Experimental Evaluation of $e$-MMC Data Recovery

Aya Fukami, Sasha Sheremetov, Francesco Regazzoni, Zeno Geradts, Senior Member, IEEE, and Cees De Laat, Member, IEEE

Abstract—In this paper, we explore the data recovery procedures from $e$-MMCs. The $e$-MMC is one of the “managed” flash memory devices that are popularly used in modern digital devices as their storage media. The $e$-MMC, which consists of flash memory and the flash memory controller, optimizes the data input/output between the host device and the non-volatile memory through its standardized protocol. Its standardized structure and protocol makes forensic physical data acquisition simpler than handling the raw flash memory. However, its secure data purging features, such as Secure Erase and Sanitize, make data recovery from $e$-MMC a challenging task. In this research, we investigate inside the $e$-MMCs, and evaluate advanced data recovery procedures. By reverse engineering the structures of $e$-MMCs and accessing the internal flash memory, we discover that securely erased data is still recoverable from the internal flash memory. In some models, more than 99% of the securely erased data can still be recoverable by accessing the flash memory inside the $e$-MMCs. The data extraction method, along with experimental data recovery evaluation, will be explored in this paper.

Index Terms—Chip-off forensic, e.MMC, physical image acquisition, mobile forensic, deleted data recovery.

I. INTRODUCTION

In digital forensics, recovering deleted data from the target devices is one of the critical tasks. Forensic data recovery is typically performed by accessing and carving the unallocated data on the storage media on the target device [1]–[5]. When a set of data is logically deleted, it is flagged as deleted by the system, but the actual data itself remains on the storage media. Therefore, forensic examiners often attempt to recover deleted data by acquiring physical data. Physical data is a bit-by-bit copy of the storage media of the target device, which allows examiners access the entire address space of the memory. By accessing the physical data, forensic examiners may recover data that has been logically, but not necessarily physically deleted data [5]–[8].

Traditionally, flash memory has been the most popular target device of the physical data acquisition in the mobile digital devices [3], [5], [9]. In the modern digital devices, instead of the raw flash memory, “managed” flash memory is commonly used as the storage device. Modern digital devices include smartphones, tablets, navigation systems, and other IoT devices. Examples of managed memory devices include SSD (Solid State Drive), UFS (Universal Flash Storage), $e$-MMC (Embedded Multi Media Card), $e$-MCP (Embedded Multi Chip Package), and uMCP (UFS-based Multi Chip Package). Of those devices, $e$-MMCs are currently most popularly embedded in mobile devices [10]–[12].

An $e$-MMC consists of flash memory and a flash controller embedded into one IC (Integrated Circuit) chip. Its specifications are defined by JEDEC (Joint Electron Device Engineering Council) Solid State Technology Association [13]–[15]. The embedded flash memory stores the data, and its operation is controlled by the flash controller. While directly controlling flash memory requires various data optimization processes and crafting vendor specific commands which vary by production, with the aid of the flash memory controller, the host system can control the target memory by simply issuing standardized commands. When acquiring the physical image of the target device from the $e$-MMC for forensic analysis, commands are therefore issued to the target $e$MMC through $e$-MMC interface, and they are translated by the flash memory controller. Then the forensic examiner can acquire the image that is already optimized by the flash controller.

While the standardized protocol makes physical acquisition simpler than handling the raw flash memory directly, the structure of the $e$-MMC gives several challenges to forensic data recovery. First, since the internal flash memory is inaccessible through the $e$-MMC interface, the data that is equivalent to the traditional physical data cannot be acquired. The physical data acquired through the $e$-MMC interface is the result of raw data processing by the flash controller. The acquired image is typically in the form which the host system can directly recognize [7]). Since the flash controller only passes the allocated area in the flash memory, unallocated area is inaccessible. Second, multiple erase commands are implemented in $e$-MMC specifications. One of them is Secure Erase, which is defined as the command to erase the data in the target addresses, as well as the copies of them, to be purged immediately when it is issued. Once this command is issued on the target device, the erased data becomes completely inaccessible through the $e$-MMC interface, which in turn greatly degrades the forensic data recovery rate.

To illustrate the difference in forensic data recovery rate, typical forensic data extraction methods are categorized in Fig. 1. When logical data extraction is performed, logically
available data, such as individual files and databases, are extracted from available user interfaces on the target devices. The data is therefore in a form which can be visualized through appropriate application software. As shown in the Fig. 1, the minimum amount of data is to be collected from the target device, and no deleted data is acquired in logical data extraction. When methods are available and deleted data needs to be recovered, a forensic examiner proceeds to acquiring wider range of data, typically at the file system level, which is shown as the second level in Fig. 1. By acquiring data at file system level, logically-deleted or some parts of unallocated data can be recovered, along with existing data. Typically no invasive hardware modification is required when performing file system based acquisition. The last and the most invasive step of data acquisition is physical data acquisition. Physical data extraction acquire the entire memory area of the storage. Therefore all the existing data at the system level becomes available, giving more chance for the examiners to recover deleted data by examining the whole system data [10]. When the physical data is acquired from a digital device in which an e-MMC is implemented as the storage, the data is acquired through its interface, either through ISP (In-System Programming), debug ports on the system, or by chip-off [10]. All of those acquisition techniques typically require hardware modification, therefore it is only performed when truly necessary. In normal circumstances, this physical data extraction is the utmost broadest data acquisition method used in forensic data recovery. However, as discussed before, given the structure of the e-MMC, the extracted data through normal physical acquisition is still a part of the real physical data that is stored inside the e-MMC. In other words, more data is possibly available in the memory device itself. This is shown as the dotted line in Fig. 1.

In this paper, we focus on this hidden data inside an e-MMC. In other words, more memory. We explore the method to directly extract data from flash memory inside e-MMCs, and evaluate how much data can be recoverable after erase. Extraction of the internal raw flash memory data can be achieved without damaging the target e-MMC, as long as the target e-MMC is detached from the PCB (Printed Circuit Board) of the target system. Through our experiments, we find that securely erased data on e-MMCs is still recoverable by extracting the internal memory data and post-processing it. Our main achievements presented in this paper are:

- We demonstrate the procedure to identify the physical structure inside an e-MMC and to extract the physical data from the internal flash memory
- We design an architecture where both e-MMC protocols and internal flash memory protocols are controlled directly
- We experimentally validate the techniques to prove that data is still recoverable after it is securely erased on the target e-MMC.

The rest of the paper is structured as follows. In section II, the structure of an e-MMC is explained. In section III, methods to access the internal flash memory inside an e-MMC are described. Then our experimental evaluation of data recovery from physical data is shown along with the test procedures in section IV and V. Finally, the results are discussed in section VI.

II. e-MMC Internals

In this section, we describe the overview of the e-MMC architectures which are defined by JEDEC Solid State Technology Association [13]–[15]. e-MMCs are typically packaged into 12mm×16mm, 12mm×18mm, 14mm×18mm, or 11.5mm×13mm BGA(Ball Grid Array) IC package.

A. e-MMC Organization

Fig. 2a and Fig. 2b each shows the top- and bottom- view of an e-MMC in 11.5mm×13mm package. 30 of the 153 ball connectors shown in 2b are assigned to the standardized signals, including Vdd and GND. The assigned pin names are shown in Fig. 2c. When controlling an e-MMC, the host system sends commands to the e-MMC through those signal pins. Signals required for communicating with the e-MMC are CMD (Command), CLK (Clock) and DAT[0-7] (Data). The CLK is the input signal from the host system, while CMD and DAT are the shared bidirectional signal between the host controller and the card. To control an e-MMC, the host system sends a 48-bit long command on the CMD line. Each bit of the command is sampled at the rising edge of the clock signal on CLK line. Responding to the commands from the host system, the e-MMC returns a response on the CMD line. The length of the response can be either 48 bits or 136 bits. When reading and writing data, the data is transferred on the DAT lines after the command is acknowledged by the e-MMC. The data bus width is configurable either 1 bit (using only DAT[0]), 4 bits (DAT[3:0]), or 8 bits (DAT[7:0]). 64 commands are available in total [14]. In addition to the standard commands, manufacturers can also implement vendor-specific commands.

![Fig. 1. Concept of categorizing data for forensic extraction. While physical data extraction is regarded as the method which provide widest range of data, we introduce a method where wider data can be recoverable.](image-url)
Fig. 2. Example of an eMMC in 11.5mm × 13mm Package. An eMMC can be controlled by 10 signals (CLK, CMD, DAT[7:0]). Each signal is assigned to one of the 153 ball connectors that are visible in 2b. The detailed pin assignment is shown in 2c.

Vendor-specific commands are typically implemented for special operation such as Smart report and RAM read/write [16].

An eMMC package consists of flash memory and a flash memory controller. The concept of the eMMC internal structure of an eMMC is shown in Fig. 3. As seen from the Fig. 3, the eMMC interface described above is the interface of the flash controller. The commands and data sent from the host system are translated by the flash memory controller, which then performs data read and write to the flash memory, where the actual data is stored in a non-volatile manner. Traditionally, flash memory and the memory controller have been packaged into individual IC chips. Therefore, consumer device vendors are required to place two components when designing their products. Furthermore, flash memory controlling protocols differ by vendors/products. With development of eMMC, device manufacturers can not only use wider area of the circuit board, but also can simplify the designing process by using the standardized controlling protocols of a storage media.

B. Flash Memory Controller

As discussed above, the flash memory controller in an eMMC is responsible for communicating with the host device and the flash memory. The flash memory controller optimizes the communication between the eMMC and the host system, and takes care of read/write operation to the internal flash memory. Not only it works as a flash translate layer (FTL) [17], the flash memory controller performs other operations in order to optimize the performance, as well as to keep the integrity of the stored data. From the digital forensic standpoint, the following operations performed by the flash memory controller are most important, as each operation modifies the original data prior to storing it to the flash memory.

- **Error correction:** NAND flash memory data reliability issues have been widely researched, and it is known that multiple bit errors occur to the stored data in NAND flash memory [18], [19]. Therefore, flash memory controllers are required to have error correcting capabilities implemented. The required error correction capability is specified by the flash memory vendors. Error Correction Codes (ECC) such as Bose-Chaudhuri-Hocquenghem (BCH) codes are popularly implemented by the flash memory controller inside e-MMCs. The parameters used in the BCH are vendor-specific and vary by models. The flash memory controller computes the ECC when storing data into flash memory, and stores the computed codewords along with the data itself.

- **Data randomization:** In a flash memory cell, data is stored in the form of electrical charge. Then those cells are connected in serial in flash memory to form pages and blocks (section II-C). Therefore low entropy in data creates imbalanced charge states between neighboring cells, making stored data more susceptible to bit errors. In order to reduce the cell-to-cell interference, the flash memory controller transforms the data into a random values before storing it into the flash memory [20]. The main purpose of data randomization is to avoid the same data being stored in succession of flash memory addresses. The flash controller inside e-MMC transforms the data by computing the XOR value between the data and the pseudo-random data. Linear Feedback Shift Register (LFSR) is typically used to produce the pseudo-random data. The structure and the initial value of the LFSR vary by models.

- **Wear Leveling:** Bit error count in flash memory increases along with the number of write/erase cycles [21], [22]. Therefore, in order to extend the lifetime of the flash memory, the flash memory controller normalizes the
TABLE I  NAND FLASH MEMORY BASIC SIGNALS

<table>
<thead>
<tr>
<th>Pin Name*</th>
<th>Input/Output</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WP_n</td>
<td>Input</td>
<td>Write Protect</td>
</tr>
<tr>
<td>ALE</td>
<td>Input</td>
<td>Address Latch Enable</td>
</tr>
<tr>
<td>CE_n</td>
<td>Input</td>
<td>Chip Enable</td>
</tr>
<tr>
<td>WE</td>
<td>Input</td>
<td>Write Enable</td>
</tr>
<tr>
<td>RY/BY_n</td>
<td>Output</td>
<td>Ready/Enable</td>
</tr>
<tr>
<td>RE_n</td>
<td>Input</td>
<td>Read Enable</td>
</tr>
<tr>
<td>CLE</td>
<td>Input</td>
<td>Command Latch Enable</td>
</tr>
<tr>
<td>I/O(7:0)</td>
<td>Input/Output</td>
<td>Data Input/Output</td>
</tr>
</tbody>
</table>

* "_n" after the signal name means that the signal is active low.

write/erase count of each physical flash memory address. To help translate the flash memory physical addresses into the logical image addresses, the flash memory controller records the logical address along with the actual data into the flash memory.

The original data sent from the host system to the e-MMC is processed at least with above mentioned operations, then the flash memory controller stores the processed data into flash memory. Therefore, when recovering the original data from the flash memory for digital forensic purposes, a forensic examiner needs to re-process the data after directly extracting it from flash memory. The typical flows of procedures are: bit error correction with ECC, data de-randomization with LFSR, and then data sorting using the logical block numbers. Identifying the required parameters for each operation requires mathematical approach, as reported in [23].

C. Flash Memory

Flash memory holds actual data after it is transformed by the controller as described in the previous section. In flash memory, data is stored in a form of electrical charge in transistors, which are typically called memory cells. The amount of trapped charge in a cell decides the data value. The data value of each flash memory cell can be 1 bit (0 or 1) for single level cell, 2 bits (0b00 to 0b11) for multi-level cell, or more bits depending on the defined technology of the target memory. Flash memory can be controlled by the signals shown in Table I [24].

Multiple CE_n and/or RY/BY_n signals may exist in one e-MMC. When multiple flash memory dies (a die is a fabricated integrated circuit) are implemented in an e-MMC, CE_n is used as a multiplexer to activate the target die, while other signals are typically shared among multiple dies. Reading or writing operation is performed by page in flash memory. A page is a group of memory cells that are connected in serial. While the page size varies on each model, typically 1 page consists of 4K to 16K bytes of data in modern flash memory. When erasing data in flash memory, however, it needs to be performed at block granularity. A block consists of multiple pages which are typically 128, 256 or 258 pages depending on the structure of the target flash memory. Due to the difference in operation granularity between writing and erasing, even when the data erase command is issued from the host system to the e-MMC, most of the time the actual data stored in the flash memory cannot be erased right away. It is typically the case that the flash memory block which stores the erased data in one of its pages also stores other valid data in other pages. In order to erase data at the flash memory level, the flash memory controller first need to move the valid data into another block, then erase the entire block of data. Therefore, there is a great chance that erased data at the e-MMC level could still remain in flash memory.

D. Erasing Data on an e-MMC

Several commands are available in the e-MMC standards to perform data erasure. By sending one of the following command sets, the host system can directly delete data stored in the target e-MMC [13], [14].

- **Erase**: Erase operation, along with other secure data purge operation, typically uses the combination of three commands: ERASE_GROUP_START(CMD35), ERASE_GROUP_END(CMD36), and ERASE(CMD38). The host defines the start and end addresses of the blocks to be deleted using CMD35 and CMD36. Then issues CMD38 in order to execute the appropriate erase operation. In Erase operation, no argument is set in CMD38. After receiving CMD38, the controller either deletes the data immediately, or at a convenient time.

- **Secure Erase**: Same as the Erase command, secure erase uses CMD38 together with CMD35 and CMD36. When issuing CMD38, argument 0x80000000 is set in order to make the e-MMC controller perform the Secure Erase operation immediately. If there exist copies of the data that is to be erased, the controller also deletes them once CMD38 is issued.

- **Trim**: Trim is similar to Erase command. Addresses are set by CMD35 and CMD36, and the Trim operation is performed by issuing CMD38 with argument 0x00000001. Once Trim is performed, the data in the target addresses is overwritten by 0 or 1. The controller can decide either performing the operation immediately, or at a convenient timing. Since Trim itself is not a secure data purge operation, it is typically performed in combination with Sanitize operation.

- **Secure Trim**: Secure Trim is performed in two steps. In the first step, the addresses where the data to be purged is stored are marked with CMD35 and CMD36. Then CMD38 with argument 0x80000001 is issued to complete the first step. Data erase is not performed at this point. This first step can be repeated until all the addresses to be trimmed are flagged. In the second step, again CMD35, 36 with addresses are issued, followed by CMD 38 with argument 0x80008000. By issuing this command, the secure purge is performed on the marked blocks. Any copies of those blocks are also purged. While CMD38 performs the erase operation right away in Secure Erase, in Secure Trim, the actual erase operation is held until the second step commands are issued. Therefore, Secure Trim can be more efficient than Secure Erase when erasing multiple fragmented erase groups [25].

- **Sanitize**: Sanitize command was introduced in the e-MMC version 4.5. In [14], the Sanitize operation is
defined as the command which physically removes data from the unmapped user address space. The sanitize operation can be initiated by writing a non-zero value into “SANITIZE_STAR” register of the extended CSD (Card Specific Data) by using SWITCH (CMD6) command. Sanitize command is recommended to use in combination with Trim command.

In [14], it is stated that e-MMC version 4.51 and higher should use Sanitize combined either with Erase or Trim instead of Secure Erase and Secure Trim. After the Sanitize operation, data should be physically removed from the unmapped user address space.

III. EXTRACTING DATA FROM FLASH MEMORY IN AN e-MMC

Since the physical data of an e-MMC is stored in the internal flash memory, acquiring the physical flash memory data lets us analyze the remnants of data after data purge operation is performed through e-MMC interface. Data recovery procedure typically requires the steps shown in Fig. 4.

For analyzing an e-MMC whose structure is unknown, typically the procedure starts with identifying the internal physical structures. Once identified physically, logical signals are examined in order to connect the target device to a flash memory reader properly. After extracting the raw data through flash memory interface, an forensic examiner needs to process the physical data to correct bit-errors, unmask randomization, and re-order the data to the logical addresses. For the detailed post data processing procedures, we refer the readers to previous research reported in [23] and [26]. In this section, we demonstrate the techniques to identify the flash memory interface in an e-MMC for physical data extraction.

A. Identifying Flash Memory Signal Connectors

The first step to access the flash memory inside an e-MMC is to identify the traces where the signal lines of the flash memory dies are connected. During the e-MMC device packaging process, the flash memory die and the flash controller die are connected to the printed circuit board (PCB) through bonding wires, then the device is assembled as an IC device with epoxy molds. An image of the cross-sectional view of an e-MMC is shown in fig. 5.

As metal material is used in bonding wires and in the PCB, the electrical path between the flash memory dies and the PCB can be visually observed through X-ray radiographic inspection. Fig. 6 shows the example of the X-ray inspection image of an e-MMC. Notice that only dense materials such as metals are visible. As Fig. 6 is taken from the top side of the target e-MMC, the bonding wires are shown as thin straight lines. By tracing the PCB trace where each bonding wire is connected, an examiner can find through which connector the flash memory is accessible. Caution must be used when using the X-ray for tracing the traces. X-ray energy can damage the flash IC, and can affect the charging state of flash memory cells, potentially damaging the stored data [27], [28]. Therefore the path tracing of flash memory in an e-MMC should be performed with a reference device when performing it for forensic purposes. After tracing the physical path from each bonding wire connected to the flash memory, the signal name of each pin can be determined by monitoring each signal with logic analyzer. A logic analyzer lets an examiner monitor the signal bus while the target device is operating. In the process of identifying the signal pins, all the non-standard pins can be connected to a logic analyzer. Then, the commands and responses between the flash memory controller and the target flash memory can be monitored. By identifying the known command and response patterns (such as chip ID read), the signal name of each connector can be identified. Through this procedure one can find which pin is connected to the signal listed in Table I. Through those procedures, we have identified connectors that are connected to internal flash memory on e-MMCs from multiple manufacturers. Typically, flash memory signal pins are connected to the connector pads that are visible on the bottom side of a eMMC package. The locations of flash signal pins on e-MMCs by multiple manufactures can be found below.

1) Samsung: Fig. 7a shows the bottom view of Samsung KLMBG4GEAC. Notice that there exist multiple connectors at every corner of the package, which are not included in the ball connectors shown in Fig. 2c. By tracing the internal connections of flash memory, we find that all the flash signal pins are connected to those non-standard connector pads.
Namely, the pads indicated in red square are connected to the signals shown in Table I. Through our experience, flash memory in all the Samsung e-MMCs is accessible through those non-standard connectors at the corners. The location of those non-standard connectors, along with the number, differ by product. Therefore tracing of flash signal pins is required every time a forensic investigator analyzes the different model of Samsung e-MMC, along with “bus-sniffing” in order to decide which signal pin is connected to each connector.

2) SanDisk: Fig. 7b shows the bottom view of SanDisk SDIN7DU2. There are non-standard connectors (shown in gold color) surrounding the standard 169-pin connectors. Flash memory connection can be traced to those pins, and based on our experience, SanDisk e-MMC always follow the same flash memory footprint. Connectors indicated with red circles are connected to the internal flash memory.

3) Toshiba: In Toshiba e-MMCs, non-standard connectors are typically under the coating layer, and they are only accessible after removing the solder masks. An example pinout is shown in Fig. 7c. Pins in red squares are connected to the internal flash memory. One can access the flash memory by exposing those connectors. At the same time, flash memory in Toshiba e-MMCs are also accessible through the standard 169-pin connectors. One of the standard pins (labeled as N2 in Fig. 2c, also indicated in blue circle in Fig. 7c) is used as the multiplexer input, which determines whether the card operates as an eMMC or a flash memory. If N2 pin is tied to Vss as defined in [15], the card operates as an eMMC. If N2 pin is pulled high, however, the card operates as a flash memory. The tracing of the internal connection reveals that connectors indicated with red circles in Fig. 7c are used for accessing the flash memory. Data bus lines are shared between the e-MMC protocol and the flash protocol on Toshiba e-MMCs.

B. Flash Memory Data Extraction

Once the locations of all the flash memory signal pins are identified, the internal data can be extracted by issuing the correct data read commands to the memory through identified signal connectors. Flash memory readers such as NFI MemoryToolKit [29], Rusolut Visual Nand Reconstructor [30], and Soft-Center Flash Extractor [31] can help examiners read the target flash memory and extract the raw data from the whole memory region.

IV. EXPERIMENTS

To investigate the data recovery rate from raw flash inside an e-MMC, we conducted two lab-controlled tests. First, we performed data recovery from Android smartphones. Second, we evaluate the effect of the e-MMC data purge operation to the internal flash memory. The detailed procedures of each test are discussed in this section.
A. Deleted Android Data Recovery

We select Xperia Z2 as our target to perform data recovery, whose data is stored with clear-text in the storage media. We purchased 6 devices from the second-hand market, within which user data still remained on the device. Sandisk SDIN7DP2-4G was mounted on each target device as the main storage media. The e-MMCs were detached from the PCB of each device, and then we first extract the physical image through the e-MMC interface. Data extraction was performed with the command “dd” after the target e-MMC is connected to a PC through an SD/MMC card reader. Following the e-MMC data extraction, we proceed to extract the physical data from the internal flash memory by connecting the flash memory reader to the flash interface on each e-MMC described in the previous section.

After acquiring the raw flash data, we process the data by performing error correction and data de-randomization to recover the original plain-text data. In order to simplify the quantification process of the data recovery comparison, we carved out the SMS messages from the acquired data using the header information. The number of recovered SMS messages are compared between the ones extracted from e-MMC interface, and the ones extracted from the raw flash data.

B. Data Recovery After Data Purge Operations

As discussed in Section II-D, multiple data purge operations are defined in e-MMC specifications. In order to investigate the impact of erasing on the e-MMC interface to the physical data, we copied the same physical e-MMC image to multiple e-MMCs from various manufacturers, and then performed data purge operation through e-MMC interface. Table II shows the list of e-MMCs used for the test. We selected Samsung, SanDisk, and Toshiba devices since we most often encounter those manufacturers’ e-MMCs in our casework. Each target e-MMC was labeled with the number shown in Table II. We use those numbers to refer each target for the rest of this paper.

4GB of Android image data was used as the original data. The data was extracted from one of the phones used for the experiment described in section IV-A. The image data includes 27 partitions, within which the userdata partition has the largest volume of 2.1GB. After confirming that data copy was successful, we issued data purge commands against userdata partition, followed by raw flash memory data extraction. We performed Secure Erase, Secure Trim, and Sanitize combined with Trim, to evaluate the impact of each operation.

For each test, we compared the raw data extracted through the flash memory interface with the original userdata partition data stored in the e-MMC. The comparison was carried out by each sector. To observe the accurate rate of possible data recovery, the number of sectors whose data still remain on the raw flash memory was counted. The physical sector address of the userdata partition in the e-MMC is between 0x31E000 and 0x747FDE, which consists of 4,366,302 512-byte long sectors. Out of 4,366,302 sectors, 920,717 blocks do not contain data (The whole block is either all logical 0 or 1). In order to count the correct number of sectors that remained in flash memory after data purge operation, we counted the recoverable data-containing sectors remained in flash memory. The rates of recoverable sectors are therefore calculated against 3,445,585 sectors.

C. Testing Environment

In order to issue the e-MMC commands directly to the target e-MMCs and control the data write and erase, we developed an e-MMC programmer with Raspberry Pi using its GPIO [32]. The e-MMC programmer first initialize the target card, which is necessary to make the card ready for IO communication. Then the required data purge commands are issued. For initial data image storing to the target e-MMC, we used the “dd” command. For acquiring the raw flash memory data, we used Rusolut Visual NAND Reconstructor throughout the experiment. Figure 8 shows the connection between one of the target e-MMC and the flash memory reader.

V. RESULTS

A. Deleted Android Data Recovery

Table III shows the overall counts of recovered SMS messages both from the e-MMC image and from the internal flash memory data. Here, the eMMC data is equivalent to the physical data described in section I. The number of SMS messages retrieved from eMMC thus already includes logically deleted data. If the traditional physical acquisition was performed and data is acquired through e-MMC interface, the numbers listed under SMS count on e-MMC in Table III is the maximum number of messages that can be recovered. Comparing it with the number shown in SMS count on flash, it is clear that traditional physical data acquisition through e-MMC interface is still missing recoverable data remaining in the internal flash memory. More message data is still available on the flash memory than what can be carved out from the e-MMC. Note that those messages that are not available on
TABLE III
SMS DATA RECOVERY FROM eMMC AND THE INTERNAL FLASH MEMORY

<table>
<thead>
<tr>
<th>Target</th>
<th>SMS count on eMMC</th>
<th>SMS Count on flash</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>41</td>
<td>116</td>
</tr>
<tr>
<td>B</td>
<td>19</td>
<td>47</td>
</tr>
<tr>
<td>C</td>
<td>20</td>
<td>105</td>
</tr>
<tr>
<td>D</td>
<td>4,723</td>
<td>4,866</td>
</tr>
<tr>
<td>E</td>
<td>82</td>
<td>118</td>
</tr>
<tr>
<td>F</td>
<td>1,174</td>
<td>1,540</td>
</tr>
</tbody>
</table>

the e-MMC is not accessible from the host system. Those messages extracted from the raw flash data therefore contain the ones deleted by the user or by the system, and cache data created by the internal flash memory controller [33].

Actual user data recovery rate depends on how each application software and the operating system of the device take care of the deleting operation. Therefore it is not feasible to directly extrapolate the userdata recovery rate from this experiment. However it is obvious that more data can be recovered if the internal flash data is explored than what is extracted through eMMC interface. For target devices A, B, and C, more than twice of user data is recovered from the internal memory. This result shows that the chance of finding deleted data can be increased by accessing the internal flash memory of an eMMC.

B. Data Recovery After Data Purge Operations

Given the results of data recovery rate from e-MMCs used in real Android devices, it is safe to assume that good amount of data still remains in the internal flash memory. Now we evaluate data recovery rate after different data purge operation is performed on the e-MMC. Through e-MMC interface, we performed Secure Erase, Secure Trim, and Sanitize against the userdata partition at e-MMC address between 0x31E000 and 0x747FDE. Viewing the physical data through eMMC interface showed us that the whole partition was overwritten with logical 0s after erasure. Therefore no user data can be recovered from physical data acquired through e-MMC interface if any of the purge operation were performed against the userdata partition. This means that, if any of the purge operation is performed on an Android device used in the previous section, the number shown in as SMS count in e-MMC in Table III becomes 0.

After performing data purge operation to the userdata partition, raw flash memory data was extracted through identified flash interface. Data recovery rate from internal flash memory after each data purge operation is shown in Table IV. Here, the number of recoverable sectors through flash memory interface is listed. Recovery rate is calculated against the total number of original data-containing sectors, which is 3,445,585 sectors.

As shown in Table IV, almost all the securely erased data still remains on the target e-MMC 2 and 3, from which the original userdata partition data can be recovered. Despite the difference of definition in Secure erase, Secure Trim, and Sanitize, the behavior of those data purge operation showed the similar results. Even after Sanitize command, which is the most secure data purge operation, most of the data remains in flash memory in the target e-MMC 2 and 3. On the other hand, if the Secure Erase is strictly implemented, majority of the data is wiped from the internal flash memory as can be seen from the result of target e-MMC 1, 4, and 5. For those devices, if data purge operation was performed from the host system, chances of recovering deleted data are significantly low even if the data is extracted from internal flash memory.

During the data purge operation, we also monitored the operation time by observing the “busy” signal returned from the e-MMC during the operation. After receiving the Erase command, the e-MMC returns a response and pulls D0 line low while performing the operation. We observed that target e-MMC 1 and 3 take longer erase operation time than other devices. However, the operation time and data recovery rate do not correlate since even after long operation time of target 3, majority of data is still recoverable.

VI. DISCUSSION

Through our experiments, it is now obvious that securely erased data can still be recovered from the internal flash memory in some e-MMC models. Given the large volume of different e-MMC models on the market manufactured by multiple manufacturers, it should be however noted that more intensive testing with much larger scale is required to build a comprehensive data recovery rate of securely erased e-MMCs. Different chip model contains different flash controller architecture, therefore internal flash memory data is treated differently. Nevertheless, extracting raw data from e-MMCs can still be effective when thorough investigation is necessary. In this section, we discuss how our results can be interpreted in digital forensic domain, as well as other environmental factors that can affect the data recovery rate.

A. Forensic Interpretation and Application

In [34] and [35], ioctl(BLKSECDISCARD) is recommended as the best practice for data sanitization process in Android factory reset. This operation issues one of the data purge commands that are used in our experiment: Secure Erase (or Secure Trim) or Trim combined with Sanitize, depending on the target e-MMC version. While each operation makes the erased data inaccessible through the e-MMC interface, as we have shown through our experiment, there is a great chance that most of the original data still remains in the underlying flash memory. Note that the flash controller is allowed to perform data erasure at its convenient timing for some of the erase commands. In [14], it is defined that “Erase” and “Trim” commands allow controllers to perform the erase at its convenient timing. In case the flash memory controllers in our target e-MMCs are configured to perform the actual erase at a system down time, we left the target e-MMCs powered on after data erase operation with the clock signals on. However, even after waiting for over an hour and repeating power cycles, the data recovery rate was unchanged. Additionally, in [12], it is proposed to execute Trim/Discard operation as a background operation in order to reduce latencies during device operation. Therefore we also send the command to manually start the background operations on the target e-MMCs. However the results remain unchanged. Based on those observations, we conclude that
all the data erasure operations are completed at the time we performed the data recovery, and no more data deletion will be performed at a later timing on the target devices. While our approach proposed in this paper is applicable to all the devices in which e-MMC is mounted, forensic examiners need to consider applying the method depending on its requirements and prioritization. Given its intense reverse-engineering in both hardware and software, the operation is time-consuming. Furthermore, details of the e-MMC controller operation are proprietary to the manufacturers, and they vary by the models. With backlog being a commonplace at digital forensics labs [36], [37], examiners need to carefully select the target eMMCs on which raw flash memory extraction to perform. There is also a great chance that the data is already destroyed at physical level if data purge operation is implemented strictly by the manufacturer. As discussed in section I and section V, recovery of deleted data is possible to some extent through file system extraction and physical extraction, which is performed through e-MMC interface. If the target data is still missing after examining the extracted data through those method, then the physical data extraction from the internal memory of e-MMCs should be considered as a next step.

B. Other Factors Affecting Data Recovery Rate

While the experimental data recovery rate is promising, various environmental factors needs to be considered when attempting data recovery from e-MMCs mounted on digital devices. The actual operating environment may greatly affect the data recovery rate. Some of the operational factors include the following:

1) Available Physical Space in the Flash Memory: Imagine a case where the eMMC in a target device is almost full, and there is not much space for a large file to be stored. In order to store some new data, the user need to discard some of the existing files. When storing a new file, since there is not much space left at the physical level, the flash controller has no choice but erase the old data remained in the flash memory. When this is the case, the recovery rate of erased data can be low. In order to prove this hypothesis, we overwrite all the blocks of the target eMMCs with logical 0s. Post data extraction from flash memory shows that only fractions of old data (less than 0.1%) remains on the flash memory after overwriting all the blocks.

2) Encryption: If the original data is encrypted when it is stored on an e-MMC, the data is also stored in the flash memory in an encrypted form. Therefore, even if the original data can be reconstructed from the data remained in flash memory, data decryption procedures are required. This operation may be challenging unless the whole data partition can be recovered error-free, and decryption method is established.

3) Data Fragmentation: In our experiments, the whole image was copied to the target e-MMC right before the data erasure was performed. In a real digital device, however, data write and erase operations are constantly performed to the e-MMC. Every time the existing file is updated, new data is written into a new physical address, leaving the old data in the existing location. As a result, fragmented data can be found in multiple locations within flash memory. While it may be hard to recover the complete files from those fragments, investigators may find some meaningful data (such as metadata) from those locations.

VII. RELATED WORK

To our knowledge, this is the first work to investigate the data recovery rate of e-MMCs after their data is securely erased. This work can be regarded as the complimentary to the previous works by [9] and [26], which demonstrate the thorough data recovery procedures from flash memory. In their research, the authors explained how to extract and reconstruct the user data from raw flash memory mounted on digital devices. Since flash memory is now embedded inside the e-MMC and other managed flash, our approach is required in addition to the techniques described in their work.

The possibility of privacy data leakage is discussed in [33] and [38]. Data management in programming and erasing operation to flash memory apparently leaves valid data in unmapped memory block, leading possible data leakage when accessed by adversaries. The problem pointed out in those works have been already explored with SSDs in [39]. In their work, authors also report that some SSD manufacturers implement secure erasing features incorrectly, leaving erased data on flash memory in a way that can be recoverable by adversaries.

Additionally, the poor data sanitization practice of flash and e-MMC based devices in the IC recycling ecosystem is reported in [40]. The authors claim that old data which belonged to the previous device still remains after the target chip is re-implemented into a new device. While old data was recovered from second-hand e-MMCs through this research, the authors did not investigate inside the e-MMC. Given that no proper data purge operation was performed on the e-MMCs before being recycled into new devices, we assume that more data than reported in [40] can be recoverable by following our approaches.

VIII. CONCLUSION AND FUTURE WORK

We have shown that securely erased e-MMC data can still be recoverable by directly extracting the data from internal flash

<table>
<thead>
<tr>
<th>Target IC</th>
<th>Secure Erase Number of Recoverable Sectors after Different Erase Operations (Rate(%)) Secure Trim</th>
<th>Sanitize</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>935 (&lt;0.001 %)</td>
<td>926 (&lt;0.001 %) NA (operation not supported)</td>
</tr>
<tr>
<td>2</td>
<td>3,425,560 (99.41 %)</td>
<td>3,425,032 (99.40 %) 3,424,652 (99.42 %)</td>
</tr>
<tr>
<td>3</td>
<td>3,444,362 (99.86 %)</td>
<td>3,444,223 (99.86 %) 3,444,424 (99.97 %)</td>
</tr>
<tr>
<td>4</td>
<td>126,033 (3.66 %)</td>
<td>125,966 (3.66 %) NA (operation not supported)</td>
</tr>
<tr>
<td>5</td>
<td>13,376 (0.0039 %)</td>
<td>11,376 (0.0033 %) 10,231 (0.0030 %)</td>
</tr>
</tbody>
</table>
memory. Extracting the flash memory data and processing it into error-free clear text require intensive investigations of the target eMMC, as the internal structure and the embedded technology vary by manufacturer and model. Nevertheless, raw flash memory data can be extracted from the target eMMC without damaging its physical structure and the function. The techniques described in this paper can be applied to wide range of digital devices, including factory reset Android devices. Data deleted by the host system may still remain on the internal flash memory of an eMMC in a recoverable manner. Since the underlying flash memory technology remains unchanged even in more advanced managed flash memory such as UFS, we assume that similar outcome can be observed in other managed flash memory devices. Those advanced managed flash memory devices can be good candidates as the future target for further analyzing the internal structure for forensic data recovery purposes.

REFERENCES