Bit^{2}RNG: Leveraging Bad-page Initialized Table with Bit-error Insertion for True Random Number Generation in Commodity Flash Memory

Wei Yan[1], Huifeng Zhu[1], Zhiyuan Yu[1], Fatemeh Tehranipoor[2], John Chandy[3], Ning Zhang[1] and Xuan Zhang[1]
[1]*Washington University in St. Louis*, St. Louis, Missouri, USA
[2]*Santa Clara University*, Santa Clara, California, USA
[3]*University of Connecticut*, Storrs, Connecticut, USA
{weiyan, zhuhuifeng, yu.zhiyuan, zhang.ning, xuan.zhang}@wustl.edu, ftehranipoor@scu.edu, john.chandy@uconn.edu

***Abstract*—Nowadays NAND flash memory is the de-facto storage technology that is widely used from compact commercial off-the-shelf (COTS) embedded devices to large-scale cloud computing facilities. Motivated by the growing demand for mobile and Internet-of-Thing (IoT) applications, researchers have proposed many innovative ways to leverage the physical characteristics of memory devices for different security functionalities. However, many existing solutions lack thorough considerations of practical factors such as device aging, implementation cost, and runtime speed, preventing them from being directly adopted for real-world industrial applications. In this work, we present a novel true random number generation method called Bit^{2}RNG that leverages the intrinsic system resources by combining the bad pages and bit errors in NAND flash as a random source. Our solution requires no hardware modifications to the memory chip, its communication interface, or the flash controller, and consumes no additional system memory space. To demonstrate the capability and benefit of the proposed Bit^{2}RNG technology, we explore several lightweight IoT applications including cryptographic key generation, device identification, and data provenance. The experimental results indicate that Bit^{2}RNG is a practical solution with better system performance trade-off compared with other state-of-the-art TRNG techniques.**

***Index Terms*—NAND flash memory, TRNG, cryptographic key generation, chip identification, image provenance, secure boot.**

I. Introduction

Flash memory has rapidly become the preferred de-facto storage solution for commercial off-the-shelf (COTS) electronic devices today. In general-purpose computing systems such as personal computers and servers, hard-disk drives are increasingly being replaced by solid-state drives, as the parallelization operation of NAND flash offers much higher read/write bandwidth. For resource-constrained systems such as mobile phones and Internet-of-Things (IoT) devices, flash is an especially indispensable component thanks to its non-volatility, high storage density, large capacity, and low cost. This resource efficiency advantage is even more prominent when flash is applied to lightweight portable mediums like SD cards and USB drives [1].

As these devices permeate many critical aspects of our society and everyday life, their security protection becomes increasingly important. However, many security primitives require additional computational power (e.g. asymmetric key cryptography), or additional hardware components (e.g. true random number generator), which ultimately increases the production cost. To enable efficient lightweight security primitives on resource-constraint devices, there have been increasing interests in the research community to leverage intrinsic physical features of hardware components to achieve the security objectives. For example, various physical characteristics of flash memory cells are used for true random number generators (TRNG) [2] or physical unclonable functions (PUFs) [3], [4]. These solutions are generally limited in two ways. First, all of the existing TRNGs make use of the physical properties that require a customized flash controller to gather low-level operating information such as operation time difference and bit-flip disturbs. Yet, the majority of COTS flash controllers do not expose such information to users, and would require extensive engineering undertaking to support it [5]. Second, many PUF solutions do not consider the aging effect in the flash devices. Experimental results demonstrate that both the program time and the erase time vary in a distinguishing way [6]. Consequently, a bad correlation coefficient can lead to false negative in the identification progress. As a result, the flash-based PUF is effective in the beginning, but the performance continuously degrades during the life cycle of the device.

In this paper, we propose Bit^{2}RNG, a design that requires no modification to commercial flash controllers, and therefore, it can be widely adopted by billions of flash-based devices. Bit^{2}RNG makes use of the bit error information that is widely available at the application level for randomness, and the bad page information for non-forgeability. As a result, besides its impact on system performance, aging also enables a non-forgeable identification, which is a necessary key property for many modern security primitives. Considering the practical TRNGs and identification requirements for COTS devices, any modification to flash controllers should be avoided. Therefore, instead of relying on obtaining low-level operation timing information that is unsupported by commodity flash memory, we turn our attention to existing high-level system information in COTS NAND flash chips that are intrinsically collected by the flash controller and made observable to the user level — bad pages and bit errors. Bad pages are due to chip manufacturing defects while bit errors are caused by a series of factors including the physical sensitivities and user operation intensity. Since both belong to uncontrollable and unpredictable faults, they are reported to the flash translation layer (FTL) or drivers with similar mapping functions through a flash controller [7]. In our work, we leverage these physical nonidealities of flash devices and leverage them for useful security primitives at no

additional design costs and without any hardware modification. Our proposed Bit^{2}RNG design is a practical true random number generator that uses a bad-page initialized table with bit-error insertion. The dynamically-updated bad page is used to initialize the seed to generate the random numbers, and the real-time progression of bit errors enhances the entropy of the TRNG output. With no additional hardware resources and simple software computation, Bit^{2}RNG guarantees a secure randomness source for lightweight embedded electronic devices. Besides, the evolution of bad pages in a flash system exhibits a delicate non-fungible characteristic that is sensitive to the Program/Erase (P/E) operations. Therefore, they provide important utility in terms of tampering awareness and content integrity, which is further explored in our work through the introduction of three applications—chip identification, image provenance, and secure boot. Our contributions are summarized as follows:

- To our knowledge, this is the first work that makes use of intrinsic high-level flash system features such as bad page and bit error to build a TRNG security primitive in lightweight devices.
- We follow the COTS flash controller standard to extract the physical property of NAND flash memories without adding extra hardware resources. We also perform comprehensive analysis and characterization of Bit^{2}RNG to experimentally demonstrate its practicality in commodity flash systems.
- We showcase several security applications of Bit^{2}RNG, each leveraging the unique strengths of the primitives, such as chip identification via aging estimation based on P/E cycles, tamper protection on multimedia via unforgeable randomness from flash bad pages. These use cases illustrate the versatility of our primitives combining aging guided nonforgeability and randomness. No measurable false positive or false negative is detected in the experiment, which is also theoretically proven.

II. Background and Preliminary Work

A. NAND Flash Cell

A floating gate memory cell is a type of metal-oxide-semiconductor field-effect transistor (MOSFET). The voltage adopted on the control gate and substrate affects the current flows in the middle layers. The floating gate is interposed between the control gate and substrate, surrounded by electrical-insulated oxide. As shown in Figure 1, information can be stored as the presence or absence of trapped charge on the floating gate by a quantum mechanical effect called Fowler-Nordheim tunneling.

As illustrated in the page program and block erase part of Figure 1, erasing the contents of a memory cell is done by placing high voltage on the silicon substrate while holding the control gate at zero. This is defined as a logic level ‘1’ in a single-level cell (SLC) flash memory cell. Relatively, the cell is programmed by placing a high voltage on the control gate while holding the source and drain regions at zero. This is regarded as a logic level ‘0’ in an SLC cell. With multi-level cell (MLC) flash memory, there are multiple programmed states. Typically, it allows two or more data bits to be stored in the memory cell. MLC memory cells tend to wear out faster than SLC memory cells because they are more sensitive to physical changes in the insulating oxide layer. MLC memory cells also experience higher levels of read errors due to variations in control gate threshold voltage and disturbances from neighboring memory cells.

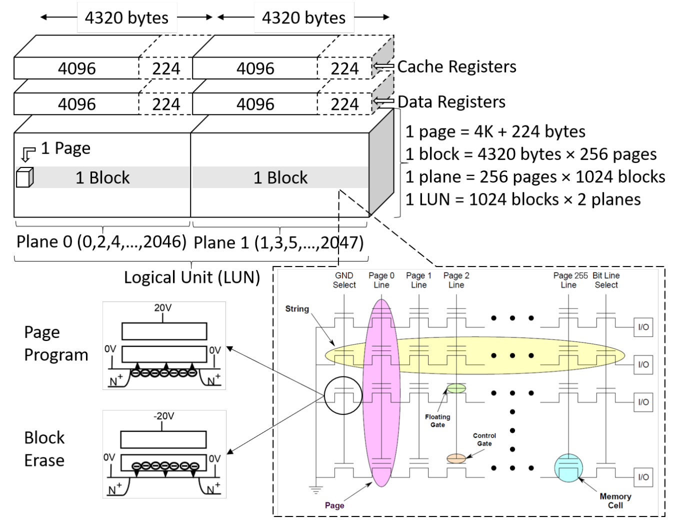

Fig. 1. Typical NAND flash array and cell structure.

B. NAND Flash Array

Flash memory cells are organized into a hierarchy of package, target, logical unit (LUN), plane, block, page, and column. For example, as illustrated in Figure 1, a 16 GB flash has a single logical unit inside a single target, which consists of two planes, divided into 1,024 even- and odd-numbered blocks. Each block contains 256 pages. A memory page contains 4,096 bytes of user data and 224 bytes of spare area. The additional area is used by the flash controller or FTL for data management such as error correcting code (ECC), bad block marking, and other system purposes.

The fundamental operations of flash memory include read, program, and erase. Because of the serial cell arrangement, it is not possible to directly access individual data bytes within a memory page. Both read and program operations have to be perfprmed at the page level. A simple read page operation takes the maximum time of 75 μs. Note that the latency of read operations is the only latency that is not significantly affected by the NAND flash lifecycle. Page program operations need a typical array latency of 1,300 μs in a new flash, and its performance degrades due to the aging effect of flash cells. Unlike a hard disk which can be overwritten directly in the same address, NAND flash requires a block erase operation before reprogramming the same page to avoid potential bit errors during the over-programming operations. An erase operation typically takes 3.8 ms or longer to finish setting

the whole block with value ‘1’. Since the erase operation is executed compulsorily before programming the existing values at the block level, it causes write amplification that merges useful data to other blocks before erasing. Consequently, this feature accelerates the aging of flash memory cells.

C. Bad Block

When failures occur during NAND flash device operation, they are categorized as either permanent or temporary failures. A permanent failure caused by a manufacturing defect or aging cannot be recovered while a temporary failure can be corrected. Most NAND flash devices include some initial bad blocks within the memory array during the manufacturing process. Factory-generated bad blocks are tested under worst-case conditions, and those blocks that fail this factory testing are marked as bad. To avoid erasing a factory-marked bad block and losing the information, every new chip is required to perform an initial read for bad blocks and create a bad-block table before issuing any program or erase commands. For example, in Micron MLC devices, any block where its first byte in the spare area of the last page does not contain “FFh” is a bad block. Since permanent bad blocks are unique to individual flash chips and located randomly, they can be regarded as fingerprints of the device.

NAND flash factories define endurance as their device lifetime. Over time, the tunnel oxide will lose its insulating properties, leading to the inability to erase or program a cell. For SLC flash, the typical endurance threshold is 100,000 P/E cycles. Most MLC chips, however, only behave reliably within 3,000 P/E cycles. An aged flash that is beyond the specification degrades by generating more bad blocks and finally wears out. Due to the different scale of the program operation and erase operation, we cannot avoid skipping any certain bad page when erasing the whole block. As this occurs, the flash controller will retire the entire block from use and replace it with a reserve block in the over-provisioning area. Alternatively, the deterioration of the increasing bad blocks is compensated for by using wear-leveling algorithms in many FTLs, which evenly spreads the P/E cycles over the entire blocks. Finally, to detect whether a block has worn out or not, one can issue a read status command after any erase or program operation. This command will report whether the previous operation is passed or failed.

D. Bit Error

Temporary failure occurs in multiple forms: data corruption by program/read disturb, data loss by retention issues, and over-programming. In the first case, bit error appears in random locations, which provides unpredictable features to be used in the proposed applications. The read/program disturb error occurs when one or more bits in cells are not intended to be charge disrupted during adjacent memory page read/program operations. NAND flash memory cells are more susceptible to disturbances from reading and programming neighboring pages due to the high density of memory cells of the silicon wafer. Moreover, the coupling may lead to random bit errors in the stored data and cannot be controlled artificially.

Data retention problem refers to flash cell value flipping due to charge gain or charge loss. High program/erase cycle counts can exacerbate the stressful conditions in the tunnel oxide by inducing trapped charge and effectively wearing out the gate oxide. To be specified, high field strength leads to structural changes in the oxide layer and creates defects that trap electrons. More structural defects behave like tiny “cracks” in the insulation that permit the charge on the floating gate to leak out into the substrate. The oxide layer breaks down over time and the stored value flips.

E. Related Work

True random number generation has attracted a significant amount of attention in the past decade [8]. Though it is possible to generate randomness from many chaotic physical processes such as thermal noise, voltage noise, and atmospheric noise, their low entropy fails to keep up with multiple security requirements. Instead, Pseudo Random Number Generator (PRNG) is developed to compensate for the lack of high-speed TRNG using the hardware noise as seed. Typically, software RNG produces cryptographic secure randomness by measuring physical events available in modern computers. The essential design is to maintain an “entropy pool” of random initial values that are assumed to be unknown to any attacker. Thus, the TRNG strategy is to maintain a stream cipher with a key and initialization vector obtained from an entropy pool. When enough bits of entropy have been collected, replace both the key and vector with new random values and decrease the estimated entropy remaining in the pool. It provides resistance against some attacks and conserves hard-to-obtain entropy. As shown in Figure 2 (a), the Linux OS transfers the entropy input to its kernel entropy pool where it is hashed and output through two interfaces. $/dev/random$ provides data with 100% entropy while $/dev/urandom$ serves for continuous output even if the pool has low entropy [9]. However, while this design is effective for desktop or server systems that constantly interact with a dynamic digital environment, this solution is often not effective for many embedded IoT devices, which do not have any of the digital randomness sources as in traditional desktop systems.

Device fingerprint is another application of NAND flash chips, which was first introduced in 2011 [10]. It evaluates the unique features extracted from program disturb, read disturb, and program operation latency. A followup work provides more experimental results to support the program latency based fingerprint and also discuss the TRNG using random telegraph noise [11]. Similar work regarding program time and erase effect based PUF application illustrates the characterization of the physical mechanism [4]. However, the existing works fail to take the device aging into consideration, which may lead to the obvious false-negative rate. This fact is also mentioned in the TRNG work using the read noise of flash cells [2]. Nonetheless, the claimed tolerance to aging and temperature effects cannot be applied to device identification

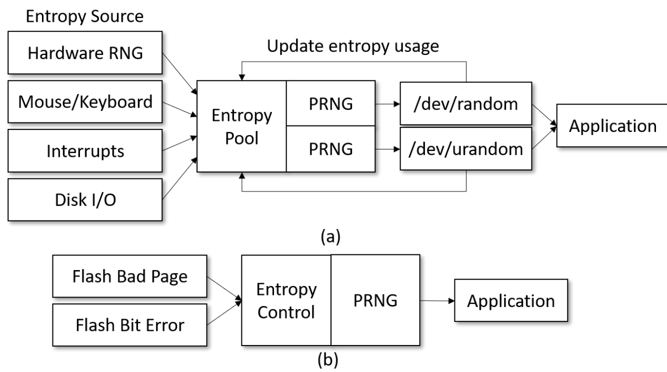

Fig. 2. (a) Typical entropy lifecycle in Linux; (b) Proposed bad-page and bit-error based entropy source and TRNG using NAND flash memory.

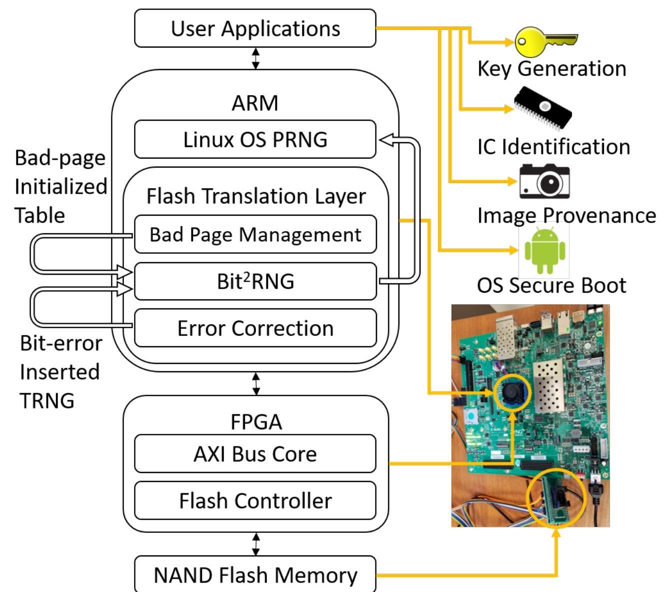

Fig. 3. Zynq SoC demo for Bit^{2}RNG design with random source, controller, FTL, and application layers.

due to the deterministic requirement. A NAND flash-based key generation solution uses bit-map and position-map methods to select reliable cells for robust outputs [3]. Considering the evaluation under various temperatures and aging conditions, a bit error rate under 10^{-6} proves the key generation application to be more realistic. Another relatively practical application is to detect recycled flash memories with the help of flash array parameters such as erase time, program time, and fail bit counts [12].

III. Bit^{2}RNG Design and Characterization

A. Bit^{2}RNG System Design

As illustrated in Figure 3, our Bit^{2}RNG design includes four layers: NAND flash memory, hardware memory control logic, software post-processing, and user applications. The bottom layer is the commercial NAND flash memory that stores system boot files, cryptographic keys, and user-generated data. While an electronic device is running regular programs that require access to the NAND flash memory, the chip notices bad-page and bit-error information to the upper layer. By doing so, it leaks the uniqueness and randomness in the meanwhile. Due to the special timing standard and mapping mechanism, users cannot read flash data from the OS directly. Instead, all the information should be collected by a hardware control layer first and then transferred to the address translation layer for post-processing. As illustrated in Figure 3, the memory layer and the hardware control layer are wire-connected. Usually, the control layer is an individual chip called the flash controller, but it can also be implemented in the FPGA. Commands from the software layer are converted to open NAND flash interface (ONFI) standard and executed in a pipeline. The flash status will be reported to the upper layer afterward. The third layer is known as the flash translation layer or flash file system, which maps the virtual address of a file system to the physical address of flash memory. FTL can be regarded as the flash driver of the Linux kernel as well. In our design, the FTL includes page-level mapping, optional wear leveling, optional garbage collection, bad block/page management, ECC, and the proposed TRNG module. Essentially, Bit^{2}RNG extracts a bad-page table from the bad block management module for the index data. It also obtains bit error messages from the ECC module before being corrected by Hamming code. With the time-stamped unique fingerprints and real-time random faults, Bit^{2}RNG is able to update the entropy dynamically. As shown in Figure 2 (b), Bit^{2}RNG sends the entropy to a software PRNG to generate high-quality random numbers. In the final layer, the operating system provides the interface of four user applications: cryptographic nonce, chip identification, image provenance, and secure boot. Those applications will either use the TRNG directly or apply part of the Bit^{2}RNG output. The details will be discussed in the later section.

B. Bad-page Initialized Table

Counting physical bad block numbers in a simple flash chip is insufficient for satisfying both the entropy and speed request of TRNGs. Instead, we choose the fine-grained units (bad pages) as the target for analysis, which can enlarge the entropy source by 64 to 256 times for a commercial flash chip. We notice the fact that in most flash bad block management strategies, the entire block can be marked as a bad block for reliability consideration, even if there is only one bad page detected and other pages are still available. Moreover, any malicious P/E operation cannot completely clone the features of a certain bad block. In other words, the bad page location, number, and the operating latency of other pages are randomly generated in the permanent locations. Given the fact that attackers measure the current chip status by physical tampering, the existing bad pages are unclonable in reality while the location of any new bad page is unpredictable. The possibility of cloning the entire chip can be estimated by the following equation:

$$P_{collision} = \prod_{j=1}^{b} \binom{p}{f_j(t)}^{-1}, p = 256, b = 2048 \quad (1)$$

Here, we set the page number p to 256 and block number b to 2,048. The failure page amount in block j at the P/E cycle t is defined as $f_j(t)$. Even for the unaged flash chip sample with one bad page over 10 blocks, the collision rate is 2^{-1640} on average, which is impractical to achieve.

Though initial bad block information can be directly obtained from the spare area of certain NAND flash pages, any new bad page or bad block has to be marked by the bad block management in FTLs. Measuring a failure of page or block operation requires the controller to issue a "Read Status" command to the current page or block, which is acceptable by the last-selected LUN even when it is busy. The 8-bit status register indicates the failure of operation if the LUN array and I/O keep busy all the time. Alternatively, one can also monitor the R/B signal of a NAND flash package to determine a bad page when it keeps low far beyond the typical operation latency. After the flash controller finishes collecting the status, the FTL can update the bad page table to generate new entropy for TRNGs.

To extract sufficient entropy from the bad-page table, we collect m groups of 8-bit page address and 11-bit block address information from the table. Ideally, the total bit length should be equal to 128 bits as an initial value, which means a minimum group number of 7 is required. If the bad pages are not enough, $14 - m$ regular pages will be selected to make up the rest of the bits. A PRNG will use the value to generate random numbers. Given I as the information content of X, the entropy is equal to the expected value of the random variable $I(X)$. For the probability mass function $P(X)$ and discrete random variable X, the entropy within m_{th} bad-page table update can explicitly be present as:

$$H(X) = E[I(X)] = \sum_{i=1}^{m} P(x_i) log_2 P(x_i) \quad (2)$$

For a RNG that only uses bad-page as the source of entropy, any binomial coefficient result has the even chance to be selected. Therefore, the probability function can be formulated as the following:

$$P(x_i) = \frac{\binom{k}{i}\binom{n-k}{k-i}}{\binom{n}{k}}, n = 256 \times 2048, k = \sum_{j=1}^{b} f_j(t) \quad (3)$$

Here n is the total page amount in the flash memory. k means the total number of bad pages in a chip, which is the sum value of each block f_j and can be affected by the P/E cycle t. According to the fitting curve of the experimental results in Figure 4, we can build the theoretical function $f_j(t)$ to estimate k:

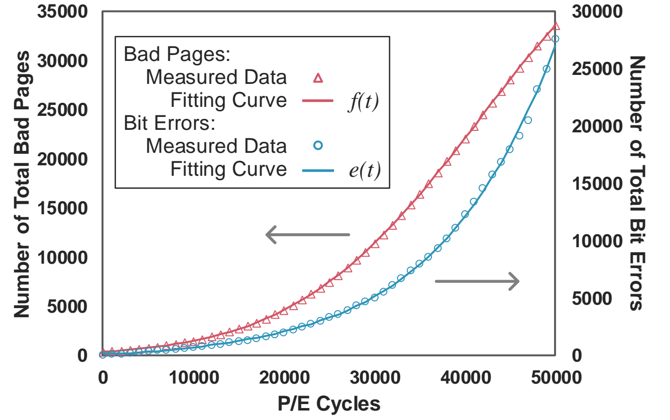

Fig. 4. Bad page and bit error cumulative quantity across different P/E cycles.

$$f_j(t) = 19.62 \times e^{-\frac{(t-6.21\times 10^4)^2}{8.17\times 10^8}} \quad (4)$$

C. Bit-error Inserted TRNG

Due to the flash endurance characteristic, bad pages are boosted after the threshold. However, the entropy pool updates slowly before that. To avoid the entropy pool drying out, we fetch bit errors from the error correction module and insert them to the initial value register of Bit^{2}RNG. We refresh the entropy intentionally in order to output 128-bit strings with a new entropy in time. As the bit error growth trend shown in Figure 4, we fit the function $e(t)$ based on the measured bit errors in each P/E cycle: $e(t) = 6.346 \times 10^{-15}t^4 - 2.466 \times 10^{-10}t^3 + 7.146 \times 10^{-6}t^2 + 0.00253t + 118.8$. By combining the bad pages and bit errors, the new entropy pool updating rate can be calculated as:

$$R(i) = \frac{(2^i - 1) \times e(t)}{\Delta t} \quad (5)$$

Here Δt is the P/E cycle count between two bad-page table updates, which can be one P/E cycle or several cycles. Note that the actual time spent during each P/E cycle is a nonlinear function $u(t)$ rather than a fixed value. According to our aging tests on Micron, Samsung, Hynix, and Toshiba flash memories, the erase latency always grows longer while program time becomes shorter. Since the entropy pool update rate is known, we can further compute the average throughput for a 128-bit TRNG:

$$TRNG\ Throughput = \frac{\sum_{i=1}^{m} R(i) \times 128}{u(t)} \quad (6)$$

Now we consider the mixed entropy $H'(X, E)$ in Bit^{2}RNG. Set $H'(X, E) = \sum_{i=1}^{m} P'(x_i) log_2 P'(x_i) = h_1 e_1 +, ..., +h_m e_{m'}$, in which $m \leqslant m'$. Meanwhile, the linear complexity of S defines the set of random sequence attempt from 1 to T. Note that the maximum number of trials

cannot exceed the current P/E cycles. Given a measurable s_t, breaking Bit^{2}RNG means to solve the equation:

$$F_t(h_1, ..., h_m, e_1, ..., e_{m'}) = s_t,\ t = 1, ..., T \quad (7)$$

s_t is the subset of S for the t_{th} attempt. The calculation of each stage is provided in term of linear function $L_t(p_1, ..., p_t)$ as follows:

$$\begin{aligned} s_1 &= a_1 p_1 = L_1(p_1) \\ s_2 &= a_2 p_1 \oplus a_1 p_2 = L_2(p_1, p_2) \\ s_3 &= a_3 p_1 \oplus a_2 p_2 \oplus a_1 p_3 = L_3(p_1, p_2, p_3) \\ &\vdots \\ s_t &= \oplus_{l=0}^{t-1} a_{t-l} p_{l+1} = L_t(p_1, p_2, ..., p_t) \end{aligned} \quad (8)$$

The coefficient a_i can be extracted from Equation 5. As long as the entropy pool is updated by bad pages or bit errors, a_t is unpredictable to the attacker. The TRNG must be resistant to exhaustive attacks on the entropy. With the entropy size of $2^m - 1$, the attack needs the knowledge of $2\Lambda_L(s) + 1$ bits, which refers to the linear complexity of Equation 8.

IV. Bit^{2}RNG Applications

Bit^{2}RNG leverages two distinct properties in commodity modern flash, unforgeable bad page and truly random bit error, to enable different security services with little impact on the performance and hardware design. In this section, four applications will be presented, each applying these two capabilities in different manners.

A. Cryptographic Nonce

A cryptographic nonce is a unique number adopted only once in a cryptographic protocol. It is often implemented using a random or pseudo-random number in a challenge and response protocol for replay attack prevention. They are also commonly applied as initialization vectors. A naive approach would be to simply use the bit error as the random number generator to produce nonce. However, it puts a strict requirement on the random number generation process from bit error to never produce the same number. One way to accomplish this is to generate a very large random number, given that true random number can be generated from the bit error, the probability of having the same number generated is very small. Depending on the targeted application, the length of the random number can be extended. One clear drawback of this approach is the computation-extensive random number extraction. Another approach to make nonce unique is to add time as a new dimension into the nonce generation process. However, for low power devices or devices in adverse environments, it is difficult to get a trustworthy time source, and this remains an open challenge in computer system. To tackle this problem, we can leverage the aging effect of flash as a notion of time, to add the temporal dimension into the nonce generation. Using the nonforgeability of bad pages and randomness of bit errors, Bit^{2}RNG is able to apply physical properties of a commodity flash chip to generate random unique nonce without additional hardware.

B. Chip Identification

Our work also contributes to the chip identification in the supply chain. For the COTS, any additional identifier like Radio-frequency identification (RFID) is not welcome due to the extra cost. Bit^{2}RNG, which simply needs the existing NAND flash memory in the embedded systems without any modification to the hardware, can be an ideal solution against the memory chip counterfeiting. By utilizing bad-page based random process variables, our method is supposed to exhibit two characteristics to work in a production environment: First, the chip can generate unique and unclonable digital ID; Second, the ID code must be repeatable correctly over the temperature, aging, and other noise. To be specific, we read the spare area of new flash chips to locate the initial bad blocks. The next step is to test all the pages in the bad blocks and determine which are the bad pages. Then we use SHA-256 to hash the entire bad-page location information and sign it with the manufacture private key, storing the signature in the One-Time Program (OTP) area, which is 10 pages large and is left for users to program as they desire. Once it is programmed, the content can be hardly modified without physical tampering. Therefore, OTP uniquely verifies the physical property of the chip, thus proving the authenticity of the component. Meanwhile, we notice the fact that even though the block level damage is clonable on other NAND flash chip by large amounts of P/E operations, attackers cannot regenerate the same variables at the page level. Both bad page number and location are not controllable on another chip. In the worst case that no bad block is detected for a new flash chip, the owner can still create a few unique bad pages by themselves easily. Since the FTL avoids operating bad blocks, there will not be any false negative. The false-positive rate can be calculated as:

$$P_{false\ positive} = max\{\frac{1}{\sum_{i=1}^{t}\binom{n}{k_i}}, \frac{1}{2^{128}}\} \quad (9)$$

Here k_i is the bad page amount in the i_{th} bad block and t is the total number of bad blocks. Thus, the false positive rate depends on the larger possibility between a successful bad page clone and a collision of SHA-256 in OTP.

C. Image Provenance

Another important application of the unforgeable property of Bit^{2}RNG is image provenance with the goal of proving the authenticity of an image or video. With the recent advancements in artificial intelligence, automatically constructing a realistic fake image or video of a subject is a capability that is ubiquitous to different users with various level of computing literacy. One way to defend this is to provide unforgeable binding that ties media to the device it was taken, which is the SD card of the recording device in this case. Note that

the application focuses on the areas that the image sensor and storage are built as a single system, where additional confidence in binding the images to a specific flash is needed.

To take a picture, the camera will first save the image into the SD card, reads the bad page information for the blocks the image is stored in, and publish the hash result of bad page information and image to non-mutable storage such as blockchain or trusted website. This published hash is often considered as the commitment. To detect any modified data, one can reproduce the hash value with the current data in the flash memory plus the pseudo serial number regenerated from the bad page. The new hash value should match the commitment in the non-mutable OTP area. If attackers copy the raw data to a different device, the unclonable bad page information will provide different serial numbers. The attacker can also attempt to modify the original image on the device, however, that will result in a change of the bad page information. As a result, the newly calculated hash value will not match the commitment, and the tampering can be detected.

D. Secure Boot

Secure boot traditionally focuses on the process of extending trust at each of the load and boot stage beginning from the root of trust. The goal is to bootstrap trust from the root to gain confidence on integrity of the booted system. Another common goal early in the secure boot process is for the software to understand and gain confidence about the environment it is executing on. When the hardware platform is not as expected, software can halt further loading process, preventing the adversary from analyzing the behavior of the system. It is also possible to take advantage of the non-forgeablity of the bad page to bind the execution of the software onto a device, enabling the software to uniquely identify the platform it is executing on. By binding the software to the platform, it is possible to prevent adversarial reverse engineering, which is an essential initial step towards further software exploitation. The idea is similar to image provenance in that software written to the flash will verify the identity of the platform by reading the bad page information from the flash itself. If the authenticity of the platform cannot be verified, the booting process will halt including the decryption of the software used for further booting stages.

V. Experimental Results and Evaluation

A. Testbed and Environmental Setup

Our flash testbed includes three objects: NAND flash socket, flash controller on the FPGA, and optional embedded FTL with application modules. To get access to the flash chip interface, we prepare different sockets to access the pins of flash packages: SOP8 clip, TSOP48 socket, and BGA63 socket, as shown in Figure 5. A selected socket is wire-connected to the Pmod interface of either Diligent Nexys 2 Board or Xilinx ZCU102 platform [13], [14]. We use the Spartan-3E FPGA on Nexys 2 board when directly dumping the stored data to PC for further analysis. The Zynq SoC on ZCU102 with embedded Linux OS is selected to evaluate

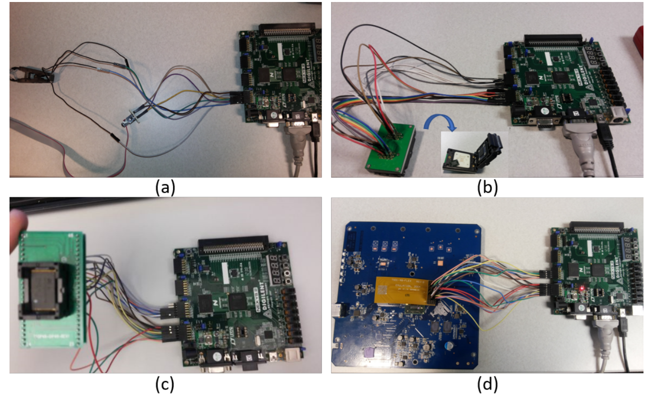

Fig. 5. Nexys 2 Testbed for various types of flash memory packages: (a) SOP8; (b) BGA63; (c) TSOP48; (d) TSOP48 without desoldering.

the user application performance. In either case, the FPGA is programmed to perform as a flash controller, which translates high-level commands to fundamental operations such as $Read\ ID$, $Read\ Page$, $Program\ Page$, $Erase\ Block$, and $Read\ Status$. The controller issues commands, addresses, and data according to some timing constraints in the ONFI 2.0 standard [15]. The flash controller communicates with a computer through the external UART port or with the processor core of Zynq SoC through internal AXI bus to obtain system instructions and exchange data. For different applications, the OS invokes Bit^{2}RNG to provide the required random number or fingerprint. All the tests are progressed under normal temperature and standard supply voltage.

B. Randomness

To prove the randomness of Bit^{2}RNG can serve as the source of entropy, we first calculate the intra-chip Jaccard index of bad pages and bit errors to evaluate their occurrence distribution. The Jaccard index is to gauge the similarity of two sets by comparing the intersection and union, and is written as [16]:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \tag{10}$$

where $0 \leq J(A,B) \leq 1$. Unlike the Hamming distance that only works for the statistic results on one dimension, the Jaccard index provides a more straightforward view of randomness under different P/E cycles. We evaluate the randomness of bad page locations on the same chip, as the blue curve shows in Figure 6 (a). In this figure, we use a dataset measured from five flash chips to present the range of the Jaccard index as the error bar and calculate the absolute error. For the randomness of intra-chip bad page, A is defined as the increased bad pages in the previous 1,000 P/E cycles, noted as $BP(N-1000)$ where N is the number of P/E cycles. B is the new bad pages over the last 2,000 P/E cycles, noted as $BP(N-2000)$. Before the endurance threshold of 3,000 P/E cycles, new bad pages are rarely generated. The initial bad pages thus take the majority, which lead to a relatively high Jaccard index. Additionally, the five chips contain different

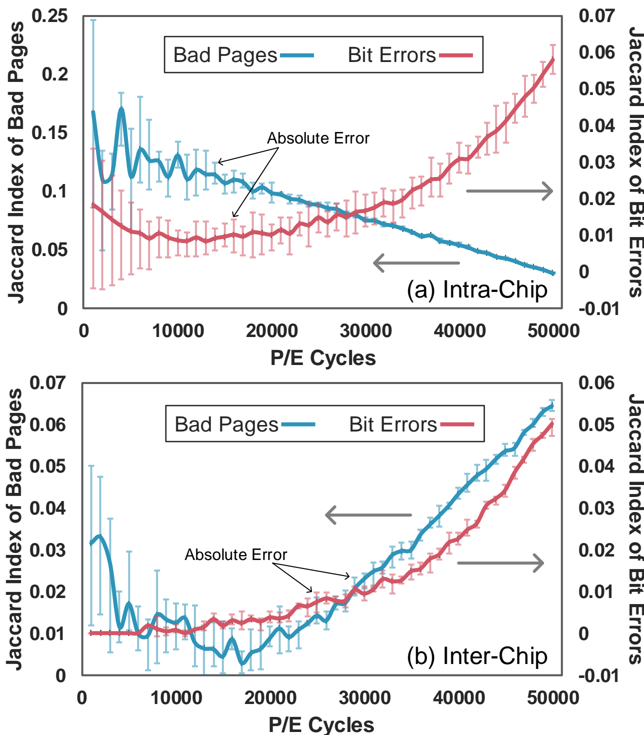

Fig. 6. Jaccard index of bad pages and bit errors from (a) intra-chip P/E experiment; (b) inter-chip P/E experiment of five NAND flash memories.

initial bad pages. Hence, the maximum and minimum values before the first 10,000 P/E cycles vary obviously. As flash cells wear out by P/E operations, the new bad pages become more significant in the fraction and make the Jaccard index decrease. Thus, we can use the bad page as an entropy source when the Jaccard index decreases after 3,000 P/E cycles.

According to our tests, the bit errors do not distribute evenly on different pages, which is illustrated in Figure 7. The lower pages in a block suffer less from errors than those higher pages. Even for the adjacent pages with higher page addresses, the reliability may vary significantly. As a result, a few pages with the high index obtain a much bigger possibility of failure bits compared to the adjacent ones, which affects the randomness to some degree.

Nevertheless, there are thousands of groups in flash memory with the same page number. The Jaccard index of intra-chip bit errors is shown as the red line in Figure 6 (a). Similar to the bad page in the same figure, five chips are tested to provide the possible range and the absolute error. For each chip, the overlapped chance between every 1,000 P/E cycles is lower than 6% regardless of the uneven bit error distribution in the high P/E cycles intersection, which means more than 94% bit error locations are not overlapped for any flash memory sample. Hence, the bit error location is proved to be a randomness source as well.

After demonstrating the randomness of Bit^{2}RNG entropy sources, we further evaluate the TRNG output by applying 15 NIST randomness tests. The p-values and proportions are

TABLE I
RESULTS OF NIST RANDOMNESS TESTS

Statistical Test	P-value	Proportion
Frequency	0.739918	5/5
Block Frequency	0.911413	5/5
Cumulative Sums	0.911413	5/5
Runs	0.122325	5/5
Longest Run	0.739918	5/5
Rank	0.437274	5/5
FFT	0.534146	5/5
Non-overlapping Template	0.066882	5/5
Overlapping Template	0.122325	5/5
Universal	0.017912	5/5
Approximate Entropy	0.911413	5/5
Random Excursions	—-	1/1
Random Excursions Variant	—-	1/1
Serial	0.350485	5/5
Linear Complexity	0.122325	5/5

listed in Table I. As the p-values are all larger than 0.01, we accept the TRNG output as random [17]. Though we use a PRNG to generate the random numbers, the proportions show that all five flash memories pass the tests. Note that two of the long results are not shown due to space constraints.

C. Uniqueness

Besides the randomness, the uniqueness among different chips is another important factor to determine the security level of random numbers. If any bad page exists, we mark the location as '1'. Otherwise, it is '0'. By comparing the overlapped amount of '1's among total bad pages, a lower ratio indicates more difficulty of regenerating the same entropy from another chip. In Figure 6 (b), the blue line stands for the Jaccard index of inter-chip bad pages by comparing the bad page table of one flash memory with the other four chips. Unlike the intra-chip case of bad pages, this Jaccard index reaches the lowest value between 10,000 and 20,000 P/E cycle, then increases sharply after exceeding the lifecycle of MLC flash. Even for the worst case, a ratio under 6% is sufficient to distinguish the entropy generated from two chips.

Apart from bad pages, bit errors also show uniqueness among multiple flash chips. While the higher frequency of error locations on some page numbers affects the intra-chip

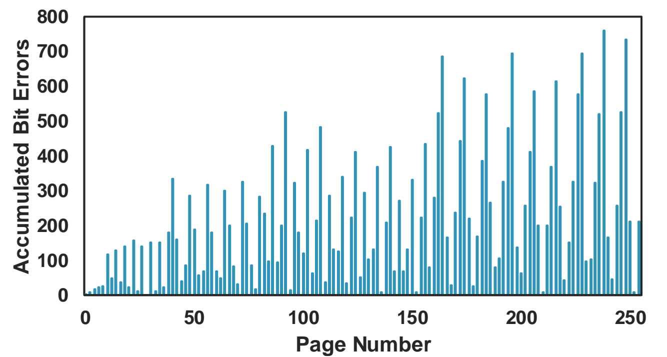

Fig. 7. Bit error distribution on different page numbers.

TABLE II
CHIP IDENTIFICATION SOLUTIONS COMPARISON

Performance Comparison	**Chip Identification Solutions**					
	RFID [18]	***PUF*** [19]	***DRAM*** [20]	***Flash*** t_{prog} [11]	***Flash mixed*** [10]	***Bit²RNG***
Additional HW	RFID	RO PUF	—-	Controller*	Controller*	—-
Removability	Possible	Hard	Possible	Possible	Possible	Very hard
False positive	$0.07\% \sim 0.28\%$	$0.06\% \sim 0.09\%$	$0.03\% \sim 0.081\%$	10^{-4} (aged)	$4.00\% \sim 9.00\%$	$1/2^{1640} \sim 1/2^{128}$
False negative	$1.71\% \sim 24.38\%$	$0.01\% \sim 0.11\%$	$0 \sim 29.3\%$	7.01×10^{-7}	N/A	0/5

randomness in Figure 7 and Figure 6 (a), the inter-chip uniqueness has smaller Jaccord index due to the physical isolation, according to the red line in Figure 6 (b). The bit error location collision rate within the first 10,000 P/E cycles keeps flat but rises significantly after 30,000 P/E cycles due to the increasing bit errors.

Note that we have verified the randomness and uniqueness on more flash chips from Micron, Samsung, Hynix, and Toshiba. According to the study on temperature effect [21], the total charge loss of the voltage threshold ΔV_t caused by random noises is almost independent of the experimental temperature within the standard operating temperature range. The offset of bit error distributions is nearly undetectable for attackers to control the randomness and uniqueness with certain patterns and predict the output accurately.

D. Cryptographic Nonce Performance

We compare three types of hardware-based TRNGs in Table III. The first column shows a DRAM TRNG solution using the inherent DRAM latency variations [22]. A recent study shows that the throughput of DRAM-based TRNGs is higher than flash-based TRNGs [24], but the approximate entropy is the lowest on the list. On the other hand, many lightweight embedded devices might not even contain a DRAM on the PCB. For systems with sufficient resources, a trusted platform module (TPM) could be a better choice [25]. Another work proposes the PUF based TRNG with high quality entropy [23]. Its output speed is more than 10 times faster than others. However, the extra resource of PUF itself is a concern for low-cost and low-power devices. We also list one existing flash memory TRNG in the next column [11]. As is claimed previously, flash-based devices such as SD cards are one of the most popular non-volatile memories for lightweight embedded systems and IoT devices. Though its entropy has been improved, the main defects are the low throughput and controller customization. To address those concerns, we implement and test the proposed Bit²RNG-based nonce and key generator on Xilinx ZCU102 platform, which liberate the hardware constraints by extracting randomness from the bad page table and bit error management in software layers. Within the NAND flash chip lifecycle, Bit²RNG provides adequate entropy from either bad pages or bit errors. Considering cost, entropy, and throughput, our solution provides desirable and balanced performance compared with other approaches.

TABLE III
CRYPTOGRAPHIC NONCE SOLUTIONS COMPARISON

Performance Comparison	**Cryptographic Nonce Solutions**			
	DRAM [22]	***PUF*** [23]	***Flash*** [11]	***Bit²RNG***
Additional HW	—-	RO PUF	Controller*	—-
Entropy	0.350485	0.637119	0.534146	0.911413
Length	512 bit	1 bit	1 bit	128 bit
Randomness	Pass	Pass	Pass	Pass
Throughput	0.47 Mbps	12.50 Mbps	1.27 Kbps	0.37 Mbps

E. Chip Identification Performance

As a well-known object identification solution, the RFID method is first list in Table II for comparison [18]. For many scenarios, RFIDs are selected for the cost reason, which can be easily removed from the target. Incorrect identification may occur due to the poor robustness and weak resistance against collisions. Another chip identification is based on Ring Oscillator PUF or Arbiter PUF [19]. Though Ring Oscillator PUFs and Arbiter PUFs take additional gates to implement, they are relatively small and can be embedded into a chip. In this case, it is hard to remove from devices. As an indeterministic primitive, the false positive and false negative still cannot be avoided after post-processing. The DRAM PUF solution does not require extra resources, but the authentication accuracy varies on different chips [20]. One flash-based TRNG and device identification work takes the program latency as the authentication benchmark [11]. However, there are two obvious limitations: the special requirement of reporting programming time to the flash controller and the sharp drop of true positive when the flash cells wear out. Hence, general flash controllers cannot support such a chip identification strategy. Moreover, users can no longer recognize the fingerprints when the flash is aged to some degree. Another work evaluates the program disturb, read disturb, program latency, erase latency, and other characteristics to extract unique signatures from flash memories [10]. But resource cost, security, and accuracy remain the major concerns to be address. The proposed Bit²RNG requires no hardware security primitives for any flash memory embedded device. It also supports commercial controllers without specific latency operations. Featuring OTP, it is impossible to overwrite the existing data without physical tamping. Hence, the stored fingerprint is robust and secure. Since a flash chip keeps permanent initial bad page locations, the generated serial number and hash output are repeatable. The measured false negative rate is a constant '0' across different chips. For the false positive, finding a hash collision within the polynomial-time is difficult in reality. The attackers

TABLE IV
IMAGE PROVENANCE SOLUTIONS COMPARISON

Performance Comparison	**Image Provenance Solutions**		
	PRNU [26]	***CFA*** [27]	***Bit^{2}RNG***
HW source	Camera sensor	Camera sensor	Flash memory
False positive	2.40×10^{-5}	3.57×10^{-2}	$\frac{1}{2^{1640}} \sim \frac{1}{2^{256}}$
False negative	2.40×10^{-2}	5.36×10^{-2}	0/5*

TABLE V
SECURE BOOT SOLUTIONS COMPARISON

Performance Comparison	**Secure Boot Solutions**		
	LEON3 [28]	***RISC-V*** [29]	***Bit^{2}RNG***
Used slices	4,078	6,793	1,093
Boot memory	Flash	ROM	Flash
Security HW	HSM	CAU	Flash

need as many as 2^{128} trials to obtain the 128-bit seed. Alternatively, reproducing the same bad page locations turns out to be impractical due to the physical uncontrollability. The false positive rate could be even lower than the hash collision rate, as demonstrated in Equation 1.

F. Image Provenance Performance

In this application, we list three camera-based image provenance solutions that leverage the hardware components such as camera sensors and memories. One provenance work with large scale tests is the camera image identification based on sensor photo-response non-uniformity (PRNU) [26]. The false positive shown in Table IV is decent while the false negative may lead to 24 failures among 100 images. Meanwhile, the clonability of noise sources is a security concern if the camera is accessible to anyone. Another research identifies the camera source of a digital image based on traces of color interpolation in the RGB color channel [27]. Though the method is universal among different cameras, it limits images that are not heavily compressed as such artifacts suppress and remove the spatial correlation between pixels due to CFA interpolation. Even the best results selected in the second column still provide relatively low identification accuracy. To optimize these solutions, we test a secure and accurate approach for user data that are intended to be kept unchanged on a specific flash chip. By hashing a raw image along with the serial number generated from the bad page part of Bit^{2}RNG, we program the value into the OTP area of five new NAND flash memories and publish the initial bad pages as a key. Whoever receives the key can verify the integrity of an image by dumping out the current image, verifying the key with the bad pages, hashing both data, and comparing the value with the OTP content on the same flash chip. Note that the image read from a flash memory may contain bit errors and lead to the verification failure. Thus, the false negative rate does exist given no ECC or aged chip scenarios. Apart from the user data, the robustness of OTP area and initial bad pages are verified on the same group of chips, which require no ECC. To test a collision of the stored hash value, we read the original data from NAND flash and flip a single bit in the image, but fail to generate the equivalent output. Similarly, the lower bound of the false positive rate is the minimum value of either the SHA-256 collision rate or the probability of cloning the entire bad page information.

G. Secure Boot Performance

Three secure boot approaches are list in Table V. The first related work comparison is a trusted boot FPGA implementation for embedded systems [28]. The processor LEON3 acts as a hardware security module (HSM) to run trusted computing firmware. However, the flash memories used in the design have no protection against malicious tampering. Even if the HSM is secure, the system still suffers from multiple vulnerabilities at the memory level. The other example introduces a lightweight RISC-V based secure boot with a centralized code authentication unit (CAU), including typical implementations such as Elliptic Curve Digital Signature Algorithm, Secure Hash Algorithm 3 (SHA3), PUF, direct memory access (DMA) and other modules. The CAU is connected to the Boot ROM for security firmware considerations. However, the entire design contradicts the claimed lightweight considerations. Ours is the first work that proposes commercial NAND flash-based secure boot for lightweight devices. To simplify the problem, we utilize the OTP area in the flash memory to authenticate the boot information. The boot file, bad page information, and hash value will be read together from the flash memory. If $hash(boot\ file + bad\ page)$ matches the OTP string, the boot file is authorized for that flash chip. To make the resource comparison, we implement the flash-based system boot on ZCU102 platform. Since this solution does not require any extra secure hardware such as TPM, this resource utilization is much smaller than other designs.

VI. CONCLUSION

In this work, we present the design and implementation of Bit^{2}RNG, a NAND flash-based true random number generator and chip fingerprint. Unlike previous work that uses the unstable low-level operation latency information, we exploit the combination of unique characteristics of the bad page and bit error, which are high-level system features readily available in commodity flash devices. The intra-chip and inter-chip measurements prove the randomness and uniqueness of the proposed system. Furthermore, four important security applications in lightweight resource-constrained embedded devices have been demonstrated experimentally: cryptographic nonce, chip identification, image provenance, and secure boot. Results show that, by combining bad pages and bit errors, Bit^{2}RNG provides sufficient entropy and decent throughput. The chip identification solution shows deterministic results compared to other hardware primitives. The flash-based image provenance has distinct advantages in terms of resource usage and accuracy. Finally, the OTP code authentication method guarantees the binding of software to the hardware system.

ACKNOWLEDGMENT

This work was supported in part by US National Science Foundation under grants CNS-1916926.

REFERENCES

[1] P. Cappelletti, C. Golla, P. Olivo, and E. Zanoni, *Flash memories*. Springer Science & Business Media, 2013.

[2] B. Ray and A. Milenković, "True random number generation using read noise of flash memory cells," *IEEE Transactions on Electron Devices*, vol. 65, no. 3, pp. 963–969, 2018.

[3] S. Jia, L. Xia, Z. Wang, J. Lin, G. Zhang, and Y. Ji, "Extracting robust keys from NAND flash physical unclonable functions," in *International Conference on Information Security*, pp. 437–454, Springer, 2015.

[4] S. Q. Xu, W.-k. Yu, G. E. Suh, and E. C. Kan, "Understanding sources of variations in flash memory for physical unclonable functions," in *2014 IEEE 6th International Memory Workshop (IMW)*, pp. 1–4, IEEE, 2014.

[5] Micron, *NAND Flash Controller on Spartan3*, 7 2014. Rev. C.

[6] S. Sakib, P. Kumari, B. Talukder, M. Rahman, and B. Ray, "Non-invasive detection method for recycled flash memory using timing characteristics," *Cryptography*, vol. 2, no. 3, p. 17, 2018.

[7] A. Gupta, Y. Kim, and B. Urgaonkar, "Dftl: A flash translation layer employing demand-based selective caching of page-level address mappings," *SIGPLAN Not.*, vol. 44, p. 22940, Mar. 2009.

[8] F. Tehranipoor, W. Yan, and J. A. Chandy, "Robust hardware true random number generators using DRAM remanence effects," in *2016 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pp. 79–84, IEEE, 2016.

[9] Z. Gutterman, B. Pinkas, and T. Reinman, "Analysis of the Linux random number generator," in *2006 IEEE Symposium on Security and Privacy (S&P'06)*, pp. 15–pp, IEEE, 2006.

[10] P. Prabhu, A. Akel, L. M. Grupp, S. Y. Wing-Kei, G. E. Suh, E. Kan, and S. Swanson, "Extracting device fingerprints from flash memory by exploiting physical variations," in *International Conference on Trust and Trustworthy Computing*, pp. 188–201, Springer, 2011.

[11] Y. Wang, W.-k. Yu, S. Wu, G. Malysa, G. E. Suh, and E. C. Kan, "Flash memory for ubiquitous hardware security functions: True random number generation and device fingerprints," in *2012 IEEE Symposium on Security and Privacy*, pp. 33–47, IEEE, 2012.

[12] P. Kumari, B. B. Talukder, S. Sakib, B. Ray, and M. T. Rahman, "Independent detection of recycled flash memory: Challenges and solutions," in *2018 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pp. 89–95, IEEE, 2018.

[13] Xilinx, *Frequency Counter for Spartan-3E Starter Kit*, 3 2006. Rev. 2014.1.

[14] Xilinx, *ZCU102 Evaluation Board User Guide*, 6 2019. Rev. 1.6.

[15] Intel, *Open NAND Flash Interface Specification*, 2 2008. Rev. 2.0.

[16] L. Hamers *et al.*, "Similarity measures in scientometric research: The Jaccard index versus Salton's cosine formula.," *Information Processing and Management*, vol. 25, no. 3, pp. 315–18, 1989.

[17] National Institute of Standards and Technology, *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*, 4 2010. Rev. 1a.

[18] C. Bertoncini, K. Rudd, B. Nousain, and M. Hinders, "Wavelet fingerprinting of radio-frequency identification (RFID) tags," *IEEE Transactions on Industrial Electronics*, vol. 59, no. 12, pp. 4843–4850, 2011.

[19] W. Yan, F. Tehranipoor, and J. A. Chandy, "PUF-based fuzzy authentication without error correcting codes," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 9, pp. 1445–1457, 2016.

[20] F. Tehranipoor, N. Karimian, W. Yan, and J. A. Chandy, "DRAM-based intrinsic physically unclonable functions for system-level security and authentication," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 3, pp. 1085–1097, 2016.

[21] D. Resnati, A. Goda, G. Nicosia, C. Miccoli, A. S. Spinelli, and C. M. Compagnoni, "Temperature effects in NAND flash memories: A comparison between 2-d and 3-d arrays," *IEEE Electron Device Letters*, vol. 38, no. 4, pp. 461–464, 2017.

[22] B. B. Talukder, J. Kerns, B. Ray, T. Morris, and M. T. Rahman, "Exploiting DRAM latency variations for generating true random numbers," in *2019 IEEE International Conference on Consumer Electronics (ICCE)*, pp. 1–6, IEEE, 2019.

[23] W. Yan, C. Jin, F. Tehranipoor, and J. A. Chandy, "Phase calibrated ring oscillator PUF design and implementation on FPGAs," in *2017 27th International Conference on Field Programmable Logic and Applications (FPL)*, pp. 1–8, IEEE, 2017.

[24] J. S. Kim, M. Patel, H. Hassan, L. Orosa, and O. Mutlu, "D-range: Using commodity DRAM devices to generate true random numbers with low latency and high throughput," in *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 582–595, IEEE, 2019.

[25] S. Bajikar, "Trusted platform module (TPM) based security on notebook PCs," white paper, Mobile Platforms Group, Intel Corporation, 2002.

[26] M. Goljan, J. Fridrich, and T. Filler, "Large scale test of sensor fingerprint camera identification," in *Media forensics and security*, vol. 7254, p. 72540I, International Society for Optics and Photonics, 2009.

[27] S. Bayram, H. Sencar, N. Memon, and I. Avcibas, "Source camera identification based on CFA interpolation," in *IEEE International Conference on Image Processing 2005*, vol. 3, pp. III–69, IEEE, 2005.

[28] O. Khalid, C. Rolfes, and A. Ibing, "On implementing trusted boot for embedded systems," in *2013 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pp. 75–80, IEEE, 2013.

[29] J. Haj-Yahya, M. M. Wong, V. Pudi, S. Bhasin, and A. Chattopadhyay, "Lightweight secure-boot architecture for RISC-V system-on-chip," in *20th International Symposium on Quality Electronic Design (ISQED)*, pp. 216–223, IEEE, 2019.