Extracting Robust Keys from NAND Flash Physical Unclonable Functions

Shijie Jia[1,2,3], Luning Xia[1,2](✉), Zhan Wang[1,2], Jingqiang Lin[1,2], Guozhu Zhang[1,2,3], and Yafei Ji[1,2,3]

[1] Data Assurance and Communication Security Research Center, Chinese Academy of Sciences, Beijing, China
{jiashijie,halk,zwang,linjq,zhangguozhu,jiyafei12}@is.ac.cn
[2] State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[3] University of Chinese Academy of Sciences, Beijing, China

Abstract. Physical unclonable functions (PUFs) are innovative primitives to extract secret keys from the unique submicron structure of integrated circuits. PUFs avoid storing the secret key in the nonvolatile memory directly, providing interesting advantages such as physical unclonability and tamper resistance. In general, Error-Correcting Codes (ECC) are used to ensure the reliability of the response bits. However, the ECC techniques have significant power, delay overheads and are subject to information leakage. In this paper, we introduce a PUF-based key generator for NAND Flash memory chips, while requiring no extra custom hardware circuits. First, we present three methods to extract raw PUF output numbers from NAND Flash memory chips, namely partial erasure, partial programming and program disturbance, which are all based on the NAND Flash Physical Unclonable Function (NFPUF). Second, we use a bit-map or a position-map to select the cells with the most reliable relationship of the size between raw NFPUF output numbers. Only the selected cells are used for key generation. Finally, we describe the practical implementations with multiple off-the-shelf NAND Flash memory chips, and evaluate the reliability and security of the proposed key generator. Experimental results show that our NFPUF based key generator can generate a cryptographically secure 128-bit key with a failure rate $< 10^{-6}$ in 93.83 ms.

Keywords: Physical Unclonable Functions (PUFs) · NAND flash · Process variation · Secret keys · Error correction

1 Introduction

As electronic devices have become interconnected and ubiquitous, people are increasingly depending on electronic devices to perform sensitive tasks and to

This work was partially supported by the National 973 Program of China under award No. 2013CB338001 and the Strategic Priority Research Program of Chinese Academy of Sciences under Grant XDA06010702.

handle sensitive information. As a result of the merits of NAND Flash memory, such as small size, low power consumption, light weight, high access speed, shock/temperature resistance and mute characteristics [12], now virtually all portable electronic devices such as smartphones, SD cards, USB memory sticks and tablets use NAND Flash memory as nonvolatile storage.

Now many electronic devices of embedded systems have become to contain more confidential information, and many applications need to identify and authenticate users. Therefore, the secret keys used by the devices and the applications should be protected to ensure the security of the communication system. However, in the real world implementations of cryptosystems, the cryptographic keys are recently revealed from nonvolatile memories by sophisticated tampering methods [3,9,21,25]. Based on the above situation, we leverage the special virtue of NAND Flash to avoid storing the secret key in the nonvolatile memory directly.

In order to prevent both the invasive and noninvasive physical attacks, Physical Unclonable Functions (PUFs) have been attracting wider attention and studied intensively in recent years. Due to the advantages of physical unclonability and tamper proof, PUFs are used to avoid storing actual bits of the secret keys in the storage memory. Generally, PUFs are engaged in two typical classes of applications, namely authentication and secret key generation. In the authentication applications, the responses of the PUFs can be designed to tolerate a certain amount of errors. While in the secret key generation applications, the responses of the PUFs need to be consistent [16]. The conventional method to ensure the robustness and the reliability of the responses is to utilize fuzzy extractors [6,14]. Traditionally, fuzzy extractors employ an Error-Correcting Code (ECC) and a cryptographic hash function. There have been several state of the art papers cite the use of ECC with PUFs to generate cryptographic keys [2,10,15,29]. However, ECC is not viable for resource constrained electronic devices. First, the error rates for PUFs across environmental variations can be as high as 25 % [5], making a straightforward use of ECC infeasible [15,24], namely the codeword sizes required will be too large in practice. Second, ECC is generally performed by specialised hardware chips, which not only requires tremendous area and power overheads, but also scales up as the number of bits of correction increases [1]. Third, ECC requires additional helper or syndrome data to be publicly stored for the regeneration of the key. The helper data of ECC reveals information about the outputs of the PUFs [23]. Thus, error reduction techniques can be applied to reduce the cost of ECC and to ensure the reliability and security of the responses.

In this work, we focus on the NFPUF and the error reduction techniques to generate cryptographic robust keys. First, we present three methods (partial erasure, partial programming and program disturbance) to extract raw NFPUF output numbers from NAND Flash memory chips. Second, we introduce two methods (the bit-map and position-map) to select the cells with the most reliable relationship of the size between raw NFPUF output numbers. In other words, the size relationship of the raw NFPUF output numbers from the selected cell pairs is almost constant during the whole lifetime of the NAND Flash chips. At last, we evaluate the reliability and security of the proposed key generator. The proposed key generator can get reliable and robust keys for the electronic devices with limited hardware resources, meanwhile it reduces the implementation costs and hardware overheads of ECC significantly.

Our Contributions. In this paper we introduce a robust key generator based on NFPUF for NAND Flash memory chips. The main contributions of this paper are as follows:

– We present the first implementation of secret key generator from unmodified commercial NAND Flash memory chips. Most importantly, the proposed key generator can be applied to any NAND Flash memory chips, extending the functionality of NAND Flash memory chips, while requiring no extra hardware circuits overheads.
– We describe three specific methods to extract raw NFPUF output numbers from NAND Flash memory chips. Particularly, the partial erasure method is proposed for the first time.
– We present two methods to select the NAND Flash memory cells with the most reliable relationship of the size between raw NFPUF output numbers. It reduces the system overheads of ECC significantly, and it is feasible for the electronic devices even with constrained hardware resources.
– We evaluate the reliability and security of our proposed key generator with multiple NAND Flash memory chips from different manufacturers by plenty of experiments.

Organization of the Paper. The organization of the rest of this paper is as follows. Section 2 introduces the related work. The background is introduced in Sect. 3. We will present the specific secret key generator in Sect. 4. Implementation details and evaluations are shown in Sect. 5. Finally, conclusions are given in Sect. 6.

2 Related Works

Pappu et al. introduced the Physical One-Way Functions (POWFs) in [18]. They used a transparent optical medium with a three-dimensional micro-structure as a POWF. The concept of silicon PUFs was introduced in [7,8]. Silicon PUFs have substantial challenge-response pairs (CRPs) owing to the manufacturing process variations, so it is impossible for the attacker to clone all the potential CRPs [13]. Our key generator also takes advantage of the manufacturing process variations.

Škorić et al. presented a key extraction method from the bit-string extraction of noisy optical PUFs in [20]. Different PUFs circuit designs based on ring oscillators were introduced in [13,15,23,29]. The first construction of a PUF intrinsic based on the power-up state of SRAM memory on current FPGAs was presented in [10] to solve the IP protection problem. An efficient helper data key extractor technique was introduced to generate secret keys on FPGAs in [2], which leverages several complicated concatenated codes (repetition code and ECC) to ensure the reliability of the keys. Our key generator does not require a power cycle or the special circuit designs that the prior PUFs need, while it can be done by any electronic devices with commercial off-the-shelf NAND Flash memory chips as nonvolatile storage.

Xu et al. introduced the sources of variations in Flash memory for PUFs in [28]. It points out that the uniqueness and robustness of the NFPUF are indeed universally applicable, rather than just a phenomenon presented in the limited selection. In general, NFPUF distributions are translated to threshold voltage distributions via tunneling current during programming and erasing operations to analyze its physical origins. Our key generator also leverages the threshold voltage distributions of the NAND Flash memory cell transistors to extract the raw NFPUF output numbers.

Prabhu et al. evaluated seven techniques to extract unique signatures from Flash devices based on the observable effects of process variations as device fingerprints [17]. They exploited formal correlation metric (the Pearson correlation) to distinguish whether the extracted signatures were from the same page or different pages, then they could uniquely identify individual Flash devices. Yang et al. took advantage of the uncertainty of Random Telegraph Noise (RTN) from Flash memory to provide two security functions: true random number generation and digital fingerprinting [27]. As a result of the high uncertainty of the random numbers and device fingerprints, neither the techniques they proposed could be used to extract unique and reproduceable secret keys with a tiny bit error rate. Our key generator leverages the specific physical characteristics of the NAND Flash memory cells to extract numbers, then we select the cells with the most reliable relationship of the size between the extracted numbers during the whole life of the chip for key generation. Our key generator ensures the reproducibility and the reliability of the key, meanwhile, it avoids the costly overheads of ECC.

3 Background

The secret keys generator that we will describe in Sect. 4 bases on the composition of NAND Flash memory cells, and how NAND Flash memory chip organize the cells into memory arrays. The specific NAND Flash memory cell composition and array organization lead to noises exist in NAND Flash memory cells universally. This section summarizes the primary characteristics of NAND Flash memory chips that we rely on for this study.

3.1 Uncertain States of NAND Flash Memory Cells

As Fig. 1(a) shows, a NAND Flash memory cell is based on a floating gate metal oxide semiconductor (MOS) transistor. There are two gates in a floating gate transistor. The top one is called control gate, which is capacitive coupled. The bottom one is the floating gate, which is surrounded by dielectrics. The special

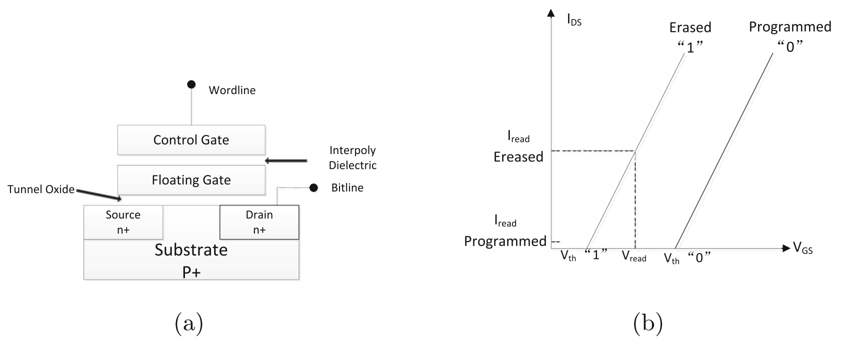


Fig. 1. (a) NAND flash memory cell structure. (b) Threshold voltage schematic diagram.

property of dielectrics makes the NAND Flash memory nonvolatile. In general, a triple layer of oxide-nitride-oxide isolates the two gates. In addition, the thin oxide between the floating gate and transistor channel is known as tunnel oxide. The source and drain electrodes are heavy doped, and they are electron-rich (n-type). While the substrate is less doped, and it is electron-deficient (p-type).

Both programming and erasing operations of common-ground NAND Flash memory cells are by Fowler-Nordheim (FN) tunneling, which is a quantum-mechanical tunneling mechanism induced by the electric field [19]. The presence or absence of trapped charge on the floating gate is expressed as logical state "0" or logical state "1" respectively.

The trapped charge affects the threshold voltage (V_{th}) of the transistor [28]. When an electron charge is stored in the floating gate, the threshold voltage of this transistor increases, and the increase amplitude is proportional to the stored charge. As illustrated in Fig. 1(b), the charge stored in the floating gate discourages the presence of current in the transistor channel, then the cell is sensed and translated into logical state "0", thus the NAND Flash memory cell will be in the programmed state. On the contrary, when the floating gate has no electron charge, then it forms a conductive path between the drain and the source electrodes, creating a current (I_{DS}) in the transistor channel, and hence the cell will be sensed and translated into logical state "1", then the NAND Flash memory cell will be in the erased state. In conclusion, by applying an appropriate voltage to the control gate and measuring the current flow through the transistor channel of the target cell, a NAND Flash memory chip can effectively measure the threshold voltage of the cells, and determine the logical states of the cells.

However, on account of variations in manufacturing processes, the threshold voltages of ($V_{th\text{“1”}}$) and ($V_{th\text{“0”}}$) vary from cell to cell. When the threshold voltage is not shifted sufficiently from the programmed state to the erased state, and vice versa, then the cell will be in an uncertain state. In these cases, the cell can be expressed as either logic state "0" or logic state "1". In this paper, we propose the partial erasure and the partial programming methods, both the methods exploit the uncertain states of the NAND Flash memory cells to extract raw NFPUF output numbers.

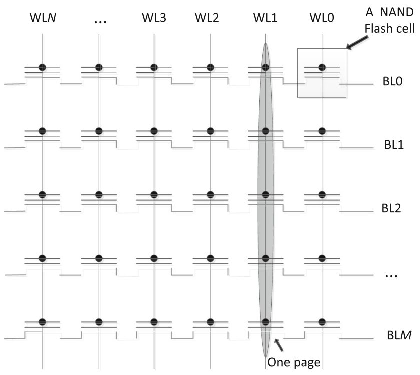


Fig. 2. NAND flash memory array organization.

3.2 Disturbance Related to NAND Flash Memory Array Organization

As Fig. 2 illustrates, the NAND Flash memory cells are arranged in a coherent and structured manner, normally in arrays, to achieve high density. Due to the array organisation, each cell can be accessed by its specific row and column address. In general, the NAND Flash memory cells are grouped into pages (typically 512 bytes-4 KB) and blocks. A block contains dozens of (typically 32–128) adjacent pages [22]. Thousands of independent blocks make up a NAND Flash memory chip. The common drain connection that the rows of cells share is called a bitline (BL), while the common poly-2 gate connection that the columns of cells share is called a wordline (WL) [9]. A single wordline connects the gates on all the transistors in a page or more than one page, and the latter case is particularly general for multi-level cells (MLC) NAND Flash memory chips. Programming and reading operations are performed on the unit of a page, whereas erasing operation must be performed on an entire block. What is more, the pages in a block must be programmed sequentially. The programmed sequence is designed to minimize the programming disturbance between the neighboring pages, which aims to avoid undesired voltage shifts in the pages despite not being selected.

However, although the array organization of the NAND Flash memory is specially designed, there still exists electrical influence between adjacent NAND Flash memory cells. During the programming and reading operations, a high voltage is applied to the wordlines of the selected pages, meanwhile producing an intermediate gate voltage to the neighbouring wordline. After multiple repeating operations, the intermediate gate voltage makes the according adjacent NAND Flash memory cells flip, which is a process of quantitative change to qualitative change. In particular, as the result of the capacitive coupling between the selected wordline and the physically adjacent wordline, the effect of programming operations is much stronger [30]. In this paper, we propose the program disturbance method, which leverages the effects between the adjacent pages to extract raw NFPUF output numbers.

4 Robust Key Generation

As Sect. 3.1 discussed the physical sources of variations in NAND Flash memories for NFPUF, not only the initial and after-erase voltages, but also the initial and after-program voltages for a same NAND Flash memory chip may vary from cell to cell due to the manufacturing process variations. Section 3.2 discussed the disturbance related to the specific NAND Flash memory array organization. The repeating programming operations to a same page, resulting in its neighbouring page unexpected bit variations.

Both the above phenomena are due to the maximum density of NAND Flash memory cells. Since process variations are beyond the manufacturers' control, small variations in tunnel oxide thicknesses and control gate coupling ratio may make a big difference in the threshold voltage of the floating gate transistor [28]. So even an adversary who has the detail information of the NFPUF principle still cannot clone the NFPUF.

4.1 Extracting Raw NFPUF Output Numbers

In this paper, we propose three methods to extract raw NFPUF output numbers, namely partial erasure, partial programming and program disturbance. We will discuss the details of the three methods in order.

Partial Erasure. This method exploits the feature of NAND Flash memory chip that the initial and after-erase voltages vary from cell to cell due to the manufacturing process variations. Algorithm 1 provides the pseudo-code. First, we erase the selected block ($BlockNum$), then we program all the cells of the selected page ($PageNum$) belonging to the selected block to logic state "0". Second, we perform fixed number ($PENum$) of partial erasure operations to the selected page. The time of each partial erasure operation (T_e) is also fixed. After each partial erasure operation, some cells in the selected page will have been erased enough to flip their states from logic state "0" to logic state "1". Therefore, we record the number of partial erasure operations that the selected cells need to flip. Third, after the fixed number of partial erasure operations, some cells may have not flipped, then the value of $PENum$ plus 1 is assigned to these cells. At last, we extract the raw NFPUF output numbers by repeating partial erasure operations from the specific block and page.

Partial Programming. Similar with the partial erasure method, the partial programming method leverages the feature of NAND Flash memory chip that the initial and after-program voltages vary from cell to cell due to the manufacturing process variations. First, we erase the selected block. Second, we perform fixed number ($PPNum$) of partial programming operations to the selected page, the time of each partial programming operation (T_p) is also fixed. After each partial programming operation, some cells may have been programmed enough to flip their states from logic state "1" to logic state "0". Therefore, we record the number of partial programming operations that the selected cells need to flip. Third, after the fixed number of partial programming operations, some cells

Algorithm 1. Partial erasure: Extract the raw NFPUF output numbers by repeating partial erasure operations to the specific block and page.

```
Require:
    The number of block to erase ($BlockNum$);
    The number of page to read ($PageNum$);
    The number of cells to record ($CellsNum$);
    The time of each partial erasure operation ($T_e$);
    The number of partial erasure operations ($PENum$);
Ensure:
    The number of partial erasure operations of each NAND Flash memory cell need
    to reach the erased state ($RawPuf[CellsNum]$).
 1: $Erase(BlockNum)$;
 2: $Program(PageNum, 0)$;
 3: for $i = 1; i <= PENum; i++$ do
 4:     $PartiallyErase$ ($T_e$, $BlockNum$);
 5:     $Read(PageNum)$;
 6:     for All the selected cells to record do
 7:         if The first observation of the cell flips from 0 to 1 then
 8:             $RawPuf$[The Position of the cell in the selected cells ]= $i$;
 9:         end if
10:     end for
11: end for
12: for The cells have not flipped after PENum partial erasure operations do
13:     $RawPuf$[The Position of the cell in the selected cells]= $PENum + 1$;
14: end for
```

may have not flipped, then the value of $PPNum$ plus 1 is assigned to these cells. At last, we extract the raw NFPUF output numbers by repeating partial programming operations from the specific block and page.

Program Disturbance. Unlike the above two methods, this method is based on the disturbance between the adjacent pages due to the specific NAND Flash memory array organization. The repeating programming operations to a same page, resulting in its neighbouring page unexpected bit variations. First, we erase the selected block. Second, we perform fixed number ($PDNum$) of programming operations to the selected page. After each programming operation, some cells in its physically adjacent page will have been programmed enough to flip their states from logic state "1" to logic state "0". Therefore, we record the number of programming operations that the selected cells in its physically adjacent page need to flip. Third, after the fixed number of programming operations, some cells may have not flipped, then the value of $PDNum$ plus 1 is assigned to these cells. At last, we extract the raw NFPUF output numbers by repeating programming operations from the specific block and pages.

4.2 Extracting Robust Keys from the Raw NFPUF Output Numbers

If the PUFs are measured repeatedly, the cell-wise extracted numbers apparently will have non-negligible fluctuations as a result of noises. Therefore, the raw PUFs output numbers are not fit as secret key directly [4]. In general, fuzzy extractors are used to ensure the reliability of the PUFs response outputs. Fuzzy extractors employ an ECC and a cryptographic hash function. As a result of the tremendous raw bits and helper data overheads are needed in real system implementations of ECC, it is expensive to implement in electronic devices with limited hardware resources [5,24].

In this work, our objective is to extract robust keys from raw NFPUF output numbers with a tiny bit error rate, meanwhile reducing the costly overheads in the implementations of ECC. Therefore, it will be feasible for NAND Flash devices even with constrained hardware resources to generate robust keys.

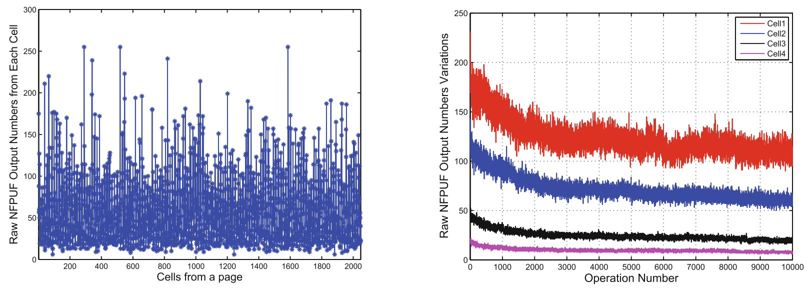
Due to the layout and spatial variations of NAND Flash memory chips, a consistent systematic variation exists among the average page NFPUF output numbers and the average block NFPUF output numbers [28]. Typically NAND Flash memory can withstand 100,000 program and erase (P/E) cycles for single-level cell (SLC) type and 10,000 for MLC type [26]. Repetitive P/E cycles can alter the raw extracted numbers of the cells due to cyclic endurance aging effects [27]. As Fig. 3(a) illustrates, the raw NFPUF output numbers from the cells of the same page present an irregular distribution and have great difference. However, as shown in Fig. 3(b), although the raw NFPUF output numbers slightly decreased over P/E cycles, the relationship between the size of raw NFPUF output numbers extracted from different cells is relatively stable during the whole lifetime of the NAND Flash memory chips.

Our key generator is to find the NAND Flash memory cells with the most reliable and stable relationship between the size of raw NFPUF output numbers. We translate the size relationship of the raw NFPUF output numbers into binary numbers as a robust secret key, meanwhile we record the according cell position information as helper data for key regeneration.

We introduce two methods to extract secret keys by selecting the NAND Flash memory cells with the most reliable relationship of the size between raw NFPUF output numbers, namely the bit-map method and position-map method. Only the selected cells are used for generation of the key.

Figure 4 describes an example of our two key extraction methods. We extract ten raw NFPUF output numbers ($RawPuf$) from ten cells, and the position of the cells starts from 0×0065 to 0×006e (due to the page size of the commercial off-the-shelf NAND Flash memory chip is from 512 bytes to 4 KB generally, so we use 16 bits to represent an address of a cell). The specific quantities of the raw NFPUF output numbers and the according bit number of the extracted key are related to the bit error rate of the key, which will be discussed in detail in the experimental section.

Bit-Map Method. First, we compare the adjacent raw NFPUF output numbers in pairs and record the absolute values of the corresponding D-values



(a) Raw NFPUF output numbers from a page.

(b) Fluctuations of raw NFPUF output numbers from four cells of a MLC type NAND Flash memory chip.

Fig. 3. The distributions of raw NFPUF output numbers.

($ADvalue$). Second, we sort the the recorded $ADvalues$ from small to large, and only a part of the cell pairs with the top largest $ADvalues$ are selected to generate a key (here three cell pairs are selected). Third, we assign "1" to the selected cell pairs, and assign "0" to the rest as helper data ($BitMap$). Fourth, if the former raw NFPUF output number is bigger than the latter one in the selected cell pairs, we allocate "1" to the key ($Key1$), if not, then we allocate "0" to the key. At last, we obtain the secret key ($Key1$) and store the $BitMap$ for regeneration of the key.

Position-Map Method. First, we sort the extracted raw NFPUF output numbers from small to large ($SortedRawPuf$). Second, we select a part of the top smallest and the top largest cells to make up the selected cell pairs (here both the cells with the top three smallest and the top three largest raw NFPUF output numbers are selected). Then we sort the raw NFPUF output numbers of each cell pairs according to the cell positions ($PairRawPuf$). Third, we record the cell position of the selected NAND Flash memory cells as helper data ($PositionMap$). Fourth, we compare the raw NFPUF output numbers of the selected cells. If the former is bigger than the latter one, we allocate "1" to the key ($Key2$), if not, we allocate "0" to the key. At last, we obtain the secret key ($Key2$) and store the $PositionMap$ for regeneration of the key.

Note that both the $BitMap$ and $PositionMap$ just represent the location of the selected NAND Flash memory cells, and they have nothing to do with the relationship of the size between the raw NFPUF output numbers. Therefore, the helper data carries no information about the polarity of the bits in the key. Hence the helper data does not leak any information about the key, unless there is a location-based correlation found in the numbers generated from the NFPUF. As the proposed key generator is based on the manufacturing process variations, which is a random process, and hence the polarity of the bits of the key is also random. Therefore, the helper data of this study is significantly more resilient to information leakage as compared to the helper data in conventional ECC.

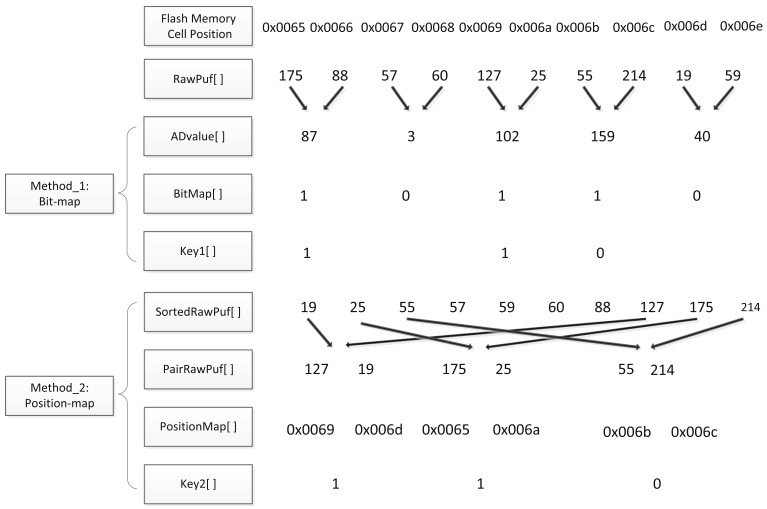


Fig. 4. Schematic diagram of the two key extraction methods.

5 Implementation and Evaluation

In this section, we present the experimental facilities used in this study, and evaluate the primary characteristics of reliability and security of the key generator based on NFPUF.

5.1 Tested Device

To extract raw NFPUF output numbers from NAND Flash memory chips, we use a custom PCB test board that contains the STM32F103VCT6 controller, which has a common ARM Cortex-M3 32-bit RISC core. With the controller, we can send program, read, and erase operations to the tested NAND Flash memory chips at will. This common device shows that our robust key generator can be applied to commercial off-the-shelf NAND Flash memory devices with no extra integrated circuits.

We evaluate NFPUF with a set of NAND Flash memory chips from different manufacturers. Table 1 shows the chips we use in this study.

5.2 Experimental Results and Evaluation

In order to evaluate the performance of the proposed key generator based on NFPUF, we need to analyse the primary characteristics of the security and reliability of the generated keys, such as speed (for performance), reproducibility (for reliability), uniqueness (for security), and randomness (for high-entropy).

Speed. Both the generation and regeneration processes of our proposed key generator need multiple program or erase operations, so the throughput of the

Table 1. Tested NAND flash memory chips.

Chip	Manufacturer	Part number	Capacity	Quantity	Technology
A	Samsung	K9K8G08U0M	8 Gbit	10	90 nm SLC
B	Samsung	K9F2G08U0B	2 Gbit	10	SLC
C	Micron	MT29F4G08ABA DAWP:D	4 Gbit	5	34 nm SLC
D	Micron	MT29F16G08CB ACAWP:C	16 Gbit	5	MLC
E	Intel	JS29F64G08AAME1	64 Gbit	5	MLC
F	Hynix	HY27UF084G2B	4 Gbit	5	SLC
G	Numonyx	NAND04GW3B2DN6	4 Gbit	5	57 nm SLC

proposed key generator varies significantly depending on the program and erase characteristics of the selected NAND Flash memory chips. Table 2 shows the parameters of the proposed three raw NFPUF output numbers extraction methods of the selected NAND Flash memory chips.

First, we find out the typical block erase time (t_{BERS}) and the typical page program time (t_{PROG}) from the datasheet of each NAND Flash memory chip. Second, we determine the time of each partial erasure operation (T_e) and the time of each partial programming operation (T_p) by trial and error until we get obviously diacritical outputs from each cell. The vast experimental results show that the T_e should be about the 1/12 of t_{BERS}, and the T_p should be about the 1/20 of t_{PROG}. To determine the number of partial erasure operations ($PENum$), we can partially erase the specific block and page with the selected T_e repeatedly until 99 % cells are erased in a page. In the same way, we can determine the number of partial programming operations ($PPNum$) by repeatedly partial programming operations with the determined T_p until 99 % cells are programmed in a page. At last, we can determine the number of repeated programming operations ($PDNum$) by normal programming operations to a specific page until 99 % cells are programmed in its adjacent page.

With the determined parameters in Table 2, we obtain the average throughput of the raw NFPUF output numbers with the NAND Flash memory chips from five manufactures, and it is shown in Table 3. The average throughput ranges from 7.35 Kbits/second to 22.38 Kbits/second. On account of the operation time, the average throughput of the partial programming method shows the highest speed, the next one is the partial erasure method, while the program disturbance method shows the slowest speed comparatively.

To get a 128-bit key with the bit error rate $< 10^{-6}$, we need 18.28 Kbits raw NFPUF output numbers for the bit-map method, so we can get a 128-bit key in 816.8 ms to 2.48 s. While we only need 2.1 Kbits raw NFPUF output numbers for the position-map method, so we can get a 128-bit key in 93.83 ms to 285.7 ms.

In our experiments, the average throughput is largely limited by the timing of the asynchronous interface, which is controlled by an ARM microcontroller with CPU frequency of 72 MHz and the 8-bit bus of the NAND Flash memory chips.

Table 2. Parameters setting of the raw NFPUF output numbers extraction methods.

Chip	$t_{BERS}(\mu s)$	$T_e(\mu s)$	$PENum$	$t_{PROG}(\mu s\ )$	$T_p\ (\mu s)$	$PPNum$	$PDNum$
A	1500	180	250	200	10	245	3912
B	1500	180	250	200	10	245	4002
C	700	85	200	200	11	200	3017
D	700	85	200	200	11	200	3123
E	3000	358	230	1200	58	220	3438
F	1500	179	240	200	10	250	3621
G	1500	181	235	200	11	240	3419

Table 3. The average throughput of the raw NFPUF output numbers(Kbits/second).

Method	Sumsung	Micron SLC	Micron MLC	Intel MLC	Hynix	Numonyx
Partial erasure	14.67	13.92	13.87	12.21	14.52	13.68
Partial programming	22.38	17.47	16.32	14.45	18.36	17.18
Program disturbance	9.72	8.78	7.89	7.35	9.74	8.29

The throughput performance can be much higher if the data can be transferred more quickly through the controller interface.

Reproducibility. In order to indicate the reproducibility of the PUF outputs, we evaluate the intra-chip variation, namely the number of bits changes when regenerated from a single PUF with or without environmental changes [23]. Ideally, the intra-chip variation should be 0 %.

To reduce the cost of ECC, we propose the bit-map and position-map methods to select the NAND Flash memory cells with the most reliable relationship of the size between raw NFPUF output numbers. For our reference implementation, we aim to obtain a 128-bit key with intra-chip variation $< 10^{-6}$, which means that our proposed key generator is applicable and reliable during the whole lifetime of the NAND Flash memory chips.

As Fig. 5 illustrates, we evaluate the average intra-chip variation with temperature and aging variations. We extract Y bits key from X raw NFPUF output numbers. The x-axis represents the ratio of Y in the X (as we use the relationship of the size between two raw NFPUF output numbers to extract a bit for the key, so the maximum of Y is $X/2$). The y-axis shows the according average intra-chip variation of the tested NAND Flash memory chips.

The variations of temperature influence thermal noise amplitude, while RTN amplitude stays almost the same [27]. Since we extract raw NFPUF output numbers based on RTN primarily, as Fig. 5(a) and (c) show, there is little difference across different temperatures.

NAND Flash memory chips wear-out over time due to program/erase (P/E) operations are performed. The average page NFPUF decreases slightly as P/E cycles increases [28]. However, we test the chips in Table 1 under different temperature conditions with our test board to verify the relationship of the size between raw NFPUF output numbers, all the chips show the same result as Fig. 3(b), namely the relationship of the size between raw NFPUF output numbers is rather stable. Therefore, we can see in the Fig. 5(b) and (d), the aging influence between different P/E cycles is also unconspicuous.

In the bit-map method, the intra-chip variation decreases from 14.42 % to less than 10^{-6} as the ratio of Y/X decreases from 0.5 to 0.055. In this case, when the ratio of Y/X is 0.055, now 2340 NAND Flash memory cells are needed to generate a 128-bit robust key, and the length of helper data is 2340/2=1.14 Kbits.

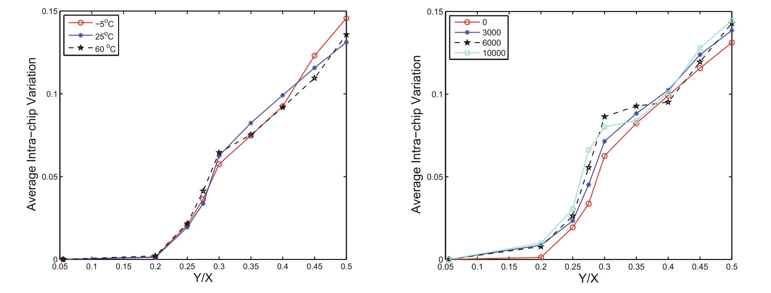
In the position-map method, the intra-chip variation decreases from 2.3×10^{-5} to less than 10^{-6} as the ratio of Y/X decreases from 0.5 to 0.474. In this case, when the ratio of Y/X is 0.474, now 270 NAND Flash memory cells are needed to generate a 128-bit robust key, and the length of helper data is $128\times16\times2$=4Kbits.

In conclusion, to generate a 128-bit key with the bit error rate $< 10^{-6}$, we can select 2340 or 270 NAND Flash memory cells for the bit-map method and the position-map method, respectively. Then we just choose the top 128 cell pairs with the maximal difference of the raw NFPUF output numbers to generate the key. Comparatively, the bit-map method needs more NAND Flash memory cells and less helper data, while the position-map method requires much less cells and more helper data. Therefore, we can select the appropriate method according to the specific implementation requirement.

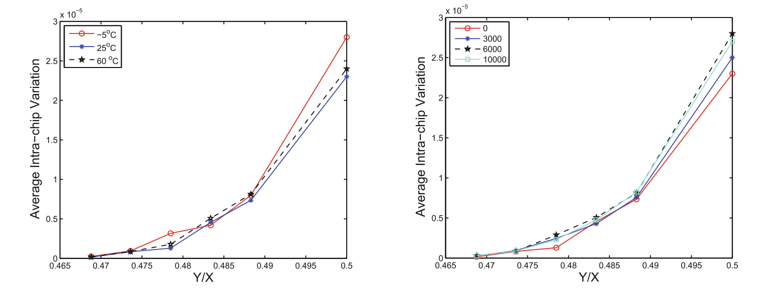
ECC is too complex and expensive to implement for efficient PUF-based key generation [5,15,24]. To generate a 128-bit key with a targeted key error rate $< 10^{-6}$, ECC implementations typically require 3 K-10 K PUF raw response bits (with bit error rate of 15 %) to generate the key, and the helper data generated for this case will be typically 3 K-15 K bits [1]. What is more, the error correcting capability of a specific ECC technology is fixed, if the number of error bits are beyond its fixed ability, the ECC would be useless. Therefore, our key generator is much more flexible, and it can achieve a 128-bit key with error rate $< 10^{-6}$ by using much less overheads compared with ECC.

Uniqueness. Uniqueness is a measure of how uncorrelated the PUFs response numbers are across different chips [1]. We evaluate the inter-chip variation, namely the number of bits which are different between two keys extracted from different PUF numbers. If the PUF produces uniformly distributed and independent random bits, the Hamming distance (HD) of a k-bit response from ideally unique chips should follow a binomial distribution with parameters $N = k$ and $p = 0.5$, and the mean of the HD distribution should be equal to $k/2$, namely the inter-chip variation should be 50 % on average [1].

The inter-chip variations of the three proposed extraction methods to generate 128-bit keys are shown in Fig. 6. The x-axis represents the different bit



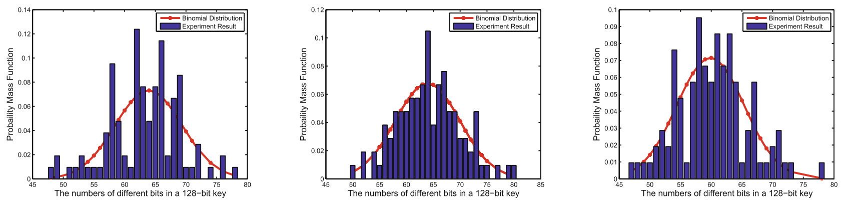
(a) Bit-map: Temperature variations influence. (b) Bit-map: Aging variations influence.



(c) Position-map: Temperature variations influence. (d) Position-map: Aging variations influence.

Fig. 5. The intra-chip variations with environmental changes.

number of the 128-bit key, and the y-axis represents the according probability. Here, the bars (blue) show the experimental results from 10000 pair-wise comparisons, and the lines (read) show a binomial distribution. As shown in the Fig. 6, the average different bits out of 128 bits are 63.91, 63.94, and 59.98, respectively. The average inter-chip variations of the three methods are 49.93 %, 49.95 % and 46.86 % respectively. The results are all pretty close to the ideal average of 50 %.



(a) Partial Programming. (b) Partial Erasure. (c) Program Disturbance.

Fig. 6. The inter-chip variations of the three proposed extraction methods.

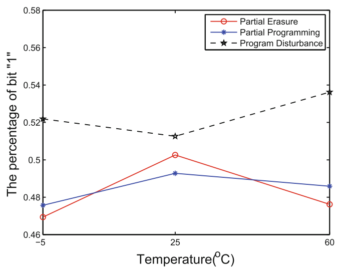


Fig. 7. The percentage of bit "1" with temperature changes.

Randomness. To ensure that the generation of the keys does not favor bits with a certain polarity, we compute the percentage of bit "1" in 10000 groups of 128-bit keys under three temperatures. As Fig. 7 shows, we find that the percentage of bit "1" ranges from 46.94 % to 53.62 %, and it is quite close to ideal 50 % with temperature variations. What's more, we can just leverage Von Neumann skew-correction algorithm to generate uniformly random bits, and use a hash function [11] to ensure the high entropy requirement.

6 Conclusion

In this work, we showed that common NAND Flash memory chips could be used to generate robust keys based on NFPUF. First, we proposed three methods to extract raw NFPUF output numbers from NAND Flash memory chips. Second, we utilized the bit-map or position-map method to select the NAND Flash memory cells with the most reliable relationship of the size between raw NFPUF output numbers. Only the selected cells are used for key generation. At last, we evaluated the primary characteristics of the generated key in various temperature and aging conditions. To our knowledge, this is the first time that a key generator based NFPUF implementation has been evaluated. Our key generator could generate a 128-bit key with a bit error rate $< 10^{-6}$ in 93.83 ms. The bit error rate ensures our key generator is reliable during the whole lifetime of the NAND Flash memory chips. Such low bit error rate is conventionally only achievable using powerful, but costly, error correction codes (ECC). Our key generator eschews the costly ECC overheads to generate robust and error-free keys. This study extends the functionality of NAND Flash memory chips, while requires no hardware change. Due to the widespread use of NAND Flash memory chips, the proposed robust key generator is potential to be widely applied to any electronic encryption devices, as long as the device leverages NAND Flash memory chip as nonvolatile storage.

References

1. Bhargava, M., Mai, K.: An efficient reliable PUF-based cryptographic key generator in 65 nm CMOS. In: Proceedings of the Conference on Design, p. 70. European Design and Automation Association, Automation and Test in Europe (2014)
2. Bösch, C., Guajardo, J., Sadeghi, A.-R., Shokrollahi, J., Tuyls, P.: Efficient helper data key extractor on FPGAs. In: Oswald, E., Rohatgi, P. (eds.) CHES 2008. LNCS, vol. 5154, pp. 181–197. Springer, Heidelberg (2008)
3. Breeuwsma, M., De Jongh, M., Klaver, C., Van Der Knijff, R., Roeloffs, M.: Forensic data recovery from flash memory. Small Scale Digital Device Forensics J. **1**(1), 1–17 (2007)
4. Delvaux, J., Verbauwhede, I.: Attacking PUF-based pattern matching key generators via Helper data manipulation. In: Benaloh, J. (ed.) CT-RSA 2014. LNCS, vol. 8366, pp. 106–131. Springer, Heidelberg (2014)
5. Devadas, S., Yu, M.: Secure and robust error correction for physical unclonable functions. IEEE Des. Test Comput. **27**(1), 48–65 (2010)
6. Dodis, Y., Ostrovsky, R., Reyzin, L., Smith, A.: Fuzzy extractors: how to generate strong keys from biometrics and other noisy data. SIAM J. Comput. **38**(1), 97–139 (2008)
7. Gassend, B., Clarke, D., Van Dijk, M., Devadas, S.: Silicon physical random functions. In: Proceedings of the 9th ACM Conference on Computer and Communications Security, pp. 148–160. ACM (2002)
8. Gassend, B.L.: Physical random functions. Ph.D. thesis, Massachusetts Institute of Technology (2003)
9. Handschuh, H., Trichina, E.: Securing flash technology. In: Fault Diagnosis and Tolerance in Cryptography, FDTC 2007, pp. 3–20. IEEE (2007)
10. Guajardo, J., Kumar, S.S., Schrijen, G.-J., Tuyls, P.: FPGA intrinsic PUFs and their use for IP protection. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 63–80. Springer, Heidelberg (2007)
11. Krawczyk, H.: LFSR-based hashing and authentication. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 129–139. Springer, Heidelberg (1994)
12. Lee, J., Heo, J., Cho, Y., Hong, J., Shin, S.Y.: Secure deletion for nand flash file system. In: Proceedings of the 2008 ACM Symposium on Applied Computing, pp. 1710–1714. ACM (2008)
13. Lim, D., Lee, J.W., Gassend, B., Suh, G.E., Van Dijk, M., Devadas, S.: Extracting secret keys from integrated circuits. In: IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 13, no. 10, pp. 1200–1205 (2005)
14. Linnartz, J.P., Tuyls, P.: New shielding functions to enhance privacy and prevent misuse of biometric templates. In: Kittler, J., Nixon, M.S. (eds.) AVBPA 2003. LNCS, vol. 2688, pp. 393–402. Springer, Heidelberg (2003)
15. Maes, R., Van Herrewege, A., Verbauwhede, I.: PUFKY: a fully functional PUF-based cryptographic key generator. In: Prouff, E., Schaumont, P. (eds.) CHES 2012. LNCS, vol. 7428, pp. 302–319. Springer, Heidelberg (2012)
16. Paral, Z., Devadas, S.: Reliable and efficient PUF-based key generation using pattern matching. In: IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), pp. 128–133. IEEE (2011)
17. Prabhu, P., Akel, A., Grupp, L.M., Yu, W.-K.S., Suh, G.E., Kan, E., Swanson, S.: Extracting device fingerprints from flash memory by exploiting physical variations. In: McCune, J.M., Balacheff, B., Perrig, A., Sadeghi, A.-R., Sasse, A., Beres, Y. (eds.) Trust 2011. LNCS, vol. 6740, pp. 188–201. Springer, Heidelberg (2011)

18. Ravikanth, P.S.: Physical one-way functions. Ph.D. thesis, Massachusetts Institute of Technology (2001)
19. Selmi, L., Fiegna, C.: Physical aspects of cell operation and reliability. In: Flash Memories, pp. 153–239. Springer, USA (1999)
20. Škorić, B., Tuyls, P., Ophey, W.: Robust key extraction from physical uncloneable functions. In: Ioannidis, J., Keromytis, A.D., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 407–422. Springer, Heidelberg (2005)
21. Skorobogatov, S.: Flash memory 'Bumping' attacks. In: Mangard, S., Standaert, F.-X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 158–172. Springer, Heidelberg (2010)
22. Subha, S.: An algorithm for secure deletion in flash memories. In: 2nd IEEE International Conference on Computer Science and Information Technology, ICCSIT 2009, pp. 260–262. IEEE (2009)
23. Suh, G.E., Devadas, S.: Physical unclonable functions for device authentication and secret key generation. In: Proceedings of the 44th Annual Design Automation Conference, pp. 9–14. ACM (2007)
24. Suh, G.E., O'Donnell, C.W., Devadas, S.: Aegis: a single-chip secure processor. Inf. Secur. Tech. Rep. **10**(2), 63–73 (2005)
25. Wang, A., Li, Z., Yang, X., Yu, Y.: New attacks and security model of the secure flash disk. Math. Comput. Model. **57**(11), 2605–2612 (2013)
26. Wang, C., Wong, W.F.: Extending the lifetime of nand flash memory by salvaging bad blocks. In: Proceedings of the Conference on Design, Automation and Test in Europe, pp. 260–263. EDA Consortium (2012)
27. Wang, Y., Yu, W.k., Wu, S., Malysa, G., Suh, G.E., Kan, E.C.: Flash memory for ubiquitous hardware security functions: true random number generation and device fingerprints. In: IEEE Symposium on Security and Privacy (SP), pp. 33–47. IEEE (2012)
28. Xu, S.Q., Yu, W.k., Suh, G.E., Kan, E.C.: Understanding sources of variations in flash memory for physical unclonable functions. In: IEEE 6th International Memory Workshop (IMW), pp. 1–4. IEEE (2014)
29. Yu, M.-D.M., M'Raihi, D., Sowell, R., Devadas, S.: Lightweight and secure PUF key storage using limits of machine learning. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 358–373. Springer, Heidelberg (2011)
30. Zambelli, C., Chimenton, A., Olivo, P.: Reliability issues of nand flash memories. In: Inside NAND Flash Memories, pp. 89–113. Springer, Netherlands (2010)