

1. Derivation of the DEK from the password in ATA Security (High mode)
2. Derivation of the DEK from the password in ATA Security (Max mode)
3. Derivation of the DEK from the password in TCG Opal
4. Derivation of the DEK from the password in proprietary standard
5. No single key for entire disk
6. Not vulnerable to ATA Master password re-enabling (only if derivation is present)
7. Randomized DEK on sanitize and sufficient random entropy
8. No wear leveling related issues
9. No DEVSLP related issues
Results

- Models studied released in 2014-2018
- Different form factors
  SATA, NVMe, USB
- Most have severe weaknesses
- Best case scenario: security guarantees are equivalent to software FDE
- Worst case: confidentiality relies on an **if-statement**

<table>
<thead>
<tr>
<th>Drive</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>Impact</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crucial MX100</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>Compromised</td>
</tr>
<tr>
<td>(all)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Crucial MX200</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>Compromised</td>
</tr>
<tr>
<td>(all)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Crucial MX300</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>Compromised</td>
</tr>
<tr>
<td>(all)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Sandisk X600</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>Probably compromised</td>
</tr>
<tr>
<td>(SATA)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Samsung 840 EVO (SATA)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>Depends</td>
</tr>
<tr>
<td>Samsung 850 EVO (SATA)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>Depends</td>
</tr>
<tr>
<td>Samsung 950 PRO (NVMe)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>Probably safe</td>
</tr>
<tr>
<td>Samsung T3 (USB)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>Compromised</td>
</tr>
<tr>
<td>Samsung T5 (USB)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>Compromised</td>
</tr>
</tbody>
</table>

1 Derivation of the DEK from the password in ATA Security (High mode)
2 Derivation of the DEK from the password in ATA Security (Max mode)
3 Derivation of the DEK from the password in TCG Opal
4 Derivation of the DEK from the password in proprietary standard
5 No single key for entire disk
6 Not vulnerable to ATA Master password re-enabling (only if derivation is present)
7 Randomized DEK on sanitize and sufficient random entropy
8 No wear leveling related issues
9 No DEVSLP related issues
Results

• Models studied released in 2014-2018
• Different form factors SATA, NVMe, USB
• Most have severe weaknesses
• Best case scenario: security guarantees are equivalent to software FDE
• Worst case: confidentiality relies on an **if-statement**
• BitLocker delegating trust amplifies the issue

<table>
<thead>
<tr>
<th>Drive</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>Impact</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crucial MX100 (all)</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>Compromised</td>
</tr>
<tr>
<td>Crucial MX200 (all)</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Compromised</td>
</tr>
<tr>
<td>Crucial MX300 (all)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Compromised</td>
</tr>
<tr>
<td>Sandisk X600 (SATA)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Probably compromised</td>
</tr>
<tr>
<td>Samsung 840 EVO (SATA)</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Depends</td>
</tr>
<tr>
<td>Samsung 850 EVO (SATA)</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Depends</td>
</tr>
<tr>
<td>Samsung 950 PRO (NVMe)</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Probably safe</td>
</tr>
<tr>
<td>Samsung T3 (USB)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>Compromised</td>
</tr>
<tr>
<td>Samsung T5 (USB)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>Compromised</td>
</tr>
</tbody>
</table>

1. Derivation of the DEK from the password in ATA Security (High mode)
2. Derivation of the DEK from the password in ATA Security (Max mode)
3. Derivation of the DEK from the password in TCG Opal
4. Derivation of the DEK from the password in proprietary standard
5. No single key for entire disk
6. Not vulnerable to ATA Master password re-enabling (only if derivation is present)
7. Randomized DEK on sanitize and sufficient random entropy
8. No wear leveling related issues
9. No DEVP related issues
Results

- TCG Opal is terrible

<table>
<thead>
<tr>
<th>Drive</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>Impact</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crucial MX100 (all)</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>Compromised</td>
</tr>
<tr>
<td>Crucial MX200 (all)</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>Compromised</td>
</tr>
<tr>
<td>Crucial MX300 (all)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
<td>x</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>Compromised</td>
</tr>
<tr>
<td>Sandisk X600 (SATA)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
<td>x</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>Probably compromised</td>
</tr>
<tr>
<td>Samsung 840 EVO (SATA)</td>
<td>x</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>Depends</td>
</tr>
<tr>
<td>Samsung 850 EVO (SATA)</td>
<td>x</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>Depends</td>
</tr>
<tr>
<td>Samsung 950 PRO (NVMe)</td>
<td>x</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>Probably safe</td>
</tr>
<tr>
<td>Samsung T3 (USB)</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>Compromised</td>
</tr>
<tr>
<td>Samsung T5 (USB)</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>Compromised</td>
</tr>
</tbody>
</table>

1 Derivation of the DEK from the password in ATA Security (High mode)
2 Derivation of the DEK from the password in ATA Security (Max mode)
3 Derivation of the DEK from the password in TCG Opal
4 Derivation of the DEK from the password in proprietary standard
5 No single key for entire disk
6 Not vulnerable to ATA Master password re-enabling (only if derivation is present)
7 Randomized DEK on sanitize and sufficient random entropy
8 No wear leveling related issues
9 No DEVLSP related issues
Results

- TCG Opal is terrible
  - Over-engineered

<table>
<thead>
<tr>
<th>Drive</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>Impact</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crucial MX100 (all)</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td>Compromised</td>
</tr>
<tr>
<td>Crucial MX200 (all)</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td>Compromised</td>
</tr>
<tr>
<td>Crucial MX300 (all)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td>Compromised</td>
</tr>
<tr>
<td>Sandisk X600 (SATA)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td>Probably compromised</td>
</tr>
<tr>
<td>Samsung 840 EVO (SATA)</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Depends</td>
</tr>
<tr>
<td>Samsung 850 EVO (SATA)</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Depends</td>
</tr>
<tr>
<td>Samsung 950 PRO (NVMe)</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Probably safe</td>
</tr>
<tr>
<td>Samsung T3 (USB)</td>
<td></td>
<td></td>
<td></td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td>Compromised</td>
</tr>
<tr>
<td>Samsung T5 (USB)</td>
<td></td>
<td></td>
<td></td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td>Compromised</td>
</tr>
</tbody>
</table>

1. Derivation of the DEK from the password in ATA Security (High mode)
2. Derivation of the DEK from the password in ATA Security (Max mode)
3. Derivation of the DEK from the password in TCG Opal
4. Derivation of the DEK from the password in proprietary standard
5. No single key for entire disk
6. Not vulnerable to ATA Master password re-enabling (only if derivation is present)
7. Randomized DEK on sanitize and sufficient random entropy
8. No wear leveling related issues
9. No DEVSLP related issues
Results

- TCG Opal is terrible
  - Over-engineered
  - Security goals not clear

<table>
<thead>
<tr>
<th>Drive</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>Impact</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crucial MX100 (all)</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>Compromised</td>
</tr>
<tr>
<td>Crucial MX200 (all)</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>Compromised</td>
</tr>
<tr>
<td>Crucial MX300 (all)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>Compromised</td>
</tr>
<tr>
<td>Sandisk X600 (SATA)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>Probably compromised</td>
</tr>
<tr>
<td>Samsung 840 EVO (SATA)</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>Depends</td>
</tr>
<tr>
<td>Samsung 850 EVO (SATA)</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>Depends</td>
</tr>
<tr>
<td>Samsung 950 PRO (NVMe)</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>Probably safe</td>
</tr>
<tr>
<td>Samsung T3 (USB)</td>
<td></td>
<td></td>
<td></td>
<td>✗</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Compromised</td>
</tr>
<tr>
<td>Samsung T5 (USB)</td>
<td></td>
<td></td>
<td></td>
<td>✗</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Compromised</td>
</tr>
</tbody>
</table>

1 Derivation of the DEK from the password in ATA Security (High mode)
2 Derivation of the DEK from the password in ATA Security (Max mode)
3 Derivation of the DEK from the password in TCG Opal
4 Derivation of the DEK from the password in proprietary standard
5 No single key for entire disk
6 Not vulnerable to ATA Master password re-enabling (only if derivation is present)
7 Randomized DEK on sanitize and sufficient random entropy
8 No wear leveling related issues
9 No DEVSLP related issues
Results

- **TCG Opal is terrible**
  - Over-engineered
  - Security goals not clear
  - No reference implementation exists

<table>
<thead>
<tr>
<th>Drive</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>Impact</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crucial MX100 (all)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>Compromised</td>
</tr>
<tr>
<td>Crucial MX200 (all)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>Compromised</td>
</tr>
<tr>
<td>Crucial MX300 (all)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td></td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>Compromised</td>
</tr>
<tr>
<td>Sandisk X600 (SATA)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>Probably compromised</td>
</tr>
<tr>
<td>Samsung 840 EVO (SATA)</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>Depends</td>
</tr>
<tr>
<td>Samsung 850 EVO (SATA)</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>Depends</td>
</tr>
<tr>
<td>Samsung 950 PRO (NVMe)</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>Probably safe</td>
</tr>
<tr>
<td>Samsung T3 (USB)</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Compromised</td>
</tr>
<tr>
<td>Samsung T5 (USB)</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Compromised</td>
</tr>
</tbody>
</table>

1. Derivation of the DEK from the password in ATA Security (High mode)
2. Derivation of the DEK from the password in ATA Security (Max mode)
3. Derivation of the DEK from the password in TCG Opal
4. Derivation of the DEK from the password in proprietary standard
5. No single key for entire disk
6. Not vulnerable to ATA Master password re-enabling (only if derivation is present)
7. Randomized DEK on sanitize and sufficient random entropy
8. No wear leveling related issues
9. No DEVSLP related issues
Results

- TCG Opal is terrible
  - Over-engineered
  - Security goals not clear
  - No reference implementation exists
  - Implementation is not even part of compliance tests

<table>
<thead>
<tr>
<th>Drive</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>Impact</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crucial MX100 (all)</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Compromised</td>
</tr>
<tr>
<td>Crucial MX200 (all)</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Compromised</td>
</tr>
<tr>
<td>Crucial MX300 (all)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>Compromised</td>
</tr>
<tr>
<td>Sandisk X600 (SATA)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>Probably compromised</td>
</tr>
<tr>
<td>Samsung 840 EVO (SATA)</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Depends</td>
</tr>
<tr>
<td>Samsung 850 EVO (SATA)</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>Depends</td>
</tr>
<tr>
<td>Samsung 950 PRO (NVMe)</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>Probably safe</td>
</tr>
<tr>
<td>Samsung T3 (USB)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>Compromised</td>
</tr>
<tr>
<td>Samsung T5 (USB)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>Compromised</td>
</tr>
</tbody>
</table>

1. Derivation of the DEK from the password in ATA Security (High mode)
2. Derivation of the DEK from the password in ATA Security (Max mode)
3. Derivation of the DEK from the password in TCG Opal
4. Derivation of the DEK from the password in proprietary standard
5. No single key for entire disk
6. Not vulnerable to ATA Master password re-enabling (only if derivation is present)
7. Randomized DEK on sanitize and sufficient random entropy
8. No wear leveling related issues
9. No DEVLSP related issues
Results

- TCG Opal is terrible
  - Over-engineered
  - Security goals not clear
  - No reference implementation exists
  - Implementation is not even part of compliance tests
  - Structural changes needed

<table>
<thead>
<tr>
<th>Drive</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>Impact</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crucial MX100 (all)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>Compromised</td>
</tr>
<tr>
<td>Crucial MX200 (all)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>Compromised</td>
</tr>
<tr>
<td>Crucial MX300 (all)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td>Compromised</td>
</tr>
<tr>
<td>Sandisk X600 (SATA)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td>Probably compromised</td>
</tr>
<tr>
<td>Samsung 840 EVO (SATA)</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Depends</td>
</tr>
<tr>
<td>Samsung 850 EVO (SATA)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Depends</td>
</tr>
<tr>
<td>Samsung 950 PRO (NVMe)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Probably safe</td>
</tr>
<tr>
<td>Samsung T3 (USB)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>Compromised</td>
</tr>
<tr>
<td>Samsung T5 (USB)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>Compromised</td>
</tr>
</tbody>
</table>

1. Derivation of the DEK from the password in ATA Security (High mode)
2. Derivation of the DEK from the password in ATA Security (Max mode)
3. Derivation of the DEK from the password in TCG Opal
4. Derivation of the DEK from the password in proprietary standard
5. No single key for entire disk
6. Not vulnerable to ATA Master password re-enabling (only if derivation is present)
7. Randomized DEK on sanitize and sufficient random entropy
8. No wear leveling related issues
9. No DEVSLP related issues
Timeline

Oct 2016  First discovery – Crucial (Mciron) MX100
Oct 2017 – Apr 2018  Attempts made contacting vendors
Apr 2018  Disclosure to Samsung – Meeting in The Hague, Netherlands
Apr 2018  Disclosure to Micron
Nov 2018  Draft paper published – Vendor responses published
Both vendors release firmware updates
Dec 2018  Presentation at 35C3
Dec 2018  Discovery of Sandisk (Western Digital) models
Timeline (2)

Today:

• Western Digital releases firmware updates available at https://www.westerndigital.com/productsecurity
  Reviewed by Trail of Bits
• “Western Digital thanks the Radboud researchers, NCSC, and CERT-CC for participating in the coordinated disclosure process. For more information on how we work with researchers - including contact details -, please go to https://www.westerndigital.com/productsecurity.”
Questions

See the paper ’Self-Encrypting Deception’

Carlo Meijer

✉️ c.meijer@cs.ru.nl
🌐 https://cs.ru.nl/~cmeijer/
🌐 https://midnightbluelabs.com/

Bernard van Gastel

✉️ b.vangastel@cs.ru.nl
🌐 https://sustailablesoftware.info/