Digital Investigation 20 (2017) S1–S11

Contents lists available at ScienceDirect

Digital Investigation

journal homepage: www.elsevier.com/locate/diin

DFRWS 2017 Europe — Proceedings of the Fourth Annual DFRWS Europe

Improving the reliability of chip-off forensic analysis of NAND flash memory devices

Aya Fukami [a, b, *], Saugata Ghose [b, *], Yixin Luo [b], Yu Cai [b], Onur Mutlu [c, b, *]

[a] National Police Agency, Japan
[b] Carnegie Mellon University, United States
[c] ETH Zürich, Switzerland

ARTICLE INFO

Article history:
Received 26 January 2017
Accepted 26 January 2017

Keywords:
NAND flash memory analysis
Chip-off analysis
Digital forensics
Read-retry
Memory errors
Memory reliability

ABSTRACT

Digital forensic investigators often need to extract data from a seized device that contains NAND flash memory. Many such devices are physically damaged, preventing investigators from using automated techniques to extract the data stored within the device. Instead, investigators turn to *chip-off analysis*, where they use a thermal-based procedure to physically remove the NAND flash memory chip from the device, and access the chip directly to extract the raw data stored on the chip.

We perform an analysis of the errors introduced into multi-level cell (MLC) NAND flash memory chips *after the device has been seized*. We make two major observations. First, between the time that a device is seized and the time digital forensic investigators perform data extraction, a large number of errors can be introduced as a result of charge leakage from the cells of the NAND flash memory (known as *data retention* errors). Second, when thermal-based chip removal is performed, the number of errors in the data stored within NAND flash memory can increase by two or more orders of magnitude, as the high temperature applied to the chip greatly accelerates charge leakage. We demonstrate that the chip-off analysis based forensic data recovery procedure is quite destructive, and can often render most of the data within NAND flash memory uncorrectable, and, thus, unrecoverable.

To mitigate the errors introduced during the forensic recovery process, we explore a new hardware-based approach. We exploit a fine-grained read reference voltage control mechanism implemented in modern NAND flash memory chips, called *read-retry*, which can compensate for the charge leakage that occurs due to (1) retention loss and (2) thermal-based chip removal. The read-retry mechanism successfully reduces the number of errors, such that the original data can be fully recovered in our tested chips as long as the chips were not heavily used prior to seizure. We conclude that the read-retry mechanism should be adopted as part of the forensic data recovery process.

Introduction

NAND flash memory continues to increase in popularity as a storage medium for a wide range of devices, such as smartphones, thumb drives, and solid-state drives (SSDs). As a result, digital forensic investigators have been encountering significantly more NAND flash memory based devices than before during the course of criminal investigations. When an operational NAND flash memory based device is received for analysis, investigators can use *logical data extraction*, where data can be read out using an interface provided by the device vendor (Ayers et al., 2014). Commercial software-based forensic acquisition tools automate logical data extraction (Ayers et al., 2014), and can yield sufficient data from the device. Unfortunately, a device received as part of an investigation may be physically damaged, or the device may not provide an interface for data acquisition, and as a result, its data may be inaccessible using the automated software-based approach. In these cases, digital forensic investigators must physically remove the NAND flash memory chip from the printed circuit board (PCB) inside the device (ENFSI, 2015). Once the chip has been removed, investigators can perform low-level analysis on the chip, at which point the data that was originally on the chip can potentially be recovered. This analysis method is commonly referred to as *chip-off analysis*.

* Corresponding authors. Carnegie Mellon University, United States.
E-mail addresses: afukami10@npa.go.jp (A. Fukami), ghose@cmu.edu (S. Ghose), onur.mutlu@inf.ethz.ch (O. Mutlu).

http://dx.doi.org/10.1016/j.diin.2017.01.011

Previous research on forensic low-level analysis of NAND flash memory chips has focused on reverse engineering the techniques implemented by the original NAND flash memory controllers, in order to access the data residing on the chip. Breeuwsma et al. (2007) established a thorough forensic data recovery procedure from a NAND flash memory chip, going from acquiring the physical image of the data on the NAND flash memory chip to reconstructing the file system used to store the data, by reverse engineering multiple techniques that are implemented in NAND flash memory controllers. For relatively old devices, where the number of raw bit errors that occurred within the device are low, this data recovery procedure is often sufficient. However, as NAND flash memory has scaled down aggressively in process technology node size to enable higher storage capacity, the number of raw bit errors that occur in NAND flash memory has increased by several orders of magnitude, as demonstrated experimentally in Cai et al. (2012a, 2013a,c). As a result, the procedure proposed by Breeuwsma et al. (2007) often cannot adequately recover the data in modern NAND flash memory.

In order to ensure that data retrieved from NAND flash memory by an end user does *not* contain any errors despite the increasing occurrence of raw bit errors, modern NAND flash memory controllers employ sophisticated *error-correcting codes* (ECC), such as BCH codes (Micheloni et al., 2008; Hocquenghem, 1959; Bose and Ray-Chaudhuri, 1960) or LDPC codes (Gallager, 1963; Zhao et al., 2013). These codes can correct up to a fixed number of raw bit errors for every read operation. Likewise, to maintain the integrity of digital evidence extracted from a device that uses NAND flash memory as its storage medium, forensic investigators need to correct errors that appear in the raw data extracted from the device. Thus, it is essential for the forensic data recovery procedure to extract the ECC information stored within the chip and use this information to correct the errors. In addition to ECC, many modern flash controllers employ *data randomization* techniques, where data that is written into memory is scrambled by XORing the data with a reproducible pseudo-random number, to reduce the impact of data value dependence on reliability (Cha and Kang, 2013; Kim et al., 2012). van Zandwijk (2015) provides a mathematical approach to reverse engineer *both* the ECC and data randomization algorithms from the raw data that is extracted from the NAND flash memory chip.

However, even when the ECC and data randomization algorithms are correctly identified and used to decode the raw data extracted from the NAND flash memory chip, digital forensic examiners may find that many chunks of data contain more errors than the ECC algorithm is able to correct. The ECC codeword contains only enough information to correct up to e bits within an n-bit data chunk (we refer to e as the *error correction capability*). If the data chunk contains more than e errors, the errors are *uncorrectable*, and the data cannot be successfully recovered. This compromises the integrity of the recovered data for forensic analysis. Our goal in this work is to develop a methodology for chip-off analysis that improves the likelihood of reliable data recovery beyond what is possible by using ECC.

To this end, this paper first investigates the impact of current forensic techniques on the number of raw bit errors that exist within a NAND flash memory chip that is removed from the PCB in a device under investigation. In order to perform chip-off analysis, forensic investigators follow best practices used by electronics manufacturers for their *rework process*, where manufacturers remove and replace faulty components in their products (JEDEC, 2007). This procedure uses hot air to heat the chip just enough to melt the solder that connects the chip to the PCB, which allows the safe removal of the chip. We refer to this technique as *thermal-based chip removal*. We demonstrate that, even though the temperature used during chip removal is the minimal temperature necessary for the solder to reach its melting point (usually more than 200 °C), this temperature is still high enough to introduce a very large number of *new* raw bit errors into the chip. Unfortunately, since the ECC data stored on-chip has a fixed error correction capability, the newly-introduced errors are often capable of rendering much of the data on a NAND flash memory chip unrecoverable using traditional low-level analysis techniques, such as those described by Breeuwsma et al. (2007) and van Zandwijk (2015).

To mitigate the impact of the new errors introduced during chip removal, we examine exploiting the state-of-the-art dynamic mechanisms implemented within the NAND flash memory itself to help reduce the raw bit error rate. In particular, we study the *read-retry* mechanism (Cai et al., 2013c). In a NAND flash memory cell, data is stored in the form of the *threshold voltage* of the cell's floating-gate transistor (i.e., the voltage at which a transistor turns on). In order to read data from the NAND flash memory cell, a *read reference voltage* is applied to the transistor. If the read reference voltage is higher than the threshold voltage, the cell turns *on*; otherwise, the cell turns *off*. As several prior works have shown (Cai et al., 2012a, 2013a,c, 2015a; Crippa and Micheloni, 2010; Degraeve et al., 2004), the threshold voltage of a floating-gate transistor can *shift over time*, due to continuous leakage of charge from the transistor. Since the standard read operation uses the *default read reference voltage*, it is unable to account for such a threshold voltage shift, and thus the read operation introduces an error (e.g., we read a bit value 1 even though the stored value was 0). The flash controller can mitigate these errors by dynamically adjusting the read reference voltage to compensate for the threshold voltage shift. This mechanism is known as *read-retry* (Cai et al., 2013c, 2015a). Modern NAND flash memory chips include several variants of read-retry, which use different techniques to adjust the read reference voltage.

We evaluate the suitability of the read-retry mechanism for forensic analysis. Our findings show that read-retry can significantly reduce the errors inside a NAND flash memory chip that are introduced over the course of chip-off analysis (by as much as 94.6%). The read-retry mechanism eliminates uncorrectable data chunks for NAND flash memory chips that have not been heavily used and thus improves the reliability of chip-off analysis based forensic data recovery. The read-retry based error mitigation techniques discussed in this paper can be implemented when extracting a raw image directly from a NAND flash memory chip during forensic analysis using a universal flash chip reader, as described in Breeuwsma et al. (2007).

This paper makes the following contributions:

- We experimentally demonstrate that thermal-based chip removal, a commonly-used technique during the digital forensic analysis of a NAND flash memory based device, can introduce a large number of errors into the raw data stored within a NAND flash memory chip, increasing the raw bit error rate by as much as 259× (Section Errors due to thermal-based chip removal). The additional errors can often be enough to prevent investigators from successfully extracting correct data from the device, hampering digital forensic analysis.
- We experimentally evaluate how the read-retry mechanism, provided by NAND flash memory manufacturers in modern flash memory chips to control the read reference voltage in a fine-grained manner, can reduce the raw bit error rate of data read from NAND flash memory by as much as 94.6% (Section Read-retry operation).
- Our evaluations show that by incorporating read-retry based error mitigation into the forensic data recovery procedure, we can mitigate the errors introduced during thermal-based chip

removal and successfully read out the data stored within a NAND flash memory chip, unless the chip has been heavily used (Section Read-retry operation).

Background: MLC NAND flash memory

We first provide necessary background on the design and operation of multi-level cell (MLC) NAND flash memory, a very common variant of NAND flash memory available in a wide number of consumer devices today. More detailed information on the operation of flash memory can be found in Cai et al. (2012a, 2013a, 2015b, 2017); Crippa and Micheloni (2010); Friederich (2010); Micheloni et al. (2010).

Flash memory organization

NAND flash memory stores data within an array of *flash cells*. A cell consists of a single *floating-gate transistor*, where the floating gate of the transistor can store some amount of charge, as shown in Fig. 1. The charge stored (i.e., *trapped*) within the floating gate determines the *threshold voltage* at which the transistor turns on. Oxide layers are placed above and below the floating gate to prevent the stored charge from leaking out of the floating gate. To program a flash cell to a specific threshold voltage, a high voltage is applied to the transistor's *control gate*.

The threshold voltage can be programmed to a voltage level within a fixed range. This fixed range is split up into multiple *voltage windows*, or *states*, where each state represents a certain bit value. Older NAND flash memory devices were made up of *single-level cells* (SLC), where each flash cell stores a one-bit value (i.e., a 0 or a 1). In order to provide higher storage density, NAND flash memory manufacturers now use *multi-level cell* (MLC) technology. A multi-level cell stores *two* bits of data within a single cell. To do this, the voltage range is split into *four* states (ER, P1, P2, and P3), with each state corresponding to one of the data values 00, 01, 10, or 11, as shown in Fig. 2. Due to variation during programming, the threshold voltage of cells programmed to the same state is distributed across the voltage window for the state. This results in a threshold voltage distribution of flash cells across the voltage range, as we show in Fig. 2.

A NAND flash memory chip contains thousands of *flash blocks*, which are two-dimensional arrays of flash cells. Fig. 3 shows the internal organization of a block. Each block contains dozens of rows (i.e., *wordlines*) of flash cells. All of the cells on the same wordline are read and programmed as a single group. MLC NAND flash

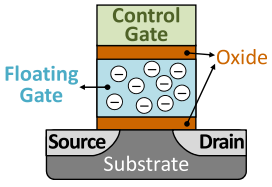

Fig. 1. A flash memory cell, which consists of a floating-gate transistor.

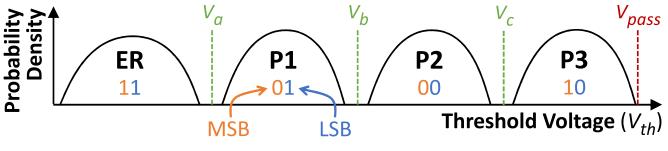

Fig. 2. Threshold voltage distribution of cells in MLC NAND flash.

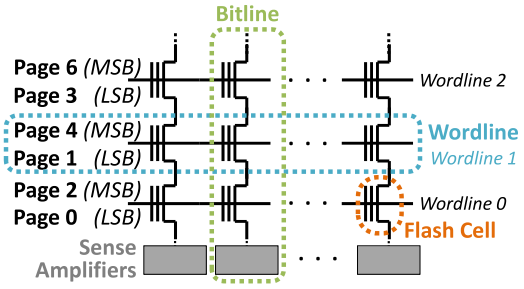

Fig. 3. Organization of a NAND flash block.

memory partitions the two bits of data in each cell across two separate *pages* (the unit of data programmed at a time). As Fig. 2 shows, the two bits stored within a multi-level cell are referred to as the *least significant bit* (LSB) and the *most significant bit* (MSB). The LSBs of all cells on one wordline form the LSB page of that wordline (e.g., Page 1 of Wordline 1 in Fig. 3), and the MSBs of these cells form the MSB page (e.g., Page 4 of Wordline 1). For 2y-nm (i.e., 20–24 nm) NAND flash memory, a single page consists of between 4 and 16 KB of data (Micron Technology, 2015). Within each column of flash cells in the block, the sources and drains of the cells' transistors are connected in series to form a *bitline*. The cells on a bitline share a common ground on one end, and are connected to a sense amplifier on the other end. Read, program, and erase operations to the block are managed by a *flash controller*.

Programming and erasing data

To program data into a flash cell, the cell needs to be in the *erased* state (i.e., no charge should be stored within the floating gate of the cell). During a program operation, electrons are injected into the floating gate by applying a high positive voltage to the control gate (see Fig. 1 for a diagram of the flash cell). NAND flash memory uses a procedure known as *incremental step-pulse programming*, or ISPP (Suh et al., 1995). During ISPP, the high programming voltage is applied for a very short period, known as a *step-pulse*. ISPP then checks the current voltage of the cell. ISPP repeats the process of applying a step-pulse and checking the voltage until the cell reaches its target threshold voltage. If we want to overwrite the data that currently exists in a cell, we must first erase the data in the cell. Within NAND flash memory, an erase operation is performed at block granularity (i.e., an entire block of flash cells is erased at once).

Over time, as a cell is programmed and erased, the cell begins to *wear out*, reducing its ability to reliably store charge within the floating gate (Pan et al., 2011; Belgal et al., 2002). As this wearout is a result of the number of times a cell is programmed and erased, we quantify the degree of wearout in *program/erase (P/E) cycles*, as done in many prior works (Cai et al., 2012a,b, 2013a,b,c, 2014, 2015a,b, 2017).

Reading data from NAND flash

To read a page of data from a block, the flash controller applies a *read reference voltage* to the cell's control gate. If the threshold voltage of a cell is lower than the read reference voltage, the cell switches *on*; otherwise, the cell switches *off*. The read reference voltage used to read a cell depends on which page is being read from the wordline. As shown in Fig. 2, to determine the LSB of a cell, the controller applies a single read reference voltage, V_b. If the threshold voltage of the cell is lower than V_b, the cell is in either the ER state or the P1 state, and holds an LSB of 1; otherwise, the cell is

either in the P2 state or the P3 state, and holds an LSB of 0. To determine the MSB of a cell, the controller applies *two* read reference voltages, V_a and V_c. The two voltages allow the controller to determine if a cell is in the P1/P2 states, and holds an MSB of 0, or if the cell is in the ER/P3 states, and holds an MSB of 1.

Since multiple cells are tied together on a single bitline, we must ensure that the cells that are *not being read* pass through the data that is being output from the cell that we want to read. In order to achieve this, the flash controller applies a *pass-through voltage* to the control gate of each unread cell (V_{pass} in Fig. 2). The pass-through voltage is higher than any threshold voltage that can be stored within a flash cell. This ensures that a cell that is *not* being read is always turned *on* during the read operation, allowing the data of the cell that we *do* want to read to successfully reach the sense amplifier.

Correcting errors

Data stored within flash cells can often contain errors. An error is introduced when the threshold voltage of a cell shifts outside of the voltage window to which the cell was originally programmed. There are a number of sources of errors within flash memory (Cai et al., 2012a, 2013a), such as retention loss (Cai et al., 2012b, 2015a), cell wearout (Cai et al., 2012a, 2013c; Luo et al., 2016), cell-to-cell program interference (Cai et al., 2013b, 2014), and read disturb (Cai et al., 2015b). As flash cells scale down to smaller process technology nodes, the total amount of charge that each cell can store decreases, which, in turn, increases the susceptibility of the flash cells to errors (Yoon and Tressler, 2012).

In order to combat the errors contained within the cells (which we refer to as *raw bit errors*), NAND flash memory makes use of *error-correcting codes* (ECC) (Cai et al., 2012a; Zhang et al., 2012). When data is programmed to a flash page, an ECC codeword is also written, which contains enough redundancy to correct e bits out of the n-bit data (van Zandwijk, 2015). We refer to e as the *error correction capability* of the codeword. When the flash page is subsequently read, the ECC codeword is sent alongside the data to the flash controller. Inside the controller, both the data and the ECC codeword are input to the ECC logic, which checks for errors using the implemented error correction algorithm. Based on the results of the algorithm, the controller fixes the erroneous bits in the data, if any, and returns the corrected data value. If the data read from the page contains no more than e raw bit errors, the controller successfully returns correct data to the end user. If the data read from the page contains more than e raw bit errors, full data correction is not possible, and the page data is said to be *corrupted* (i.e., it is uncorrectable) (Meza et al., 2015; Schroeder et al., 2016). A block that contains corrupted data is marked by the flash controller as a *bad block* (Micron Technology, 2011), and is no longer used for storing data.

NAND flash error sources during forensic analysis

As discussed in Section Correcting errors, errors can occur when the threshold voltage of a flash cell shifts. The probability of occurrence of such shifts has increased due to continued device scaling, which allows manufacturers to increase the flash storage density (Yoon and Tressler, 2012). The reliability of NAND flash memory has been widely researched (e.g., Belgal et al., 2002; Brand et al., 1993; Cai et al., 2012a,b, 2013a,b,c, 2014, 2015a,b, 2017; Chimenton et al., 2008; Degraeve et al., 2004; Luo et al., 2016; Meza et al., 2015; Mielke et al., 2008; Papandreou et al., 2014, 2015; Park et al., 2008; Parnell et al., 2014; Prodromakis et al., 2015; Schroeder et al., 2016; Yang et al., 2006). For a detailed overview of MLC NAND reliability, we refer the reader to Zambelli et al. (2010) and Cai et al. (2013a). Cai et al. (2012a) investigated multiple error factors on MLC NAND flash memory, including program/erase errors, program interference errors, retention errors, and read errors. Digital forensic investigators should assume that the majority of NAND flash memory devices they receive for analysis already contain multiple errors at the time the device is obtained.

In this section, we focus on the two major sources of errors that can be *introduced* during digital forensic analysis, *in addition to* the preexisting errors within the NAND flash memory device. First, a device is often stored unused for several days or even weeks before investigators are able to examine its contents, due to issues such as transport time or lab backlog (Casey et al., 2009). During this time, additional *retention errors* can occur, which we discuss in Section Retention errors. Second, for devices where thermal-based chip removal is required, a high temperature must be steadily applied to the device. This high temperature rapidly accelerates the effect of retention errors, as we discuss in Section Thermal effect on error rate.

Retention errors

Retention errors in a flash cell occur when the charge stored within the floating gate of cell transistor *leaks*. Due to the structure of the cell transistor (see Fig. 1), where the floating gate and substrate are separated by an oxide layer, a small amount of charge *tunnels* through the oxide, causing the leakage. This trend accelerates as the P/E cycle count increases (Cai et al., 2012a; Yoon and Tressler, 2012), as repeated programming and erasing of a cell degrades the oxide layer, which in turn allows charge to tunnel through the oxide layer at an increasing rate (Cai et al., 2015a). This tunneling occurs whether or not a NAND flash memory device is powered up, causing retention errors to accumulate when the device is stored unused with or without power.

Imagine a hypothetical case where a device was seized at a crime scene, and the device is received for analysis at a digital forensics lab three weeks later. Let us assume that this device had been used over the course of three years, and that the NAND flash memory within the device endured 10 P/E cycles each day during these three years of usage, adding up to a total of approximately 10^4 P/E cycles over its lifetime. Fig. 4 shows the relationship between the P/E cycle count and the retention error rate found by Cai et al. (2012a) for real 3x-nm MLC NAND flash memory chips. The figure demonstrates that the error rate grows with (1) the P/E cycle count and (2) the *retention age* (i.e., the time elapsed since the data was programmed). As we can see by comparing the curve labeled *1 Day* with the curve labeled *3 Weeks*, the number of errors in the device increases by 38× over the three-week period between

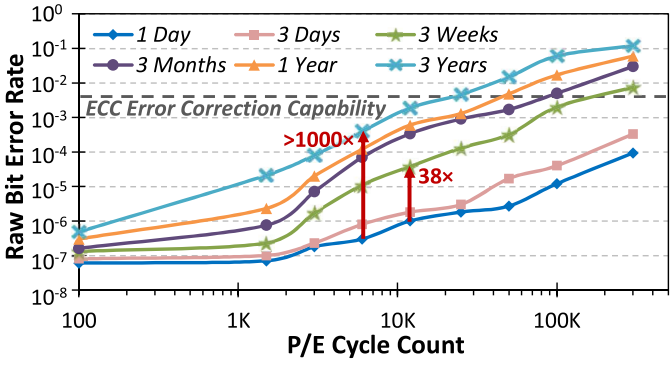

Fig. 4. Raw bit error rate of a 3x-nm NAND flash memory chip for different retention ages vs. P/E cycle count. Reproduced from Cai et al. (2012a).

device seizure and lab delivery, while the device is being stored and transported.

To provide more insight into retention errors, we perform an experimental analysis on the retention error rate of modern 2y-nm (i.e., 20–24 nm) MLC NAND flash memory in Section Errors introduced due to retention time.

Thermal effect on error rate

After the device is received by the digital forensics lab, investigators must extract the data from the device. In many such cases, the device might have been damaged prior to seizure, and cannot be accessed using software-based analysis techniques. In these cases, a lower-level *chip-off analysis* (Billard and Vidonne, 2015) must be performed, where investigators physically remove the NAND flash memory chips from the device and use hardware that can extract data directly from the chips (ENFSI, 2015). In order to remove the chips, investigators must melt the solder connecting the chips to the PCB, at which point the chips can be pulled off.

Unfortunately, this thermal-based chip removal procedure greatly accelerates the number of retention errors that occur. Mielke et al. (2006) states that when NAND flash memory is exposed to high temperature, Arrhenius' Law (Laidler, 1984; Arrhenius, 1889) can be used to convert the effects of high temperature into additional data retention time at normal operating temperature for the memory. Let t_b denote the amount of time that heat is applied to the chip, and t_r denote the equivalent retention age at the normal operating temperature. According to Arrhenius' Law, t_r can be calculated as:

$$t_r = t_b \cdot exp\left[\frac{E_a}{k}\left(\frac{1}{T_r} - \frac{1}{T_b}\right)\right] \quad (1)$$

where k is the Boltzmann constant, T_r denotes the normal operation temperature, and T_b denotes the *baking temperature* (e.g., the high temperature applied to the chip during the removal process). E_a is the activation energy, which we set to 1.1 eV according to Mielke et al. (2004).

Following standard rework procedures used by electronics manufacturers (JEDEC, 2007), thermal-based chip removal requires investigators to apply 250 °C of heat for a duration of approximately two minutes. Using Equation (1), we see that applying this heat adds the same number of retention errors as if we had left the NAND flash memory untouched at room temperature for *833 years*. Given that the amount of retention errors after *only three years*, as shown in Fig. 4, already introduces more than 1000× the number of errors into NAND flash memory, compared to the number of retention errors after one day, we can extrapolate that 833 years' worth of retention errors would be significantly and prohibitively larger (over $10^5\times$ the number of retention errors after one day). Such a large rate of errors could easily overwhelm the error correction capability of ECC algorithms employed in modern flash controllers.

We experimentally quantify the exact impact of high-temperature baking on modern 2y-nm MLC NAND flash memory in Section Errors due to thermal-based chip removal.

Reducing NAND flash errors with read-retry

As we saw in Section NAND flash error sources during forensic analysis, the number of errors that exist in the raw data can increase by several orders of magnitude even under best practices used in forensic investigation techniques. If left unaddressed, the number of errors can quickly exceed the total error correction capability of the flash device, which in turn results in partial or complete data loss during forensic data recovery. In this section, we examine a mechanism that exists in modern MLC NAND flash memory chips, called *read-retry*, which can be used as part of the recovery process to mitigate the high number of errors introduced during data recovery.

Recall from Section Background: MLC NAND flash memory that errors occur when the threshold voltage of a flash cell shifts, causing the read reference voltages to incorrectly interpret the state of the cell. Prior work has demonstrated that retention errors are the dominant source of errors in MLC NAND flash memory (Cai et al., 2012a). As a result, the threshold voltage of a flash cell tends to *decrease* as the data retention age increases. To combat this decrease in threshold voltage, NAND flash memory manufacturers provide a *read-retry operation*, which can adjust the read reference voltages used to read data from a cell, and thus potentially reduce the number of raw bit errors in the data (Cai et al., 2013c).

Fig. 5 shows an example of a threshold voltage distribution that has shifted downwards due to charge leakage over retention time. For the sake of simplicity, we show only the ER state and P1 state distributions in the figure. If we had applied the normal read reference voltages (i.e., V_a, V_b, and V_c in Fig. 2) to the original distribution (i.e., before the distribution shifted), we would be able to read out the values of all cells in the distribution correctly. Once the distribution shifts, the read reference voltages no longer fall in between the shifted distributions of each voltage state, but instead fall *inside* the distributions of some of the states. For example, we use the default read reference voltage V_a to distinguish between cells in the ER state and cells in the P1 state. We see in Fig. 5 that, after the distribution shifts, some of the cells that belong to the P1 state now fall to the left of V_a. If the controller still uses V_a, it incorrectly classifies these cells as being in the ER state. As we can see, the default read reference voltages (V_a, V_b, V_c) can introduce many raw bit errors when the threshold voltage distribution shifts.

We can apply the read-retry operation to our example shifted distribution to compensate for the effects of the voltage distribution shifts caused by charge leakage. The basic goal of the read-retry operation is to adjust the read reference voltages up or down *with the goal* of minimizing the errors *that are introduced* due to misclassification. We denote the *optimal read reference voltages* (i.e., the voltages that are exactly in the middle of the distance between two neighboring distributions, which minimizes the number of errors) as V_a^{opt}, V_b^{opt}, and V_c^{opt}. One example read-retry mechanism can adjust the voltages down one step at a time during a read operation, checking to see whether the number of errors goes down with each subsequent step. We illustrate how this example mechanism works in Fig. 5. Here, the read-retry mechanism tries to adjust the voltage used to distinguish between cells in the ER state and cells in the P1 state. The mechanism tries several voltages (V_{RRn} in the figure, where n represents the nth voltage tried). As we can see in Fig. 5, the mechanism eventually finds a voltage (V_{RR3}) where there are no cell classification errors (because V_{RR3} falls *between* the threshold voltage distributions of the ER and P1 states). Even

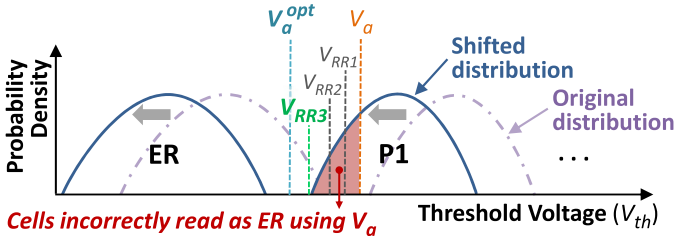

Fig. 5. Effect of read-retry operation on a shifted threshold voltage distribution (showing the distributions for only the ER state and the P1 state).

though V_{RR3} is higher than V_a^{opt}, we can still use V_{RR3} to safely extract data from the NAND flash memory *without any errors*.

Note that the read-retry operation itself does *not always* reduce errors. While retention errors usually shift the threshold voltage of a cell down, the threshold voltage of a cell can sometimes *increase* from its original state, as a result of cell-to-cell interference that occurs during programming, reading, and erasing (Cai et al., 2013b, 2014, 2015b). The direction and magnitude of the change in threshold voltage vary for each cell, due to factors such as the retention age of the data in the cell, the number of read and program operations performed to neighboring cells, and manufacturing process variation. As a result, the threshold voltage distributions of each state do *not* shift uniformly, and can overlap with each other. Therefore, it is possible that simply shifting the read reference voltage up or down (without checking the resulting number of errors) could unintentionally introduce more errors than the normal read that is done with the default read reference voltage. A read-retry based mechanism that uses the flash controller to check the change in the number of errors for every tried read reference voltage can ensure that it never increases the number of errors during a read operation.

Some NAND flash memory chips expose a number of different modes for the read-retry operation. The details of these modes are, unfortunately, often publicly unavailable, making it difficult to know, with certainty, whether a mode supports the ability to ensure that the number of read errors never increases. As we show in Section Experimental results, since the thermal-based chip removal procedure introduces a large number of retention errors (i.e., it shifts the threshold voltage distribution down significantly), we can improve the read error rate even for read-retry modes that do *not* check or expose the error rate change after a read is performed (though the improvements could be larger if the read-retry mode exposes more information to the controller).

Testing methodology

In order to investigate the raw bit errors that digital forensic investigators would encounter during their chip-off analysis (see Section Thermal effect on error rate), and to evaluate the effectiveness of using the read-retry operation (see Section Reducing NAND flash errors with read-retry) to overcome these errors, we examine the effects of data retention and thermal-based chip removal using two new 2y-nm NAND flash memory chips manufactured by two different vendors. We refer to the two chips in the rest of the paper as Chip A and Chip B. We use an Altera DE0 field-programmable gate array (FPGA) board (Terasic, 2013) to design a controller that communicates with the target chips, similar to Cai et al. (2011) and Breeuwsma et al. (2007). Fig. 6 shows a photograph of our testing environment. The FPGA issues commands to and receives data from the target NAND flash memory chip. The USB microcontroller sends the data received from the NAND flash memory chip to the PC, where we collect and analyze the data. We conduct all of our testing at room temperature, unless otherwise noted.

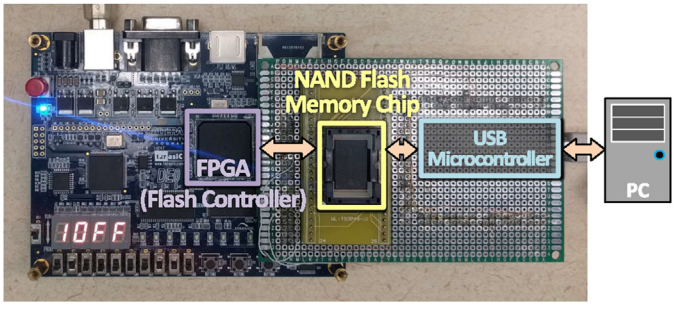

Fig. 6. Photograph of our test infrastructure used to extract data from MLC NAND flash memory chips.

First, we examine the retention errors that forensic examiners would encounter when a device is received for forensic examination. We choose multiple blocks randomly from the target chips, and we program pseudo-random data into each block. We use pseudo-random data to mimic the data scrambling employed by modern flash controllers (Cha and Kang, 2013; Kim et al., 2012). To simulate the different impact that retention errors have on relatively new devices and on heavily-used devices, we divide the blocks into multiple subsets, and perform a fixed and different number of P/E operations/cycles to each subset. Each program operation writes pseudo-random data to the blocks. We choose P/E cycle counts of 10, 300, 1000, 2500, and 4000 to examine a wide range of device wear. Once all of the blocks reach their target P/E cycle count, we perform reads at set retention ages to capture the effects of retention age on the error rate. We study six retention ages: Day 0 (i.e., immediately after programming), Day 1, and Weeks 1, 2, 3, and 4.

Next, we simulate the thermal-based chip removal procedure by baking the chips once the data stored in the chips reaches a certain retention age. Using the same parameters that are used for chip removal during chip-off analysis, we apply a temperature of 250 °C to the chips for two minutes, using a heat gun. After this baking procedure is complete, we immediately perform one more set of data read operations to measure the error rate. We study the effects of baking at three different retention ages: Day 0 (i.e., immediately after programming), Week 1, and Week 4.

One of the target chips (Chip B) has the read-retry operation implemented. For this chip, we perform multiple read operations at every retention age, performing one read for each available read-retry mode, and one read without read-retry. The read-retry modes in Chip B do not enable us to check or observe whether the number of errors is lower than that when the default read reference voltages are used. Each mode simply shifts the read reference voltage to a different fixed level.

Experimental results

We evaluate the reliability impact of chip-off digital forensic analysis on real MLC NAND flash memory chips. First, we examine the effects of data retention on the raw bit error rate (RBER) of two real MLC NAND flash memory chips (Section Errors introduced due to retention time). Next, we examine how applying high temperature to the two chips, under the same conditions as thermal-based chip removal in chip-off analysis, affects the raw bit error rate (Section Errors due to thermal-based chip removal). Finally, we demonstrate how the error rate changes when we use the read-retry mechanism available on one of the chips (Section Read-retry operation).

Note that error characteristics differ by manufacturer and by chip model, and thus the RBER of *other* NAND flash memory chips may *not* always be consistent with the results that we observe here. However, given that digital devices are exposed to varying levels of stress (e.g., mobile phones left in a car on a hot day, solid-state drives salvaged from CCTV cameras found at a fire scene, dash-cams that write to and erase data from NAND flash memory constantly), digital forensic investigators should assume that the NAND flash memory they are investigating can possibly contain *even more errors* than what we show in this paper.

Errors introduced due to retention time

Fig. 7 shows how the RBER of the tested NAND flash memory chips varies with (1) the P/E cycle count of the chip (i.e., how much

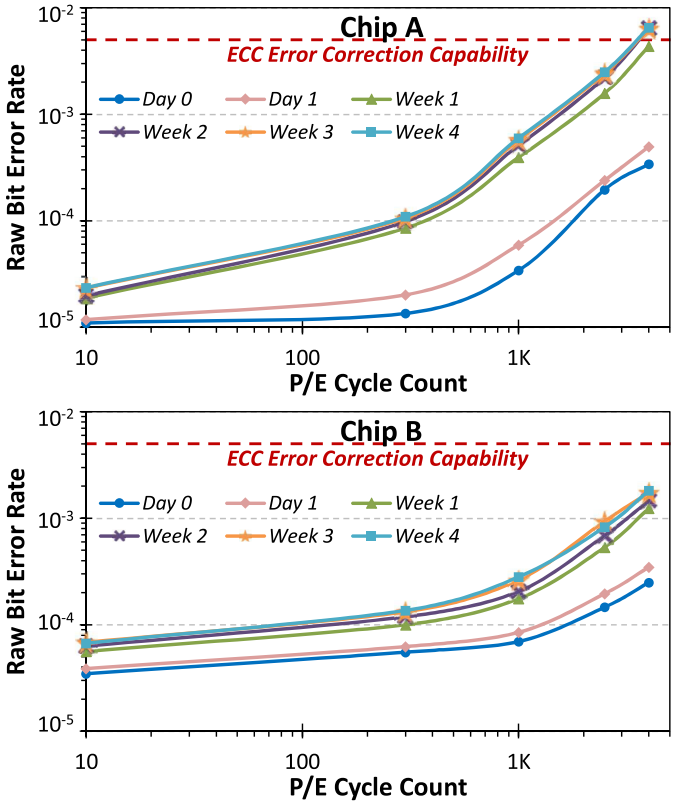

Fig. 7. Raw bit error rate at different data retention ages, for different P/E cycle counts.

the flash memory cells on the chip have been worn out), and (2) the *retention age* of the data (i.e., the amount of time that elapses after the data was written). We make three major observations about the impact of retention age on MLC NAND flash memory.

First, we observe that in both chips, the RBER grows as the retention age increases. The Day 0 results in Fig. 7 show the number of errors that exist in flash blocks immediately after the data is programmed to the NAND flash chips. We see that the RBER increases over time, but that the largest increases occur soon after the data is programmed. For example, the increase in RBER is much greater during the first week (6.34× for Chip A and 1.81× for Chip B at 300 P/E cycles, compared to Day 0 of each respective chip) than the second week (1.15× for Chip A and 1.19× for Chip B), and greater during the second week than the third week (1.08× for Chip A and 1.12× for Chip B).

Second, we observe that in both chips, the RBER grows as the P/E cycle count increases. In other words, as a NAND flash memory chip is worn out, its susceptibility to raw bit errors due to retention increases, for data with the same retention age. Note that the y-axis in Fig. 7 is in log scale. A chip at a higher P/E cycle count (i.e., a chip with greater wearout) accumulates retention errors at a much faster rate than a chip at a lower P/E cycle count (Cai et al., 2015a).

Third, we observe that while the RBER grows with both wearout and retention age, the overall RBER of the chip does *not* exceed the error correction capability of ECC unless the P/E cycle count significantly exceeds the endurance guaranteed by manufacturers. For 2y-nm MLC NAND flash memory, a controller that employs BCH codewords (Micheloni et al., 2008; Hocquenghem, 1959; Bose and Ray-Chaudhuri, 1960) for ECC can typically correct 40 bits of errors for every 1 KB of data (Cypress Semiconductor Corp, 2013) (i.e., it can correct errors for an RBER of up to 4.9×10^{-3}). As our results from Fig. 7 show, the overall RBER stays lower than this error correction capability through a retention age of four weeks, for P/E cycle counts below 3000 cycles, which is the typical endurance of commercial 2y-nm MLC NAND flash memory (Cypress Semiconductor Corp, 2013).

Uncorrectable data

We implement a 70-byte BCH codeword for ECC within our experimental platform, to provide the expected 40 bits of correction capability for each 1 KB chunk of data. Note that a data chunk is smaller than a page (van Zandwijk, 2015), which can range from 4 to 16 KB of data (Micron Technology, 2015). When we read data from our test chips, we use the BCH codeword to determine how many of the data chunks *cannot* be successfully corrected by ECC.

Table 1 shows the fraction of pages that contain at least one uncorrectable data chunk. As shown in Table 1, even before chip-off

Table 1
Fraction of pages containing uncorrectable 1 KB data chunks. A dash (–) indicates that no data chunks are uncorrectable.

Chip	Retention Age (days)	P/E Cycle Count				
		10	300	1000	2500	4000
A	0	–	–	–	–	–
	1	–	–	–	–	–
	7	–	–	–	0.9%	64.9%
	14	–	–	–	9.6%	92.2%
	21	–	–	–	14.4%	91.8%
	28	–	–	–	15.9%	91.9%
B	0	–	–	–	–	–
	1	–	–	–	–	–
	7	–	–	–	–	–
	14	–	–	–	–	–
	21	–	–	–	–	0.0039%
	28	–	–	–	–	0.0065%

analysis is performed, if a chip has been worn out significantly (e.g., after 2500 P/E cycles for Chip A), it can contain some uncorrectable pages even at a retention age of just one week. However, for less worn-out chips (e.g., a chip at 1000 P/E cycles), none of the pages are uncorrectable even after a retention age of four weeks.

For perspective, a device can reach 2190 P/E cycles if all of the pages in the NAND flash memory chip are written to twice a day, every day, over a period of three years. We observe that once we exceed the expected endurance of 2y-nm MLC NAND flash memory (which is 3000 P/E cycles), the fraction of pages with uncorrectable errors grows rapidly for Chip A. At 4000 P/E cycles, 64.9% of the pages in Chip A contain uncorrectable errors after a retention age of *only* one week.

We observe that even when the *overall* RBER stays lower than the ECC error correction capability, errors in *some pages* become uncorrectable over time. For example, Chip A's RBER at a retention age of one week (i.e., seven days) at 2500 P/E cycles is 1.6×10^{-3} (see Fig. 7), which is lower than the ECC error correction capability of 4.9×10^{-3}. At the same time, 0.9% of the pages in Chip A contain uncorrectable errors, as we see in Table 1. We see similar behavior for Chip B, even though its overall RBER *always* remains below the error correction capability (see Fig. 7).

We conclude that even if we assume that all of the data inside the device is refreshed immediately before the device was confiscated, a worn-out device can quickly accumulate errors, and some of those errors become uncorrectable over time (as we have shown in Fig. 7 and Table 1). Thus, in order to avoid data loss due to uncorrectable data pages, data needs to be extracted from a NAND flash memory based device at the *earliest possible time after the receipt of the device*.

Error characterization

By investigating the state of each cell at various retention ages, we can characterize a number of trends in the threshold voltage distribution shift. We say that a cell belongs to the set $[S_O, S_M]$ if the cell was originally programmed to state S_O, but is misread as belonging to state S_M when we use the default read reference voltages (see Section Reducing NAND flash errors with read-retry). The graphs in Fig. 8 show the fraction of cells in Chip A that are in the set $[S_O, S_M]$, for all neighboring (S_O, S_M) pairs, out of the total number of cells originally programmed to state S_O, across a range of retention ages.

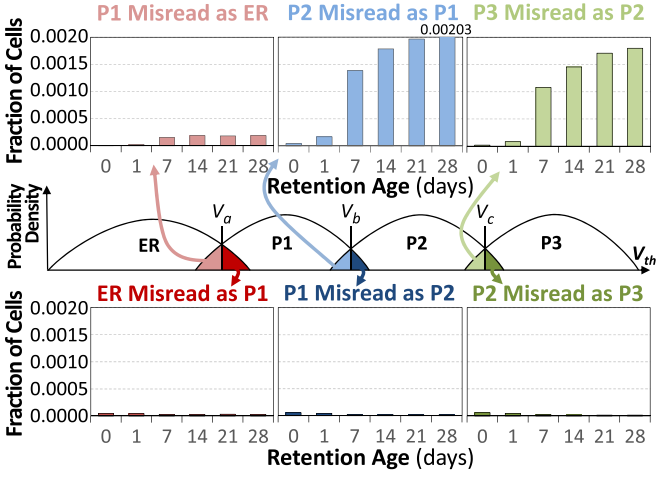

Fig. 8. Fraction of cells in Chip A that were programmed to state S_O but are misread as belonging to state S_M, out of the total number of cells originally programmed to state S_O.

We observe from Fig. 8 that when S_M is a *lower* voltage state than S_O, a greater number of cells belong to $[S_O, S_M]$ as the retention age increases. For example, after a retention age of four weeks (i.e., 28 days), 0.20% of cells that were originally programmed to the P2 state are misread as belonging to the P1 state (i.e., the cells are in the set [P2, P1]), as opposed to only 0.02% after a retention age of one day. We find that regardless of the retention age, when S_M is a *higher* voltage state than S_O, only a very small number of cells belong to the set $[S_O, S_M]$ (e.g., [ER, P1]). From these results, we find that the threshold voltage of a misread cell tends to be lower as the retention age increases. We conclude that the threshold voltage reduction, which occurs as a result of charge leakage from the floating gate of a flash memory cell, is the dominant source of errors that are introduced by retention age.

Errors due to thermal-based chip removal

We now study how the RBER due to retention errors changes *after* the thermal-based chip removal procedure is performed as a part of chip-off analysis. Fig. 9 shows the RBER after we perform the chip baking process (see Section Testing methodology) to emulate the removal procedure. Note that the RBER data for Day 0 (i.e., immediately after programming), Week 1, and Week 4 *before baking* is the same as the data shown in Fig. 7.

We observe that simply applying the heat required for chip removal causes the RBER to increase significantly. When the heat is applied immediately after the data is written (*Day 0, After Baking* in the figure), at 1000 P/E cycles, the RBER increases by 432× for Chip A, and by 17× for Chip B, compared to the RBER before baking (*Day 0, Before Baking*). When the heat is applied four weeks after the data is written at the same P/E cycle count, (*Week 4, After Baking*), the RBER increases by 47× and 54× for Chips A and B, respectively,

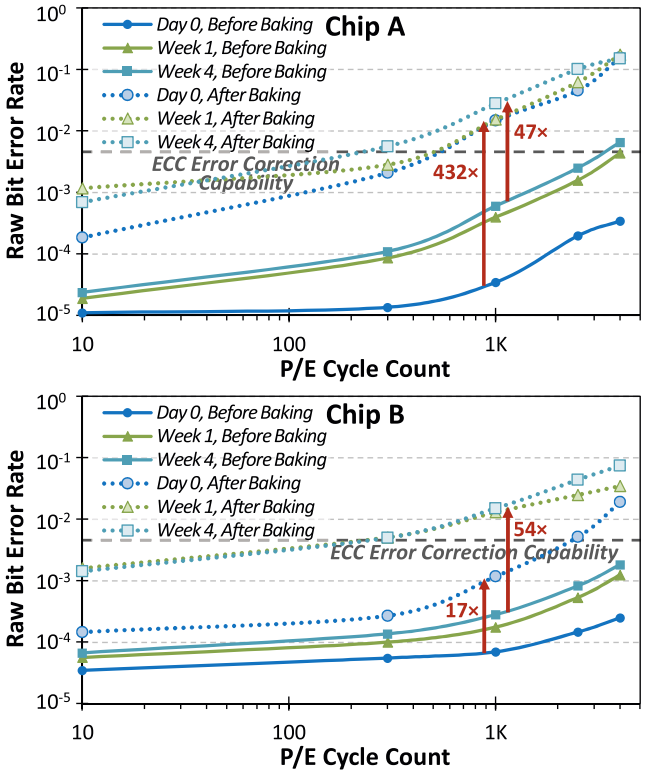

Fig. 9. Raw bit error rate before and after baking NAND flash memory chips.

Table 2
Fraction of pages containing uncorrectable 1 KB data chunks after applying heat to the chips. A dash (–) indicates that no data chunks are uncorrectable.

Chip	Retention Age (days)	P/E Cycle Count				
		10	300	1000	2500	4000
A	7	–	29.1%	99.8%	100.0%	100.0%
	28	0.7%	84.2%	96.9%	100.0%	100.00%
B	7	–	78.1%	96.5%	96.9%	96.9%
	28	–	83.6%	99.7%	100.0%	100.0%

compared to the RBER before baking (*Week 4, Before Baking*). Starting at 300 P/E cycles, the RBER exceeds the ECC error correction capability when heat is applied to the chip.

The impact of the heat applied during chip removal can cause critical damage to the data stored within the NAND flash memory chip that is being analyzed. Suppose that a digital forensic investigator starts the chip-off procedure four weeks after a device has been seized, and that the NAND flash memory chip inside the device was only lightly used (e.g., 300 P/E cycles) prior to seizure. Before applying heat, the RBER remains safely within the error correction capability of contemporary ECC, as shown in Fig. 9 (*Week 4, Before Baking* at 300 P/E cycles), with a raw bit error rate of 1.1×10^{-4} for Chip A and 1.4×10^{-4} for Chip B. However, after applying the chip removal temperature, the RBER exceeds the error correction capability of ECC (*Week 4, After Baking* at 300 P/E cycles). The RBER becomes 5.6×10^{-3} and 5.0×10^{-3} for Chip A and Chip B, respectively. At such a high RBER, it is impossible to correct all of the errors in the data with the given ECC error correction capability. Thus, the integrity of the data recovery is compromised. As a point of comparison, we extrapolate the increase in RBER between Week 1 and Week 4 before baking for both chips, to see how long it would take for the chips to reach the RBER after baking if we had not baked the chips. Baking can increase the RBER by 113× for Chip A, and by 38× for Chip B on average, while the increase in RBER between *Week 1, Before Baking* and *Week 4, Before Baking* is 1.43× for both chips. Therefore, applying heat to a chip induces approximately *two to five years*' worth of retention errors at room temperature.

Table 2 shows the percentage of uncorrectable data chunks *after* the chips are baked (compare this to Table 1, which shows the uncorrectable data chunks *before* baking). Nearly all of the pages stored within the NAND flash memory contain uncorrectable errors after the baking process. At only 300 P/E cycles, 84.2% of the pages in Chip A and 83.6% of the pages in Chip B contain uncorrectable errors, and *all* pages contain errors when we reach 2500 P/E cycles for both Chip A and Chip B, when the heat is applied four weeks after the data is written. From our analysis, we conclude that, when left unmitigated, the thermal-based chip removal procedure is prohibitively destructive, as it greatly decreases the amount of data that can be successfully retrieved from NAND flash memory during forensic recovery.

Read-retry operation

We now investigate the ability of the read-retry operation (see Section Reducing NAND flash errors with read-retry) to mitigate the errors introduced during the thermal-based chip removal process. We are able to exploit the read-retry mechanism built into one of our chips (Chip B). In Chip B, we have access to two read-retry modes, which we refer to as Mode A and Mode B.[1]

Fig. 10 shows how the read-retry mechanism affects the RBER as the retention age increases, *ignoring the effects of the chip removal process for now*. We observe that across all P/E cycle counts, while the RBER increases with retention age when we use the default read operation, the RBER actually *decreases* with retention age if we use either of the read-retry modes. However, we find that the read-retry modes appear to be basic: *only* Mode A can outperform the default read operation *only* at high P/E cycle counts. We find that we do not benefit from read-retry for most of the time, as we are unable to check or control the behavior of the implemented read-retry modes when they lead to a higher RBER than the default read reference voltages.[2] Fortunately, without the thermal removal process, the RBER after using Mode A typically remains within the ECC error correction capability, as shown in Fig. 7. When the chip is worn out beyond the manufacturer's endurance specification (e.g., at 4000 P/E cycles), Mode A effectively reduces the RBER compared to the default read operation after a retention age of two weeks (i.e., 14 days). In fact, we find that some of the uncorrectable pages observed in Section Errors introduced due to retention time become correctable when we use Mode A (as we explain below).

When we apply our baking process to emulate thermal-based chip removal, we find that the read-retry modes, even with their uncontrollable behavior, are very successful at reducing the RBER, as shown in Fig. 11. As we saw in Section Errors due to thermal-based chip removal, baking a chip induces 2–5 years' worth of retention errors. These errors are the result of a significant downward shift in the threshold voltage distribution of the flash cells. The two read-retry modes are able to adapt the read reference voltages to the shifted distribution such that *they can reduce the post-baking RBER to a level that is within the error correction capability of the implemented ECC algorithm*, even at high P/E cycle counts. For example, while the RBER for Chip B at 1000 P/E cycles (1.5×10^{-2}) is significantly over the ECC error correction capability after the chip is baked, Modes A and B can reduce the RBER by 88.6% (i.e., to 1.7×10^{-3}) and 94.6% (i.e., to 8.2×10^{-4}), respectively.

The read-retry mechanism significantly reduces the number of uncorrectable data chunks after the chip is baked. Table 3 shows the number of uncorrectable data chunks when we use the default read operation, and when we use the available read-retry modes, for data with a retention age of four weeks. We make two observations from the table. First, at low P/E cycle counts (e.g., 1000 P/E cycles), Mode B can *completely eliminate* the uncorrectable data chunks that were introduced by baking. Second, at high P/E cycle counts, while read-retry cannot fully eliminate the uncorrectable data chunks, it can significantly reduce the number of them. For example, at 2500 P/E cycles, the default read operation produces uncorrectable errors in *every* data chunk. Read-retry Mode B reduces the number of uncorrectable data chunks to 49.5% of all chunks. We believe these results can be significantly improved if the read-retry mode implemented by the chip is modified to expose more information to the flash controller.

We conclude that digital forensic investigators should employ the read-retry mechanisms built into NAND flash memory chips, as read retry is able to overcome the large number of uncorrectable errors introduced due to the exposure of chips to very high temperatures during chip-off analysis.

[1] No documentation is available from the manufacturer on how the modes operate.

[2] As explained in Section Reducing NAND flash errors with read-retry, it is difficult to know, with certainty, whether the read-retry modes implemented in the test chip check the error rate after read-retry is performed.

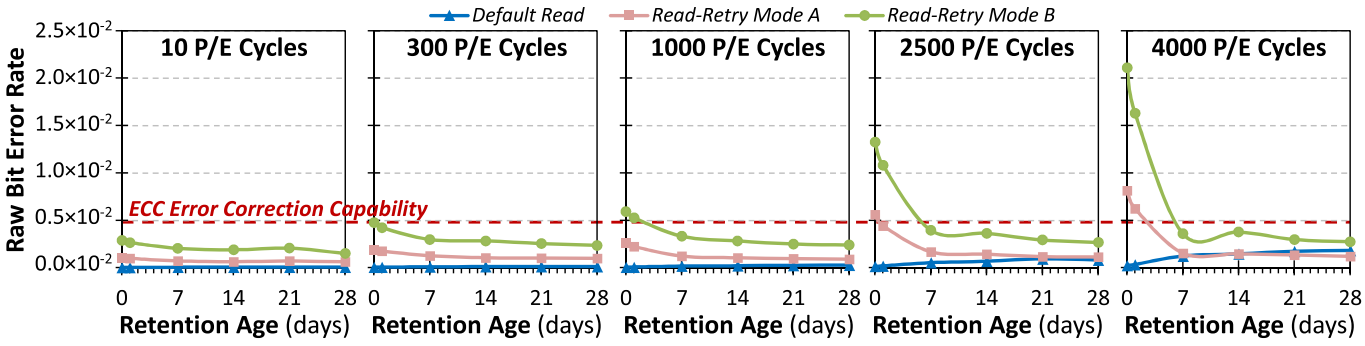

Fig. 10. Effect of read-retry modes on the RBER of Chip B as retention age increases.

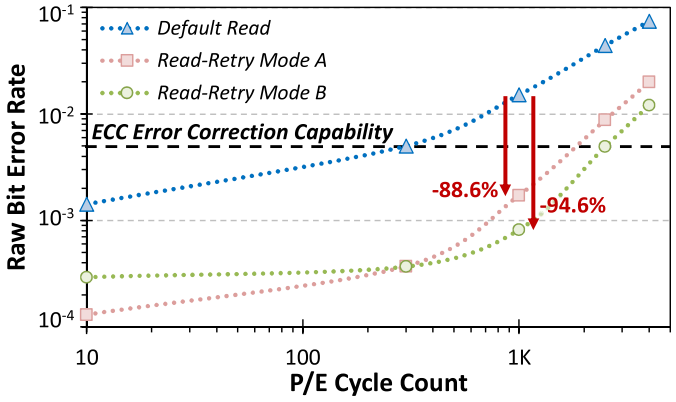

Fig. 11. RBER with read-retry after Chip B is baked.

Table 3
Fraction of pages containing uncorrectable 1 KB data chunks after Chip B is baked, with and without read-retry.

Read mode	P/E cycle count				
	10	300	1000	2500	4000
Default read	0.0%	83.6%	99.7%	100.0%	100.0%
Read-retry mode A	**0.0%**	**0.0%**	12.1%	69.0%	99.1%
Read-retry mode B	**0.0%**	**0.0%**	**0.0%**	49.5%	90.6%

Values in bold indicate cases where read-retry eliminates all of the uncorrectable chunks.

Related work

To our knowledge, this is the first work to (1) demonstrate that the high temperatures used during thermal-based chip removal in chip-off analysis greatly increase the error rate in NAND flash memory chips, and is thus detrimental to forensic data recovery if left unmitigated; and (2) analyze the effect of the read-retry mechanism, commonly implemented in modern NAND flash memory chips, on recovering the data stored in the chips from a digital forensics perspective.

There are several works that characterize (1) various types of errors in MLC NAND flash memory (e.g., Cai et al., 2012a,b, 2013a,b,c, 2014, 2015a,b; Luo et al., 2016; Papandreou et al., 2014, 2015; Park et al., 2008; Parnell et al., 2014; Prodromakis et al., 2015), and (2) the effect of temperature on NAND flash memory reliability (Yang et al., 2006; Tressler et al., 2011). None of these works (1) quantify the largely detrimental effects of high temperatures on reliable recovery of data from MLC NAND flash memory chips, (2) examine these issues from the perspective of digital forensic data recovery, or (3) demonstrate the value of read-retry in mitigating the NAND flash errors introduced as a result of thermal effects.

Other works in digital forensics analyze data recovery from flash memory chips (Breeuwsma et al., 2007; Klaver, 2010; van Zandwijk, 2015). However, none of these examine the effects of (1) the powerful read-retry mechanism, or (2) high temperatures on error rates during the data recovery procedure.

Conclusion

With the increasing popularity of NAND flash memory as a storage medium, digital forensic investigators today are required to perform data recovery from a growing number of NAND flash memory based devices that are seized during the course of criminal investigations. Prior works have documented a series of procedures that can be used by investigators to extract data from physically-damaged devices, including the use of chip-off analysis. In this work, we find that the large amount of time that may elapse between device seizure and data extraction increases the error rate of data stored within the device. Often, this error rate can exceed the number of errors that can be corrected by the internal error correction mechanisms originally implemented by the NAND flash memory controller. In many cases, a majority of the data extracted from the device can contain uncorrectable errors. Therefore, we conclude that it is critical for digital forensic investigators to perform data extraction from NAND flash memory based digital devices at the earliest point of time after seizure of the target device.

In situations where chip-off analysis is required, a chip is exposed to high temperatures during the thermal-based chip removal process needed for chip-off analysis. We find that these high temperatures can further increase the error rate by more than two orders of magnitude, despite the use of best practices taken from electronics rework procedures. We demonstrate that using the read-retry mechanism built into modern NAND flash memory chips, instead of simply using the default read operation when extracting data, provides a promising solution. As our experimental results show, the read-retry mechanism can significantly mitigate the error rate increase caused by the thermal-based chip removal process. We conclude that forensic data recovery from NAND flash memory should adopt the read-retry mechanism as part of the data recovery procedure.

References

Arrhenius, S., 1889. Über die Reaktionsgeschwindigkeit bei der Inversion von Rohrzucker durch Säuren. Zeitschrift für Physikalische Chemie.
Ayers, R., Brothers, S., Jansen, W., 2014. Guidelines on Mobile Device Forensics. National Institute of Standards and Technology.
Belgal, H.P., et al., 2002. A new reliability model for post-cycling charge retention of flash memories. In: IRPS.
Billard, D., Vidonne, P., 2015. Chip-off by matter subtraction: Frigida via. In: SADFE.
Bose, R., Ray-Chaudhuri, D., 1960. On a Class of Error Correcting Binary Group Codes. Information and Control.

Brand, A., et al., 1993. Novel read disturb failure mechanism induced by FLASH cycling. In: IRPS.
Breeuwsma, M., et al., 2007. Forensics data recovery from flash memory. Small Scale Digital Device Forensics J.
Cai, Y., et al., 2011. FPGA-based solid-state drive prototyping platform. In: FCCM.
Cai, Y., et al., 2012a. Error patterns in MLC NAND flash memory: measurement, characterization, and analysis. In: DATE.
Cai, Y., et al., 2012b. Flash correct-and-refresh: retention-aware error management for increased flash memory lifetime. In: ICCD.
Cai, Y., et al., 2013a. Error analysis and retention-aware error management for NAND flash memory. Intel Technol. J.
Cai, Y., et al., 2013b. Program interference in MLC NAND flash memory: characterization, modeling, and mitigation. In: ICCD.
Cai, Y., et al., 2013c. Threshold voltage distribution in MLC NAND flash memory: characterization, analysis, and modeling. In: DATE.
Cai, Y., et al., 2014. Neighbor-cell assisted error correction for MLC NAND flash memories. In: SIGMETRICS.
Cai, Y., et al., 2015a. Data retention in MLC NAND flash memory: characterization, optimization, and recovery. In: HPCA.
Cai, Y., et al., 2015b. Read disturb errors in MLC NAND flash memory: characterization and mitigation. In: DSN.
Cai, Y., et al., 2017. Vulnerabilities in MLC NAND flash memory programming: experimental analysis, exploits, and mitigation techniques. In: HPCA.
Casey, E., et al., 2009. Investigation delayed is justice denied: proposals for expediting forensic examinations of digital evidence. J. Forensic Sci.
Cha, J., Kang, S., 2013. Data randomization scheme for endurance enhancement and interference mitigation of multilevel flash memory devices. ETRI J.
Chimenton, A., Atti, M., Olivo, P., 2008. Reliability of floating gate memories. In: Error Correction Codes for Non-volatile Memories. Springer Netherlands.
Crippa, L., Micheloni, R., 2010. MLC storage. In: Inside NAND Flash Memories. Springer, Netherlands.
Cypress Semiconductor Corp, 2013. SLC versus MLC NAND Flash Memory. URL. http://www.cypress.com/file/209181/download.
Degraeve, R., et al., 2004. Analytical percolation model for predicting anomalous charge loss in flash memories. IEEE Trans. Electron Devices.
ENFSI, 2015. Best Practice Manual for the Forensic Examination of Digital Technology. URL. http://enfsi.eu/wp-content/uploads/2016/09/enfsi-bpm-fit-01_1.pdf.
Friederich, C., 2010. Program and erase of NAND memory arrays. In: Inside NAND Flash Memories. Springer, Netherlands.
Gallager, R.G., 1963. Low-density Parity-check Codes. MIT Press.
Hocquenghem, A., 1959. Codes Correcteurs d'Erreurs. Chiffres.
JEDEC Solid State Technology Association, 2007. Moisture/reflow sensitivity classification for nonhermetic solid state surface mount devices. In: IPC/JEDEC J-STD-020D.1.
Kim, C., et al., 2012. A 21 nm high performance 64 Gb MLC NAND flash memory with 400 MB/s asynchronous toggle DDR interface. IEEE J. Solid-State Circuits.
Klaver, C., 2010. Windows mobile advanced forensics. Digit. Investig.
Laidler, K.J., 1984. The development of the Arrhenius equation. J. Chem. Educ.
Luo, Y., et al., 2016. Enabling accurate and practical online flash channel modeling for modern MLC NAND flash memory. IEEE J. Sel. Areas Commun.
Meza, J., et al., 2015. A large-scale study of flash memory failures in the field. In: SIGMETRICS.
Micheloni, R., Marelli, A., Commodaro, S., 2010. NAND overview: from memory to systems. In: Inside NAND Flash Memories. Springer, Netherlands.
Micheloni, R., Marelli, A., Ravasio, R., 2008. BCH hardware implementation in NAND flash memories. In: Error Correction Codes for Non-volatile Memories. Springer.
Micron Technology, Inc., 2011. Bad Block Management in NAND Flash Memory. URL. https://www.micron.com/~/media/documents/products/technical-note/nand-flash/tn2959_bbm_in_nand_flash.pdf.
Micron Technology, Inc., 2015. How Micron SSDs Handle Unexpected Power Loss. URL. https://www.micron.com/~/media/documents/products/white-paper/ssd_power_loss_protection_white_paper_lo.pdf.
Mielke, N., et al., 2004. Flash EEPROM threshold instabilities due to charge trapping during program/erase cycling. IEEE Trans. Device Mater. Reliab.
Mielke, N., et al., 2006. Recovery effects in the distributed cycling of flash memories. In: IRPS.
Mielke, N., et al., 2008. Bit error rate in NAND flash memories. In: IRPS.
Pan, Y., Dong, G., Zhang, T., 2011. Exploiting memory device wear-out dynamics to improve NAND flash memory system performance. In: FAST.
Papandreou, N., et al., 2014. Using adaptive read voltage thresholds to enhance the reliability of MLC NAND flash memory systems. In: GLSVLSI.
Papandreou, N., et al., 2015. Enhancing the reliability of MLC NAND flash memory systems by read channel optimization. ACM Trans. Des. Autom. Electron. Syst.
Park, K.-T., et al., 2008. A zeroing cell-to-cell interference page architecture with temporary LSB storing and parallel MSB program scheme for MLC NAND flash memories. IEEE J. Solid-State Circuits.
Parnell, T., et al., 2014. Modelling of the threshold voltage distributions of Sub-20nm NAND flash memory. In: GLOBECOM.
Prodromakis, A., Korkotsides, S., Antonakopoulos, T., 2015. MLC NAND flash memory: aging effect and chip/channel emulation. Microprocess. Microsyst.
Schroeder, B., Lagisetty, R., Merchant, A., 2016. Flash reliability in production: the expected and the unexpected. In: FAST.
Suh, K.-D., et al., 1995. A 3.3 V 32 Mb NAND flash memory with incremental step pulse programming scheme. IEEE J. Solid-State Circuits.
Terasic, Inc., 2013. Altera DE0 Board. URL. http://de0.terasic.com/.
Tressler, G., Vanstee, D., Griffin, T., 2011. Enterprise MLC NAND industry comparison. In: Flash Memory Summit.
van Zandwijk, J.P., 2015. A mathematical approach to NAND flash-memory descrambling and decoding. Digit. Investig.
Yang, H., et al., 2006. Reliability issues and models of Sub-90nm NAND flash memory cells. In: ICSICT.
Yoon, J.H., Tressler, G.A., 2012. Advanced flash technology: status, scaling trends, and implications to enterprise SSD technology enablement. In: Flash Memory Summit.
Zambelli, C., Chimenton, A., Olivo, P., 2010. Reliability issues of NAND flash memories. In: Inside NAND Flash Memories. Springer, Netherlands.
Zhang, L., Tan, Y., Zhang, Q., 2012. Identification of NAND flash ECC algorithms in mobile devices. Digit. Investig.
Zhao, K., et al., 2013. LDPC-in-SSD: making advanced error correction codes work effectively in solid state drives. In: FAST.