Integrated Circuit Identification and True Random Numbers using 1.5-Transistor Flash Memory

Lawrence T. Clark, James Adams and Keith E. Holbert
School of Electrical, Computer, and Energy Engineering, Arizona State University
{Lawrence.clark,keith.holbert}@asu.edu; jameswilliamsadams@gmail.com

Abstract

Flash memory bits, like other integrated circuit (IC) devices, are prone to random variability in their actual vs. nominal characteristics. We present the use of 1.5-T flash memory cells in physically unclonable functions (PUFs) leveraging their erase speed variability. This type of memory is interesting for the internet of things (IOT) due to its wide availability as IP at foundries. Using experimentally measured results, we show simple methods that provide high reliability with no or limited need for helper data and error correction. High quality fingerprints for IC identification are demonstrated. Moreover, techniques to remove systematic variations from the array response are shown, allowing the resulting binary strings to pass all National Institute of Standards and Technology tests for randomness. Consequently, with low complexity helper functions, true random numbers can be readily produced.

Keywords

True random number generation, physically unclonable functions, flash memory, systematic mismatch.

1. Introduction

Using physical variations in integrated circuit (IC) constituent components, such as transistors, is an increasingly popular technique for IC identification and generating true random numbers. The latter circuits are known as true random number generators (TRNGs). Since the resulting random numbers are based on the as-fabricated characteristics of a specific IC, any two will have different random fabrication process induced variations. Thus, they are unclonable—the resulting random values cannot be guessed. Hence, the generating circuits are also known as physically unclonable functions (PUFs) [9]. Given a sufficient number of bits, two ICs are distinguishable based on the resulting fingerprint (ID) which differs for each IC.

1.1. Physically unclonable functions

Figure 1 shows a PUF enclosed in a helper algorithm logic. Initially, the PUF produces a response RE without or with a partial input code. This is obfuscated or otherwise combined with the partial input code to produce a full output code for later use. In an interrogation or check for a specific device, the input code is used to interrogate the PUF. After the helper function performs error detection and correction (EDAC) and compares the challenge CH to the PUF response RE, an output is provided. Since helper functions may include correction bits, etc., they can leak information out of the PUF and thus be used in attacks [4][12][16]. Alternatively, because the PUF should exhibit near perfect randomness, it can be used as a TRNG, often then used to produce high quality seeds for pseudo-random number generators or keys for encryption codes.

Delay racing (arbiter) based PUFs have been extensively analyzed and generate one bit at a time based on the outcome of racing paths through multiplexer paths selected by the input code or its result through a helper function [4][16].

Memories can also be used as a PUF, based on a number of characteristics [2][6][7][11][13][15]. Holcomb, *et al.* used SRAM power-up state [6]. Chellappa, *et al*. showed improved circuits that work by destabilizing continuously powered SRAM cells [2]. DRAM was effective as a PUF in [15]. Here, the authors showed that the rate at which cells failed when not refreshed could be used to produce a random IC fingerprint. An important issue with memories is systematic mismatch, which was shown to affect both of these designs. It can derive from poly misalignment in older technologies, and threshold implant screening in more modern SRAM technologies. Layout, e.g., location of word line tap cells, can also affect the nominal local matching.

Wang, *et al*. used standard flash memory to produce random strings based on the variability of the flash cells [21]. Specifically, they purposely programmed cells to a point where the cell could flip either way based on whether or not trapping, in the form of random telegraph noise (RTN), caused the cell to flip either way when sensed. A string is then generated based on those bits that are found to be unstable. They also suggest fingerprints based on the order that the cells program, when programming is interrupted.

1.2. Helper functions and reliability

One significant problem with PUFs is reliability and repeatability. In the latter, the same sequence may not be produced by the PUF for the same input in all cases. For instance, when delay or mismatch values are very close, the result is a function of noise rather than the mismatch. These errors can require that error detection and correction (EDAC) be applied to the result so that it is consistent (Figure 1). The former can manifest as a time-varying response, for instance due to circuit aging changing circuit delays.

Memory based PUFs are referred to as weak [3][20], while delay based, e.g., arbiter, PUFs are considered strong.

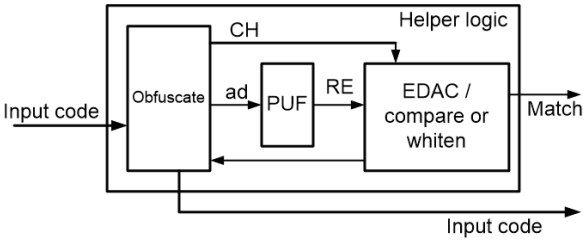

Figure 1: A PUF enclosed in the helper data algorithm (function) that protects access to the PUF and corrects unreliable bits, responding to the challenge with a match signal and otherwise obfuscating the actual PUF access.

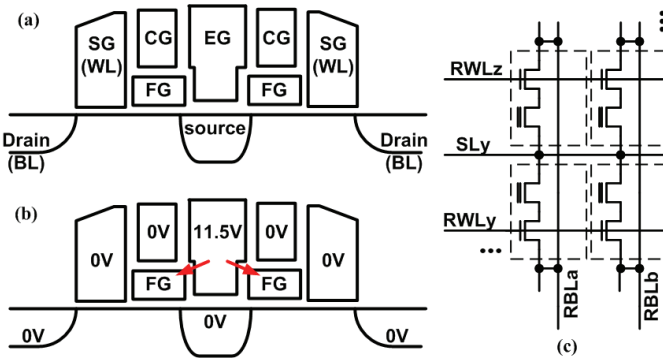

Figure 2: 1.5-T cell cross-section (a) and normal erase voltages (b). Circuit configuration (c) where NOR bit lines are perpendicular, while the erase gate and source line are parallel to the WL. The memory may be erased in small sections of 256 to 1k bits.

This is not due to one having greater variability or reliability, but rather derives from the ability to completely read a memory because it has finite locations. Nonetheless, memory based PUFs are commercially used and available as embedded blocks. In contrast, trying all combinations of an arbiter based PUF can be impossible in finite time. However, machine learning algorithms have been shown effective at determining underlying delay PUF values and are aided when some bits are not reliable, which is common [16].

Consequently, either type of PUF is usually protected from interrogations by a circuit that obfuscates the actual from the applied interrogations. These are commonly referred to as helper data algorithms [4][10] and help ensure meeting required reproducibility, entropy, and control specifications. The first is accomplished by temporal replication or EDAC to allow or remove inconsistent bits, respectively. The entropy requirement must be primarily met by the physical mechanism controlling the PUF. The control function generally includes shielding the raw PUF data from the user to mitigate attacks. In the memory case, it can for instance function to disallow continuous read out, an essential feature for a memory based PUF.

When using a PUF for authentication, the response of the same IC to the same challenge, e.g., accessing the same physical characteristics, is an intra-class access and should produce an identical (or identical post-helper corrected) response. Conversely, a challenge to a different location (inter-class) or the same challenge to a different IC should produce a different response. The most common method of determining response similarity is the Hamming distance.

1.3. PUF entropy and TRNG quality

The PUF randomness properties are generally characterized by the min-entropy, which is defined for a binary source stream i as

$$H_{\min} = -\log_2\left(\max\left(p_0^i, p_1^i\right)\right), \quad (1)$$

where p_0 and p_1 are the 0 and 1 probabilities, respectively. The maximum ensures that values above 0.5 are used. Ideally, $H_{\min}$ approaches the max-entropy and both approach one.

The National Institute of Standards and Technology (NIST) publishes random number generator tests [17]. The NIST tests provide measures of the binary string randomness. The tests calculate a P-value representing the confidence in whether or not that test's null hypothesis should be accepted, i.e., that the data are in fact random. The alternative hypothesis is that the sequence is non-random. The goal of the tests is to minimize the probability of type II error, that the data are not random but pass. For each test, the bit sequence is analyzed and the output X is input to either a complementary error function or upper incomplete gamma function. The result is compared with the significance level α that is a measure of the type I error probability.

The tests make two primary assumptions. First is uniformity: the number of 0's and 1's should be approximately equal throughout the string. The second is scalability: extracted subsequences should not match.

1.4. Paper significance and organization

In this paper we describe for the first time a PUF based on 1.5-T flash memory erase characteristics. The target is the Microchip Technology SST Superflash 1.5-T memory available as standalone and foundry available embedded memories [8][19]. While random telegraph noise (RTN) has been shown in these memories, we choose not to use that mechanism, which was used in a 1-T flash memory based TRNG [21] as it is not stable in time and thus not appropriate for IC identification (ID). We present a reliable flash erase approach to using the resulting strings as a unique chip identifier with minimal computation and helper data.

Like many memory PUFs the resulting codes (or fingerprints) are prone to systematic offsets and due to granularity in the erase process, do not have consistently equal numbers of 1's and 0's. We also show, for the first time, that these systematic offsets can be removed so that the resulting TRNG binary strings pass all of the NIST randomness tests. Passing the NIST tests indicates that the PUF circuit can function as a TRNG unit in an IC. We believe these techniques are useful for removing systematic variations in other memory based PUFs. Finally, we show that the IDs produced are robust to aging effects, although the PUF may need to be isolated to avoid them.

Section 1 establishes the motivation for this work. In Section 2 we briefly describe embedded flash memory cell architecture, functionality and our experimental approach. Section 3 introduces a novel method to validate the challenge and response (or determine matching device IDs (fingerprints)) using the erase monotonicity with time. We describe whitening helper functions to reduce bias in Section 4, which allow high quality TRNG. Section 5 presents the impact of program/erase stress on the quality of response (stressed) vs. challenge (pre-stress). Section 6 summarizes the results presented in this paper.

2. 1.5-T Flash Array PUF

2.1. Memory cell

The split-gate flash cell (Figure 2) is referred to as a 1.5-T cell since it has separate select gate (word line) and floating gate devices as a composite between the source and drain [1][8][19]. The cell transistor pairs share a common source

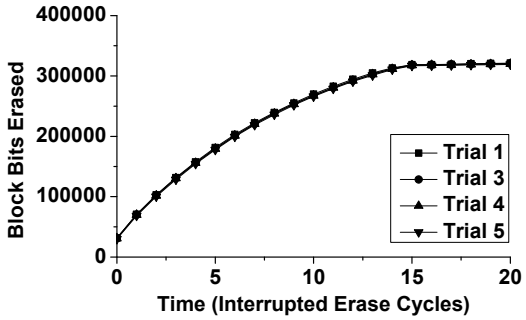

Figure 3: Flash block low V_{DD} erase characteristics for multiple trials. After each erase cycle, the block is read out to determine the number of cells erased to the logic 1 state.

line (source in Figure 2(a)). From a circuit point of view, the design is two series devices as in Figure 2(c)—the floating gate (FG) at the source line (SL) provides programmability. To read, V_{DD} is applied to the WL and $V_{CG} = V_{DD}$ to couple the FG up slightly. The source line is at ground and the BL is biased to about 1 V, producing the read current. A large IOT benefit of the 1.5-T cell is high voltage is not needed for read.

The FG is erased by removing electrons via poly to poly Fowler-Nordheim (F-N) tunneling, raising the FG potential. To erase, the erase gate (EG) is driven to a high voltage (~11.5 V) while the bit line (BL), WL and control gate (CG) are grounded (Figure 2(b)). The WL and CG couple the FG to a low potential, increasing the tunneling probability to the EG. The F-N tunneling current density is

$$J = C_1 \, E^2 \, \exp\left(-E_0 / E\right) \tag{2}$$

where here E is the electric field and C_1 and E_0 are constants related to the effective mass and barrier height [18]. Since E is inversely proportional to the inter-poly erase tunnel oxide equivalent thickness, the rate at which a cell erases is a measurement of the oxide thickness t_{ox}. Thus, the erase rate is an exponential function of the effective as-fabricated floating gate to erase gate inter-poly dielectric thickness at the interface shown by the red arrows in Figure 2(b). While other variations affect the erase speed [19], this variation is the first order basis of the 1.5-T PUF in this paper.

2.2. Experiments

The experiments use 512k-bit blocks in Microchip Technology Inc. 16 Mbit SST39VF1601C devices. We begin by fully programming a block to all 0's. The 1.5-T flash cells have very high program and erase efficiency [8]. The former takes only one clock and so cannot be interrupted. Erase requires many cycles, so we interrupt the erase operations as fast as our test system can (in one memory instruction). At nominal voltages, most array cells erase in the single cycle even when interrupted as fast as possible. Consequently, to slow the erase further, we reduce the flash V_{DD} during erase. Lower IC V_{DD} reduces the internal charge pump output voltage, diminishing the erase gate to floating gate potential. Since F-N tunneling current density is exponentially related to the field (Eq. 2) the erase rate is slowed significantly.

Figure 3 shows the number of 1 state (erased) cells vs. partial erase cycles for one device under test (DUT) array. We use the array states that are closest to, but over 50% 1's vs. 0's to determine the codes. Note the saturation in the values after approximately 15 cycles—many of the remaining un-erased cells will not erase at the low voltages used. A bit map shows that the erase characteristics appear to be random (Figure 4). The operations are highly repeatable and the erase operations are monotonic with time, until the saturation point where further erase operations have little effect (see Figure 3). There are no bits changing from 1 to 0 with subsequent erase operations. This suggests an identifier scheme following the DRAM PUF and flash program approaches [15][21].

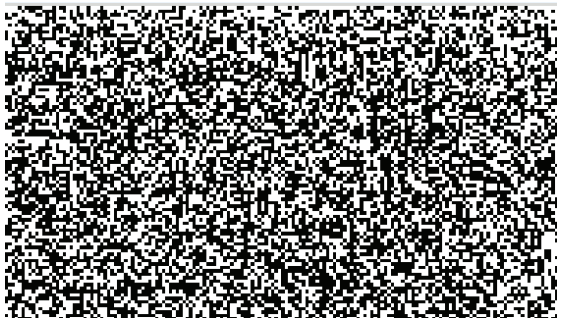

Figure 4: Partial bit map of the array state at approximately 50% erased bits.

3. Integrated Circuit Device Identification

The use of PUFs as IC fingerprints allows secure identification of devices and users, as well as the ability to trace ICs, e.g., to determine grey market devices. In this section we evaluate the resulting IC fingerprint quality. The key to a reliable fingerprint is that it must be repeatable and its randomness (the PUF entropy) must ensure that other fingerprints are sufficiently far from the repeatable matching fingerprint. A post-helper function value would be stored elsewhere. It comprises the challenge and therefore must include the location to be accessed and the appropriate response, ideally obfuscated since it is publicly accessible. In this section we evaluate the quality of the 1.5-T flash codes.

3.1. Novel IC fingerprint compare

Following other memory based fingerprint analyses, we compared every 256-bit string generated in the flash memory with each other. We generated the same fingerprints in ten separate trials on multiple DUTs. The intra-class keys exhibited between 0 and 65 mismatches and the inter-class keys had 92 to 102 mismatching bits for cases with up to 10% fewer erased bits in the response vs. the challenge. Thus, a direct compare scheme will have to allow some mismatches, as do most memory PUF schemes. A key issue is the ability to stop the erase at the same point. In a real application this may be impossible, so there will always be some mismatches, given by the control that can be exhibited by the erase process. In these experiments it is purposely crude, but the results are nonetheless quite good.

At the top Figure 5 shows the bits for a code subset over multiple partial erase cycles, labeled EC. Erase is monotonic so a code generated with say 45% 1's as the response can be effectively compared with a challenge code generated with 55% 1's. The challenge code (**CH**) is produced by stopping the erase when the flash array has more than 50% 1's and the result code (**RE**) has been erased to less than 50% 1's. In

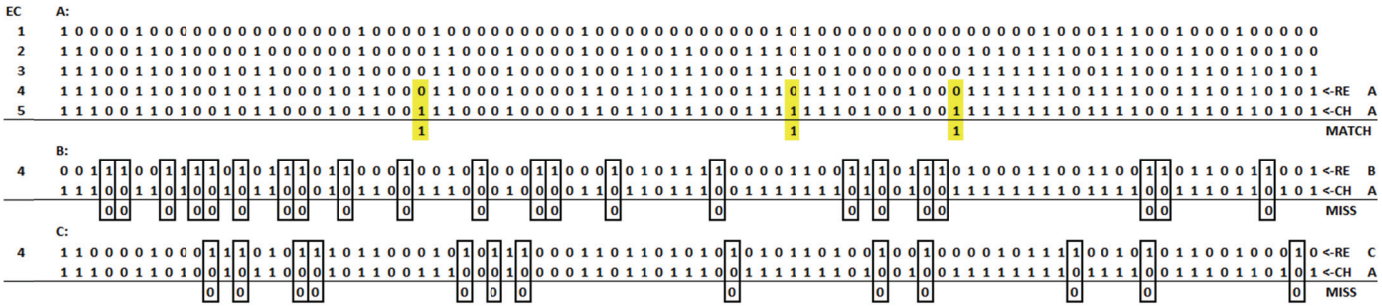

Figure 5: Partial fingerprint compare based on the monotonicity of the erase operations (scheme MON1). The challenge (**CH**) is chosen as codes after the memory block reaches 50% ones content. The monotonicity of cells becoming 1 with more erase cycles (EC) is evident in the top (A) codes. The response (**RE**) is reproduced by the flash as a cycle before 50% ones content on demand. A matching challenge bit that is a 0 cannot be a 1 in the response (yellow). Conversely, mismatching challenges (intra-code on B and C) have many 0's that are 1's in the response (boxes). This experiment purposely has fast (coarse) erasing.

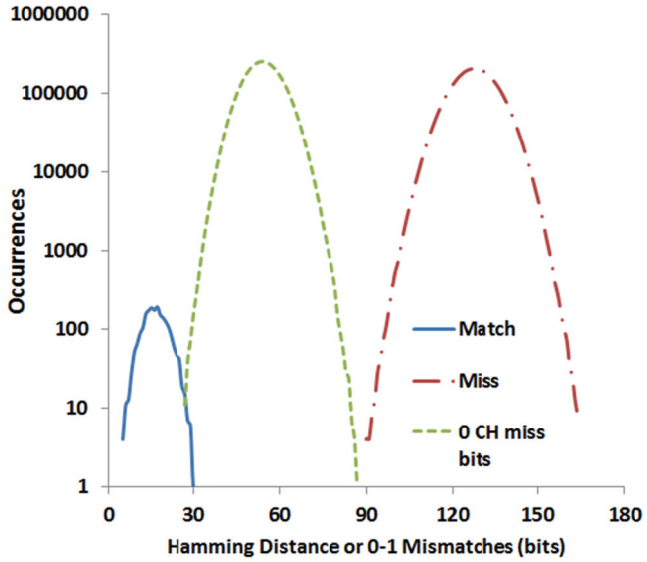

Figure 6: 256 bit code compare of 2k samples using MON1. All mismatching bits are 1 in the challenge for all codes that should match. A minimum of 15 (CH bit = 0) cases occur for mismatching bits in codes that should not match (unstressed).

Figure 5, we use code A for **CH**. The matching code subset with lesser erase has a few mismatching bits, but challenge bits are 1 in this intra-class case. This suggests a simple scheme whereby for each (RE[x] != CH[x]) if CH[x] == 1, the mismatch is ignored or conversely, if any CH[x] == 0 on a mismatch, the codes do not match. Figure 5 bottom shows that many mismatching bits have CH[x] == 0 for mismatching bits when the challenge is compared to the response of other codewords, i.e., intra-class, with two representatives labeled B and C.

Figure 6 shows the resulting Hamming distances for 256 bit codes. Intra-class, i.e., matching codes, never have more than 30 mismatching bits. The results are very Gaussian, confirmed by Q-Q plots to ±5 σ. The intra-class codes have Hamming distances of 22 to 65 (Figure 6 labeled Match). This seems a poor result, but the number of bits that are 0 in the challenge that mismatch ranges from 15 to 79 (Fig. 6 labeled "0 CH miss bits"). *We never observed a challenge bit zero corresponding to a response one bit in any intra-class comparison*. Thus the scheme of detecting a mismatch as zero challenge and one response bit always works.

We ensure the monotonicity of erased bits by avoiding the shallow portion of the curve in Figure 3 by choosing the voltage to be high enough that the 50% point is below the erase saturation. While bits may still fluctuate if they are at the input referred offset of the sense circuits [14], we have not seen such fluctuations until the flat portion of the curve in over 4M bits. In the shallow region, only a few hundred fluctuating bits have been observed on multiple DUTs.

3.2. Raw erase data as random numbers

Following the analysis used on SRAMs in [3] the max-entropy was determined using the Unix zip program, where the compression ratio estimates the max-entropy. In all cases, no compression was provided, and analysis of the .zip file header showed that the resulting file was uncompressed. Thus, we estimate the max-entropy to be unity. The min-entropy (Eq. 1) was calculated for the raw flash data. The min-entropy for all blocks on all DUTs tested was greater than 0.9. This min-entropy is considerably higher than the 0.75 to 0.76 reported for SRAM powerup [3].

We used the NIST tests to evaluate the raw output as true random numbers. Using a naïve approach, very few of the tests pass (Table 1 column "raw"). Tests that were not run are left blank. For the 150 non-overlap template tests, we considered anything with less than 5 failures a pass (the fail count is also in Table 1). The tests evaluate deviations from expected randomness and search for patterns, which systematic variations might produce if the bits are chosen in row order, as we did here. Most importantly, nearly 50:50 ones and zeros are required, specifically for the Frequency test, but also for others. This would require very small steps in the erase cycles. We have, with very careful V_{DD} selection for each device achieved values within 2k matching in 500k bits (0.4%) but most experiments here use much coarser granularity (as mentioned) to ensure worst-case analysis.

4. Systematic variations and helper functions

A closer examination of the spatial variation by mapping multiple 512k-bit arrays (raw data) shows systematic erase rate variability across a block, as shown in Figure 7(a). Rows read in address order from left to right show a systematic bias towards 0's (un-erased) at the edges. This was revealed by

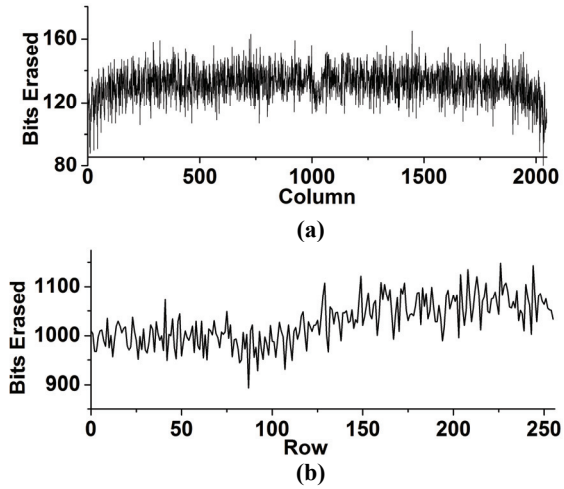

Figure 7: Systematic results across one arrays in the (a) row (along WL) and (b) column (along BL) directions**.**

summing the ones per array column. Thus, each 2048 bit substring has a lower likelihood of a one at its ends. All blocks on a device and across devices show very similar behavior, so we attribute it to the specific array design. For instance, the slower erase at outer columns may be due to IR drop in the erase voltage across the array. We speculate as we do not have access to the specific design, but simulations show such a droop if driven only at the middle. The mechanism behind the by row behavior in Figure 7(b) is less obvious. The by column variability clearly affects 0/1 frequency, as evidenced by the FFT test.

While this did not disrupt the IC fingerprint scheme in Section 3.1 it does show that the entropy is less than perfect, and that use as a true random number generator will require helper data algorithms to ensure better one vs. zero distribution and to counteract the systematic offsets. That is the topic of the next section.

4.1. Systematic offset mitigation and data whitening

We investigated numerous helper data algorithms to improve the 1.5-T flash response with the goal of passing the NIST tests for TRNG application. Our goal was very simple: helper functions that could be easily implemented in circuits.

Since the cause of most of the NIST test failures was the uneven distribution of 1 and 0 bits, we initially tried inverting every other bit, i.e., whitening the string. Since an input bit has a 50% likelihood of being inverted, the resulting string has very nearly 50:50 one:zero distribution. The resulting strings, labeled **EOinvert** in Table 1, pass many more tests, but not all. Specifically, the patterns of Figure 7 produce obvious spectral bounces picked up by FFT.

Scramble uses a pseudo-random number generator to change the bit order of each **EOinvert** generated codeword. This passes all the tests (see Table 1) but is not a simple hardware solution, since it requires a large buffer memory. **Scramble16** performs the same function with 16 bit groups. We surmise that the Runs test fails this helper algorithm since the edge 0 to 1 frequency slopes are retained but moved with this scheme. **Complex** drops the problematic columns at the outer edges (and some in the array middle columns) and interleaves two-bits at a time from two 512k-bit arrays with

Table 1: NIST randomness test results of the flash generated random strings with different whitening helper algorithms.

Test	raw	EOinvert	Scramble	Scramble16	Complex	RandomInvert	LFSR invert
Approximate Entropy	F	F	P	P	P	P	P
Block Frequency	F	F	P	P	P	P	P
Cumulative Sums	F	P	P	P	P	P	P
FFT	F	F	P	P	P	P	P
Frequency	F	P	P	P	P	P	P
Longest Run	F	P	P	P	P	P	P
Non-overlap Template	F 78	P 4	P	P 1	P	P 2	P 2
Overlapping Template	F	P	P	P	P	P	P
Random Excursions		P	P	P	P	P	P
Rand. Exc. Variant		P	P	P	P	P	P
Rank	P	P	P	P	P	P	P
Runs	P	F	P	F	P	P	P
Serial	P	P	P	P	P	P	P
Universal	F	P	P	P	P	P	P

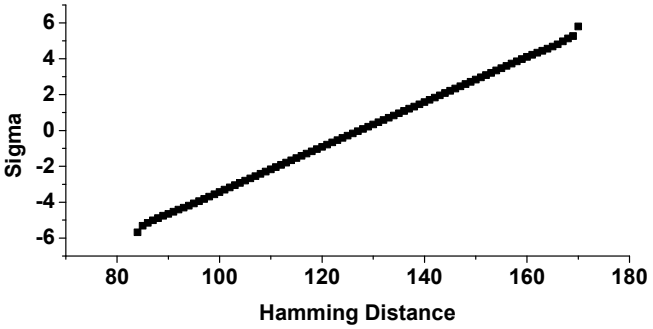

Figure 8: Hamming distance between non-matching 256-bit fingerprints. Results are Gaussian to over ±5 sigma, i.e., exhibit nearly perfect randomness.

bits from the second array inverted. It passes all of the tests. The H_{min} of this result is over 0.99 and the Q-Q plot of the intra-class codes (Figure 8) is nearly ideal.

Finally, **RandomInvert** uses a pseudo-random number generator to decide which bits to invert (with 50:50 behavior). It passes all the tests (see Table 1). This result suggested the final very simple procedure **LFSRinvert**. This scheme inverts bits in the raw string based on the LSB of a 127 cycle linear feedback shift register (LFSR). It also passes all tests, showing the good 1.5-T flash entropy requires a minimal helper data algorithm for nearly ideal results. The H_{min} of the resulting string is over 0.99 and the max-entropy is 1 based on no compression by the zip program.

5. Impact of aging

Delay PUFs and some SRAM PUFs are substantially impact by process, voltage and temperature (PVT) variations [12]. Temperature greatly affects delay, some SRAM, and DRAM based designs [7][11][21]. This is important since many IOT and certainly automotive applications require wide temperature range. The FN tunneling rate is not significantly impacted by temperature and we saw no significant temperature impact in experiments varying it.

We also investigated how the codes (IDs) withstood repeated program/erase cycles. Embedded flash allows saving the generated code post-helper algorithm directly, this violates the premise of a PUF. Ideally, if program/erase stress does not impact the response, then any flash sector (the minimum erase block) can be used effectively. This would maximize the possible codes. Of course for a TRNG, small changes are not important

We stressed two DUTs with 10k full high voltage program/erase cycles. The pre-stress MON1 code distances of one DUT comprise Figure 6. The stressed results show very little impact from the aging (Figure 9). However, post-aging, there are some intra-code bits that deviate from our MON1 scheme, i.e., *up to three bits exhibit non-monotonic behavior when compared to the pre-stress challenge code*. These are evident in dotted black line near the Y-axis. Consequently, some form of error detection will be required. This suggests that a dedicated embedded flash block, which is more easily isolated by the helper function logic, may be required. If the embedded PUF and helper function block use a precision, internally isolated erase voltage, via say an internally regulated voltage, with fine erase time control (not software, as used here) any stress or aging impact may be mitigated.

6. Analysis and summary

This paper demonstrated that 1.5-T erase rate produces effective device IDs. The resulting codes, based on the as-fabricated tunnel oxide variability, are highly repeatable. Thus, repeated erase cycling is an effective PUF approach. The min-entropy of the 1.5-T flash memory used as a PUF as described is quite good, varying between 0.90 and 0.91 across DUTs. This suggests that further work truly embedding it as a PUF with better voltage and timing control could improve this to near ideal. As it is, the large Hamming distance avoids bit-aliasing causing false-positives. Relatively poor SRAM PUF reliability has suggested soft-decision encoding [4][10]. Here we demonstrated much simpler schemes make adequate helper data algorithms for this type of flash memory PUF.

The multiple simple data helper (whitening) functions effectively alleviate systematic offsets so the resulting codes pass the NIST tests, proving applicability as a TRNG circuit. Weak PUFs have been suggested for choosing keys in cryptographic blocks [20]. In this context, the data whitening operations are particularly valuable. The whitening schemes should be effective for any memory based PUF.

7. References

[1] P. Cappelletti and A. Modelli, "Flash memory reliability," in *Flash Memories*, P. Cappelletti, C. Golla, P. Olivo, E. Zanoni, Eds., New York: Springer, 1999, pp. 399-441.

[2] S. Chellappa, A. Dey, and L. T. Clark, "SRAM-based unique chip identifier techniques," *IEEE Trans. VLSI Sys.*, vol. 24, no. 4, pp. 1213–1222, Apr. 2016.

[3] M. Claes, V. van der Leest, and A Breaken, "Comparison of SRAM and FF PUF in 65 nm technology," in *Nordic Conference on Secure IT Systems*, pp. 47-64, Oct. 2011.

[4] J. Devaux, D. Gu, D. Schellekens, and I. Verbauwhede, "Helper data algorithms for PUF-based key generation: overview and analysis," *IEEE Trans. CAD of Int. Circ. and Syst.*, vol. 34, no. 6, pp. 889-902, June 2015.

[5] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, "Controlled physical random functions," *Proc. Computer Security App. Conf.*, pp. 149-160, 2002.

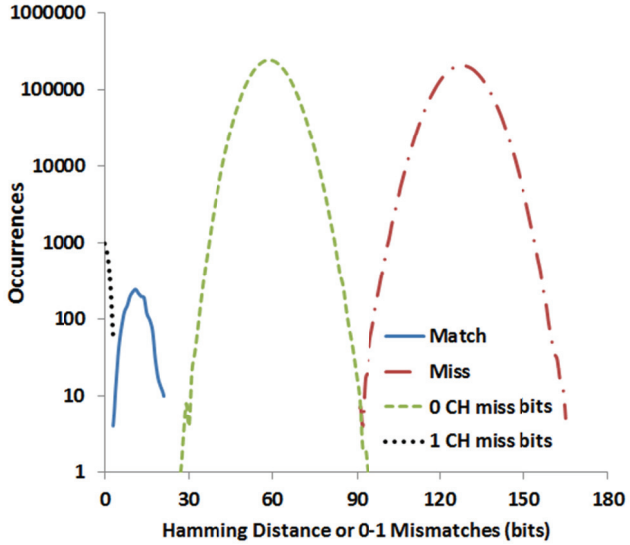

Figure 9: Post-stress 256 bit 2k code compare using MON1. Four mismatching bits are 1 in the wort-case challenges for many matching codes. A minimum of 26 (**CH** bit = 0) cases occur for mismatching bits in codes that should not match**.**

[6] D. Holcomb, W. Burleson, and K. Fu, "Power-up SRAM state as an identifying fingerprint and source of true random numbers," *IEEE Trans. Comp.*, vol. 58, no. 9, pp. 1198–1210, Sep. 2009.

[7] J. Jang and S. Ghosh, "Design and analysis of novel SRAM PUFs with embedded latch for robustness," *Proc. ISQED*, pp. 298-303, May 2015.

[8] A. Kotov, "Three generations of embedded SuperFlash split gate cell: scaling progress and challenges," *Memory Workshop*, June 2013.

[9] K. Lofstrom, W. R. Daasch, and D. Taylor, "IC identification circuit using device mismatch," *Proc. ISSCC*, pp. 372-373, Feb. 2000.

[10] R. Maes, P. Tuyls, I. Verbauwhede, "A soft decision helper data algorithm for SRAM PUFs," *IEEE Int. Symp. Inf. Theory*, pp. 2101-2105, 2009.

[11] R. Maes, V. van der Leest, E. van der Sluis, and F. Willems, "Secure key generation from biased PUFs," *Int. Workshop on Cryptographic Hardware and Embedded Systems*, pp. 517-534, Sept. 2015.

[12] A. Maiti, I. Kim, and P. Schaumont, "A robust physical unclonable function with enhanced challenge-response set," *IEEE Trans. Inf. Forensics and Security*, vol. 7, no. 1, pp. 333-345, Feb. 2015.

[13] S. Mathew, *et al*., "A 0.19pJ/b PVT-variation-tolerant hybrid physically unclonable function circuit for 100% stable secure key generation in 22 nm CMOS," *Proc. ISSCC*, pp. 278-279, Feb. 2014.

[14] R. Micheloni, L. Crippa, M. Sangalli, and G. Campardo, "The flash memory read path: building blocks and critical aspects," *Proc. IEEE*, vol. 91, no. 4, pp. 537-553, Apr. 2003.

[15] S. Rosenblatt, S. Chellappa, A. Cestero, N. Robson, T. Kirihata, and S. S. Iyer, "A self-authenticating chip architecture using an intrinsic fingerprint of embedded DRAM," *IEEE J. Solid-state Circ.*, vol. 48, no. 11, pp. 2934-2943, Nov. 2013.

[16] U. Ruhrmair, *et al*., "Modeling attacks on physical unclonable functions," *Proc. ACM Conf. on Computer and Comm. Security*, pp. 237-249, 2010.

[17] A. Rukhin, *et al*., A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications," NIST Special Publication 800-22 Revision 1a, Apr. 2010.

[18] S. M. Sze, *Physics of Semiconductor Devices*. New York, NY: John Wiley and Sons, 1981.

[19] Y. Tkachev, X. Liu, and A. Kotov, "Floating-gate corner-enhanced poly-to-poly tunneling in split-gate flash memory cells," *IEEE Trans. Elec. Dev.*, vol. 59, no. 1, pp. 5-11, Jan. 2012.

[20] A. Vijayakumar, V. C. Patil and S. Kundu, "On testing physically unclonable functions for uniqueness," *Proc. ISQED*, pp. 368-373, May 2016.

[21] Y. Wang, *et al*., "Flash memory for ubiquitous hardware security functions: true random number generation and device fingerprints," *IEEE Security and Privacy Symp*., pp. 33-47, 2012.