cryptography

MDPI

Review

Flash-Based Security Primitives: Evolution, Challenges and Future Directions

Holden Gordon [1], Jack Edmonds [1], Soroor Ghandali [1], Wei Yan [2], Nima Karimian [3] and Fatemeh Tehranipoor [1,*]

[1] Electrical and Computer Engineering, Santa Clara University, Santa Clara, CA 95053, USA; hgordon@scu.edu (H.G.); jsedmonds@scu.edu (J.E.); sghandali@scu.edu (S.G.)
[2] Electrical and Computer Engineering, Clarkson University, Potsdam, NY 13699, USA; wyan@clarkson.edu
[3] Computer Engineering, San Jose State University, San Jose, CA 95192, USA; nima.karimian@sjsu.edu
* Correspondence: ftehranipoor@scu.edu

Abstract: Over the last two decades, hardware security has gained increasing attention in academia and industry. Flash memory has been given a spotlight in recent years, with the question of whether or not it can prove useful in a security role. Because of inherent process variation in the characteristics of flash memory modules, they can provide a unique fingerprint for a device and have thus been proposed as locations for hardware security primitives. These primitives include physical unclonable functions (PUFs), true random number generators (TRNGs), and integrated circuit (IC) counterfeit detection. In this paper, we evaluate the efficacy of flash memory-based security primitives and categorize them based on the process variations they exploit, as well as other features. We also compare and evaluate flash-based security primitives in order to identify drawbacks and essential design considerations. Finally, we describe new directions, challenges of research, and possible security vulnerabilities for flash-based security primitives that we believe would benefit from further exploration.

Keywords: flash memory; flash-based physical unclonable function; physical unclonable function (PUF); true random number generator (TRNG); integrated circuit counterfeit detection; hardware security primitives; survey

Citation: Gordon, H.; Edmonds, J.; Ghandali, S.; Yan, W.; Karimian, N.; Tehranipoor, F. Flash-Based Security Primitives: Evolution, Challenges and Future Directions. *Cryptography* **2021**, *5*, 7. https://doi.org/10.3390/cryptography5010007

Received: 7 December 2020
Accepted: 27 January 2021
Published: 4 February 2021

1. Introduction and Background

Attacks on cyberinfrastructure and electronic devices become more and more advanced each year, costing companies and countries around the globe millions of dollars in time and resources. Consequently, enhanced methods of information protection are all the more important. In a world that relies as heavily on technology to function as we do on a daily basis, individuals must also trust that it is safe to use and that our information will be protected. Whether it be our cars, laptops, phones, or smart thermostats, we often use devices to assist us in our daily efforts, without giving much thought to how the information collected could be used against us if it falls into the wrong hands. Constituting anything from a computer virus to identity theft, attacks via weaknesses in device security are more prevalent than ever today. This places considerable responsibility on the designers of such devices to ensure that private information stays private, and this is where the constantly evolving field of hardware security comes into play.

While technology continues to grow in its omnipotence, the hardware devices themselves continue to shrink in size. This has led to an increased interest in hardware-based security primitives, such as PUFs and TRNGs, because hardware implementations are less exposed to attackers than software ones. Intrinsic implementations, which do not require extra hardware components, have been proposed as a lightweight and cost-efficient basis for security solutions. Both PUFs and TRNGs offer a promising solution in this regard, as they can provide authentication and validation that do not require heavy cryptographic measures and implementations [1–4]. To the best of our knowledge, we perform the first

systematic classification, analysis, and assessment of works regarding flash-based security primitives. We aim in this way to present a thorough and transparent overview of this field, provide clear insights, and collectivize the current and future trends. In this regard, we evaluate the efficacy of flash memory-based security primitives and categorize them based on the inherent process variation they exploit. We display where flash memory-based security primitives are advantageous, as well as what challenges and vulnerabilities they are faced with. Furthermore, we offer a discussion on new directions of research in this area that we hope will spark new ideas among readers and researchers that will expand the boundaries of flash-based security.

The motivation of memory-based security primitives is that they have several advantages over time-delay or other security primitive architectures due to their ubiquity, difficulty to model, simplicity, and reliability. Memory is an essential component in nearly every computing system—this guarantees that memory on-board a device can be used for security primitives rather than relying on periphery hardware. Furthermore, time-based PUFs, such as arbiter PUFs, are significantly more vulnerable to modeling-based attacks whereas memory-based PUFs are significantly more difficult to model. Next, memory primitives are often lightweight and simple since they leverage the on-board memory that has relatively simple architectures. Finally, research in this area has demonstrated the reliability of memory-based primitives even for stringent applications such as cryptographic key generation.

Flash memory has become a promising candidate as a memory-based primitive due to its popularity, density and architecture, and unique features (e.g., programmability and adaptability). Flash memory has arisen as an extremely common storage platform due to its very low cost. This has resulted in the widespread adoption of flash memory in many computing devices. Secondly, flash memory provides extremely high density and a variety of architectures. Therefore, flash can be used in a variety of different applications due to the different architectures and has an extremely high density of flash cells. Finally, flash memory is also programmable, allowing for different programming techniques to obtain the most efficient security primitive construction, and the programming commands can be adapted over time to provide advanced features such as aging resilience.

1.1. Our Contributions

Specifically, we make the following five major contributions in this paper:

1. We provide an overview of flash memory architecture and detail the various forms, including two dimensional (2D) NAND, three dimensional (3D) NAND, and NOR flash. We also discuss the different storage configurations used in flash memories.
2. We provide a detailed overview of various process variations exhibited through disturbs and other entropy sources.
3. We provide a comprehensive literature review relevant to flash-based security primitives, in order to allow for a clear and thorough view into the current state-of-the-art works.
4. We provide a thorough cost-benefit evaluation and comparison of flash-based security primitives to bring to light the advantages and limitations of each.
5. We elaborate on some open challenges and new directions of research in flash memory-based security primitives that can provide new opportunities for further research in this field.

1.2. Physical Unclonable Functions

A PUF is ideally unreproducible by manufacturers and attackers alike. PUFs take advantage of the unique physical characteristics of an integrated circuit (IC) that result from manufacturing variation on the micro and nano scales, giving each its own fingerprint. Armed with a unique set of challenge (input) and response (output) pairs (CRPs), a PUF relies on inherent entropy as well as stability/reliability to provide uniquely secure yet consistent operation [5]. Evaluating a PUF's effective uniqueness involves determining

whether or not it provides a different enough signature for its given IC to clearly differentiate it from other ICs of the same kind. Computing the Hamming distance (HD) between a pair of PUF identifiers (or the number of bit positions that differ in value) is one way this can be done. No IC should have the same signature as any other. Reliability refers to how well a PUF is able to provide a consistent response to the same challenge. This is important because a PUF is intended to be used again and again and should not be easily worn out or affected by environmental factors. Another important characteristic of a PUF is its generated ID's randomness. The randomness is necessary in order to prevent an attacker from reconstructing the ID bits because of a poor entropy rate within an ID.

There are many forms of existing PUFs that have been researched extensively, including arbiter PUFs [6], ring-oscillator PUFs [7], DRAM PUFs [8], and SRAM PUFs [9]. Flash-based PUFs are a relatively lesser-known type of PUF and could benefit from additional investigation. Flash-based PUFs rely on the concept of process variations, which will be discussed in detail in this paper. So-called "weak and strong PUFs" are the two subtypes of PUFs. Weak PUFs are used for storing secret keys to non-volatile memories [10]. They show some internal, unclonable physical disorder, and they are involved in some form of challenge-response mechanism which should be access-restricted. It is considered that even by having the PUF-carrying hardware, the adversaries cannot access the weak PUF's responses. Formerly, weak PUFs were toward special purpose circuits; now, they are based on intrinsic PUFs built from CMOS parts, which are more affordable. SRAM PUFs are the most popular type of weak PUF and a variety of other weak PUFs have been proposed, including Flash [11] and DRAM [2,12]. The most important advantage of weak PUFs is that it is harder for adversaries to obtain CRPs. There are two basic cases to derive secret key in Weak PUFs. The first one is "shared secret key" which is known to the manufacturer of the hardware and sometimes to a limited number of parties. One of the perfect uses of shared key is encryption of the design bitstream [10]. The second one is "unshared key" in which the key is unknown to any party outside the hardware and must be inside the system. Memory encryption is one simple application of such an internal, unshared key [10]. Strong PUFs have numerous CRPs which each time provide a new CRP for the procedure of authentication. They exhibit a more complex challenge-response. They are unpredictable, which means even if an adversary knows a large subset of CRPs, it is impossible for them to predict the other unknown CRPs. Strong PUFs have public access for challenge response mechanisms that allow everyone with physical possession of the PUF to apply challenges to the strong PUF which lead to some downsides, such as the need for many CRPs to remain secure [10].

1.3. True Random Number Generators

A TRNG is a device that generates random numbers from a physical process by utilizing random "noise" signals, such as random telegraph noise, a chip's power supply noise, etc. [13–16]. The aim of a TRNG is to generate random numbers that are utterly unpredictable (and often with lightweight hardware). These randomly generated bit streams can be used for cryptographic purposes, such as serving as a cryptographic key in an encryption or authentication process. Hardware security primitives built into device memories can provide cost-efficient and practical security solutions, especially for resource-constrained devices (such as in IoT applications). This is because memories are an intrinsic part of most contemporary computer systems, and using them as security hardware does not require the inclusion of additional hardware dedicated purely to device security [1].

1.4. IC Counterfeit Detection

As the supply chain grows, more diverse manufacturers and components are used throughout the various stages of the supply chain. However, there are few ways of verifying the correct production of these components [17]. Consequently, many counterfeit products were and continue to be injected in the supply chain, creating a huge threat [17]. These recycled, reused, or or cloned parts often have reduced life expectancies and can

pose serious reliability concerns. Furthermore, these products reduce the profitability of legitimately created Integrated Circuit (IC) products, resulting in an estimated 100 billion dollars lost in global revenue every year [18]. Moreover, 1% of semiconductor sales are estimated to be from counterfeit units [19]. Detecting counterfeit products in non-invasive and efficient ways has become imperative in creating stronger trust between parties in the supply chain and providing critical infrastructure assurances in the quality of sourced components. Example techniques of counterfeit detection focus on exploiting the aging biases in SoCs in order to detect counterfeit [20].

1.5. Memories

Computer memories come in both volatile and non-volatile forms. Upon losing power, volatile memory loses all stored data over a short period of time, while data stored in non-volatile memory remain. To date, many hardware security primitives have been implemented in various types of volatile memory, including dynamic random access memory (DRAM) [2,21] and static random access memory (SRAM), which are fast to program and very dense [22,23]. These allow for authentication, key generation, tamper detection, and other security schemes that rely purely on existing IC resources in devices instead of additional external hardware modules but they lose the data contents when the power supply is turned off. Flash memory is a rather ubiquitous form of non-volatile memory that can be electrically programmed and erased. It is typically faster and consumes less power than its primary competitor today, hard disc drives (HDDs), which also keep data content after losing power. Unlike an HDD, flash memory has no moving parts and this is termed a solid-state form of memory. Because of its high memory density and electrically programmable features, it has become widely adopted in lightweight embedded hardware, including IoT technologies, and due to the process variations that come about in the flash manufacturing process being user-programmable, flash memory-based PUFs and TRNGs are prime candidates for lightweight hardware/device security. The sources of flash memory process variations are discussed in detail in Section 5. These stable yet unpredictable characteristics allow for the implementation of reliable hardware security primitives unique to a given flash module [3,24–26].

1.6. Paper Organization

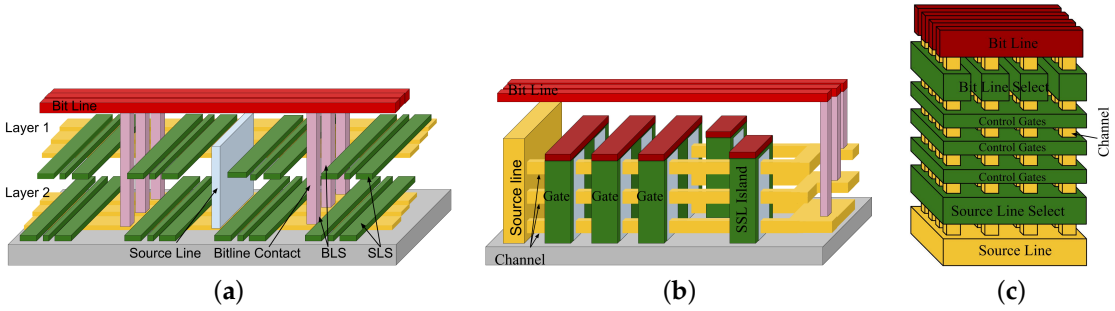
The rest of the paper is organized as follows. In Section 2, we provide an overview of flash memory architectures, their types, and flash memory storage configurations in Sections 2 and 3. In Section 4, we discuss sources of process variation in flash memory useful for security purposes, and detailed methods of how these variations can be induced. In Section 5, we provide a comprehensive literature review and categorize them into different groups. In Section 6, we elaborate on new avenues and challenges of research in flash memory-based security, and we conclude the paper in Section 7.

2. Flash Memory Architecture

As illustrated in Figure 1, flash memory generally consists of four main components: dies, planes, blocks, and pages. Each memory package contains at least one die, which is the smallest component that can independently execute commands. Within a die lies planes, where concurrent operations take place. Blocks are erasable units that are subdivided into pages, in which the individual memory cells are located. Blocks can also be further divided first into sectors, which will each contain a set of pages, as shown in Figure 1a. The flash I/O bus can be used for carrying data, addresses, and commands (as in this example), and many of the control signals allowing for basic operations are active-low. The chip enable signal ($\overline{CE}$) prepares a page for an operation, the read and write enable ($\overline{RE}$ and $\overline{WE}$) allow for reading and writing data from and to memory (latching commands, addresses, and data when the signals are pulsed), and the write-protect ($\overline{WP}$) must be high for write operations to occur. The command latch enable (CLE) latches commands into a command register from the I/O port, and the address latch enable (ALE) similarly latches addresses into the

address register. The ready/busy output pin ($\overline{R/B}$) indicates whether or not the device is ready for operation. Flash memory can be either asynchronous, where operations are driven purely by the control signals, or synchronous, where a free-running clock controls when operations occur [27,28].

Figure 1b illustrates an expanded visual of flash architecture features. In the upper half of the figure, the data buffer section holds data before they are either released to the data bus or sent to a sector from the data bus to be stored in memory. Tri-state buffers can be used to accomplish this task, which passes or hold data based on a signal sent to them from a control logic unit. The control signals, previously mentioned in describing Figure 1b, are fed into the logic unit, which asserts the corresponding signals needed to carry-out operations. The requested read/write address is stored in an address register until the logic unit enables the address decoder, which then points to a location in memory to be read from or written to [27,28].



(**a**) (**b**) (**c**)

Figure 1. (**a**) 3D NAND stacked architecture [29], (**b**) 3D NAND vertical gate (3dVG) architecture [30], and (**c**) 3D NAND BiCS architecture [29].

3. Types of Flash Memory

The two dominant forms of flash memory are NAND flash and NOR flash. Developed initially by Intel in 1988, NOR flash is typically used for code execution and storage, while NAND flash is often better suited for data storage and was first introduced by Toshiba in 1989. Three-dimensional NAND is a relatively new variant of NAND flash, which increases the traditional 2D NAND flash storage capacity by expanding its planar memory structure into a three-dimensional space. In general, writing and erasing can be accomplished more quickly with NAND flash, while reading is faster with NOR flash.

3.1. NAND Flash Memory

The NAND-type flash memory [31] was originally developed to target solid-state mass storage applications. Toward this end, NAND offers a small cell size, low power consumption, and fast page-based read/program operations. In addition, the small erase block size and fast erase times of NAND make it a very manageable memory.

3.1.1. 2D NAND Flash

Figure 2 (left) illustrates the basic structure of planar (two-dimensional) NAND flash memory, which gets its name from its visual appearance. It is organized in strings of cells connected in series and these strings resemble the pull-down network of a NAND gate. A combination of lines driven high or low allow for reading, writing, and erasing operations. Each string has a bit line select transistor on one end that connects to a bit line, and a source line select transistor on the other that connects to a source line. Cells store data in floating gate transistors through electron tunneling, and all cells connected to a given word line comprise a page within a block. All cells in a page are operated on together and typically allow for 512 bytes + 16 spare bytes, 2048 bytes + 64 spare bytes, or 4096 bytes + 128 spare bytes of storage in the page [32]. A memory block often contains either 32 or 64 pages. The configuration results in slow reading times relative to NOR flash (NOR flash will be described in Section 3.2), as entire pages must be read at once

to access sequential data. On the other hand, it does allow for faster writing and erasing cycles. The series configuration of cells in the NAND architecture also allows for higher storage capacity than a NOR architecture of similar size, which is an attractive quality for space-constrained electronics. The bulk read and write operations offered by NAND flash make it useful in a wide range of storage applications that involve sequential data archiving, such as audio and video storage. NAND flash is commonly used in storage devices such as SSDs (solid-state drives) for secondary storage in computers, USB flash drives, and SD (secure digital) cards for portable devices such as digital cameras. NAND flash memory is generally cheaper than NOR flash.

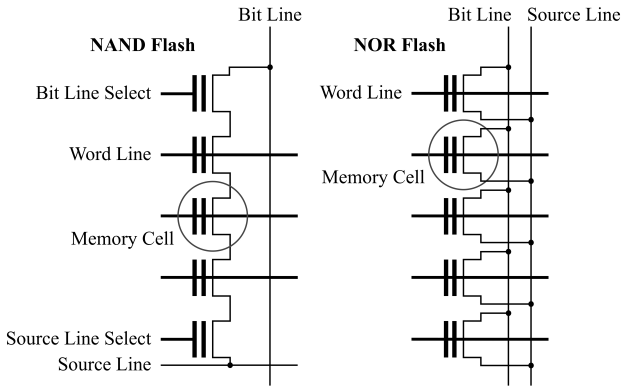


Figure 2. Internal structure of 2D NAND flash (**left**) and NOR flash (**right**).

3.1.2. 3D NAND Flash

Although most of the 2D NAND flash memories are based on floating gate (FG) technology, most of the three-dimensional (3D) NAND flash memories for the cell transistors are based on charge trapping (CT) technology. Both floating gates and charge traps accomplish the same goal of storing electrons and have similar structures, but charge traps generally have a simpler fabrication process that tailors itself well to the 3D architecture. The first 3D integration of a memory chip was presented in 2006 [33] and consisted of vertically stacking two identical planar NAND arrays with horizontal strings and word lines (WLs). The 3D architecture using etching processes has been known as a new advanced technology for CT-based and FG-based NAND design to increase the die density and to have better performances [34]. Figure 1a depicts a straight-on view of the basic structure of one possible 3D architecture, which consists of vertical arrays of NAND strings similar to what is shown in Figure 2 (left). With bit line above and source line below, the memory cells are sandwiched in-between and connected by channels that extend from top to bottom. Similar to planar NAND, bit and source line selects allow connection between a string's memory cells and the bit and source lines.

The technologies of 3D architectures can be divided into three main categories: (I) horizontal channel and gate, (II) horizontal channel and vertical gate, (III) vertical channel and horizontal gate. The first category is known as *3D stacked structure* [35], which is the preliminary attempt to achieve 3D integration starting from planar technology. Figure 1a shows a view of a 3D stacked NAND flash integrating two separate planes. The second architecture (3DVG) has been investigated and extensively studied in [36] and is shown in Figure 1b. In this architecture, the NAND cell string runs horizontally and is made of several wordlines plus the dummy wordlines and the select transistors. A silicide layer can be easily deposited on the top of the polysilicon gate to achieve a lower wordline resistance and a very fast access time despite the long wordlines.

The vertical channel solution has rapidly become the mainstream integration scheme for 3-D NAND flash arrays. For this architecture, the Bit Cost Scalable (BiCS) was presented for the first time by Toshiba [37,38]. The improved versions of BiCS are called Pipe-shaped Bit Cost Scalable (P-BiCS) [39], and the pathway to the V-NAND architecture presented by Samsung in [40,41]. The BiCS 3D NAND flash architecture is described in Figure 1c.

The first element of the architecture is the control gate (CG) stack shown by the different rectangle elements piled on top of each other, whereas the bottom rectangle plate is the ensemble of source line selectors (SLS) terminating the flash string. Multiple holes are drilled through the stacks and filled with polysilicon in order to form a series of vertically arranged NAND flash memory cells. Bitline selectors (BLS) and bitline (BL) contacts are on top of the structure [42]. Each cell in the BiCS architecture works in depletion mode [43] since the polysilicon constituting the body of the transistor is lightly n-doped with a uniform profile or even left undoped. This reduces the manufacturing complexity of the p-n junction along the vertical direction of the plugs (also called pillars). The CG plate intersection with a pillar maps a single memory cell. Each NAND flash string of cells is connected to a BL contact via BLS, whereas the bottom of the string is connected to a common source diffusion formed directly on the process substrate made of silicon.

3.2. NOR Flash Memory

Figure 2 (right) illustrates the basics of NOR flash memory as well, which gets its name from its resemblance to the pull-down network of a NOR gate. While the NOR structure does contain pages, it does not access data by entire pages at a time as in NAND flash. The individual memory cells are connected in parallel, and enough connections are made among bit, word, and source lines to make random access (or byte-level reading and writing) possible. Reading is faster in NOR flash than it is in NAND flash for this reason. As an example of operation, to read a value from a cell, the source lines and a specific word line are asserted. The addressed cell's value can then be determined from its corresponding bit line via a sense amplifier. The downside of the NOR structure is that the memory density is lower, writing and erasing takes longer, and its random access capability makes it more expensive than NAND flash. NOR flash is commonly used for code execution, networking device memory, and can serve as a stable replacement for EEPROM (electrically erasable programmable read-only memory) integrated into microcontrollers. It is also used in medical devices and various scientific instruments.

Flash Memory Storage Configurations: At the cell level, flash memory will usually have either a single-level cell (SLC), multi-level cell (MLC), triple-level cell (TLC), or quad-level cell (QLC) storage capability. These options allow for either one, two, three, or four bits to be stored per cell, respectively. As far as storage density goes, QLC wins the day and is also the cheapest of these four options. However, as more bits are packed into one cell, reliability and speed are sacrificed. SLC storage is the most reliable and fastest overall, as writing a single bit to a cell takes less time than writing multiple. Figure 3 displays the basic concept of these four storage options.

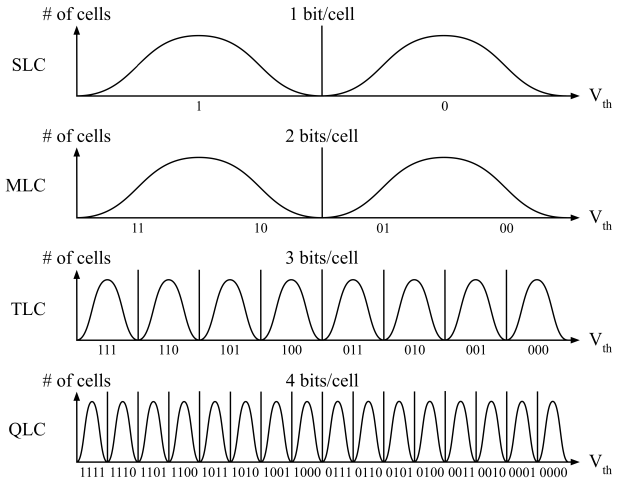


Figure 3. Flash memory cell capacities (SLC, MLC, TLC and QLC).

4. Sources of Process Variation in Flash Memories

Process variations are intrinsic, naturally occurring sources of entropy created in the fabrication process of ICs. Process variations are exploited for various hardware security primitives due to their uncontrollable but reproducible randomness [44]. They are also common in flash memories and mainly come from random dopant fluctuations (RDF). The RDF fluctuations can be induced by traditional flash programming commands that are standardized in the Open NAND Flash Interface (ONFI) [44]. This leads to more flexible flash memory-based hardware security primitive solutions that can be introduced through a variety of implementations such as Read Disturb, Program Disturb, Program/Erase Interrupt, Program/Erase Latency, and Random Telegraph Noise (RTN). The following subsections will discuss each of these in detail (concept and impact).

4.1. Read Disturb

Concept: *Read disturb errors are caused by repeated readings from flash memory pages.* The page that is repeatedly read will cause the threshold voltages to subtly increase for surrounding cells in a neighboring page. Figure 4a highlights the neighboring flash cells that are affected by read disturb. This threshold voltage drifting arises because NAND flash cells are organized in series. Therefore, each cell must be turned on to a high voltage to allow a selected cell to be read. This voltage can be high enough to unintentionally cause electrons to be tunneled across the oxide layer of cells in neighboring pages [24]. This induces bit flips from the logical "one" to the "zero" states that disturb the original contents in those cells. As flash density increases, this effect is intensified as the memory is scaled down to smaller sizes [44].

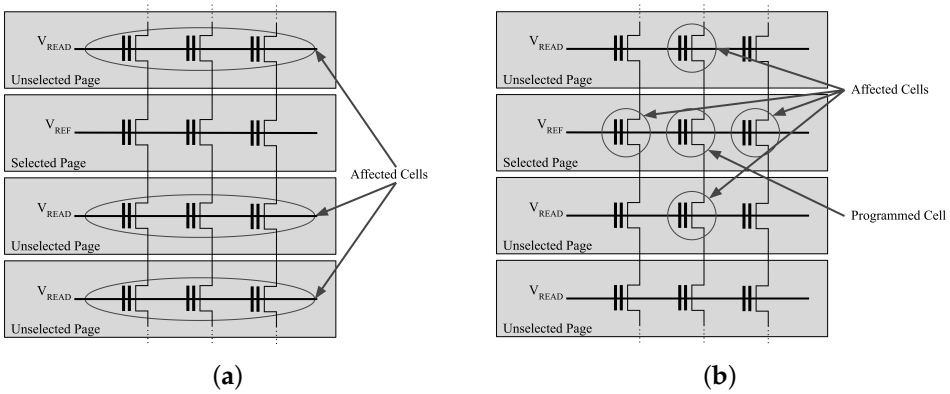


Figure 4. (**a**) Affected cells from Read Disturb and (**b**) affected cells by Program Disturb.

Impact: As a source of process variation, read disturb can be implemented through a standard interface common to all flash devices. However, read disturb can take many consecutive reads to have a pronounced effect on the logical cells of flash cells. In [11], 10 million reads were needed to create effective differentiation between signatures from different blocks in MLC chips, which took around six hours. This approach may not be suitable to generate hardware security primitives alone and may require other operations to put flash cells in more unstable states to visualize the effects of read disturb.

4.2. Program Disturb

Concept: *Program disturb is created by erasing a single block and then repeatedly reprogramming a page in the block to physically affect adjacent cell values.* Like read disturb, program disturb will affect cells in the nearby vicinity of the programmed cell. However, program disturb affects only physically adjacent cells due to parasitic capacitive coupling. Figure 4b shows the flash cells that are affected by a program operation to a single cell [45]. This coupling causes elevated voltage stress that alters the threshold voltage level of neighboring cells channeling electrons onto the floating gate of the flash cells.

Impact: As an exploitable process variation, program disturb results in unintended programming of nearby cells. Since the integrity of the oxide layers in nearby cells is also degraded over time (due to increased program/erase (P/E) cycle stress), this can increase the unintentional tunneling of electrons onto the floating gate. Therefore, program disturb has been used as a source of entropy for both TRNG and PUF applications.

4.3. Program/Erase Interrupt

Concept: *Program/Erase Interrupt is a method of aborting the "programming" or "erase operations" of a particular flash module. This can cause cells to be in intermediate states due to a partial program or partial erase.* Due to flash modules taking more than a single clock cycle to execute either a program or erase, a "RESET" command can be issued to interrupt the execution of the program/erase through the standard flash interface. This interrupts the programming to allow for flash cells to be partially programmed or erased and to put into intermediate states that can allow other, less noticeable process variations to flip the logical values of flash cells [46].

Impact: This process variation extraction method has been used for a wide variety of characterizations, ranging from quantifying radiation damage to unique signature generation. Program/erase interrupt is visualized in Figure 5. Here, the data line "DX" is writing data to a flash block. The *Cycle Type* line is used to visualize what commands are being used to write or program these data to the flash block. As the "10 h" program command is sent to the flash module, a "RESET" command can be sent while the programming is taking place. This programming can be interrupted, leaving flash cells in an intermediate and sensitive state [47]. To measure this interruption, the R/B register, also known as the ready/busy register, will go low as the operation is being performed. However, when the operation is interrupted, it will return to high as the command is interrupted and the device has completed the programming and is in the ready state. This can allow for a speed-up of signature and random number generation by putting flash cells into intermediate unsteady states that are extremely sensitive. It can also be used to provide tunability to specific hardware security primitives. For example, Wang et al. [45] used partial programming to tune their PUF construction to allow for variable generation and robustness.

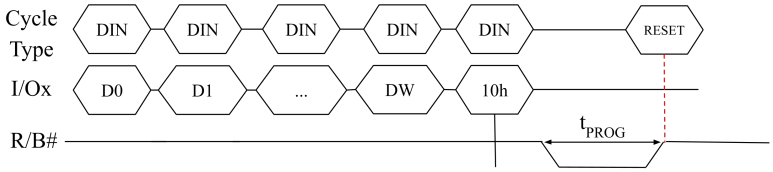


Figure 5. Program/Erase Interrupt Scheme [48].

4.4. Program/Erase Latency

Concept: *Program/Erase Latency measures the latency between particular programming schemes of various blocks and pages.* Latency is described as the required waiting time before another operation can be executed and this time varies during a flash block's lifetime. Figure 6 highlights the timing diagram for a programming/erase latency scheme. This is visualized by monitoring the ready/busy (R/B) register in a flash chip as shown. While programming is occurring, the R/B register will go low for a set period of time until the specified operation is complete. This time frame is called the "program/erase execution time". The program/erase execution time can be utilized to measure the latency values of a previous operation.

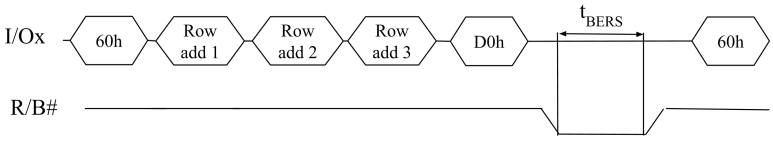


Figure 6. Program/Erase Latency Scheme [48].

Impact: Latency has been used as flash security feature in [11,49]. As devices age, their latency values will also change due to oxide degradation from increased P/E stress. This latency value can also vary for a variety of flash devices, making it difficult to predict [48,50]. However, latency values can also lack a wide range of values required for the successful digitization of the latency information [11]. This can be problematic for latency-based hardware security primitives.

4.5. Random Telegraph Noise (RTN)

Concept: *RTN is a type of noise found in semiconductors and very thin oxide gate films.* It is caused by electron capture/emission traps that have become more pronounced as flash memory has scaled down and bit per cell density is increased [51]. Figure 7 shows a general idea of RTN process variation. In this figure, the threshold voltage changes for a given trap, *emission and capture*. This trap occurs due to oxide defects in the floating gate structure. This trapped electron can either be captured from the drain current or released into the drain current. This added or removed electron causes an abrupt change in the drain current, which is demonstrated in Figure 7. Furthermore, this voltage swing from RTN is worsened by increased program and erase cycling wearing down the quality of the tunnel oxide, causing pronounced RTN effects during a flash cell's lifetime [52]. RTN was originally not a severe source of noise until recent advances have aggressively scaled down the thin oxide films. For instance, in recent NAND flash implementations, the threshold voltage swing has been as high as a full volt *(one volt)* at a single node [51].

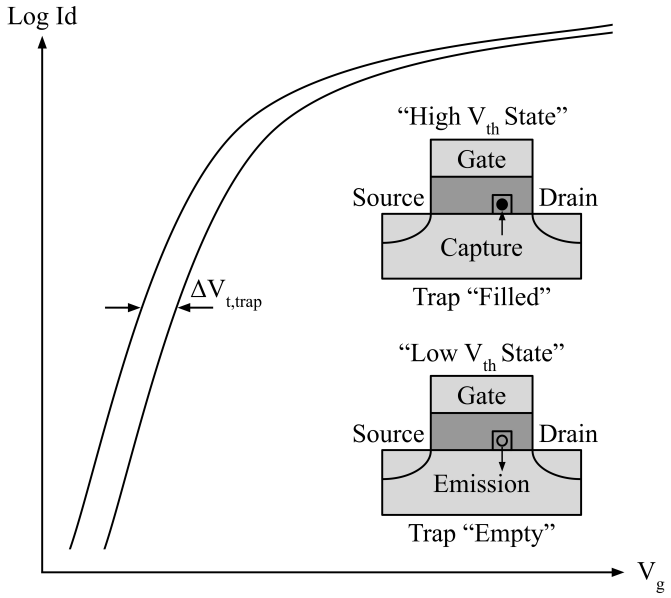


Figure 7. Random Telegraph Noise from trap, emission, and capture [53].

Impact: RTN has rarely been used for flash hardware security primitive applications due to its recent prominence. However, a key contribution of RTN as a source of entropy is that it is based on quantum mechanical probabilities, which is independent of thermal noise or other external properties. This makes it a very reliable entropy that can be considered for hardware security primitives within flash, especially due to the recent aggressive scaling down of flash memories.

Table 1 highlights which of these process variations are exploited as sources of entropy in both the flash PUF and TRNG literature. A detailed review of these constructions will be presented in the following section.

Table 1. History of the development of flash memory PUF and TRNG based on different process variations.

Year	Publication	PUF/TRNG	Program Disturb	Partial Progr-Amming	Erase Interrupt	Program Erase Latency	Read Disturb	RTN
2011	Prabhu et al. [11]	PUF	X			X	X	
2012	Wang et al. [53]	PUF/TRNG		X				X
2015	Kim et al. [54]	PUF	X				X	
2015	Jia et al. [46]	PUF	X	X				X
2017	Saito et al. [55]	PUF					X	
2018	Milenkovic et al. [3]	TRNG	X				X	
2018	Wu et al. [56]	PUF					X	
2019	Clark et al. [57]	PUF/TRNG			X			
2019	Poudel et al. [58]	TRNG		X			X	
2019	Mahmoodi et al. [59]	PUF		X				
2020	Sakib et al. [45]	PUF	X					
2020	Chakbatory et al. [49]	TRNG				X		
2020	Larimian et al. [60]	TRNG						X

5. Comprehensive Literature Review

This section is divided into three major subsections: flash memory-based PUFs, flash memory-based TRNGs, and IC counterfeit detection techniques for flash memory.

5.1. Flash Memory-Based PUFs

The overall challenges related to PUFs are achieving advanced characteristics such as high reliability, uniqueness, high entropy, abnormal environmental conditions resistance (aging, voltage variations, high-low temperature, etc.), and cost effective architectures or designs. Previous research has shown that memory-based PUFs are one potential PUF architecture that can meet almost all requirements for creating cutting edge, attack-resilient PUFs. In particular, flash memory-based PUFs have seen rapid development from 2012 to 2020, with two major chronological time periods of development denoted as the first and second phase. Flash memory PUFs have been created for a variety of flash architectures and have a variety of design parameters. As more flash PUF technology matures, more emphasis is placed on tuning advanced PUF characteristics such as aging resistance, throughput, and ease of implementation.

- **First Phase:** This phase comprises the first PUF constructions. These constructions may often be proofs of concept, be based on one single process variation, and do not deal with more advanced considerations for PUF constructions such as hackability, throughput, or temperature/aging resistance.
- **Second Phase:** This phase tries to combat these shortcomings and comprises constructions made within the last three years. These PUFs consider more dynamic factors in their constructions, such as resistance to machine learning, expanding the CRP space, and the aforementioned considerations: throughput, aging, etc.

5.1.1. First Phase of Development

One of the first definitive instances of a flash-based PUF occurred in 2011 by Prahbu et al. [11]. This work highlights seven separate mechanisms for PUF generation with NAND flash memories. Prabhu et al. [11] characterized read disturb, program disturb,

and then several different forms of latency such as "page-write latency", "programming latency", and "reading latency". These PUFs solely used standardized flash programming commands ubiquitous to all flash chips, required no peripheral hardware, and did not mix different programming variations. The results demonstrated that flash PUFs could be created from repeated read operations, repeated program operations, or by analyzing the latency of various operations. However, many of these PUFs were not able to generate PUF signatures in reasonable time frames, and the PUFs that were generated quickly started to lose accuracy and were vulnerable to hacking. For example, read disturb flash PUFs had the highest accuracy but took over 6 h to generate. Conversely, program latency only took one to three seconds to generate 100 measurement signatures, with greater than 10% reduction in hamming distance. Furthermore, the digitization of the latency values made the construction vulnerable to forging attacks. Although a great step in proving the viability of flash PUFs, these constructions had severe throughput limitations, making them hardly usable in a real-time system.

Another paper from this time frame was from Kim et al. [54], who created a flash PUF in 2015 by programming a flash page to the statistical median of its threshold voltage. This threshold voltage range was then divided into a high and low section, with the lower half decoded as a logical "1" and the upper half as a logical "0". After cells were programmed to this statistical median, the cells were read, and depending on the program/erase efficiency, cells were interpreted as logical "1" or "0". This paper produces a proof of concept for the possibility of more sophisticated PUF constructions. Since this construction only performs a single partial program, it is a faster PUF construction than that proposed in [11], which requires millions of reads. However, this method lacks a detailed explanation of its estimated throughput, and the method of approximating the statistical mean of the threshold voltage for the logical "0" state, since threshold voltages are proprietary as shown in [24].

The next paper from Wang et al. [53] proposes a physical implementation of a technique that was successful in generating unique PUF signatures with intra-page Pearson coefficients, with an average Pearson coefficient that rivaled those presented in Prahbu et al. [11] and Kim et al. [54]. Most importantly, the throughput for this device was around 20 kb/s. This shows a substantial improvement from the generation delays within the PUFs presented by [11] that took multiple hours to generate signatures with comparable robustness and uniqueness. The security of this construction was also investigated. In regard to hackability, this construction is vulnerable to replay attacks. However, the attacker would have to store the fingerprints from every page in order to ensure a complete modeling attack which improves upon the forging attacks represented in Prahbu et al [11]. Since the bits are stored as a 10-bit number, this would require $10\times$ the chip storage. Enhanced modeling attacks such as machine learning or power analysis were not considered.

After Wang et al. [53], Jia et al. [46] used a technique in a very similar manner to the proposed method in [53]. They added two post-processing techniques, "Bit Mapping" and "Position Mapping", to extremely enhance bit reliability with error rates less than 10^{-6}, allowing for processed outputs to be used as cryptographic keys with high fidelity. This enhanced the work from Wang et al. [53] and allowed for more reliable responses on PUFs extracted from program disturb or partial programming. Similar to the previous papers, Jia et al. [46] do not deeply consider the hackability of their constructions and do not consider voltage, temperature, or modeling attacks.

This first phase resulted in some of the first flash PUFs and their proofs of concept; however, it is characterized by implementations that have practical implementation drawbacks. This is because the PUFs solely relied on a single process variation in [11,46,53,54] or they lacked detailed analysis of the hackability of their constructions [46,54]. This is primarily due to the novelty of flash PUFs. Later in the literature, phase two commences, where many more considerations are taken into account. This upcoming phase differs from the first one as it has PUF constructs that use multiple programming variations for process variation extraction, consider energy consumption, modeling attacks, temperature

resilience, and build architectures around specific flash memory types as phase one solely deals with NAND flash memory.

5.1.2. Second Phase of Development

Wu et al. [56] provides the first construction within this time period. This construction provides a programming burst to the gate voltage, stressing the flash cells. This burst induces current leakage into a neighboring cell. This is then connected to a sense amplifier and extra logic to process a PUF response bit. This construction has phenomenal uniqueness, with an inter-hamming distance of 0.499999 (the ideal is 0.5). The randomness was also excellent, with a hamming weight of approximately 50%. The intra-ID hamming distance for the responses was also practically negligible, highlighting the phenomenal reliability of the construction. Differing from the first phase PUFs, this PUF construction is resistant to temperature stress, voltage stress, and can withstand extreme operating temperatures ranging from 40 to 150 degrees Celsius. This improves the hackability of this construction significantly, and it also allows for this PUF to be able to extend to automotive applications that have extremely high temperatures.

Another paper from phase two that Saito et al. [55] published in 2017 provides a flash PUF construction that utilizes 28-nm SG-MONOS embedded flash. This construction is based on differential comparison between two flash cells, a posi-cell and a negi-cell. The cells are compared without a reference voltage to avoid having to apply a threshold voltage to the gate of the cell which prevents read disturb by allowing a read without applying voltage to the gate. Figure 8 describes the SG-MONOS structure. Built around this flash, Saito et al. [55] created a PUF construction with very strong uniqueness and randomness. Finally, the reliability was measured using an intra-PUF hamming distance which yielded around 8.3%, which could be corrected by a novel correction scheme. This PUF also improves to near zero reliability error and makes the construction resistant to temperature changes. This works by increasing resistance to temperature attacks and making it applicable for automotive applications since this PUF can withstand temperatures ranging from 40 to 170 degrees Celsius. This is done by using an offset read scheme that attaches to the SG-MONOS flash cells to allow for effective "masking" of unstable bits. A drawback to this implementation is that, on average, 47% bits were deemed unstable. Losing this many bits may not be the most efficient use of space on the SG-MONOS flash chips since complicated ECC may impose smaller space overhead and ensure reliability. Similar to Saito et al. [55], Clark et al. [57] proposed a very similar method that used high performance $1.5 - T$ flash cells in 2019. These flash modules condense programming execution to a single clock cycle, making programming interrupt difficult [57]. Therefore, this PUF construction utilizes erase interrupt, which follows very similar design principles as programming interrupt.

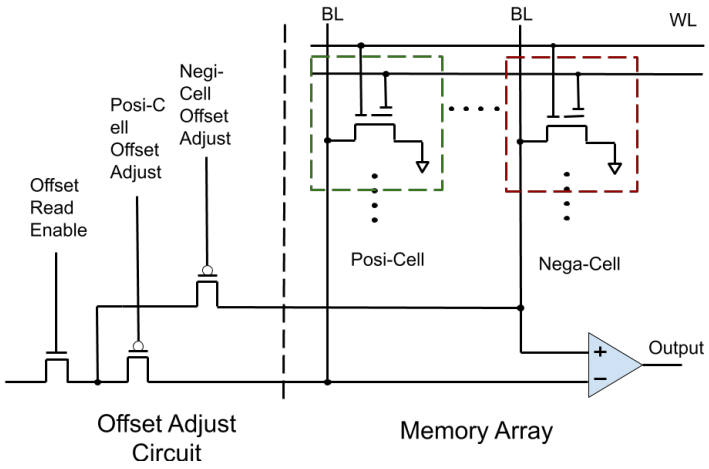


Figure 8. Structure of Posi-Cell and Negi-Cell from Saito et al. [55].

In this PUF construction, bits were erased to "1" in multiple cycles that were interrupted. The interrupted erases were repeated until the distribution of cells was half 1s and half 0s. Clark et al. [57] proposed a helper function known as MON1 that measures the numbers of "1"s and "0"s and if a response has a skewed number of "1"s—say, 55 percent—the authors generate another response with 45 percent to balance out the number of "1"s and "0"s, creating sufficient randomness. This is done by varying the erase interrupt timing to adjust and tune the percentage of "1"s and "0"s. Furthermore, this work also provides temperature and voltage tamper resistance. The next construction in this period is ChipSecure, which is a novel reconfigurable eFlash created in 2019 by Mahmoodi et al. [59]. This construction provides a novel implementation that provides enhanced post-processing techniques to expand the CRP space to 10^{211} to provide resilience to machine learning attacks and an extremely energy efficient architecture able to operate at 0.5 pJ/b. This paper is a clear representative example of the second time period of flash PUFs. This construction was also highly reliable, with a bit error rate lower than 5%. This was achieved by measuring gate leakage currents and induced current leakage from gate voltage variation. With this current leakage used as a process variation, the response bits are multiplexed and then fed into primitive blocks and then into a hidden shift register (HSR). By using the HSR and multiplexing various response bits, the CRP space becomes significantly larger. Furthermore, the time-multiplexing also allows for an equivalent throughput of 192.3 Mbps. Mahmoodi et al. [59] then used a multi-layer perceptron network to attempt to attack their construction. More details of the network can be found in [59]. The success rate of predicting a particular output bit is around 50%, which is close to the ideal of a random 50% accuracy for predicting a "1" or "0". This construction tackles hackability via machine learning and a massive CRP space, presents a powerful post-processing technique, and has very close to ideal uniqueness, reliability, and randomness metrics. Another paper, Larimian et al. [60], will be mentioned in more detail in the next TRNG section. However, this paper takes the construction from the Mahmoodi et al. [59] paper and builds an authentication scheme around it. It does not focus on an additional novel PUF application and so will be left out until the TRNG section.

Another second generational PUF scheme for flash memory was released in 2020 by Sakib et al. [45], where researchers only utilized a program disturb method. This construction does not apply extra periphery hardware or multiple new process variation extraction techniques. However, the traditional program disturb behavior was sufficiently characterized for individual chips to allow for a tunable PUF construction that was aging-resistant. This scheme is easy to implement, requires no peripheral hardware, is resilient to aging, and is extremely reliable. This PUF generator took approximately 2.2 s to generate PUF responses from a single page. Furthermore, PUF responses of length 128 bits had an error of less than 10^{-6}, and responses of 20,000 bits had an extremely low error rate of 0.2%.

Furthermore, this construction has 3% to 8% intra-chip hamming distance distribution where the ideal is 0%. This highlights the phenomenal reliability of PUF construction. This construction has randomness within the range of 42% to 51% of "1"s within the construction, close to the ideal value of 50%; however, it is more skewed towards "1"s than "0"s. Finally, the PUF demonstrated phenomenal uniqueness, with results between 49% and 51% inter-chip hamming distance distribution. This solution was enabled by creating a model that describes the effects of aging on the bit error rate with program disturb-based PUF techniques. Furthermore, Sakib et al. [45] created a "Golden PUF" that allows the response bits that are noisy to be filtered out in a manner similar to Saito et al. [55]. Although this solution does not include many of the post-processing techniques mentioned in the previous paper from Mahoodie et al. [59], it provides a lightweight, aging-resistant, and extremely reliable construction that requires no peripheral hardware, making it an easy-to-implement, lightweight solution. This is due to the deeper characterization of program disturb to generate new signatures.

This second phase deeply focuses on practical PUF implementations and results in PUF constructions that are much more practical for real-time hardware security measures.

For example, Mahmoodi et al. [59] consider new ways to drastically increase the CRP space while also being one of the only works to perform modeling attacks on their PUF construction with an advanced deep learning attack. Sakib et al. [45] provides a deeper characterization of program disturb's effects over a device's age, providing an aging-resistant PUF, which is an extremely important consideration for any security primitive designed within flash memory. This is still a very active area of research that has tremendous room for growth, and new research directions will be discussed in greater detail in Section 6.

5.2. Flash Memory-Based TRNG

The overall challenges for TRNG models are (i) designing cost-effective and lightweight techniques that require minimal peripheral hardware as well as (ii) providing high entropy and throughput with a large volume of random bit strings. There have been various research and studies in this regard and memory-based models are becoming popular and potential sources of entropy. In particular, the principle focus in advancing flash TRNG constructions is maximizing throughput while using as minimal extra post-processing as possible. Furthermore, many flash architectures need different TRNG models; therefore, various flash memory designs require novel and unique entropy extraction techniques. This section is broken up into sections as the literature is more sparse and has fewer publications than flash PUFs. However, the development of flash TRNGs is still clearly seen throughout the progression of the literature.

Some of the first work with flash TRNGs came from Wang et al. [53] in 2012. This first TRNG construction exploits thermal and RTN noise from NAND flash memory to create unique random numbers. This technique initially utilizes partial programming to characterize whether cells are experiencing solely RTN or RTN and thermal noise combined. With this characterization, the cells are programmed a sufficient amount to become sensitive to slight perturbations from RTN or thermal noise. This characterization provides two important timing characteristics for TRNG behavior. The first is up-time, which is a sequence of time in the erased state, and the second is down-time, which is a sequence of time in the programmed state. To produce random numbers, these two time sequences are used to make a binary number sequence. Next, Von Neumann debiasing is used as a post-processing technique. However, the duration of the partial program is dependent on whether the flash cell is experiencing solely RTN noise or both RTN and thermal noise. Combining the up-down, down-time, and the characterization of the source of the noise, the TRNG is able to generate a random number. The TRNG had average performance throughput anywhere between 848 bits per second to 3.37 Kbits/s. This variation is due to the intense pre-processing requirements that are required to identify the type of noise present in the flash cells. However, the TRNG passed all of the NIST test requirements for a TRNG without having to do any post-processing. Furthermore, temperature and aging only increase the effects of the RTN or thermal noise. This increase in randomness does not hinder the construction, which is different from the tight uniqueness, randomness, and reliability requirements for PUFs. This construction is a phenomenal stepping stone for other constructions that will exploit multiple process variations, enhance throughput, provide specialized implementations, or mix multiple process variation extraction techniques.

In a 2018 paper by Milenkovic et al. [3], the authors create a TRNG based on program disturb and read noise, which combines program and reading disturbs. A block is first erased and then a page is programmed in a checkerboard manner, alternating between "0"s and "1"s. This pattern is repeatedly reprogrammed to induce program disturbs in neighboring cells within the page. Then, the cells are read a repeated number of times, classifying cells that are unstable and oscillate between state "1" and "0". These cells had their threshold voltages moved so close to the logical "0" state that subtle noises such as RTN and thermal noise caused these threshold voltage fluctuations. This TRNG also passed all NIST randomness tests. Furthermore, this TRNG construction was functional through temperatures ranging from 85 to -10 °C, and there was an increase in TRNG performance as the device was aged through additive P/E cycling. Regarding throughput,

this construction reported an estimated throughput of 1 Mb/s. This exceeded previous work from Wang et al. [53]. This combination of using program disturb and read disturb enhances the work from Wang et al. [53], making the construction more implementable. In 2019, Clark et al. [57] extended a partial programming technique to create a TRNG based on interrupted erasure on a novel flash architecture. Since high-performance 1.5-T flash chips are able to execute programming in a single clock cycle, it is not possible to interrupt these commands from the digital interface. Therefore, an interrupted erasure, which takes more than one clock cycle, was characterized and interrupted to generate random numbers. By slightly reducing the erase voltage, random numbers were able to be created that passed the NIST test for randomness. However, several tests did initially fail and helper data had to be used to refine the generated random numbers to ensure true statistical randomness. After all of these changes, the minimum entropy was 0.90 to 0.91, making the construction a strong TRNG. Even though a throughput was not given, the TRNG construction was successful under a variety of aging, temperature, and voltage tests.

This highlights the maturation of the literature covering flash TRNGs, as more sophisticated considerations are taken into account for novel architectures. In 2019, Poudel et al. [58] also used a new technique to perturb split-gate flash memory cells. This is a flash cell that is commonly used within embedded microcontrollers. Since NOR flash is byte-addressable, these flash cells are commonly used for code execution. By using onboard NOR flash, Poudel et al. [58] create a TRNG engine. This construction programs the NOR flash as close as possible to its threshold voltage and then applies multiple reads to the cells, allowing read noise which is a combination of thermal and RTN noise to perturb the cells randomly. This interrupt is done by issuing several timed NOP commands from the microcontroller and then issuing an emergency exit command. Poudel et al. [58] also incur an extra software delay and specifically time the emergency exit command—this is optimized to provide the highest throughput for the TRNG. Furthermore, this delay is also based on the processor speed on the microprocessor itself. This is also calibrated to provide more tuning to programming interrupt, and the results from Poudel et al. [58] show that the finer clocking resolution does lead to more random numbers being generated, leading to a higher TRNG throughput. The construction passed all NIST tests for randomness after Von Neumann debiasing, and the TRNG engine achieved a peak of 68,200 bits per second. Furthermore, this construction also endured aging and temperature variation. Although lower temperatures reduced the number of "good" bits, the RTN is a quantum noise that continues to create sufficiently random bits for the TRNG engine. This work not only provides a TRNG engine in an IC, but it implements the IC in a microcontroller. By using the onboard microcontroller memory, this construction moves the literature closer to providing fully realizable TRNG engines embedded in flash on commodity microcontrollers.

One of the most recent papers discussing flash TRNGs was released by Chakbatory et al. [49]. This paper further seeks to analyze the process variations seen through write and erase latency operations in NOR flash and offers a TRNG engine that is implementable in off-the-shelf NOR flash memory. This greatly expands the ease of integration for designers, as the construction does not require peripheral hardware and overly complex pre/post-processing. Their approach even generalizes this analysis to other forms of non-volatile memory, namely resistive non-volatile memory, to extend the discussion and knowledge, exploiting process variations seen in writing and erasing latency. The latency values are extracted by polling the write-in-progress bit in non-volatile memories. This bit is "0" for an uncompleted write/erase operation and is "1" for a completed operation. Then, the random numbers are generated by measuring the write latency of various data patterns. This paper also used an XOR-based post-processing technique that reverses and returns the bit-wise XOR of the raw latency values. All the NIST tests were passed and the NOR flash had a throughput of 0.05 kb/s. However, this throughput may have been limited by the SPI interface in the test bench set-up.

Finally, the literature reaches one of the most recent constructions that designs an IoT authentication scheme with the TRNG engine implementation as well. This flash TRNG

implementation is used in conjunction with an aforementioned flash PUF design from Mahmoodi et al. [59]. The TRNG engine is used as a nonce for the PUF challenge, which adds an extra layer of security to prevent replay attacks and masking PUF challenges and responses by being encrypted. Furthermore, by condensing both the TRNG engine and the PUF engine, energy-efficient designs that use 25% less space than standalone designs can be used. Although this method does utilize symmetric encryption and decryption schemes, which can incur extra overhead, it utilizes this together with PUF and TRNG constructions. More on the construction can be read in [60]. This construction utilizes a TRNG engine based on RTN and thermal noise (also known as flicker). This is done by applying a read to the same source line in two consecutive cycles on a NOR flash chip. This TRNG engine had a Pearson coefficient of 0.003 with PUF responses. This ensures that nonce from the TRNG and PUF response are not interrelated. Furthermore, the Shannon entropy was measured as 0.99958 and 0.99998 on the same dataset for the PUF and TRNG responses; this reinforces the claim that the PUF and TRNG outputs are not correlated and have maximum uncertainty. Furthermore, the probability mass function of the TRNG output highlights the 0.5 probability that the generated number will be "0" or "1". The autocorrelation of the 10-k-bit-long TRNG outputs perform ideally and have minimal correlation with lagged versions of the same input. This paper also further validates the TRNG construction by measuring its resistance to aging, temperature changes, and machine learning attacks, specifically multilayer perceptron (MLP) and long short-term memory models (LSTM). Neither of these is successful in predicting TRNG outputs. Finally, the output of the system is 192.3 Mbps and the TRNG passed all NIST tests. By providing a fully realizable construction tested against sophisticated machine learning attacks and an authentication scheme for a real IoT system, the literature has moved the flash TRNG into a much more realizable construction closer to being integrated into mainstream real-time systems.

Tables 2–4 illustrate the various qualities of flash-based PUFs and TRNGs. Table 2 is a classification of PUF and TRNG constructions based on their resistance to a variety of environmental effects such as temperature, voltage, and aging and also highlights the various quality tests used to gauge PUF/TRNG performance. An important note regarding the quality tests is that the statistical tools used to characterize their performance vary per implementation. Therefore, IDEAL values were added to give more context to the statistical test. Table 4 classifies PUF/TRNG constructions based on their configurations and performance metric. This table highlights whether a particular construction is updatable through software/firmware, whether periphery hardware or ECC is used, and investigates reliability error and estimated throughput.

Table 2. Classification of publications on flash-based PUFs and TRNGs according to various environmental conditions and quality tests.

Author(s)	Aging Resistant	Temperature Resistant	Voltage Resistant	Inter-Page Dependency	Intra-Page Accuracy	NIST Test
Physical Unclonable Function						
Prahbu et al. [11]	No	No	No	Pearson Coefficient. Program Disturb: 0.012 Read Disturb: 0.0 Program Latency: 0.02 to 0.03 IDEAL: 0.0	Pearson Coefficient. Program Disturb: 0.94 Read Disturb: 0.98 Program Latency: 0.83–0.84 IDEAL: 1.0	N/A
Wang et al. [53]	No	Yes	No	Pearson Coefficient Average: 0.0076	Pearson Coefficient Average: 0.9722	N/A

Table 2. *Cont.*

Author(s)	Aging Resistant	Temperature Resistant	Voltage Resistant	Inter-Page Dependency	Intra-Page Accuracy	NIST Test
Kim et al. [54]	No	No	No	None given	None given	N/A
Jia et al. [46]	Yes	Yes	No	Inter Chip Variation. Partial Program: 49.93% Partial Erasure: 49.95% Program Disturb: 46.86% IDEAL: 50%	None given	N/A
Clark et al. [57]	Yes	No	No	None given	None given	N/A
Sakib et al. [45]	Yes	Yes	No	Inter-Chip Hamming Distance: 49% to 51% IDEAL: 50%	Intra-Chip Hamming Distance: 0.2% to 1.7% IDEAL: 0%	N/A
Wu et al. [56]	Yes	Yes	Yes	Inter-Chip Hamming Distance: 0.499999 IDEAL: 0.5	Intra-Chip Hamming Distance: 0% IDEAL: 0%	N/A
Saito et al. [45]	Yes	Yes	Yes	Inter-Chip Hamming Distance: 49.4% IDEAL: 50%	Intra-Chip Hamming Distance: 0% IDEAL: 0%	N/A
Mahmoodi et al. [59]	Yes	Yes	No	Inter-Chip Hamming Distance: 50.3% IDEAL: 50%	Intra-Chip Hamming Distance: <5% IDEAL: 0%	N/A
True Random Number Generator						
Wang et al. [53]	Yes	Yes	No	N/A	N/A	Yes
Clark et al. [57]	Yes	Yes	No	N/A	N/A	Yes
Ray et al. [3]	Yes	Yes	No	N/A	N/A	Yes
Chakraborty et al. [49]	Yes	Yes	No	N/A	N/A	Yes
Larimian et al. [60]	Yes	Yes	No	N/A	N/A	Yes
Poudel et al. [58]	Yes	No	No	N/A	N/A	Yes

Table 3. Classification of publications on flash-based PUFs and TRNGs according to configuration and performance.

Author(s)	Software/Firmware Updatable	Estimated Throughput	ECC	Peripheral Hardware	Reliability Error
Physical Unclonable Function					
Prabhu et al. [11]	Yes, through standardized flash commands (RESET, READ, PROG, etc.)	None given	No	No	Not investigated. Each PUF signature is run until achieved sufficient accuracy.
Wang et al. [53]	Yes, through software-based classification of process variations (RTN and /or thermal noise)	848 bits/s to 3.37 kb/s	No	No	Aging Error: 10^{-4} exceeding 500,000 P/E cycles
Kim et al. [54]	Yes, through standardized NAND flash commands (PROG, READ, etc.)	None given	Yes, fuzzy extractor	No	Reliability Error: 2% in raw PUF generation

Table 3. *Cont.*

Author(s)	Software/Firmware Updatable	Estimated Throughput	ECC	Peripheral Hardware	Reliability Error
Jia et al. [46]	Yes, through standardized flash commands (ERASE, READ, and PROG)	7.35 kb/s to 22.38 kb/s	No	Yes, bit and position mapping	Reliability error: 128 bit key generated $<10^{-6}$
Clark et al. [57]	Yes, through 1.5 T type flash interfaces (ERASE, RESET, etc.)	None given	No	Yes, helper function MON1	None given
Sakib et al. [45]	Yes, adaptive through standardized flash command (PROG).	~16 kb/s	No	No	None given
Wu et al. [56]	No, uses custom alterations of flash cells	None given	No	Yes	<500 ppm in differential mode. 0 in single end mode
Saito et al. [55]	No, uses custom alterations of flash cells	None given	No	Yes	approximately 0%
Mahmoodi et al. [59]	No, uses custom alterations of flash cells	192.3 Mbps	No	Yes	<5%
True Random Number Generator					
Wang et al. [53]	Yes, through software-based classification of process variations	None given	No	No	N/A
Clark et al. [57]	No	None given	No	No	N/A
Milenkovic et al. [3]	Yes, through standardized flash commands (PROG and RESET)	None given	No	Yes	N/A
Chakraborty et al. [49]	No	700 k cycles: 7.2×10^8 bits	Yes, XOR circuit	No	N/A
Larimian et al. [60]	No	192.3 Mbps	Yes	Yes	N/A
Poudel et al. [53]	Yes, uses microcontroller flash memory.	123 processor clock cycles	No	No	N/A

Table 4. Characterization of performance qualities and types of different IC counterfeit detection strategies.

Author and Year	Passive/ Active/ Sensor	Cost	Programming Technique	Accuracy	Usage
IC Counterfeiting Detection					
Tehranipoor et al. [11] (2014)	Active	100%	None relies on physical inspection.	High	Any type of usage can be detected by physical inspection.
Huang et al. [53] (2015)	Passive	Low	Program variation and current leakage.	100% accurate	Test is designed for extremely aged components

Table 4. *Cont.*

Author and Year	Passive/ Active/ Sensor	Cost	Programming Technique	Accuracy	Usage
He et al. [54] (2016)	Sensor	Medium	None, uses EM probe to measure counterfeit	Highest Z score is 70 times greater for counterfeit device.	Test is designed for extremely aged components
Ye et al. [46] (2017)	Sensor	Ring oscillator, EM-aging sensor, and antifuse memory with a novel sensing architecture	Wire degradation and RO frequency failure	100% accuracy	Over 3 months of aging
Guo et al. [57] (2017)	Passive	One page required to characterize P/E stress	Bit error rate from Program/ Erase Cycling	100% accuracy	Detected with as little as 5% of flash life.
Kumari et al. [45] (2018)	Passive	Low	Bit Error Rate and P/E Latency	100% accuracy	Detected with as little as 3% of flash lifetime
Chattopadhyay et al. [45] (2019)	Passive	Low	Bit Error Rate and P/E Latency combined with machine learning	Greater than 97.5% accuracy	Detected with as little as 0.05% to 0.95% of flash lifetime.
Liu et al. [54] (2019)	Sensor	Five extra cells	Two exposed floating gate cells.	Bit line fluctuations were observed when changes in temperature, humidity and dust were induced.	N/A
Poudel et al. [45] (2020)	Passive	Low	Partial program combined with erasure latency	A watermark is imprinted to detect counterfeits	Applicable to any NOR flash chip.

5.3. Flash IC Counterfeit Detection

Integrated circuit (IC) counterfeiting is a severe problem affecting the global supply chain that introduces reliability and security concerns that affect a wide range of industries, from automotive electronics to sensitive military constructions. Furthermore, it increased by a factor of four just from 2009 to 2014—this has had extremely damaging effects, from fiscal loss to destroyed user confidence. Counterfeit electronics are generally classified into seven main categories: recycled, remarked, overproduced, defective, forged documentation, and tampered.

Furthermore, there are typically two main methods for counterfeit detection: physical inspection and electrical testing. Physical inspection can be one of the most advanced and comprehensive ways of detecting counterfeits; however, physical inspection is more costly and difficult to automate. Research in IC counterfeit detection faces various challenges such as (i) creating the least invasive techniques for successful detection since many detection methods damage the memory units which results in having aged chips; (ii) many of the counterfeit detection techniques also need expensive equipment. This is a big challenge since academic research laboratories need to be equipped with costly hardware.

The first paper for detecting IC counterfeiting with flash chips uses the physical inspection methodology proposed in 2014 [60]. In this paper, the advanced imaging processing techniques were considered, utilizing electron scanning microscopy coupled with

3D X-ray microscopy to detect counterfeits in five Intel flash memory ICs. The 3D scanning electron microscope technology (SEM) was able to detect the effects of sanding—this can facilitate the process of automating IC device scanning to detect counterfeit electronics that have been sanded. Furthermore, roughness patterns and coated/filled dimples were also measured to identify the effects of chip recycling. In addition, 3D and 2D X-ray results were used to distinguish incorrect die and chip orientation, which is another sign of IC counterfeiting. Along with improper orientations, die face delamination was also exposed to counterfeit chips through this technique. By combining both X-ray techniques and scanning electron microscopy techniques, physical inspection was able to identify every counterfeit intel flash chip.

This physical inspection presents strong results; however, it requires expensive equipment, subject matter experts to analyze the images, and is difficult to automate. In fact, the same authors would produce another method that relies on process variations extracted from partial programming in order to noninvasively detect IC counterfeits. This paper, released in 2017 from Guo et al. [61], is validated with 200,000 flash memory pages that are able to detect as little as 5% usage in flash memory chips with 100% accuracy. This is done by stressing one page to the end of its life in order to characterize the P/E cycle endurance of the entire chip. This then allows for characterization for a model of the ideal programming duration that varies throughout the lifetime of the flash memory. The amount of cells that flip during a varied programming time is recorded and each iteration is stored. This builds the enrollment model which is used later on to detect the counterfeit chips. This is the verification step which randomly chooses other papers to apply a partial program of a particular duration to randomly selected pages to see if the chip conforms to the model. This ID generation used with the counterfeit model that verifies if a chip is counterfeit or not is much less invasive and allows researchers to move away from the inherent limitations of advanced physical inspection.

Another paper from Kumari et al. [62] uses less invasive techniques to extract latency values from aged flash chips in order to detect counterfeits. Instead of a partial programming burst, the latency of different operations such as program and erase is used. If a page is aged through P/E cycling, the oxide layers in the flash cells wear down due to hot electron injection or quantum tunneling stress. Therefore, this drastically increases the erase time since the electric field is weakened. The program is slightly lengthened; however, it also increases the bit error rate (BER) since this induces reprogramming. Furthermore, the failed bit count is also increased because read noise increases in aged chips due to the defects in the oxide layers. These process variations bring great insight into the age of the chip, extending previous work by relying on more than one process variation and having greater accuracy in IC counterfeit detection. This work provides an IC counterfeit detection with 100% accuracy that only requires around 3% of the total endurance.

The next paper, released in 2019 from Chattopadhyay et al. [63], actually builds on the same technique presented above. However, this implementation uses machine learning to assist in the counterfeit detection of flash chips. Logistic regression (LR), support vector machines (SVM), and artificial neural networks (ANN) were used. This expanded the work from Kumari et al. [62], enhancing their detection scheme by achieving a counterfeit detection accuracy greater than 97.5% , only requiring 0.05% to 0.96% of the total P/E cycling endurance used. Furthermore, the author used three extra features besides the erase, programming, and bit error metrics used in the previous work to increase the accuracy. These three features measure the difference between neighboring blocks on the same chip. This allowed for greater accuracy and enhanced the process variations by introducing block differences within the same chip. These results were experimentally validated on Micron MLC and SLC chips and Toshiba MLC chips. This work does a phenomenal job of extending their technique to different SLC and MLC architectures and different IC vendors. This is crucial for extending existing techniques to allow them to be implementable on a wide assortment of flash memory chips.

The next detection technique from 2020 is from Poudel et al. [58], who also released a novel TRNG implementation for flash chips. However, this technique is a watermarking scheme for NOR flash chips to detect flash counterfeits. This technique imprints a watermark on NOR flash memory cells and then extracts process variations through the digital interface of NOR chips. Furthermore, this construction is completely agnostic to the device type. As long as the device is a NOR chip, it is able to be watermarked without requiring peripheral hardware through standardized flash memory commands. Secondly, this implementation is particularly useful as it does not require maintenance of chip-specific databases or require authentication with the original chip manufacturers. These are huge improvements that make this watermarking scheme much more adaptable and enhance previous works. This scheme performs a full program and then performs a partial erasure that is interrupted. This results in cells that are only partially programmed, and these cells are read N times, where N is an odd number. Then, odd parity will be chosen for the bit. The erasure interruption is increased and the results are recorded until the time reaches the normal erasure time. Since oxide degradation due to P/E cycling is irreversible, it will be extremely difficult for attackers to watermark circuits that are recycled. Circuits that are recycled will have greater P/E cycling stress and therefore it will be much more difficult to accurately imprint a forged watermark. This scheme was very successful in providing a watermarked IC counterfeit detection scheme for flash memory.

Another approach to counterfeit detection is creating counterfeit detection sensors rather than exploiting changes with flash chips themselves. Works have been proposed highlighting embedded sensors for IC counterfeit detection.

A work detailing this sensor is from Liu et al. [64], released in 2017. This work uses an eflash-based powerless non-volatile sensor that uses floating gate technology exposed on the integrated circuit. Without power, this sensor still experiences charge fluctuations from changes in humidity, temperature spikes, or increased dust particle density. When checking for tampering, this sensor can be powered on and analyzed. This sensor is designed with five floating gates comprising two counterfeit detection sensors. One of the sensors is the testing sensor while the second is the reference sensor. The testing sensor requires one program in order to have electrons deposited on its floating gate; after this, though, the sensor does not need power and is not programmed again. By using a reference sensor, the testing sensor has its threshold voltage changes amplified by thirty times what they usually are. Then, the bit line voltage is converted into a frequency output by a voltage controlled oscillator digitized as read out. Temperature, humidity, and dust particulate tests were done and successfully characterized by fluctuations in the frequency output. Similar ideas with exposed sensors have been used for IC counterfeiting using EM emanations and statistical methods from Huang et al. [65] and Ye et al. [66], respectively, in order to detect counterfeit electronic goods.

6. Potential Future Research Directions

The extent to which flash memory-based security primitives have been explored has just begun to scratch the surface. There are many potential research directions to this date, and in this section, we attempt to provide insight into those areas. We highlight some of the challenges that currently exist in flash-based hardware security solutions and offer up what we believe could benefit from future research and development.

6.1. Enhancing Existing 2D Flash Memory Features

In many of the presented PUF/TRNG constructions in Tables 2 and 4, signature/number generation time has been decreased by mixing multiple approaches or by adding a deeper characterization to an existing approach. However, characterizing what programming schemes to mix and combine is not thoroughly understood. Researchers in [11] allowed for a great first step in understanding how each of these programming commands can be used individually. However, ***some schemes that employ programming disturb and read disturb may be enhanced through a different combination of various programming methods.***

This has happened in the literature; however, there is little characterization or exploration of combining multiple programming processes to increase PUF or TRNG throughput by exploiting their trade-offs. On the other hand, deeper characterization of existing disturbs can also provide ways to enhance PUF/TRNG designs as well. Sakib et al. [45] were able to enhance program disturb-based PUFs by characterizing the program disturb effect on threshold voltages and by analyzing the effect of program disturb through a device's aging. This allowed for a high-throughput PUF that also had aging resistance due to this characterization. By deeply characterizing the effects of these programming styles, it may add greater understanding of how process variations occur and how they change over time. This deeper characterization can be used to enhance many features of existing PUF/TRNG constructions. This development was seen in the "third phases" and the "second phase", respectively, of PUF and TRNG development in Section 5. In more detail, TRNGs in [3,49,57] extended flash TRNGs to be applicable to other forms of non-volatile memory, to high-performance flash modules, and mix multiple flash programming styles to increase throughput. Flash PUFs in [45,46,54] also provide higher throughput, PUF tunability, and great aging resistance by combining existing programming styles to extract process variations. Further characterizing process variations sources and combining multiple programming styles can significantly enhance PUF and TRNG constructions. ***Achieving the correct "mix" of these modifications requires a deeper investigation and holds promise in future research applications.***

A research project could be easily done investigating different programming combinations to optimize PUF accuracy, resiliency, or throughput. For example, aging is still a huge problem for flash-based PUFs since repetitive P/E cycles degrade the quality of flash cells over time. Creating adaptive aging constructions by observing programming latency or other process variations could help extend aging-resistant flash PUFs beyond the program disturb based construction in [45]. A deeper characterization of how aging affects these process variations is required to develop more aging-resistant PUFs. This could extend aging resistance outside of just program disturb-based PUFs. Furthermore, when looking at Table 2, voltage variation is not considered for all of the PUF constructions. This may make these PUFs vulnerable to voltage-based attacks, and this provides researchers an opportunity to investigate the effects of varying voltage on these process variations. This can provide more voltage-tampering-resistant PUFs that have enhanced security and greater resistance to sudden voltage changes in a given deployment environment. Another research avenue could also be just experimenting different types of flash-based PUFs with various combinations of process variations. For example, [11] remarked how read disturb-based PUFs were infeasible to practically use because of long signature generation times. However, what if programming interrupt or partial programming were used to prime flash cells into unsteady or unstable states? This can significantly increase PUF throughput while also leveraging the high accuracy from a read disturb-based PUF as shown in [11].

6.2. Leveraging 3D Flash Memories for New Hardware Security Applications

Three-dimensional NAND flash technology is a new storage solution offering even higher memory density than 2D planar NAND flash. Since 3D flash technology is very new, with the first V-NAND technology created in 2012 and first being shipped into an enterprise in 2016, there has been little work even in the academic setting for applying hardware security primitives to 3D flash devices. However, 3D flash memories are prime candidates for hardware security due to new exploitable process variations from the geometric properties of 3D memories. When discussing possible 3D flash security primitives, this discussion will focus on CT (charge trap) 3D flash rather than on FG (floating gate) 3D flash. T3D FG flash does not provide fundamental advantages to 2D flash in cell programming performance. It does increase storage density; however, CT 3D flash provides additional benefits due to the cylindrical shape of the CT cells. For this reason, possible CT 3D flash security primitives will be the assumed 3D flash for hardware security primitive applications.

Another concern is that 3D CT devices include the reliability issues in planar flash technologies such as various disturbs, RTN, and aging problems. However, the vertical channel in CT flash causes vertical challenge loss through the top and bottom oxides and lateral charge migrations towards spacers. This extra unreliability causes irregular electric field distributions in the bottom and top oxides and causes enhanced charge loss in 3D memories that is much worse compared to 2D architectures.

Due to the difficulty of separating the charge-trap layer between each layer, each cell's active charge area can leak towards cells on the same string. The charge loss causes band bending in the top oxide (TBO) and in the bottom oxide (BTO). In the lateral charge migration, the charge can leak through the lateral spacers, which can be accelerated by high temperatures. Furthermore, these lateral spaces are very difficult to position because of the difficulty in cutting the cylindrical geometries in the CT cells. This causes severe charge migrations that have promise for sources of uniqueness or entropy in flash security primitives. To further highlight this, [67] used a Monte Carlo simulation of a 3D NAND flash module to demonstrate the variations in the current string of 3D flash due to process variations in the geometry of CT flash memory cells. The simulation was able to generate excellent uniformity, diffuseness, and uniqueness for a simulated PUF structure based on the variation of the string currents in 3D CT flash. Furthermore, due to the high density of the flash, the amount of challenge and response pairs (CRPs) was sufficiently large to resist machine learning simulation-based attacks.

Due to the immense CRP space and the severe charge leakage in 3D structures vs. 2D structures, 3D flash memory is an extremely promising candidate for hardware security primitives. This is another active area of research that is only just starting because of the novelty of 3D flash. However, it offers great promise for future research in leveraging the severe charge loss present in 3D flash for a fruitful security function.

6.3. Exploring New Process Variation for Flash Memories

As flash memory cells scale down in size and bit density increases, unique opportunities arrive for exploiting new process variations in flash chips due to an increased sensitivity of flash cells. These new process variations are not new in the strict sense of the word. They have always existed. However, because the scaling down process of flash cells makes the cells more sensitive, these process variations now have a much larger effect on bit error rates, which are observable through a programming interface. An example of this can be seen in TLC flash memory [68]. TLC flash memory has eight possible states storing three bits per state. Due to the increased sensitivity of TLC flash memory, a well-defined programming sequence needs to be used to avoid cell-to-cell interference. This is known as foggy-fine programming. Foggy-fine programming initially uses very large programming steps to increase its voltage level on its LSB bit. Then, the next two bits are programmed to intermediate temporary values [68]. Finally, fine partial programming steps are applied to program the TLC cells to the correct narrow voltage windows. This programming process has immense potential in extracting process features in TLC flash because of the multiple programming stages [68]. A well-placed interrupt could be used to extract process variations for the highly sensitive TLC cells.

Furthermore, this increased sensitivity in TLC and QLC flash also leads to other unique process variations that occur in exploitable trends. An example of this is known as "symbol error rate." In higher density flash memories, the bit error rate increases over time, and this is expected due to increased P/E cycle aging. However, these high-density flash chips also have fluctuating bit error rates throughout the flash chip lifetime [69]. In TLC flash, the bit error rate increases at different rates depending on which bit, LSB, CSB (the middle bit), or MSB, is programmed. Furthermore, 96.17% of all bit errors were due to only one bit flip per cell [69]. Very few cells ever had two or three bit errors. Furthermore, errors in flash blocks also tend to arise from a small subsection of cells within the block. This localizes and concentrates bit errors into a single locality. These trends offer great opportunities for exploitation in flash security primitives.

Although many of these new variations have been brought up in the context of TLC and QLC flash chips, similar variations can be discovered and exploited in MLC chips, as well. This is due to the aggressive device downsizing that has shrunken floating gate transistors' sizes down exponentially. Despite this being traditionally associated with negative process variations, this allows for great sources of entropy that are observable through the flash digital interface for hardware security primitives.

6.4. Discovering the Vulnerabilities of Existing Flash Memory-Based Security Primitives

Many of the existing flash-based PUFs and TRNGs have very little resistance regarding possible attacks or vulnerabilities in their design. However, we wish to enumerate and explain security vulnerabilities in flash chips that could be used to damage the integrity of flash security primitives. The first concern is a secure data deletion. When the erase function is completed, flash chips often experience data remanence. This can lead to the recovery of already erased data. For example, attackers in [70,71] were able to locate programmed/erased blocks vs. non-programmed blocks due to differences in the threshold voltages of the cells after they were erased. The threshold voltages were easily differentiated between programmed/erased because of a 0.5-volt difference in threshold voltages. This data recovery can make flash PUFs susceptible to modeling attacks and flash TRNGs could have their random numbers determined. To prevent this attack, secure flash cell overwriting is necessary to prevent data remanence effects.

Another attack vector for flash memory is based on the two step-programming sequences of MLC flash. In an effort to reduce the cell-to-cell programming interference in flash memory, designers employ a unique two-step programming sequence that separates the programming of the LSB and MSB of MLC flash cells (as MLC flash memory holds two bits per cell) [68]. This is done to help reduce the threshold voltage swing induced by writing both at the same time since this swing induces severe cell-to-cell interference. However, this programming sequence can be manipulated for fault injection. When the LSB is programmed for a particular page, the MSB will be programmed next. However, while this program is happening, researchers were able to maliciously program neighboring pages to induce severe program disturbs that corrupted the MSB values of the victim page [68]. These errors went undetected and caused severe raw bit errors in these cells.

Finally, advanced optical attacks were also demonstrated to zero out confidential information from secure flash memory modules. This was done through novel selective bumping attacks [72]. These attacks allow for bypassing traditional verification of flash memory blocks. This was done by "bumping" flash cells to a specific value, which was able to expedite brute force searching attacks on flash memory cells and enhanced reverse engineering efforts of the flash memory units. Furthermore, memory control logic was able to be identified in flash chips through OBIC imaging, and this control logic was modified through laser attacks [72]. This could possibly leak PUF information and divulge obfuscation techniques to attackers, compromising PUF and TRNG constructions.

These attack methods against flash memory units could also be adopted to attack flash security primitives; therefore, it is of imperative importance for researchers to tackle existing vulnerabilities and discover new weaknesses in these security primitives or the flash chips themselves. The literature is also very undeveloped in this area, allowing for great research opportunities into attacks on flash security primitives.

7. Conclusions

As shown by this survey through a detailed presentation and discussion of the evolution of flash memory-based PUFs/TRNGs, flash memory has the potential to be one of the main hardware security primitive modules. However, there are many challenges that remain before practical flash memory-based security solutions can be used. PUF/TRNG constructions themselves have many challenging barriers to widespread adoption, such as aging dependence, lower throughput, and environmental dependencies, especially voltage variation. Furthermore, attacks to these existing constructions need a deeper investigation

to validate the efficacy of flash security primitives. Despite these challenges, flash memory is a great platform for security primitives as it is one of the most common forms of non-volatile memory used in industry. Moreover, it is constantly being innovated through various advanced architectures, such as 3D flash, QLC, and TLC memory modules, to adapt to modern data storage demands. These various new platforms also show even greater promise in integrating novel security primitives, as highlighted in Section 6's discussion of developing research areas. In these changing times for hardware security, quality research is key to affirm non-volatile memories, particularly flash memory, as a viable alternative to the most famous approaches. Hopefully, following the challenges discussed in this survey, research can address the open issues, taking advantage of current opportunities, and propose increasingly competitive and applicable flash memory-based hardware security primitives.

Author Contributions: Conceptualization, H.G. and F.T.; methodology, H.G. and F.T.; software, H.G. and J.E.; validation, H.G., J.E., and F.T.; formal analysis, H.G.; investigation, F.T., N.K., and W.Y.; resources, J.E., W.Y., and N.K.; data curation, H.G. and S.G.; writing—original draft preparation, H.G.; writing—review and editing, S.G. and F.T.; visualization, H.G., J.E., and S.G.; supervision, F.T., N.K., and W.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: This study did not report any data.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Anagnostopoulos, N.A.; Katzenbeisser, S.; Chandy, J.; Tehranipoor, F. An overview of DRAM-based security primitives. *Cryptography* **2018**, *2*, 7.
2. Tehranipoor, F.; Karimian, N.; Xiao, K.; Chandy, J. DRAM based intrinsic physical unclonable functions for system level security. In Proceedings of the 25th edition on Great Lakes Symposium on VLSI, Pittsburgh, PA, USA, 20–22 May 2015; pp. 15–20.
3. Ray, B.; Milenković, A. True random number generation using read noise of flash memory cells. *IEEE Trans. Electron Devices* **2018**, *65*, 963–969.
4. Karimian, N.; Tehranipoor, F. How to Generate Robust Keys from Noisy DRAMs? In Proceedings of the 2019 on Great Lakes Symposium on VLSI, Tysons Corner, VA, USA, 9–11 May 2019; pp. 465–469.
5. 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, Washington, DC, USA, 18–22 November 2002; pp. 148–160.
6. Sushma, R.; Murty, N.S. Feedback Oriented XORed Flip-Flop Based Arbiter PUF. In Proceedings of the 2018 International Conference on Electrical, Electronics, Communication, Computer, and Optimization Techniques (ICEECCOT), Msyuru, India, 14–15 December 2018; pp. 1444–1448, doi:10.1109/ICEECCOT43722.2018.9001605.
7. Cui, Y.; Wang, C.; Liu, W.; Yu, Y.; O'Neill, M.; Lombardi, F. Low-cost configurable ring oscillator PUF with improved uniqueness. In Proceedings of the 2016 IEEE International Symposium on Circuits and Systems (ISCAS), Montreal, QC, Canada, 22–25 May 2016; pp. 558–561, doi:10.1109/ISCAS.2016.7527301.
8. Anagnostopoulos, N.A.; Arul, T.; Fan, Y.; Hatzfeld, C.; Lotichius, J.; Sharma, R.; Fernandes, F.; Tehranipoor, F.; Katzenbeisser, S. Securing IoT Devices Using Robust DRAM PUFs. In Proceedings of the 2018 Global Information Infrastructure and Networking Symposium (GIIS), Thessaloniki, Greece, 23–25 October 2018; pp. 1–5, doi:10.1109/GIIS.2018.8635789.
9. Urien, P. Innovative ATMEGA8 Microcontroler Static Authentication Based on SRAM PUF. In Proceedings of the 2020 IEEE 17th Annual Consumer Communications Networking Conference (CCNC), Las Vegas, NV, USA, 10–13 January 2020; pp. 1–2, doi:10.1109/CCNC46108.2020.9045502.
10. Rührmair, U.; Holcomb, D.E. PUFs at a glance. In Proceedings of the 2014 Design, Automation Test in Europe Conference Exhibition (DATE), Dresden, Germany, 24–28 March 2014; pp. 1–6, doi:10.7873/DATE.2014.360.
11. Prabhu, P.; Akel, A.; Grupp, L.M.; Wing-Kei, S.Y.; Suh, G.E.; Kan, E.; Swanson, S. Extracting device fingerprints from flash memory by exploiting physical variations. In Proceedings of the International Conference on Trust and Trustworthy Computing, Pittsburgh, PA, USA, 22–24 June 2011; Springer: Berlin/Heidelberg, Germany, 2011; pp. 188–201.
12. Tehranipoor, F.; Karimian, N.; Yan, W.; Chandy, J.A. DRAM-based intrinsic physically unclonable functions for system-level security and authentication. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2016**, *25*, 1085–1097.
13. Eckert, C.; Tehranipoor, F.; Chandy, J.A. DRNG: DRAM-based random number generation using its startup value behavior. In Proceedings of the 2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS), Boston, MA, USA, 6–9 August 2017; pp. 1260–1263.

14. Brederlow, R.; Prakash, R.; Paulus, C.; Thewes, R. A low-power true random number generator using random telegraph noise of single oxide-traps. In Proceedings of the 2006 IEEE International Solid State Circuits Conference-Digest of Technical Papers, San Francisco, CA, USA, 6–9 February 2006; pp. 1666–1675.
15. Tehranipoor, F.; Wortman, P.; Karimian, N.; Yan, W.; Chandy, J.A. DVFT: A Lightweight Solution for Power-Supply Noise-Based TRNG Using Dynamic Voltage Feedback Tuning System. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2018**, *26*, 1084–1097.
16. Tehranipoor, F.; Yan, W.; Chandy, J.A. Robust hardware true random number generators using DRAM remanence effects. In Proceedings of the 2016 IEEE International Symposium on Hardware Oriented Security and Trust (HOST), McLean, VA, USA, 3–5 May 2016; pp. 79–84.
17. Guin, U.; Huang, K.; DiMase, D.; Carulli, J.M.; Tehranipoor, M.; Makris, Y. Counterfeit Integrated Circuits: A Rising Threat in the Global Semiconductor Supply Chain. *Proc. IEEE* **2014**, *102*, 1207–1228, doi:10.1109/JPROC.2014.2332291.
18. Pecht, M.; Tiku, S. Bogus: Electronic manufacturing and consumers confront a rising tide of counterfeit electronics. *IEEE Spectr.* **2006**, *43*, 37–46, doi:10.1109/MSPEC.2006.1628506.
19. Nathalie, K.-N.; Stephanie, P. Qualification and Testing Process to Implement Anti-Counterfeiting Technologies into IC Packages. In Proceedings of the 2013 Design, Automation & Test in Europe Conference & Exhibition (DATE), Grenoble, France, 18–22 March 2013; pp. 1131–1136, ISBN 978-1-4673-5071-6, doi:10.7873/DATE.2013.237.
20. Guin, U.; Wang, W.; Harper, C.; Singh, A.D. Detecting recycled socs by exploiting aging induced biases in memory cells. In Proceedings of the 2019 IEEE International Symposium on Hardware Oriented Security and Trust (HOST), McLean, VA, USA, 5–10 May 2019.
21. Schaller, A.; Xiong, W.; Anagnostopoulos, N.A.; Saleem, M.U.; Gabmeyer, S.; Škorić, B.; Katzenbeisser, S.; Szefer, J. Decay-Based DRAM PUFs in Commodity Devices. *IEEE Trans. Dependable Secur. Comput.* **2018**, *16*, 462–475.
22. Holcomb, D.E.; Burleson, W.P.; Fu, K. Power-up SRAM state as an identifying fingerprint and source of true random numbers. *IEEE Trans. Comput.* **2008**, *58*, 1198–1210.
23. Wang, W.; Guin, U.; Singh, A. Aging-Resilient SRAM-based True Random Number Generator for Lightweight Devices. *J. Electron. Test.* **2020**, *36*, 301–311.
24. Cai, Y.; Luo, Y.; Ghose, S.; Mutlu, O. Read disturb errors in MLC NAND flash memory: Characterization, mitigation, and recovery. In Proceedings of the 2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, Rio de Janeiro, Brazil, 22–25 June 2015; pp. 438–449.
25. Che, W.; Plusquellic, J.; Bhunia, S. A non-volatile memory based physically unclonable function without helper data. In Proceedings of the 2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), San Jose, CA, USA, 2–6 November 2014; pp. 148–153.
26. De Castro, C.G.; de Medeiros Câmara, S.; da Costa Carmo, L.F.R.; Boccardo, D.R. EVINCED: Integrity Verification Scheme for Embedded Systems Based on Time and Clock Cycles. In Proceedings of the 2017 IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing, 15th Intl Conf on Pervasive Intelligence and Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech), Orlando, FL, USA, 6–10 November 2017; pp. 788–795.
27. Bindal, A. *Fundamentals of Computer Architecture and Design*; Springer: Berlin/Heidelberg, Germany, 2017.
28. Oh, J.W. *Reverse Engineering Flash Memory for Fun and Benefit*; Blackhat US: Las Vegas, NV, USA, 2014.
29. Micheloni, R. *3D Flash Memories*; Springer: Dordrecht, The Netherlands, 2016.
30. Chang, K.P.; Lue, H.T.; Chen, C.P.; Chen, C.F.; Chen, Y.R.; Hsiao, Y.H.; Hsieh, C.C.; Shih, Y.H.; Yang, T.; Chen, K.C.; et al. Memory architecture of 3d vertical gate (3dvg) nand flash using plural island-gate ssl decoding method and study of it's program inhibit characteristics. In Proceedings of the 2012 4th IEEE International Memory Workshop, Milan, Italy, 20–23 May 2012; pp. 1–4.
31. Shirota, R.; Nakayama, R.; Kirisawa, R.; Momodomi, M.; Sakui, K.; Itoh, Y.; Aritome, S.; Endoh, T.; Hatori, F.; Masuoka, F. A 2.3 mu m/sup 2/memory cell structure for 16 Mb NAND EEPROMs. In Proceedings of the International Technical Digest on Electron Devices, San Francisco, CA, USA, 9–12 December 1990; pp. 103–106.
32. Campardo, G.; Scotti, M.; Scommegna, S.; Pollara, S.; Silvagni, A. An overview of flash architectural developments. *Proc. IEEE* **2003**, *91*, 523–536, doi:10.1109/JPROC.2003.811705.
33. Jung, S.M.; Jang, J.; Cho, W.; Cho, H.; Jeong, J.; Chang, Y.; Kim, J.; Rah, Y.; Son, Y.; Park, J.; et al. Three dimensionally stacked NAND flash memory technology using stacking single crystal Si layers on ILD and TANOS structure for beyond 30 nm node. In Proceedings of the 2006 International Electron Devices Meeting, San Francisco, CA, USA, 11–13 December 2006; pp. 1–4.
34. Shijun, L.; Xuecheng, Z. Analysis of 3D NAND technologies and comparison between charge-trap-based and floating-gate-based flash devices. *J. China Univ. Posts Telecommun.* **2017**, *24*, 75–96.
35. Micheloni, R.; Crippa, L.; Zambelli, C.; Olivo, P. Architectural and integration options for 3d NAND flash memories. *Computers* **2017**, *6*, 27.
36. Chowdhur, M.A.H.; Ki-Hyung Kimy. A survey of flash memory design and implementation of database in flash memory. In Proceedings of the 2008 3rd International Conference on Intelligent System and Knowledge Engineering, Xiamen, China, 17–19 November 2008; Volume 1, pp. 1256–1259, doi:10.1109/ISKE.2008.4731123.

37. Fukuzumi, Y.; Katsumata, R.; Kito, M.; Kido, M.; Sato, M.; Tanaka, H.; Nagata, Y.; Matsuoka, Y.; Iwata, Y.; Aochi, H.; et al. Optimal integration and characteristics of vertical array devices for ultra-high density, bit-cost scalable flash memory. In Proceedings of the 2007 IEEE International Electron Devices Meeting, Washington, DC, USA, 10–12 December 2007; pp. 449–452.
38. Tanaka, H.; Kido, M.; Yahashi, K.; Oomura, M.; Katsumata, R.; Kito, M.; Fukuzumi, Y.; Sato, M.; Nagata, Y.; Matsuoka, Y.; et al. Bit cost scalable technology with punch and plug process for ultra high density flash memory. In Proceedings of the 2007 IEEE Symposium on VLSI Technology, Kyoto, Japan, 12–14 June 2007; pp. 14–15.
39. Ishiduki, M.; Fukuzumi, Y.; Katsumata, R.; Kito, M.; Kido, M.; Tanaka, H.; Komori, Y.; Nagata, Y.; Fujiwara, T.; Maeda, T.; et al. Optimal device structure for pipe-shaped BiCS flash memory for ultra high density storage device with excellent performance and reliability. In Proceedings of the 2009 IEEE International Electron Devices Meeting (IEDM), Baltimore, MD, USA, 7–9 December 2009; pp. 1–4.
40. Jeong, W.; Im, J.W.; Kim, D.H.; Nam, S.W.; Shim, D.K.; Choi, M.H.; Yoon, H.J.; Kim, D.H.; Kim, Y.S.; Park, H.W.; et al. A 128 Gb 3b/cell V-NAND flash memory with 1 Gb/s I/O rate. *IEEE J. Solid State Circuits* **2015**, *51*, 204–212.
41. Kang, D.; Jeong, W.; Kim, C.; Kim, D.H.; Cho, Y.S.; Kang, K.T.; Ryu, J.; Kang, K.M.; Lee, S.; Kim, W.; et al. 256 Gb 3 b/cell V-NAND flash memory with 48 stacked WL layers. *IEEE J. Solid-State Circuits* **2016**, *52*, 210–217.
42. Aochi, H. BiCS flash as a future 3D non-volatile memory technology for ultra high density storage devices. In Proceedings of the 2009 IEEE International Memory Workshop, Monterey, CA, USA, 10–14 May 2009; pp. 1–2.
43. Nishi, Y.; Magyari-Kope, B. *Advances in Non-Volatile Memory and Storage Technology*; Woodhead Publishing: Cambridge, UK, 2019.
44. Xu, S.Q.; Yu, W.k.; Suh, G.E.; Kan, E.C. Understanding sources of variations in flash memory for physical unclonable functions. In Proceedings of the 2014 IEEE 6th International Memory Workshop (IMW), Taipei, Taiwan, 18–21 May 2014; pp. 1–4.
45. Sakib, S.; Milenković, A.; Rahman, M.T.; Ray, B. An Aging-Resistant NAND Flash Memory Physical Unclonable Function. *IEEE Trans. Electron Devices* **2020**, *67*, 937–943.
46. Jia, S.; Xia, L.; Wang, Z.; Lin, J.; Zhang, G.; Ji, Y. Extracting robust keys from nand flash physical unclonable functions. In Proceedings of the International Conference on Information Security, Trondheim, Norway, 9–11 September 2015; Springer: Cham, Switzerland, 2015; pp. 437–454.
47. Roach, A.H.; Gadlage, M.J.; Duncan, A.R.; Ingalls, J.D.; Kay, M.J. Interrupted PROGRAM and ERASE operations for characterizing radiation effects in commercial NAND flash memories. *IEEE Trans. Nucl. Sci.* **2015**, *62*, 2390–2397.
48. Heidecker, J. *Flash Memory Reliability: Read, Program, and Erase Latency versus Endurance Cycling*; Technical Report; Jet Propulsion Laboratory, National Aeronautics and Space Administration: Pasadena, CA, USA, 2010.
49. Chakraborty, S.; Garg, A.; Suri, M. True Random Number Generation From Commodity NVM Chips. *IEEE Trans. Electron Devices* **2020**, *67*, 888–894.
50. Fayrushin, A.; Seol, K.; Na, J.; Hur, S.; Choi, J.; Kim, K. The new program/erase cycling degradation mechanism of NAND flash memory devices. In Proceedings of the 2009 IEEE International Electron Devices Meeting (IEDM), Baltimore, MD, USA, 7–9 December 2009; pp. 1–4.
51. Joe, S.M.; Yi, J.H.; Park, S.K.; Shin, H.; Park, B.G.; Park, Y.J.; Lee, J.H. Threshold voltage fluctuation by random telegraph noise in floating gate NAND flash memory string. *IEEE Trans. Electron Devices* **2010**, *58*, 67–73.
52. Puglisi, F.M.; Padovani, A.; Larcher, L.; Pavan, P. Random telegraph noise: Measurement, data analysis, and interpretation. In Proceedings of the 2017 IEEE 24th International Symposium on the Physical and Failure Analysis of Integrated Circuits (IPFA), Chengdu, China, 4–7 July 2017; pp. 1–9.
53. 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 Proceedings of the 2012 IEEE Symposium on Security and Privacy, San Francisco, CA, USA, 20–23 May 2012; pp. 33–47.
54. Kim, M.S.; Moon, D.I.; Yoo, S.K.; Lee, S.H.; Choi, Y.K. Investigation of physically unclonable functions using flash memory for integrated circuit authentication. *IEEE Trans. Nanotechnol.* **2015**, *14*, 384–389.
55. et al., T.S. High-Temperature Stable Physical Unclonable Functions with Error-Free Readout Scheme Based on 28nm SG-MONOS Flash Memory for Security Applications. In Proceedings of the 2017 IEEE International Memory Workshop (IMW), Monterey, CA, USA, 14–17 May 2017; pp. 1–4.
56. Wu, M.; Yang, T.; Chen, L.; Lin, C.; Hu, H.; Su, F.; Wang, C.; Huang, J.P.; Chen, H.; Lu, C.C.; et al. A PUF scheme using competing oxide rupture with bit error rate approaching zero. In Proceedings of the 2018 IEEE International Solid—State Circuits Conference (ISSCC), San Francisco, CA, USA, 11–15 February 2018; pp. 130–132, doi:10.1109/ISSCC.2018.8310218.
57. Clark, L.T.; Adams, J.; Holbert, K.E. Reliable techniques for integrated circuit identification and true random number generation using 1.5-transistor flash memory. *Integration* **2019**, *65*, 263–272.
58. P. Poudel, B.R.; Milenkovic, A. Microcontroller TRNGs Using Perturbed States of NOR Flash Memory Cells. *IEEE Trans. Comput.* **2019**, *68*, 307–313.
59. Mahmoodi, M.; Nili, H.; Larimian, S.; Guo, X.; Strukov, D. ChipSecure: A Reconfigurable Analog eFlash-Based PUF with Machine Learning Attack Resiliency in 55nm CMOS. In Proceedings of the 2019 56th ACM/IEEE Design Automation Conference (DAC), Las Vegas, NV, USA, 2–6 June 2019; pp. 1–6.
60. Larimian, S.; Mahmoodi, M.R.; Strukov, B.D. Lightweight Integrated Design of PUF and TRNG Security Primitives Based on eFlash Memory in 55-nm CMOS. *IEEE Trans. Electron Devices* **2020**, *67*, 1586–1592.

61. Zimu Guo, X. Xu, M.M.T.; Forte, D. FFD: A framework for fake flash detection. In Proceedings of the 2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC), Austin, TX, USA, 18–22 June 2017; Volume 54, pp. 1–6.
62. Kumari, P.; Talukder, B.M.S.B.; Sakib, S.; Ray, B.; Rahman, M.T. Independent detection of recycled flash memory: Challenges and Solutions. In Proceedings of the 2018 IEEE International Symposium on Hardware Oriented Security and Trust (HOST), Washington, DC, USA, 30 April–4 May 2018; pp. 89–95.
63. Chattopadhyay, S.; Kumari, P.; Ray, B.; Chakraborty, R.S. Machine Learning Assisted Accurate Estimation of Usage Duration and Manufacturer for Recycled and Counterfeit Flash Memory Detection. In Proceedings of the 2019 IEEE 28th Asian Test Symposium (ATS), Kolkata, India, 10–13 December 2019; pp. 49–495.
64. Liu, M.; Kim, C.H. A powerless and non-volatile counterfeit IC detection sensor in a standard logic process based on an exposed floating-gate array. In Proceedings of the 2017 Symposium on VLSI Technology, Kyoto, Japan, 5–8 June 2017; Volume 68, pp. T102–T103.
65. He, K.; Huang, X.; Tan, S.X.-D. EM-based on-chip aging sensor for detection and prevention of counterfeit and recycled ICs. In Proceedings of the 2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), Austin, TX, USA, 2–6 November 2015; pp. 146–151.
66. Ye, Y.; Kim, T.; Chen, H.; Wang, H.; Tlelo-Cuautle, E.; Tan, S.X.-D. Comprehensive detection of counterfeit ICs via on-chip sensor and post-fabrication authentication policy. In Proceedings of the 2017 14th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD), Giardini Naxos, Italy, 12–15 June 2017; pp. 1–4.
67. Sahay, S.; Klachko, M.; Strukov, D. Hardware Security Primitive Exploiting Intrinsic Variability in Analog Behavior of 3-D NAND Flash Memory Array. *IEEE Trans. Electron Devices* **2019**, *66*, 2158–2164.
68. Cai, Y.; Ghose, S.; Luo, Y.; Mai, K.; Mutlu, O.; Haratsch, E.F. Vulnerabilities in MLC NAND flash memory programming: Experimental analysis, exploits, and mitigation techniques. In Proceedings of the 2017 IEEE International Symposium on High Performance Computer Architecture (HPCA), Austin, TX, USA, 4–8 February 2017; pp. 49–60.
69. Wanner, L.; Lai, L.; Rahimi, A.; Gottscho, M.; Mercati, P.; Huang, C.H.; Sala, F.; Agarwal, Y.; Dolecek, L.; Dutt, N.; et al. NSF expedition on variability-aware software: Recent results and contributions. *it-Inf. Technol.* **2015**, *57*, 181–198.
70. Xin, R.; Ye, M.; Wang, J.; Hu, K.; Zhao, Y. Data deletion method for security improvement of Flash memories. *IEICE Electron. Express* **2018**, *15*, 20180152.
71. Wang, J.; Zhao, Y.; Xin, R.; Ye, M. A study of residual characteristics in floating gate transistors. *Sci. China Inf. Sci.* **2018**, *61*, 069402:1–069402:3.
72. Skorobogatov, S. Flash memory 'bumping'attacks. In Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems, Santa Barbara, CA, USA, 17–20 August 2010; Springer: Berlin/Heidelberg, Germany, 2010; pp. 158–172.