Data Sanitization on eMMCs

Fukami, A.; Regazzoni, F.; Geradts, Z.

DOI
10.1145/3566097.3568349

Publication date
2023

Document Version
Final published version

Published in
ASP-DAC 2023

Data Sanitization on eMMCs

Aya Fukami
a.fukami@uva.nl
Netherlands Forensic Institution and
University of Amsterdam
The Netherlands

Francesco Regazzoni
f.regazzoni@uva.nl
University of Amsterdam and
Università della Svizzera italiana
The Netherlands

Zeno Geradts
Z.J.M.H.Geradts@uva.nl
Netherlands Forensic Institution and
University of Amsterdam
The Netherlands

ABSTRACT

Data sanitization of modern digital devices is an important issue given that electronic wastes are being recycled and repurposed. The embedded Multi Media Card (eMMC), one of the NAND flash memory-based commodity devices, is one of the popularly recycled products in the current recycling ecosystem. We analyze a repurposed devices and evaluate its sanitization practice. Data from the formerly used device can still be recovered, which may lead to an unintentional leakage of sensitive data such as personally identifiable information (PII). Since the internal storage of an eMMC is the NAND flash memory, sanitization practice of the NAND flash memory-based systems should apply to the eMMC. However, proper sanitize operation is obviously not always performed in the current recycling ecosystem. We discuss how data stored in eMMC and other flash memory-based devices need to be deleted in order to avoid the potential data leakage. We also review the NAND flash memory data sanitization schemes and discuss how they should be applied in eMMCs.

CCS CONCEPTS

• **Security and privacy → Security in hardware**.

KEYWORDS

security, digital forensics, data sanitization, data recovery

ACM Reference Format:
Aya Fukami, Francesco Regazzoni, and Zeno Geradts. 2023. Data Sanitization on eMMCs. In *28th Asia and South Pacific Design Automation Conference (ASPDAC '23), January 16–19, 2023, Tokyo, Japan.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3566097.3568349

1 INTRODUCTION

In order to protect sensitive information from accidental leakage, proper data sanitization is an important step when disposing digital devices. The ultimate goal of data sanitization is to delete physical data in a way that the original data is not recoverable. If an adversary who is given some manner of access to a device can recover the deleted data, then the data of the device is not properly sanitized [6, 21]. In modern digital devices, when data is deleted from the user interface, only the metada of the file is deleted in order to make the data *unlinked*, leaving the actual data in the storage media [6, 16, 21]. At this point, the deleted data is easily recoverable since the original data remains untouched. However, users of the system are not exposed to the physical memory and do not have direct access to it. Typically, there exist multiple layers between the user input and the physical storage media on a digital device. The request from the user application (the highest layer) modifies the file system, which can communicate with the controller of the physical storage media (the lowest layer). Then the controller can finally perform physical data deletion. Therefore proper data sanitization relies on this chain of communication on the host system. In other words, unless the proper deletion command is issued against the storage device, the physical data still remains on the system in an recoverable way.

Due to its low cost-per-bit and portability, flash memory is the most popularly used storage media in the modern digital devices. Additionally, commodity flash memory products are gaining their popularity recently instead of the raw flash memory. Most widely used product is called embedded Multi Media Card (eMMC). An eMMC is a non-volatile memory product which has flash memory and a flash memory controller packaged into one IC chip. Its interface is standardized by JEDEC, and an eMMC can be easily controlled by the standardized commands [12].

Deleting the flash memory data requires the proper erase command issued against the address where the data is stored. Flash memory stores data by trapping charges in the floating gate in a transistor. By issuing the erase command, those charges are discharged, erasing the stored data as a result. This requirement is hard to achieve on digital devices because of the two reasons. First, flash memory chips are directly soldered on the circuit board of a digital device. Therefore detaching the memory and directly send the erase command needs physical processing performed by experts. Second, if the flash memory is used in a form of a commodity device such as an eMMC, even after detaching the memory product, operators do not have access to the raw flash memory.

Proper data sanitization becomes important especially when the owner of the system changes from one to another. An example is when a device is to be sold on a second-hand market. If an unsanitized system is purchased by an adversary, data remnant can be extracted to recover original data, leading to an unintentional data leakage. Additionally, issues can be observed when a flash memory is repurposed for another device. With the current global chip shortage, IC (Integrated Circuit) chips in digital devices are often recycled and repurposed, and flash memory chips are also one of the popularly recycled chips in the current ecosystem. It is reported that they are frequently recycled while holding data from the old device. However its data is not always properly deleted in the current recycling ecosystem [22]. As a result, sensitive data including personally identifiable information (PII) can be unintentionally leaked. According tothe study by Schneider et al. [22], eMMCs are also also frequently recycled and reused in low cost digital devices. Unlike

ASPDAC '23, January 16–19, 2023, Tokyo, Japan

ACM ISBN 978-1-4503-9783-4/23/01.
https://doi.org/10.1145/3566097.3568349

the data on flash memory, the data extracted from an eMMC is already in the form of the file system used in the target host system. Thus makes it easier for adversaries to carve the old data.

In this paper, we discuss how flash memory-based devices are poorly sanitized when recycled and repurposed. Using a real-world example, we show that data from the formerly used device is still recoverable from the newly purchased storage device. Throughout the paper, we focus on the eMMC as the target, since eMMCs are the most commonly used flash memory-based device in modern digital devices [2]. The rest of this paper is organized as follows. In Section 2, we discuss data sanitization issues on flash memory-based devices. Then we demonstrate data recovery from an unsanitized and repurposed device in Section 3. We then proceed to review the currently available flash memory sanitization procedures to discuss their applicability in Section 4. In section 5, we review the related work, before concluding in Section 6.

2 DATA SANITIZATION ISSUES ON NAND FLASH MEMORY-BASED DEVICES

2.1 Poor Data Sanitization in Second-hand Market

As reported by Blancco and Ontrack [3], recycled storage devices are sold in second-hand market without proper data sanitization. As a result, it is reported that 42 percent of the collected drives contain data in a recoverable way. While this is an example in a second-hand market, it is also reported that old data can be found in devices sold as *new* [22]. According to Schneider et al. [22], flash memory chips are recycled from one device and re-mounted onto another device, and they are sold as new low-cost devices. In many cases, those repurposed memory devices still hold data from the formerly used devices, allowing data exposure to unintended users. Flash memory ICs are recycled from wide varieties of devices, including Android devices, smart TVs, car navigation systems, and other IoT devices. After their life cycles, digital devices are recycled as e-wastes, their memory ICs are removed from their circuit boards, and then re-mounted onto other "new" devices. In this cycle, data sanitization seems not always to be performed properly [22]. Therefore, by carving the physical data extracted from the memory ICs on those new devices, an adversary can collect data which belongs to different device where the memory was mounted one life cycle ago.

2.2 Data Sanitization on NAND Flash Memory

While data can remain undeleted due to poor handling, NAND flash memory itself has a drawback when it comes to data deletion. When programming and erasing data on flash memory, the data size needs to be crafted according to the page and block size of the target flash memory. A page is a series of flash memory cells, and its size is typically between 2K to 16K bytes. Page size is the unit used for data write and read on flash memory. Data erase, on the other hand, needs to be performed at a block granularity. A block consists of multiple pages, i.e. 128 or 256 pages. On top of those constraints, when performing a data write on NAND flash memory, the target page data needs to be empty. Therefore, NAND flash memory uses an erase-before-write procedure. Since directly updating the page of data is not possible, when the data is updated on the host system, the new data that needs to be written into a new empty page of the flash memory. This scheme is typically called out-of-place update. When the new data is stored, the old data is unlinked and flagged as deleted, so that the flash translation layer (FTL) can recognize only the new data as the valid data [1]. However, the old data itself remains untouched since most of the time there exists valid data on different pages in the same block. Therefore the controller cannot issue the data erase command directly. As a result, deleted data remains in the unallocated area of the flash memory.

Additionally, in order to optimize the writing speed, a multi-plane copy operation is commonly used in flash memory based-devices. When writing data from the host system, the data is written into the cache area of the flash memory, then it is copied internally to data areas. For the cache area, typically the SLC (single level cell) area is used, while the MLC (multi-level cell) area is used for storing data. Therefore, the copy of the old data can still remain on multiple areas of the flash memory as cache data.

Finally, the erase operation on NAND flash memory is time and power consuming [1]. First, valid and invalid data needs to be placed into separated blocks, in order to avoid accidental data deletion of the allocated data. Since a data write operation requires high voltage, flash memory controllers try to minimize the count of the operation to achieve low power performance. Second, the block erase command needs to be issued to the blocks which have invalid data. Erase operations on NAND flash memory requires even higher voltage. Additionally, since NAND flash memory cells wear out through repeated program and erase (PE) cycles, flash memory controllers try to optimize the erase counts of each block. The combination of these constrains creates a time gap between the delete operation from user interface and the actual data purge of the physical data on the flash memory.

2.3 Data Sanitization on eMMCs

The same issue discussed in the previous section also applies to the eMMC, a flash memory-based commodity device. An eMMC consists of flash memory and the flash memory controller. The host system interacts with the controller through standardized commands. As a result, the host system cannot control the data at the flash memory level. The last interface where the host system can control to sanitize the physical data is one layer further from the flash memory.

Nevertheless, on the eMMC, multiple data purge commands are defined [12]. *Erase*, *Trim*, *Discard*, *Secure Erase*, *Secure Trim*, and *Sanitize*. When either an *Erase*, *Trim*, or *Discard* command is issued, the target data is flagged as deleted, and the controller can erase the actual data at its convenient timing. Therefore the deleted data can remain on the device. Meanwhile, *Secure Erase*, *Secure Trim*, or *Sanitize* command requires the controller to execute the data purge operation immediately. While performing the operation, no other command can be executed. Therefore those commands may impact the performance of the system. Up to eMMC version 4.50, only *Secure Erase* and *Secure Trim* commands are defined. For eMMC version 4.51 or higher, it is required to use the *Sanitize* operation instead of those commands. For *Secure Erase* and *Secure Trim* commands, the eMMC manufacturer can define the physical data

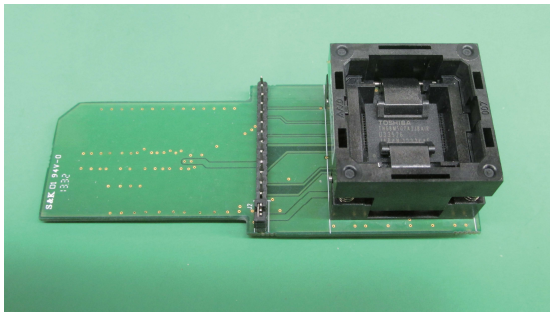


Figure 1: An example eMMC adaptor. An eMMC can be recognized as SD/MMC card by the PC when connected.

removing method. The available secure data removal type includes data erase and overwriting data. When operating *Sanitize*, the controller is required to remove data physically from "the unmapped user address space" [12].

3 DATA RECOVERY FROM eMMCS

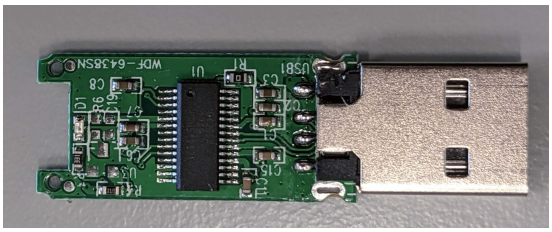
3.1 Data Recovery Process

In principle, acquiring the physical data from NAND flash memory lets the adversary collect all the hidden data in a NAND flash memory-based device [1, 4, 7, 23]. Before storing the data into the NAND flash memory, the flash memory controller modifies the data sent from the host system. This modification includes error correction, data randomization, and wear-leveling. For error correction, error correcting code (ECC) is computed and attached to the original data, using the dedicated algorithm defined by the controller. Data randomization modifies the original data with the random pattern in order to generalize the data distribution of 0s and 1s. Finally, by wear-leveling, the controller chooses the physical address where the logical data is to be stored. Those addresses are selected in a way that the PE cycle counts of each block distributes evenly. Given this modification performed by the controller, additional data processing is required after extracting the physical data in order to retrieve the original data [4, 23, 25].

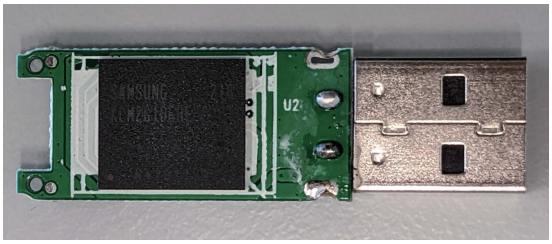
Compared to the raw NAND flash memory, data recovery from an eMMC is straightforward. All the data processing described above is already performed through the flash memory controller, and the pinout of the eMMC is defined by JEDEC [11]. Therefore once connected to the personal computer (PC) through an appropriate SD/MMC controller and adapter, an eMMC can be recognized as a standard storage media. An example eMMC adaptor is shown in Fig. 1. The eMMC is placed into the socket, which is connected to the SD-like interface. The adaptor can be inserted into an SD card slot in the PC. A user can then extract the data from the whole area of the eMMC by issueing the read command to all the address space.

3.2 Example Product with Recycled eMMC

Fig. 2 shows an example low-cost USB thumb drive which is disassembled from its casing. The drive was purchased as a new device. The device consists of a SD/MMC controller (Fig. 2a) and an eMMC (Fig. 2b). Unlike a normal flash memory controller, since the eMMC contains its own flash controller, the controller in Fig. 2a does not



(a) USB thumbdrive with a recycled eMMC. Top side with the controller visible



(b) Sample Product Bottom View

Figure 2: Example Product with Recycled eMMC

work as the FTL of the storage memory. Instead, it controls the eMMC as a general SD-like card. Therefore this device can function with any other eMMCs without a special manufacturing process. Typically, when manufacturing a USB thumbdrive with a flash memory chip, the controller needs its dedicated firmware stored on the flash memory. This requirement makes the manufacturing process more complex. Given that this manufacturing process can be skipped, eMMC-based USB thumbdrives such as this example can be manufactured with lower cost.

3.2.1 Hidden Data in Repurposed eMMC. We detached the eMMC chip from the PCB and directly extracted its content before connecting the drive to a PC. Quick data carving of the extracted data with forensic software shows multiple jpeg files and location entries. Additionally, many strings are available to show that the target eMMC was once used in a marine GPS navigation system. While the data on this particular example might be trivial and not necessarily sensitive, since old data is available in clear-text, we claim that the proper data deletion was never performed between the time when the old device is disposed and the time when the thumb drive was manufactured. This claim can be supported by the past large-scale study conducted by Schneider et al. [22]. They report that private pictures and videos, as well as chat messages can be found on the similar devices. Additionally, the fact that the eMMC in a new device is not clean and contaminated with the data with which the owner of the device has nothing to do, could also be a problem during forensic investigations. Recognizing the actual owner of the extracted file can be difficult if the file is extracted from the unallocated area.

In order to search for more evidence that the target eMMC was once used in another device, we checked the "smart report" of the target eMMC. The smart report is available on Samsung eMMCs through a vendor-specific command. It is a report feature where

```
---Smart Report---
Error Mode: 0xd2d2d2d2
Super Block Size: 0x00200000
Super Page Size: 0x00004000
Optimal Write Size: 0x00004000
Number of Banks: 0x00000001
Bank0 Initial Bad Block: 0x00000004
Bank0 Runtime Bad Block: 0x00000000
Bank0 Remain Reserved Block: 0x00000038
Max Erase Count: 0x000000c4 (196)
Min Erase Count: 0x00000000 (0)
Avg Erase Count: 0x0000008c (140)
Read Reclaim Count: 0x00000000
Optimal Trim Size: 0x00002000
Max Erase Count (SLC): 0x0000009c (156)
Min Erase Count (SLC): 0x00000000 (0)
Avg Erase Count (SLC): 0x00000069 (105)
Max Erase Count (MLC): 0x000000c4 (196)
Min Erase Count (MLC): 0x00000070 (112)
Avg Erase Count (MLC): 0x0000008d (141)
```

Figure 3: The result of smart report reading from the eMMC mounted on the device shown in Fig. 1

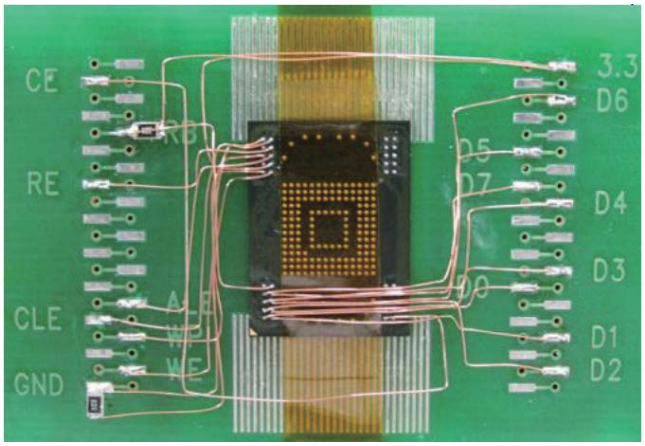


Figure 4: Wiring flash memory interface of an eMMC for direct access to internal flash

users can check the health status of the target. Through the smart report, the user can learn the status of the internal flash memory such as the number of bad blocks and program/erase (P/E) cycles. The smart report of the eMMC taken from the device in Fig. 2 is shown in Fig. 3. As shown here, the maximum erase count of the flash memory blocks is 196 times, and the average erase count of all the blocks is 140 times. Those numbers show that the target eMMC is not new, and it has obviously been used on another device. Also, since the internal flash memory has already gone through more than 100 P/E cycles, the internal physical data could be already susceptible to bit errors.

3.2.2 Extracting Internal Flash Memory Data. Figure 4 shows the flash memory interface on an eMMC. By issuing the proper flash memory data read command through the interface, an adversary can get access to the unallocated data. On the eMMC we examined, 654,197 sectors out of the total 3,907,584 sectors contain data. The amount of data is thus 334.949 megabytes. The extracted raw data from the internal flash memory has 2,264,924,160 bytes. Since a page size of the target flash memory is 8,640 bytes (8,192 data bytes and 448 spare bytes), the target flash memory consists of 262,144 pages. Out of all the pages, 46,090 pages contain data. Thus overall available data from the flash memory is around 377.569 megabytes. Almost 42 megabytes of unallocated data still remains in the flash memory, giving adversaries more chance of recovering the old data. Additionally, even if the target eMMC data is sanitized through data purge commands mentioned in section 2.3, former study shows that the physical data is not always deleted. According to Fukami et al. [8], even after the *Sanitize* operation is performed, the internal flash controller does not always issue the data erase command to the flash memory, leaving the old data on the memory.

4 TOWARDS SECURE DATA MANAGEMENT ON eMMCS

4.1 Operational Requirements

4.1.1 Standards on Data Sanitization. National Institute of Standards and Technology [18] issues a guideline on secure data sanitization. According to the guideline, SD cards and MMCs are recommended to be overwritten with an approved data overwriting tool. Since eMMCs belong to the same category, when an eMMC is recycled, at least overwriting procedure needs to be performed. However, as it is stated in the guideline, no tool is evaluated for its ability to securely erase the data remnant. Therefore, users need to either rely on the open source tools or outsource the sanitization procedure to the third party.

When performing the data purge on a device embedded with an eMMC, it is recommended to issue either the Secure Trim or Secure Erase command to securely delete its sensitive data. If this procedure is properly performed at the end of the life cycle of a device, data becomes inaccessible through the eMMC interface. As a result, the risk of the data leakage becomes significantly low.

4.1.2 Cryptographic Erase. In the same guideline, Cryptographic Erase (CE) is proposed as a fast and effective sanitization solution. With CE, all the user data is encrypted by the system before being stored on the storage device. When performing the data sanitization, the encryption key is destroyed to prevent data leakage. CE is promising since data is already encrypted when it is stored in the storage media. Therefore data is already protected against the adversaries who have access to the physical data. Data sanitization can be completed only by erasing the key data, which greatly optimizes the sanitization process compared to the complete data erasure.

On the other hand, while the CE is cost effective, fast and secure way of data sanitization, when applied to an eMMC, the expected security level might not be achieved. As discussed, deleted data on an eMMC can still be recoverable by extracting data from internal flash memory. If the key data remains in flash memory after deletion, an adversary may directly access the internal flash memory

and recover the key data. Since the ciphertext still remains on the storage media after CE is performed, an adversary can decrypt the ciphertext offline using the obtained key data, gaining access to the original data as a result. If CE is to be implemented on the device with an eMMC, secure key management system is required in a way that the key data will not be recovered after the sanitization procedure.

4.2 Studies on Effective Flash Memory Data Sanitization

Issues in data sanitization on NAND flash memory-based devices have been pointed out through multiple studies [1, 5, 26]. Traditionally, garbage collection [24] has been performed to erase unallocated data through the FTL. However, due to the significant overhead coming from time-consuming data processing, this scheme is not entirely reliable for proper data sanitization. In order to improve garbage collecting operation, "Scrubbing" technique, which overwrites the old data with zeros, has been considered as one of the solution to destroy data at the memory cell level [13, 26]. However, this technique can cause more bit errors to the valid data due to over-programming, thus it is not stable enough when applied[15]. Additionally, Hasan and Ray [9] demonstrated that through analog reading of the difference in threshold voltage in the updated flash memory cell, data can still be recoverable even after data is erased by scrubbing.

One of the more effective data erase schemes is called one-shot sanitization [17]. In the MLC flash memory cell, the 2-bit stored data consists of bits from two different pages. The idea is to preserve the data from one page, while sanitizing the bit from another page. Depending on the location of the bit to sanitize (Least Significant Bit (LSB) or Most Significant Bit (MSB)), the different level of data write voltage is applied. The applied voltage changes the amount of charges trapped in the floating gate of the cell. This way, data from one page becomes 0, while the original data from another page is preserved. The applied voltage needs to be carefully selected in order to keep the other bit of data. While this method is promising, creating the fine-tuned voltage requires vendor-specific commands. Therefore it is not generic enough to implement this scheme into the FTL. Meanwhile, Raquibuzzaman et al. [20] proposes another data sanitization method that works instantly when performed. Their method makes use of the fact that the threshold voltage of a flash memory cell can be increased by performing a write command. When one page of data needs to be deleted, the data on the same memory cell (data on another page) is cached, and then the new value which is the combination of 0 and the cached data is again written into the same memory cell. This way, only one bit of the data in a memory cell is erased, preserving the data from another page which shares the same flash memory cell. Through this procedure, the old data can be deleted real-time just using the standard read and write commands.

Another recent study by Kim et al. [15] suggests more robust data protection technique. The authors use the new flash commands called page-lock and block-lock commands. By locking the page or block immediately after the data becomes invalid, the data becomes inaccessible even through the flash memory interface. As a result, the data is protected even against the adversaries who have direct access to the NAND flash memory interface.

Although those promising techniques are shown to have minimum overhead, flash memory controlling protocols tend to differ greatly among manufacturers. Not only ONFI (Open NAND Flash Interface [19])- defined protocols, vendors implement vendor-specific protocols which optimize the performance of their products. Therefore applying those techniques in the real world can be challenging. In addition, TLC (triple-level-cell) technologies and 3D NAND flash memory technologies pose more challenges in developing effective data sanitization procedures.

4.3 Transparency in Data Sanitization on Flash Memory Based Devices

Given that the eMMC and other managed flash memory storage have an internal flash memory controller, it is the effort of the flash memory controller manufacturers to apply the secure data purge operations in order to keep the data from possible leakage. In JEDEC standard, it is defined that physical data including data copy needs to be erased after *Secure Erase*, *Secure Trim*, and *Sanitize*. However, physical data allegedly remains in the internal flash memory in some eMMC models even after issuing those commands [8]. Given the black-box nature of the flash memory controller, users do not have control over the communication between the controller and flash memory. Meanwhile, with the current demands for speed and power performance of the memory systems, the designing trend of the flash memory controller tends to focus on the performance optimization. From the security perspective, proper handling of the old data should also be one of the main features during the flash memory controller designing process.

5 RELATED WORK

The global flow of e-waste has been studied from a waste management standpoint. According to the research conducted by Ilankoon et al. [10], it is the fastest growing waste stream in the world. It is also pointed out that even though the components of electric equipment are recycled and refurbished, data security of those components is a "largely overlooked area in e-waste management" [10]. Another study conducted by Kapoor et al. [14] points out that e-waste can be illegally traded, and sensitive data is collected by cybercriminals. A large-scale study using the actual refurbished flash memory-based devices conducted by Schneider et al. [22] shows that variety of data can also be found in eMMCs. Although they did not investigate the data remnants at the internal flash memory level, according to Fukami et al. [8], more data can remain on the internal flash memory even after data deletion on the eMMC. As stated by Wu et al. [27], the sanitization method used for hard disk drives does not work properly for flash memory-based devices due to its FTL design. Therefore multiple research has been conducted in order to securely erase data on NAND flash memory through the FTL [6, 15, 17, 20, 26, 27]. However, actual application of those techniques to the flash memory-based commodity devices such as eMMCs is still unclear.

6 CONCLUSION

We have shown an example where an eMMC has been repurposed into a new device without going through the proper data sanitization process. This poor sanitization practice in the e-waste recycling ecosystem can lead to an unintentional leakage of sensitive data. Either the secure data purge operation, or the encryption-based data protection/sanitization needs to be performed at the proper timing to minimize the risk of data leakage. Additionally, to keep the data safe against adversaries who have access to the internal flash memory interface, proper data sanitization using the newly developed schemes needs to be provided through the flash memory controller. Secure data handling at the physical level is expected inside the eMMC.

REFERENCES

[1] Na-Young Ahn and Dong Hoon Lee. 2019. Schemes for Privacy Data Destruction in a NAND Flash Memory. *IEEE Access* 7 (2019), 181305–181313. https://doi.org/10.1109/ACCESS.2019.2958628

[2] Paolo Amato, Danilo Caraccio, Emanuele Confalonieri, and Marco Sforzin. 2015. An Analytical Model of eMMC Key Performance Indicators. In *2015 IEEE International Memory Workshop (IMW)*. 1–4. https://doi.org/10.1109/IMW.2015.7150276

[3] Blancco and Ontrack. 2019. *Privacy for Sale: Data Security Risks in the Second-Hand IT Asset Marketplace.* Technical Report.

[4] Marcel Breeuwsma, Martien de Jongh, Coert Klaver, Ronald van der Knijff, and Mark Roeloffs. 2007. Forensic Data Recovery from Flash Memory. *Small Scale Digital Device Forensics Journal* (2007).

[5] Bo Chen, Shijie Jia, Luning Xia, and Peng Liu. 2016. Sanitizing Data is Not Enough! Towards Sanitizing Structural Artifacts in Flash Media. In *Proceedings of the 32nd Annual Conference on Computer Security Applications* (Los Angeles, California, USA) *(ACSAC '16)*. Association for Computing Machinery, New York, NY, USA, 496–507. https://doi.org/10.1145/2991079.2991101

[6] Sarah Diesburg, Christopher Meyers, Mark Stanovich, An-I Andy Wang, and Geoff Kuenning. 2016. TrueErase: Leveraging an Auxiliary Data Path for Per-File Secure Deletion. *ACM Trans. Storage* 12, 4, Article 18 (may 2016), 37 pages. https://doi.org/10.1145/2854882

[7] Aya Fukami, Saugata Ghose, Yixin Luo, Yu Cai, and Onur Mutlu. 2017. Improving the reliability of chip-off forensic analysis of NAND flash memory devices. *Digital Investigation* 20 (2017), S1 – S11. https://doi.org/10.1016/j.diin.2017.01.011

[8] Aya Fukami, Sasha Sheremetov, Francesco Regazzoni, Zeno Geradts, and Cees De Laat. 2022. Experimental Evaluation of e.MMC Data Recovery. *IEEE Transactions on Information Forensics and Security* 17 (2022), 2074–2083. https://doi.org/10.1109/TIFS.2022.3176187

[9] Md Mehedi Hasan and Biswajit Ray. 2020. Data Recovery from "Scrubbed" NAND Flash Storage: Need for Analog Sanitization. In *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, 1399–1408. https://www.usenix.org/conference/usenixsecurity20/presentation/hasan

[10] I.M.S.K. Ilankoon, Yousef Ghorbani, Meng Nan Chong, Gamini Herath, Thandazile Moyo, and Jochen Petersen. 2018. E-waste in the international context – A review of trade flows, regulations, hazards, waste management strategies and technologies for value recovery. *Waste Management* 82 (2018), 258–275. https://doi.org/10.1016/j.wasman.2018.10.018

[11] JEDEC Solid State Technology Association. 2007. *Embedded MultiMediaCard (eMMC) Mechanical Standard.* JEDEC Standard JESD84-C43. https://www.jedec.org/system/files/docs/JESD84-A41.pdf

[12] JEDEC Solid State Technology Association. 2015. *Embedded Multi-Media Card (e·MMC) Electrical Standard (5.1).* JEDEC Standard JESD84-B51. https://www.jedec.org/system/files/docs/JESD84-B51.pdf

[13] Shijie Jia, Luning Xia, Bo Chen, and Peng Liu. 2016. NFPS: Adding Undetectable Secure Deletion to Flash Translation Layer. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security* (Xi'an, China) *(ASIA CCS '16)*. Association for Computing Machinery, New York, NY, USA, 305–315. https://doi.org/10.1145/2897845.2897882

[14] Neeti Kapoor, Pradnya Sulke, and Ashish Badiye. 2021. E-waste forensics: An overview. *Forensic Science International: Animals and Environments* 1 (2021), 100034. https://doi.org/10.1016/j.fsiae.2021.100034

[15] Myungsuk Kim, Jisung Park, Genhee Cho, Yoona Kim, Lois Orosa, Onur Mutlu, and Jihong Kim. 2020. Evanesco: Architectural Support for Efficient Data Sanitization in Modern Flash-Based Storage Systems *(ASPLOS '20)*. Association for Computing Machinery, New York, NY, USA, 1311–1326. https://doi.org/10.1145/3373376.3378490

[16] Jaeheung Lee, Junyoung Heo, Yookun Cho, Jiman Hong, and Sung Y. Shin. 2008. Secure Deletion for NAND Flash File System. In *Proceedings of the 2008 ACM Symposium on Applied Computing* (Fortaleza, Ceara, Brazil) *(SAC '08)*. Association for Computing Machinery, New York, NY, USA, 1710–1714. https://doi.org/10.1145/1363686.1364093

[17] Ping-Hsien Lin, Yu-Ming Chang, Yung-Chun Li, Wei-Chen Wang, Chien-Chung Ho, and Yuan-Hao Chang. 2018. Achieving Fast Sanitization with Zero Live Data Copy for MLC Flash Memory. In *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 1–8. https://doi.org/10.1145/3240765.3240773

[18] National Institute of Standards and Technology. 2014. *Guidelines for media sanitization.* Technical Report SP 800-88 Rev. 1. U.S. Department of Commerce, Washington, D.C. https://doi.org/10.6028/NIST.SP.800-88r1

[19] ONFI. 2021. *Open NAND Flash Interface Specification.* Technical Report. http://www.onfi.org/

[20] Md Raquibuzzaman, Matchima Buddhanoy, Aleksandar Milenkovic, and Biswajit Ray. 2022. Instant Data Sanitization on Multi-Level-Cell NAND Flash Memory. In *Proceedings of the 15th ACM International Conference on Systems and Storage* (Haifa, Israel) *(SYSTOR '22)*. Association for Computing Machinery, New York, NY, USA, 85–95. https://doi.org/10.1145/3534056.3534941

[21] Joel Reardon, David A. Basin, and Srdjan Capkun. 2013. SoK: Secure Data Deletion. *2013 IEEE Symposium on Security and Privacy* (2013), 301–315.

[22] Janine. Schneider, Immanuel. Lautner, Denise. Moussa, Julian. Wolf, Nicole. Scheler, Felix. Freiling, Jaap. Haasnoot, Hans. Henseler, Simon. Malik, Holger. Morgenstern, and Martin. Westman. 2021. In Search of Lost Data: A Study of Flash Sanitization Practices. In *Digital Forensics Research Conference Europe (DFRWS EU)*.

[23] Igor Sestanj. 2016. *NAND Flash Data Recovery Cookbook.*

[24] Che-Wei Tsao, Yuan-Hao Chang, and Ming-Chang Yang. 2013. Performance enhancement of garbage collection for flash storage devices: An efficient victim block selection design. In *2013 50th ACM/EDAC/IEEE Design Automation Conference (DAC)*. 1–6.

[25] Jan Peter van Zandwijk. 2015. A Mathematical Approach to NAND Flash-Memory Descrambling and Decoding. *Digital Investigation* (2015).

[26] Michael Wei, Laura Grupp, Frederick E. Spada, and Steven Swanson. 2011. Reliably Erasing Data from Flash-Based Solid State Drives. In *9th USENIX Conference on File and Storage Technologies (FAST 11)*. USENIX Association, San Jose, CA. https://www.usenix.org/conference/fast11/reliably-erasing-data-flash-based-solid-state-drives

[27] Chin-Hsien Wu, Po-Ling Lin, Yu-Hun Hu, and Ming-Yang Du. 2019. A Data Sanitization Method for Mobile Devices with NAND Flash Memory. In *Proceedings of the Conference on Research in Adaptive and Convergent Systems* (Chongqing, China) *(RACS '19)*. Association for Computing Machinery, New York, NY, USA, 7–13. https://doi.org/10.1145/3338840.3355639