FFD: A Framework for Fake Flash Detection

Zimu Guo, Xiaolin Xu, Mark M. Tehranipoor and Domenic Forte
ECE Department, University of Florida
zimuguo@ufl.edu, {xiaolinxu,tehranipoor,dforte}@ece.ufl.edu

ABSTRACT

Counterfeit electronics have become a big concern in the globalized semiconductor industry where chips might be recycled, remarked, cloned or overproduced. In this work, we advance the state-of-the-art counterfeit detection of flash memory, which is widely used in electronic systems. Fake memories may be used in critical systems, such as missiles, military aircrafts and helicopters, thus diminishing their reliability. In addition, there are countless stories of fake flash drives in the general consumer market. We propose a comprehensive framework called FFD to detect fake flash memories (i.e., recycled, remarked and cloned parts). FFD is validated with 200,000 commercial flash memory pages. Experimental results show that our framework performs well in: 1) nearly 100% detection accuracy of flash with as little as 5% usage, 2) estimating the flash memory usage with high resolution ($\leq$ 5% of its maximal endurance). Another contribution of this work is a chip ID generation technique that can generate unique flash fingerprints with greater than 99.3% reliability.

1. INTRODUCTION

The globalization of the semiconductor industry makes it difficult to track electronic chips. Under these circumstances, the counterfeit electronic market continues to grow each year. According to Industryweek, consumers and industrial businesses are losing approximately $250 billion per year because of counterfeit components [5]. Besides the detrimental impact on profit, the overall reliability of critical infrastructures also decreases if built with counterfeit electronics. In 2012, a Senate Armed Services Committee uncovered more than 1 million "bogus parts" in the Pentagon supply chain [3], in which suspect components were found from mission computers of important missiles, military aircrafts, and helicopters.

Non-volatile memories (e.g., flash and solid-state disk) are commonly utilized in today's electronic systems. Thus a fake (recycled, remarked, cloned, etc) flash memory may widely devastate system reliability and security. For instance, 1,500 flash memory chips bought by Raytheon for missile systems were discovered as counterfeit [2]. According to a report from eBay [1], fake flash drives and SSDs usually possess less than half of their labeled capacity and slower access speed. Moreover, these counterfeit types contribute to more than 80% of all the counterfeit types [9]. To address this problem, the authors in [7] presented a flash-based PUF by manipulating the reading operation. However, it is impossible to directly apply this approach on most commercial off-the-shelf (COTS) flash chips since access to on-chip voltages is not commonly available. Programming [10] and reading disturb [8] were also proposed to implement hardware security primitives. One drawback of the programming disturb approach lies in the low robustness against flash aging. Though reading disturb does not significantly degrade the flash memory, it may take 6 hours to generate one useful ID [10].

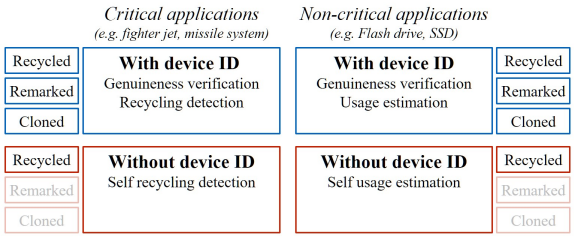

Figure 1: Detection coverage analysis.

In this paper, a fake flash memory detection (FFD) framework is presented. This framework has three major contributions: (i) recycled flash memory detection, (ii) usage estimation, and (iii) aging-resilient flash ID generation, the coverage of FFD is shown in Figure 1. If a flash memory has an enrolled ID, its recycling, remarking and cloning status can be easily verified. Our proposed FFD framework facilitates using the non-volatile nature of flash memory, that enables the capability to store the model and helper data. This feature makes it possible to conduct the off-line recycling detection and usage estimation, in which all the helper data stored locally in the flash memory. Our major contributions in this paper are as follows:

- A comprehensive framework (FFD) that addresses counterfeit detection of external COTS flash memory chips as well as SoCs with embedded flash. The performance is evaluated with more than 200,000-page data from 20 commercial flash memory chips.
- Recycled flash detection based on a score-usage model. A 100% detection accuracy can be achieved when the flash memory is used for 5% of its maximal endurance. Remarked and cloned flash detection based on saturation property for unclonable ID generation. The reliability is greater than 99.3% with excellent uniqueness.

The rest of the paper is organized as follows. The background of flash memory is introduced in Section 2. Section 3 presents the partially programming effects which we observe at the page and floating gate levels. The recycled flash detection framework is proposed in Section 4. In Section 5, we

DAC '17, June 18-22, 2017, Austin, TX, USA

DOI: http://dx.doi.org/10.1145/3061639.3062249

present a reliable ID generation approach. The experimental results are provided in Section 6. Finally, we conclude and provide future research directions in Section 7.

2. FLASH MEMORY PRELIMINARIES

This section briefly introduces the flash memory architecture and the working mechanism of floating gate transistors.

2.1 Floating Gate Transistors

The schematic of a floating gate transistor is shown in Figure 2.(a) [10]. The non-volatile property of flash memory is realized by the charging and discharging of this floating gate [6]. When the floating gate is discharged, it behaves like a MOSFET and the threshold voltage remains relatively low (the blue curve in Figure 2.(b)). If the floating gate is charged, the threshold voltage is shifted (the red curve in Figure 2.(b)). A *read* signal is designed to monitor whether the floating gate is charged or not. Based on the storage capacity, the floating gate transistors can be classified into single-level cell (SLC), multiple-level cell (MLC) and triple-level cell (TLC). The latter two floating gate transistors can be charged to three or seven potentials.

2.2 Flash Organization and Operations

Flash memory can be classified into two types: NAND flash and NOR flash. In a NAND flash, the bit line is pulled low only if all word lines are high, while in a NOR flash, each cell has one end grounded, another end is connected to a bit line. Flash memory has three major operations: *read, erase,* and *program (write)*. The threshold voltage of a charged transistor is driven high after programming. Thus logic "0s" can be stored in the selected locations. *Erase* operation discharges the floating gates to reduce the threshold voltage. After erasing, logic "1s" are restored. To *read* a flash cell, the corresponding transistor is turned on, and the amount of current is detected. Several parameters of NAND flash memory include: *Page*, the basic unit for programming and *Block*, the basic units for erasing. A block commonly consists of 32 to 256 pages. A *ready* signal will be presented to notify the host controller when these operations are completed.

3. PARTIALLY PROGRAMMING

The programming operation of the flash memory is monitored by internal sensors, and the memory cells will be kept busy until the program/erase operations are completed. This process can be referred to as *fully programming*. However, the fully programming process can be interrupted both externally (e.g., disconnect power) and internally (e.g., circuit reset). If such interruption occurs, the programming operation will be terminated immediately. As a result, some data will not be programmed (referred as bit errors). This interrupted programming is named as *partially programming*. We define the time interval between the start and interruption of programming as the *partially programming time*.

For any COTS NAND/NOR flash memory, the programming time can be controlled by issuing a RESET instruction.

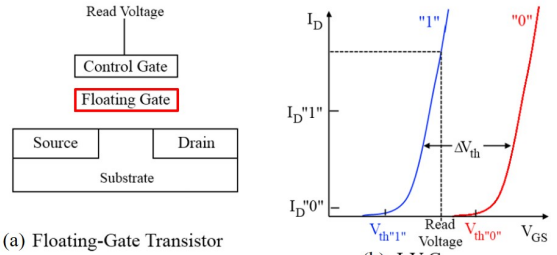

Figure 2: Floating gate transistors [10].

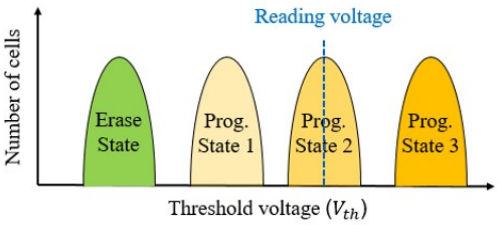

Figure 3: Potential floating gate transistor programming states for SLC flash memory.

In Figure 3, three programming states of floating gate transistors and their threshold voltages are depicted. Among all the states, the erase state indicates whether a transistor is charged or not. Prog. state 1 means that the transistors are barely/not charged, such transistors store"1s" in erase state and will introduce bit errors. If the transistor is charged to Prog. state 2, its threshold voltage will be close to the nominal reading voltage. However, such transistor may also induce errors due to the threshold voltage variations. If the transistor is fully or almost fully charged (Prog. state 3), there will not be any bit errors. The states of memory cells: fully charged, partially charged or barely/not charged depend on the following two factors:

- Page programming scheduling: the programming controller determines the order and number of the floating gate transistors being charged at different time.
- Physical process variations: the time length required to fully charge each floating gate transistor varies due to the process variations.

3.1 Programming Time Sweeping

In general, we refer to the above errors as *floating gate transistor failures (FG failures)* for various flash memory cells. One floating gate transistor is labeled as "failed" if its output bit differs from the one programmed. To investigate the floating gate transistor failures under various partially programming times, we swept the partially programming time from 0 microseconds to 300 microseconds with a step of 200 nanoseconds. For each programming, the whole page is written with "0"s. In this section, this operation is executed on five pages from five MLC flash chips. For each page, the stored data are read 40 times consecutively, and the corresponding *FG failure* rate is calculated as

$$\text{FG failure rate} = \frac{\#\text{ of failed FG transistors}}{\text{Total }\#\text{ of FG transistors}} \quad (1)$$

The results collected from one chip are presented in Figure 4. From this figure, the following properties of partially programming are verified by all flash memory under test:

- When the programming time is relatively short (i.e., $\leq$ 125 microseconds), all the floating gate transistors are not or barely charged (Prog. state 1).
- When the programming time is long (i.e., $\geq$ 180 microseconds), most of the floating gate transistors are fully charged (Prog. state 3).
- When the programming time is medium (i.e., 125 to 180 microseconds), around 40% of the floating gate transistors are partially charged (Prog. state 1 and 2).

These observations can be also found in SLC flash memory. According to these failure rate observations under different partially programming times, neither too short nor too long programming time is useful for extracting the IDs and detection the recycled flash memory, since no significant discrimination is presented.

Besides sweeping the partially programming time, the aging effects are studied. More than 8,000 times partially programming and erasing are performed on one page. The

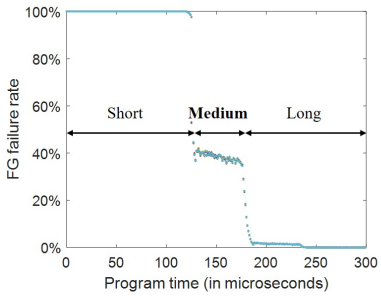

Figure 4: Programming time sweeping analysis.

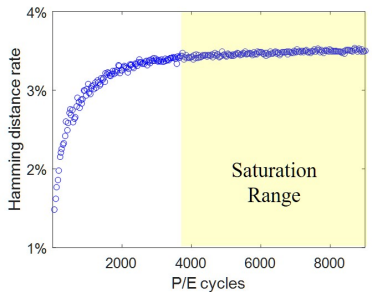

Figure 5: Hamming distance rates saturation observation.

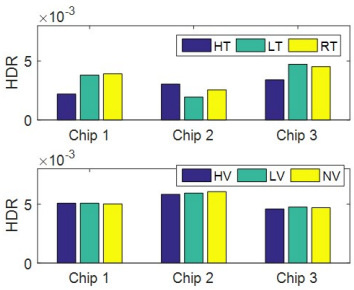

Figure 6: Temperature and supply voltage variation analysis.

Hamming distance rates are computed between the first partially programming and the rest of them. This result is presented in Figure 5. In this figure, the rates increase as more program/erase (P/E) cycles are performed. When the number of P/E cycles performed is high, where is the shaded range in Figure 5, the Hamming distance rate becomes nearly unchanged. This shaded ranged range is referred as the **saturation range**. In this range, the locations of the floating gate transistor failures are relatively stable even though the usage increases. Thus, this property can be utilized to generate reliable device ID.

3.2 Temperature/Supply Voltage Variations

Aside from the partially programming under nominal condition, various ambient temperature and supply voltage testing cases are also considered. To mimic the noise from the power supply, we utilized two different supply voltages: 3.6V (+10% high supply voltage, HV) and 3.0V (-10% low supply voltage, LV). In parallel, we conducted the experiments by applying low temperature (LT, 0 °C), room temperature (RT, 20 °C) and high temperature (HT, 80 °C) using a thermostream device. Thus, 9 test corners (e.g. HTHV, LTNV, etc.) are evaluated. For each test corner, the page is read 40 times after partially programmed for 150 microseconds.

Figure 6.(a) presents the results of the comparison of different temperatures and voltage corners. The intra-page Hamming distance rates (HDR) computed under HT and LT are close to the ones under room temperature (RT). According to this result, we can conclude that the temperature variations have negligible effects on the flash memory. A similar conclusion can be made for the supply voltage variations from Figure 6.(b). These properties make the partially programmed flash memory as a good PUF/ID generator.

4. RECYCLED FLASH DETECTION

In this section, the framework for detecting recycled flash memory and estimating usage is provided. In general, this framework models the flash memory aging by investigating the floating gate transistor failures. These failures are characterized by partially programming a page as discussed in Section 3. The capabilities of our framework can be categorized into three parts: (i) **Recycled flash memory determination** makes a yes-or-no decision on whether the flash memory under test is recycled or not. (ii) **Rough usage estimation** generates a rough assessment about the number of P/E cycles the flash memory has experienced. (iii) **Accurate usage estimation** refines the estimation result in rough usage estimation by applying a *slope analysis*. The recycled flash memory determination capability is crucial in critical applications since it distinguishes low-quality from high-quality flash memory chips. Both rough and accurate usage estimation capabilities allow non-critical applications to monitor the usage of the flash memory. The proposed framework is broken into two parts: enrollment phase and verification phase.

4.1 Enrollment Phase

The enrollment phase consists of two sub-phases: **Model enrollment** employs one memory page to build the aging model. In this phase, the enrollment will wear out (note that this sacrifice is negligible since a flash memory consists of an enormous number (more than 1 million) of pages). the enrolled page by consecutively applying P/E cycles. **Page enrollment** selects multiple pages across the flash memory and enrolls their partially programmed page contents. The enrolled data (model and page contents) can be stored somewhere in the flash memory under enrollment. Alternatively, these data can be stored in a remote database.

4.1.1 *Model Enrollment*

The *programming time sweep analysis* introduced in Section 3 should be performed first to determine a proper partially programming time length (i.e., the medium range). Once the partially programming time is determined, the model enrollment in Figure 7 can be executed. This model enrollment procedure can be split into three processes which are illustrated at the top right of this figure. During the *partially programming process*, the page is partially programmed with the predefined time. In total, this process will be performed for m times and the data programmed are all "0s". Then, the page content is read out and collected in a temporary storage. These stored data will be used to formalize the model (*post process*). After each partially programming, the *normal P/E cycles process* conducts n fully P/E cycles to mimic the regular operations of the flash memory. During this process, random data are programmed into the page under enrollment. The parameters (m and n) satisfying Equation (2) are selected according to

$$m \times (n+1) = E \tag{2}$$

where E stands for the endurance (i.e., the maximum number of P/E cycles allowed) of the page under enrollment. Besides the endurance information, another criteria, *usage*, is exploited to indicate how much a page is used. The usage

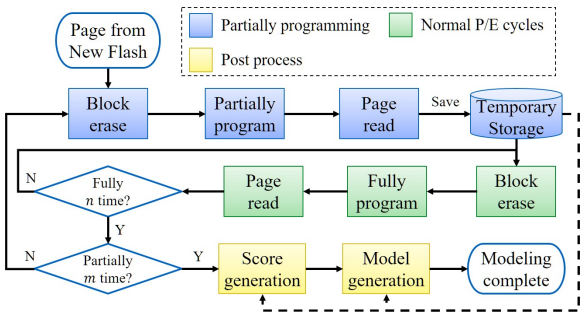

Figure 7: Model enrollment flow.

of a page is defined as the percentage of the P/E cycles it has experienced out of the maximal endurance.

In this framework, the partially programmed page content is exploited as the indicator of this page's current usage. According to Equation (2), the page usage u, after the k^{th} partially programming can be expressed as:

$$u_{i,j}(k) = \frac{k \times (n+1)}{E} = \frac{k}{m}, \quad 0 \leq k \leq m \tag{3}$$

where, $k = 0$ refers to the first partially programming which is performed on a brand new page. The subscripts i and j indicate the chip and page indexes respectively.

After all the m partially programming trials, the *floating gate transistor failure map* is formulated based on the collected page data. In this map, the FG transistors are sorted by their logic address and the failures are marked as "1". Thus, for each page, a binary failure map, FM, can be built. During the post process, a score $s_{i,j}$ will be computed for the j^{th} page from the i^{th} chip based on its failure map:

$$s_{i,j}(k) = \frac{HDR(FM_{i,j}(0), FM_{i,j}(k))}{FR_{i,j}(0)}, \quad 0 \leq k \leq m \tag{4}$$

where $HDR(A, B)$ refers to the Hamming distance rate between two binary vectors A and B. $FM_{i,j}(k)$ stands for the failure map after the k_{th} partially programming from the j^{th} page of i^{th} chip. $FR_{i,j}(0)$ presents the floating gate transistor failure rate obtained from the first partially programming, this value can be computed by Equation (1). The denominator in Equation (4) normalizes the Hamming distance rate between the failure map of the 1^{st} and k^{th} partially programming for each page. The comparisons between with and without normalization are provided in Figure 8. The Hamming distance rates vary from page to page even when their usages are the same, such variation hinders building a general model. To tackle this issue, we normalized the Hamming distance rates into scores, which converge when the usage is the same, as shown in (Figure 8.(b)). This relationship between actual memory usage $u_{i,j}(k)$ and our normalized score $s_{i,j}(k)$ is expressed as

$$s_{i,j}(k) = f_i(u_{i,j}(k)), \quad 0 \leq k \leq m \tag{5}$$

where $f_i(.)$ has a polynomial form as Equation (6).

$$f_i(x) = p_1 x^r + p_2 x^{r-1} + \cdots + p_r x + p_{r+1} \tag{6}$$

The value r can be determined by the designer and the coefficients $p_1, \ldots, p_{r+1}$. The fitting results from one page are provided in Figure 8.(b) in solid curve. For the i^{th} flash memory chip, only one model ($f_i(x)$) should be enrolled using one arbitrary page. These coefficients can be stored either in the flash memory itself or a remote database.

4.1.2 Page Enrollment

Following the model enrollment algorithm, the framework randomly enrolls an arbitrary number of pages and these pages are exploited as the usage "checking points". Each selected page will be partially programmed with the same time length. The data retrieved from each partially programmed page is converted into the floating gate transistor failure map. Note that the page enrollment phase does not completely wear out any memory pages. The number of enrolled pages can be determined by trading off the desired performance and required storage.

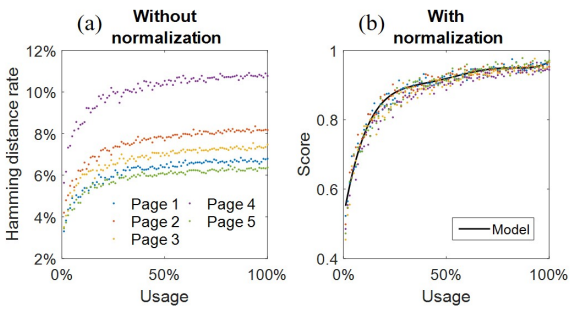

Figure 8: Normalizing effects in score generation.

4.2 Verification Phase

For **recycled flash memory determination**, a yes-or-no decision is made on whether the flash memory under test is recycled or not. In order to accomplish this task, a threshold is computed based on the enrolled model. By setting $u_{i,j}(k)$ as 0 in Equation (4), we can obtained this threshold for the i^{th} flash memory chip, thr_i.

$$thr_i = f_i(0) \tag{7}$$

After the threshold is determined, partially programming is performed and the failure maps can be extracted from the enrolled pages. Then, a score ($s_{i,j}(v)$) should be generated for each of these pages by the following equation.

$$s_{i,j}(v) = \frac{HDR(FM_{i,j}(e), FM_{i,j}(v))}{FR_{i,j}(e)} \tag{8}$$

Where, $FM_{i,j}(e)$ refers to the enrolled failure map, while $FM_{i,j}(v)$ represents the failure map generated during the verification. The floating gate transistor failure rate $FR_{i,j}(e)$ can be computed from the enrolled failure map $FM_{i,j}(e)$ by Equation (1). Finally, the decision is made by comparing thr_i and $s_{i,j}(v)$. If $s_{i,j}(v)$ is smaller than thr, the page under test is new, otherwise, this page is labeled as used. An overall decision of the flash memory under test can be made by combining the decisions from its enrolled pages. Either a majority voting or estimated usage averaging can be exploited to make the decision.

Rough usage estimation produces a rough assessment of the usage of flash memory under test. Instead of comparing this score with a threshold, the usage of the page under test can be estimated by the following equation.

$$u_{i,j}(v) = f_i^{-1}(s_{i,j}(v)) \tag{9}$$

where, $f_i^{-1}(.)$ indicates the inverse version of Equation (6). $u_{i,j}(v)$ refers to the estimated usage.

However, this estimation is not accurate enough in practice, especially in the "high usage" situation (reasons can be found in Section 6). To improve the accuracy, a new approach, **Accurate usage estimation**, which exploits the slope difference in score curve at different usage range is proposed. According to Figure 8.(b), the score grows faster in the low usage range than in the high usage range. If more accurate results are required to determine the remaining endurance, the accurate usage estimation should be applied by performing additional P/E cycles. An example of utilizing accurate usage estimation can be found in Figure 9. The solid black curve indicates the enrolled model of the i^{th} flash memory and the blue circles refer to the scores generated from the j^{th} page of this flash memory. The area in the dashed rectangle is shown in the inset on the right side. Assuming that the current page usage is $u_{i,j}(A)$. According to Equation (8), $s_{i,j}(A)$ can be computed by partially programming this page. After q P/E cycles, the usage of this page becomes $u_{i,j}(B)$ and the corresponding score at this time point is $s_{i,j}(B)$. Even though both $u_{i,j}(A)$ and $u_{i,j}(B)$ are unknown, the horizontal coordinate difference between points A and B can be calculated:

$$AC = u_{i,j}(B) - u_{i,j}(A) = \frac{q}{E} \tag{10}$$

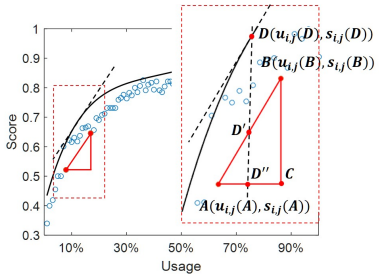

Figure 9: Example of accurate usage estimation.

where, E stands for the maximal page endurance (this information can be found in the flash memory datasheet). Then, the slope of segment AB can be obtained:

$$slope = \frac{BC}{AC} = \frac{E \times (s_{i,j}(B) - s_{i,j}(A))}{q} \quad (11)$$

By matching this slope with the points on the solid curve, point D is found of same slope (dashed line) as the segment AB. Since enrolled model is known, the usage at point D can be computed as $u_i(D)$. Points D' and D'' share the same horizontal coordinate as D. Additionally, the point D'' is approximately located in the middle of segment AC. Thus, the unknown usage $u_{i,j}(A)$ can be inferred by Equation (12).

$$u_{i,j}(A) = u_i(D) - \frac{AC}{2} = u_i(D) - \frac{q}{2E} \quad (12)$$

Comparing with the rough usage estimation, the accurate usage estimation provides a more precious result while paying more costs (e.g. performing additional P/E cycles).

5. ID GENERATION

The saturation range in Figure 5 can be utilized in ID generation, which is composed by two phases: enrollment and verification phase. In **enrollment phase**, P/E cycles are conducted on one page. If the recycled flash detection framework (presented in Section 4) has been implemented in advance, the page for model enrollment can be reused in this phase. Generally, t partially programming operation should be done to produce t floating gate transistor failure maps. Based on these failure maps, the following two types of floating gate transistors are identified:

- **Always-fail FG transistors**: transistors always experience failures in failure maps (i.e., stable "1"s in the failure maps).
- **Never-fail FG transistors**: transistors never fail in any failure map (i.e., stable "0"s in the failure maps).

Once these two types of FG transistors and their addresses are determined, the device ID can be enrolled following the procedures in Figure 10. Since the number of always-fail FG transistors differs from never-fail FG transistors, a transistor pool should be constructed with equal number of these two types of transistors. The enrolled ID with a length of w bits is split into two parts: always-fail and never-fail. These transistors are randomly selected and their addresses are in ascending order. These enrolled addresses can be stored in the flash memory after enrollment as helper data, the enrolled ID is stored in a remote database for verification. After the enrollment, these pages should be reserved for the dedicated purpose of device ID generation. This page address information should be stored in the flash memory under enrollment.

During the **verification phase**, the enrolled page is partially programmed. The device ID can be generated by the failure map. Then, this regenerated device ID can be compared with the enrolled ID for the verification purpose. Note that differing from the previous work in [11] [12] that employ

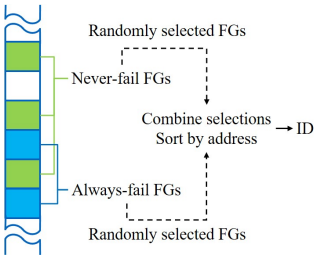

Figure 10: ID and address enrollment.

the natural bias of memory circuit, our method leverages the aging phenomenon of flash memory. Implementing the proposed ID generation framework benefits both the critical and non-critical applications by providing the remarked and cloned flash memory detection capability. Especially for the critical applications, the helper data for recycling detection (e.g. the enrolled model and page contents) can be remotely stored in the database. These data can be retrieved only when the device's ID is verified.

6. EXPERIMENTAL RESULTS

In this section, the performance of the proposed framework is evaluated with COTS flash memory, the configuration of which is listed in Table 1. 200,000 memory pages from 20 flash memory chips are tested in total. As shown in Section 3.2, the flash memory is insensitive to temperature and supply voltage variation, we conducted experiments under room temperature and nominal voltage condition.

Table 1: Experiment flash memory specifications

Manufactory	Micron
Model #	MT29F32G08CBACA
Page/Device size	4320 bytes/32Gb
Technology node	65 nm MLC NAND
Endurance	3,000 P/E cycles [4]

6.1 Recycled Flash Detection

In our experiment, 60 pages from 10 flash memory chips are utilized. For each chip, one page is employed to generate the model and the other five pages for evaluation. During the model enrollment phase, 100 partially programming steps are performed for each page. The partially programming time is set to be 150 microseconds. This model is represented in the 5-order polynomial from (i.e., $r = 5$ in Equation (5)). Additionally, 29 normal P/E cycles are executed before each partially programming. For **recycled flash memory determination**, the average detection accuracies of 50 pages are provided in Figure 11. The results denote a 100% detection accuracy even though the page is slightly used (around 5% of the maximal endurance).

The **rough usage estimation** method estimates the usage, $u_{i,j}(v)$, according to Equation (9). For each page under verification, 100 scores, $s_{i,j}(v)$, are generated by Equation (8). The estimated usage of each flash memory chip can be computed by averaging the estimated usages. The comparisons between actual and estimated usages from 10 flash memory chips are provided in Figure 12. The dots in different colors refer to the results from different chips. The solid line indicates the errorless estimation scenario. According to this figure, the estimation accuracy decreases with higher usage. This is because when the usage becomes higher, the growth of score keeps slowing down (Figure 8.(b)). As a result, a small fluctuation in the score leads to a bigger error. In our proposed framework, the **accurate usage estimation** improves the estimation accuracy in high usage scenario by taking the score slope variations into consideration. A significant improvement can be seen in Figure

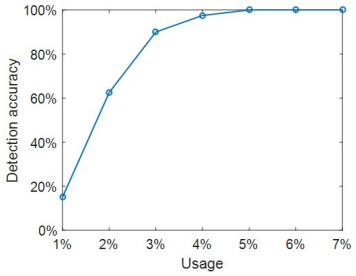

Figure 11: Recycled flash memory determination accuracy analysis.

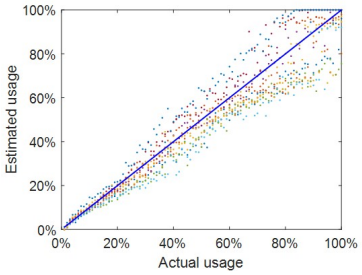

Figure 12: Rough usage estimation accuracy analysis.

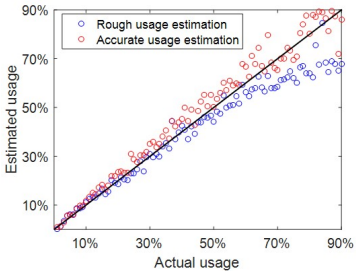

Figure 13: Accurate usage estimation accuracy analysis.

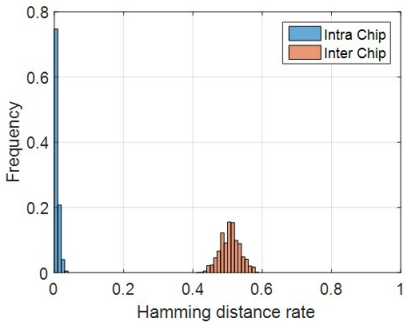

Figure 14: ID Uniqueness and robustness analysis.

13: the estimated usages are close to the errorless estimation when the usage is higher than 50%. Note that in our framework, additional P/E cycles (i.e., 10% of the maximal endurance) are required to accomplish the usage estimation, therefore, the usage estimation coverage is from 0% to 90%.

6.2 ID Generation

To evaluate the proposed ID generation method, the uniqueness and reliability of the IDs generated from 10 flash memory chips are analyzed. The enrollment is performed after 4,000 P/E cycles, which is 1.3 times of the chip's maximal endurance [4]. A 256-bit ID and the corresponding memory cell addresses are enrolled for each chip from each page. For verification, the enrolled pages are stressed by 12,000 more P/E cycles, which is four times of the maximal endurance. Then, 60 IDs for each chip are regenerated and their reliability is evaluated. The inter-chip Hamming distance rate is obtained by comparing the IDs generated from different chips. The evaluation results obtained from 1,000 IDs are presented in Figure 14. According to this figure, the reliability of generated ID is no less than 99.3% while the uniqueness is close to 50%.

Compared to ECID, the ID generated by the proposed framework is completely unclonable. Its uniqueness also eliminates the chip identification errors. Additionally, since this ID is very reliable, its output can also be used for the key generation with little need for error correction codes (ECCs). If ECC is implemented, the helper data can be directly stored in the flash memory due to its non-volatile nature. This makes key generation from embedded flash more realistic than SRAM in SoCs.

7. CONCLUSION AND FUTURE WORK

In this paper, a comprehensive fake flash memory detection framework (FFD) is proposed for combatting recycled, remarked and cloned flash chips. This framework utilizes the floating gate transistor failure, which is induced by partially programming, to achieve recycling detection and ID generation. 200,000-page data collected from 20 flash memory chips are used to evaluate the performance of FFD. The results demonstrate that our framework performs well in three perspectives scenarios: 1) 100% fake flash memory detection accuracy when the flash memory is slightly used. 2) Usage characterization with high accuracy, $\geq 95\%$. 3) The device ID generated by the proposed method show excellent reliability and uniqueness after large numbers of ID inquires. In the future, we will extend this analysis to other flash memory types, such as TLC NAND and NOR flash memory.

8. REFERENCES

[1] All about fake flash drives 2013 | ebay. http://www.ebay.com/gds/All-About-Fake-Flash-Drives-2013-/10000000177553258/g.html.

[2] Feds close huge chip counterfeiting case. http://venturebeat.com/2011/09/25/feds-close-the-books-on-a-huge-chip-counterfeiting-scheme/.

[3] The global impact of counterfeit parts. https://www.nts.com/ntsblog/global-impact-counterfeit-parts/.

[4] Micron technology, inc. - mt29f32g08cbacawp-z. https://www.micron.com/parts/nand-flash/mass-storage/mt29f32g08cbacawp-z.

[5] The 'ticking time bomb' of counterfeit electronic parts. http://www.industryweek.com/procurement/ticking-time-bomb-counterfeit-electronic-parts.

[6] R. Bez, E. Camerlenghi, A. Modelli, and A. Visconti. Introduction to flash memory. *Proceedings of the IEEE*, 2003.

[7] M.-S. Kim, D.-I. Moon, S.-K. Yoo, S.-H. Lee, and Y.-K. Choi. Investigation of physically unclonable functions using flash memory for integrated circuit authentication. *IEEE Trans. Nanotechnol*, 2015.

[8] P. Prabhu, A. Akel, L. M. Grupp, S. Y. Wing-Kei, G. E. Suh, E. Kan, and S. Swanson. Extracting device fingerprints from flash memory by exploiting physical variations. In *TRUST*. Springer, 2011.

[9] M. M. Tehranipoor, U. Guin, and D. Forte. *Counterfeit Integrated Circuits: Detection and Avoidance*. 2015.

[10] Y. Wang, W.-k. Yu, S. Wu, G. Malysa, G. E. Suh, and E. C. Kan. Flash memory for ubiquitous hardware security functions: true random number generation and device fingerprints. In *S&P*, pages 33–47. IEEE, 2012.

[11] X. Xu and D. E. Holcomb. Reliable puf design using failure patterns from time-controlled power gating. In *Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), 2016 IEEE International Symposium on*, pages 135–140. IEEE, 2016.

[12] X. Xu, A. Rahmati, D. E. Holcomb, K. Fu, and W. Burleson. Reliable physical unclonable functions using data retention voltage of sram cells. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 34(6):903–914, 2015.