Robust Hardware True Random Number Generators using DRAM Remanence Effects

Fatemeh Tehranipoor, Wei Yan and John A. Chandy
Department of Electrical and Computer Engineering
University of Connecticut, Storrs, CT USA
(f.tehrani, wey140004, chandy)@engr.uconn.edu

***Abstract*—A True Random Number Generator (TRNG) is an important security primitive used in a variety of applications including cryptographic algorithms, communication systems, simulations, etc. It is critical to be able to produce outputs consisting of fully unpredictable and unbiased bits in a cost-effective manner. In this paper, we present a robust hardware TRNG based on the Dynamic RAM (DRAM) remanence effect, which is a condition whereby information remains in a DRAM even after powering it down. The advantage of our hardware TRNG is that it forms from existing components with no extra circuitry. We assessed and tested the randomness of our proposed hardware TRNG by applying the NIST Statistical Test which indicates the unpredictability and nonrepeatability of our data. Given its strong NIST results, we believe that there is a potential for immediate cryptographic applications.**

***Index Terms*—True Random Number Generator, Cryptography, Dynamic RAM, Data Remanence, Randomness, NIST Statistical Test.**

I. Introduction

With the crucial need for information and data security, cryptography has become more and more important. Cryptographic systems, in order to ensure their security guarantees, are highly dependent on the availability of a truly random numbers. Random Number Generators (RNGs) have thus become an indispensable primitive in security systems and applications. RNG systems can be categorized into two types: Pseudo Random Number Generators (PRNGs) and True Random Number Generators (TRNGs). PRNGs depend on deterministic computational algorithms to produce long sequences of random bits based on an initial input called the seed. The determinism causes PRNGs to be predictable. On the other hand, TRNGs are non-deterministic random number generators that can provide full entropy. They generate high-speed and true random sequences based on physical processes such as phase noises, thermal noise, jitter, random telegraph noise, photoelectric effect and other phenomena, that are statistically random in nature. The randomness of a TRNG is affected by limited process variation when inner random noise source cannot provide enough. For cryptographic applications, the security of a TRNG relies heavily on its entropy rate. The unpredictability of the output implies that the TRNGs must be neither observable nor manipulable by any attacker. Maintaining high entropy randomness across environmental variations is also paramount. The main goal of all RNG principles is to maximize the entropy of each bit in the generated bit-stream. The random source can be constructed of dedicated hardware devices (hardware-based RNGs) or extracted from software procedures (software-based RNGs). Hardware RNGs present the advantage of having a clearer model of the source since they have interactions with the outside. Normally, Hardware-based TRNGs are faster, higher in quality, more protected, and get much higher throughput compared to one obtained from software-based RNGs [1]. The unpredictability of TRNGs offers better randomness and since a TRNG lacks periodicity or data dependencies, making it the best option for meeting the stringent requirement for communication and encryption. Although TRNGs have many advantages over PRNGs, PRNGs are very popular due to their flexibility, low cost and generation time [2].

Data remanence is the residual information that remains on a storage medium even after erasure (data clearing) or powering off the device. Data remanence as a problem was first discovered in magnetic media [3], [4]. Thus, there is a possibility that sensitive information still can be extracted from non-volatile memory that had presumably been erased. Additionally, supposedly volatile memories, such as DRAM, are examples of storage components that they do not lose their data immediately after power is removed. Furthermore, these memories exhibit remanence effects wherein data stored for an appreciable amount of time in the same location can leave a permanent recoverable trace in the memory cells. Data remanence effects can play an important role in extracting the secret stored information from volatile memories. As mentioned in [3], data remanence is not a directly evaluated criterion of trusted computing systems, but it is critical to the safeguarding of information used by trusted computing systems. In [5], Halderman et al. have shown that DRAMs retain their contents for several seconds after power is lost.

In this paper, we introduce a new approach for constructing a robust hardware TRNG based on DRAM remanence effects. A robust hardware TRNG should have a high level of randomness or unpredictability in the output. Otherwise, an adversary can simply query the generator and predict the sensitive data. Our novel approach focuses on creating a TRNG from DRAM devices or components that are already present in most computing systems. This feature effectively allows for DRAM to generate a TRNG without any extra hardware required to be added to the system. Using this knowledge we designed

978-1-4673-8826-9/16/$31.00 ©2016 IEEE

Authorized licensed use limited to: ETH BIBLIOTHEK ZURICH. Downloaded on November 19,2020 at 13:01:18 UTC from IEEE Xplore. Restrictions apply.

an inexpensive cryptographic key generator, implemented the findings and analyzed the results. The main advantage of this approach over other TRNGs is that we can produce highly random bits without incurring any additional hardware costs.

Motivations and Objective: Specifically, we have the following motivations and objectives:

1) Proposal and implementation of a method to generate robust TRNG from DRAM that requires no extra circuitry.
2) Validation of the quality of random bit sequences generated by the proposed TRNG using the NIST Test.
3) The goal is to produce an unpredictable TRNG which is subject to experimental tests.

Contribution and Paper Organization: Our primary contribution is the use of the remanence effect property of DRAM to produce a TRNG. The remainder of the paper is organized as follows. The next section will briefly cover previous work, observations, and challenges that have been encountered while designing TRNGs over the past few years, including preliminary information to help illuminate the workings of DRAM. Section III describes our novel remanence-based TRNG model in DRAM. Section IV presents the results of NIST statistical testing. Section V will discuss two main points: the limitation of our work and other system variations caused by operating environments. The purpose of section VI is to highlight application areas for our discovery. At the end, we conclude the paper with some final remarks in Section VII.

II. Related Work

Cryptography and security applications make extensive use of random numbers and random bits. Random numbers are useful for a variety of purposes, such as generating data encryption keys, simulating and modeling complex phenomena, selecting random samples from larger data sets, and even for gambling. Generally, PRNGs fit most of the application needs but there are demanding situations where PRNGs are substituted by TRNGs and those are the applications where it is important that the numbers be really unpredictable, such as data encryption, games, generating cryptographic keys, generating lists of lottery winner tickets, or stochastic simulations. The reason being the PRNG's lack of strong statistical properties [2], which can be seen via statistical tests. Hence, the output sequences from TRNGs should have good statistical properties verified with the use of statistical tests, e.g. from NIST 800-22 statistical test suite. However, the generation of true random bits is problematic in many practical applications of cryptography. Currently, several techniques are used for implementing TRNGs. In [6], Bruyninckx et al. analyzed the security requirements of TRNGs, demonstrated the real-life attacks performed on various types of TRNGs and proposed solutions for generating safe cryptography random bits using TRNGs from untrusted vendors. In [7], the authors exploited random behavior from nearly-metastable operation of a group of FPGAs. An oscillator-based TRNG has been proposed in [8] that can automatically adjusts the generated unbiased random

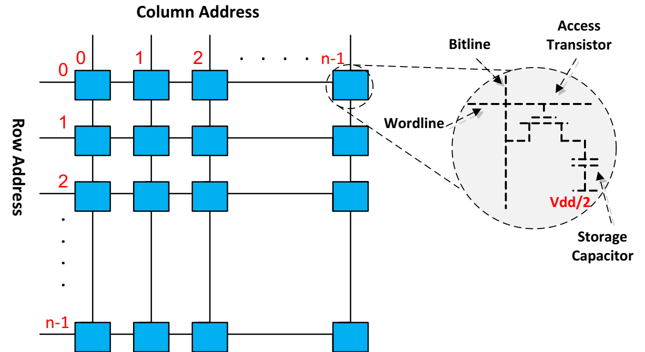

Figure 1: DRAM cells array with a typical single MOSFET transistor and a storage capacitor.

numbers produced by process variation and dynamic temperature. Mudit et al. proposed a TRNG design based around sense amplifier circuits that are balanced in the metastable region using hot carrier injection in [9]. Recently, new TRNG models such as Technology Independent (TI) TRNG, TRNG using hot-carrier injection balanced metastable sense amplifiers, Portable TRNG for personal encryption application based on smart phone cameras, and Highly Efficient TRNG in FPGA Devices Using Phase-Locked Loops are investigated in [10], [11], [12], and [13], respectively.

The issue when examining the challenges of previous TRNG work is that there is a clear line of acceptance between a working TRNG and a non-working TRNG device. There is, however, a range of success for TRNG models that meet the required standard. Some TRNGs have better results in terms of randomness while others do not. Our proposed remanence-based TRNG has the advantage of being simplistic, in that, our model does not need any extra hardware or overhead when compared to existing solutions for TRNGs. For example a ring oscillator TRNG requires extra circuitry and design for the production of the ring oscillator network.

III. DRAM Remanence-based TRNG

In this section, we describe our methodology of using the DRAM remanence effect to propose a new TRNG model. We start with a brief description of a typical DRAM architecture. DRAM is a low cost and high density memory, therefore it is widely used for the main memory in computers. A DRAM memory cell uses a single transistor and a capacitor to store a bit of data. It has the advantage of a high memory density per chip due to the fewer number of MOSFETs needed per cell. Cell information (voltage) is degraded mostly due to a junction leakage current at the storage node. Therefore, the cell data must be read and rewritten periodically even when memory arrays are not accessed. Essentially, the DRAM controller must refresh each cell's voltage before it decays to the point where the bit information gets lost. Normally, the refresh rate is so high that each cell gets refreshed several times per second. Figure 1 shows a DRAM cell array that are arranged in rows (0 to n-1) and columns (0 to n-1) of memory cells called

Authorized licensed use limited to: ETH BIBLIOTHEK ZURICH. Downloaded on November 19,2020 at 13:01:18 UTC from IEEE Xplore. Restrictions apply.

wordlines and bitlines, respectively. Each memory cell has a unique location or address defined by the intersection of a row and a column. One memory cell has been magnified in Figure 1 which consists of a access transistor and a storage capacitor that is charged to produce a 1 or a 0.

The processes of extracting random bits from DRAMs, while considering the remanence effect and startup behavior of DRAMs is briefly laid out as follows. As shown in Figure 2, the process is first we write the value '1' to all cells of the available memory; this can be seen as step 'a)' in Figure 2. After the write operation, a delay function (step b) has been applied to turn OFF the DRAM for certain amount of time, which is in milliseconds, and then turn the DRAM back on after a specific delay time. In the next step, the entire 1 Mbit of data are read (step c) and stored.

The expectation was that, depending on the delay, a certain percentage of the '1's written at startup would have flipped to '0' and the remainder would have stayed '1' because of remanence. Presumably, which bits flipped would be random and that could provide the bits for our TRNG sequence. However, it turns out that DRAMs do not actually settle to all '0' when powered off completely. In [14], it was noticed that certain DRAMs exhibit behavior similar to static RAMs (SRAMs), i.e. they have seemingly random startup values after being powered on. The reason for this behavior in DRAMs can be explained by the interaction of precharge, row decoder, and column select lines when the device is powered up. Thus, while reading from DRAMs, the sense amplifier is equally likely to read a '1' or '0'. However, because of the manufacturing variations, the storage capacitance of each bit will have slight differences, which leads to biasing of each bit to either a '1' or a '0'. Thus, for determining the effect of remanence after various delays, in our experiment we found that instead of flipping to '0' the bits will settle to the original startup value at power on.

We performed this processes trying several delay values to determine the effect of remanence on the bit values. We used a Xilinx Spartan-6 XC6SLX45 FPGA experimental platform similar to [15]. A power-cycle circuit was implemented in our experimental setup as to provide a method to deliver power to the DRAM chip and as well as enabling (turning ON the DRAM) and disabling (turning OFF the DRAM) the on-board DRAM chip located on the Xilinx Spartan-6 FPGA board. We collected approximately 400 measurements of DDR2 SDRAMs based on various delay time and under nominal environment conditions (temperature: 25 °C and DRAM voltage: 1.8 volt).

Figure 3 shows the results of our experiments where the x-axis shows the delay times from 1 milliseconds to 2000 milliseconds and the y-axis illustrates the percentages of '0' bits observed at each delay time. As can be seen, Figure 3 is divided in to three regions. Specifically, we note that before the DRAMs settle into their startup values where the data is quite stable, the DRAM values somewhat "overshoot" in that slightly more bits flip to '0' before flipping back to '1' in the startup state. We call this region the "fluctuation" region. In the following subsections, we will discuss and model the three distinct regions.

a)	Write	11111111111111111111111111111111 11111111111111111111111111111111 11111111111111111111111111111111
b)	Delay(1)	D1
c)	Read(1)	11111110111110111011111111101011 11111010111110111111011110111111 11110111111111110111111011011111
	Delay(2)	D2
		. . .
	Read(n)	11001010100101010010101001001101 10100101010100110011010010100011 01101000110110000101010001001010
	"Settling down to Startup value"	
d)	Read(m)	11100101100101101000101000100011 01110100000100110101101100010000 01011000110001101010000110110101

Figure 2: Real data snippets that illustrate the DDR2 SDRAM operations: a) Write. b) Delay(1). c) Read(1). d) Read(m).

A. *Our TRNG Model*

*1) **Flip Region**:* The Flip Region is the place that the percentage of flip bits increases from below 1% to 56% as the delay time goes from 1 milliseconds up to 350 milliseconds. We got 5 measurements at each delay time, and showed the percentage of bit flip with an error bar. In total, we got 175 measurements at this region. Using MATLAB cftool, we draw the fit data that is represented by the red line. The fit equation is a Linear model Equation with the below specifications:

$$b_0 = 0.211t - 5.909 \tag{1}$$

where t is the delay time, and b_0 is the expected number of zero bits at each delay time within the Flip-Region.

*2) **Fluctuation Region**:* The Fluctuation Region starts from 350 milliseconds and ends at 1180 milliseconds. This region has more variations and most of our measurements are focused on this region. In total we took 168 measurements of this region. The percentage of bit flips ranges from 58% to 64% which is more than the percentage of '0's at startup. That indicates that there are more '0's in the fluctuation region than the startup. This unexpected phenomenon became the focus of our investigation. A potential cause for this unexpected behavior is the temperature effect that we will discuss more about it in subsection III-B. The expected data of this region is drawn by a red line and the fit equation is a Linear model as seen with Equation 2.

$$b_0 = 3.744e^{-5}t + 61.27 \tag{2}$$

(3). Startup Region: Finally, the Startup Region is where the DRAM's behavior settles down to its expected startup behavior. This region is where the delay goes from around 1200 milliseconds and beyond. The linear model of this region

Authorized licensed use limited to: ETH BIBLIOTHEK ZURICH. Downloaded on November 19,2020 at 13:01:18 UTC from IEEE Xplore. Restrictions apply.

is as follows:

$$b_0 = 0.0002923t + 56.3 \quad (3)$$

B. Temperature Effects on DRAM data Remanence

Previous works have characterized DRAM and SRAMs memory remanence as a function of temperature. In fact, there is a correlation between DRAM remanence and DRAM temperature. Temperature plays an important role in data remanece time and the amount of bit flips [16] by affects power leakage at the transistor level. Particularly in DRAM, it significantly decreased with increasing the operation temperature of the System-on-Chip (SoC). For this reason, we also tried to test DDR2 SDRAMs in high temperature condition. When the temperature goes up, the more bits flip and the remanence time is much shorter. In other words, at high temperatures, the degradation process is accelerated and is very soon. On the other hand, the cooler the memory, the less bits flip and memory can retain its data for a longer period of time. Therefore the rate of RAM decay is largely a function of temperature. This poses a serious threat to hardware which operates with secret information (typically security key) in a secure environment. We got 5 measurements at some delay time in the Fluctuation Region. Using Thermostat, we measured the DRAM package temperature after the first measurement and it was 28°C. But the temperature of the package after the 5th measurement was 55°C. Note that the package surface temperature is not the actual temperature at the die - which will be much hotter. Based on the experiments, the percentage of flip bits at 5th measurement was much higher than the first one. For example, in one sets of 5 measurement continuously, the first measurement has around 58% flip bits, but the 5th measurement had almost 64% flip bits at the same delay time, since the DRAM is much hotter at 5th measurement than the first measurement.

C. Potential Attacks

Remanence characterization results can be used to build secure memories that are less vulnerable to various attacks. Cold boot attack is such an example that makes use of the property that the remanence effect is prolonged by cooling down RAM chips. Cold boot attacks rely on DRAM remanence that allow keys to be recovered from memory even after a machine has been powered down. These attacks can retrieve unencrypted data from RAM; The cold boot attack requires physical access to the machine, a boot disk (or specialized hardware), and quick timing (as data remains resident in RAM for only a short time after it's powered down). The extent and predictability of memory remanence and report that remanence times can be increased dramatically with simple cooling techniques [17].

IV. Data analysis and Evaluation

A. Random Number Generation

Given the data we collected, the next step is to extract random numbers from the data. Our initial attempt was to select an arbitrary delay time during the fluctuation region,

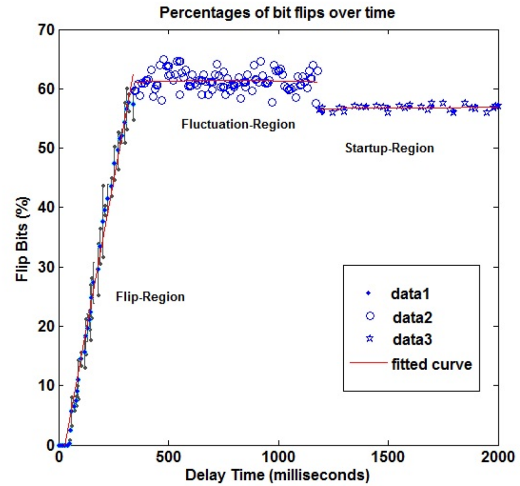

Figure 3: Our DRAM Remanence based TRNG Model using MATLAB Curve Fitting Tool (CFTool).

and use the memory that was read at that time as our set of random data. However, our analysis found that this data did not reliably generate suitably random data. The data was heavily biased with approximately 60% of bits being '0'.

As a secondary approach, we examined the differences between fluctuation and startup regions. Simply taking an XOR of the data returns the differential bits from the two regions. In the fluctuation region, we had 168 measurements and these were XOR'ed with a measurement taken from the startup region. Note that the startup region measurements were stable, so we only compared with one measurement from the startup region. Figure 4 shows the distribution of the 168 measurements taken from the Fluctuation Region XOR'ed with the measurements collected from the Startup Region. The mean of percentage of '1's among the 168 measurements is 0.5090 which is close to 0.5, the ideal case. The standard deviation of the data is 0.0201 and because it is close to zero, indicates that the data points tend to be very close to the mean of the sets of our measurements. As shown, the shape of the histogram clearly illustrates that the distribution of '1's and '0's is close to normal distribution. Therefore the distribution indicates that DRAM clearly demonstrates the characteristics of a TRNG as it relates to randomness.

We were also interested in the distribution of XOR'ed data relative to the delay time (Figure 5). We can see that the data is centered around 50% as expected. Moreover, other than the first 200 ms, the variation is not time dependent either, meaning that one could select any measurement in the fluctuation region to use as our random data generator.

B. Results of the NIST Statistical Test Suite

The NIST statistical test suite is used to evaluate the "randomness" of the bit strings produced by DRAM cells. Based on this test, we can determine whether a data set has a recognizable pattern or the process that has been generated is

Authorized licensed use limited to: ETH BIBLIOTHEK ZURICH. Downloaded on November 19,2020 at 13:01:18 UTC from IEEE Xplore. Restrictions apply.

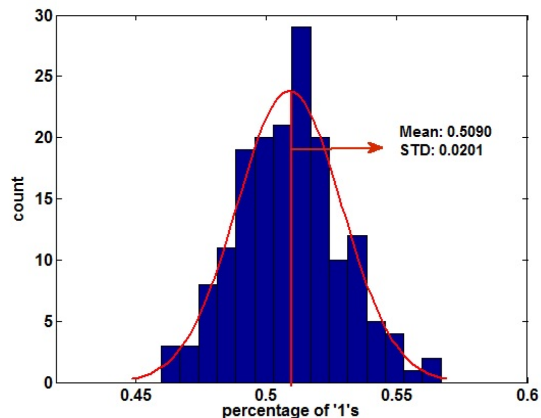

Figure 4: The histogram of the XORed data between Fluctuation and Startup regions.????

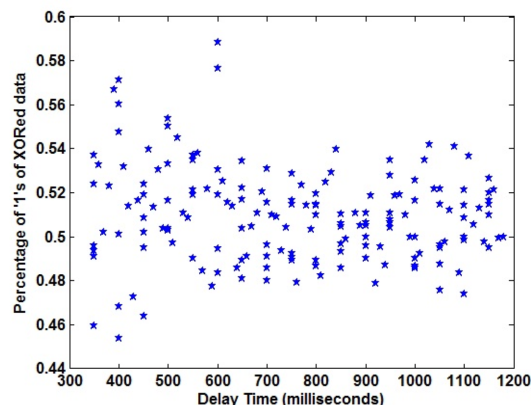

Figure 5: scatter plot of percentage of '1's of XORed data that are taken from Fluctuation and Startup regions.

significantly random. The NIST Test Suite (NTS) is a statistical package consisting of different types of tests to evaluate the randomness of binary sequences. Each statistical test is employed to calculate a P-value that shows the randomness of the given sequences based on that test. If a P-value for a test is determined to be equal to 1, then the sequence appears to have perfect randomness. A p-value $>= 0.01$ (normally 1%) would means that sequence would be considered to be random with a confidence of 99% [18].

To evaluate our collected data from DRAM, we apply NIST tests to the differential bits as discussed in subsection IV-A. Table I displays the average results of a suite of 15 performed NIST tests at room temperature. Table I shows that all the NIST tests p-value are greater than 0.01, this indicates that the measurements pass the requirements for randomness. We also applied NIST tests to the data gathered at high temperature. For space reasons, we do not show the NIST results, but even at high temperature condition, the data streams passed the NIST tests and high temperature does not have effect on the randomness of the data even though it has a huge impact on the remanence time and decay rate. Note that we were able to get 420 Kbit of data from 1 Mbit of DRAM cells (XOR'ed of data from Fluctuation Region and Startup Region). Therefore, extrapolating to a system with 8 GB of memory, we could generate approximately 3 GB of random data.

Table I: NIST Statistical Tests Suite Results at Room Temperature Condition.

NIST Tests	P-value	Result
Frequency	0.6247	passed
Block Frequency	0.6734	passed
Runs	0.7731	passed
Longest Run	0.6457	passed
Cumulative Sums	0.7035	passed
Rank	0.8241	passed
FFT	0.4872	passed
Linear Complexity	0.6576	passed
Overlapping Template	0.5594	passed
Non Overlapping Template	0.9184	passed
Approximate Entropy	0.6943	passed
Universal Statistical Test	0.4208	passed
Random Excursions Variant	0.5557	passed
Random Excursions	0.2843	passed
Serial	0.5208	passed

V. Application Areas

The main application for electronic hardware TRNGs is in cryptography and particularly in data encryption, where they are used to generate random cryptographic keys to securely transmit data. Our proposed DRAM remanence-based TRNG is a feasible solution for cryptographic systems since the randomness of the gathered data from DRAM proved based on the results we got from NIST statistical tests. In addition to security applications, we can also use our TRNG for games, gambling, lotteries, draws, random sampling, etc.

DRAMs are excellent TRNG candidates because they are ubiquitous in most computer systems and are a source of a large number of random bits due to the large memory size. However, a drawback of using DRAMs is that the random bits can only be generated when the DRAM is powered off. In normal operation, this would mean that any data that was stored in the DRAM would be lost. We envision, however, that the DRAM-based TRNG procedure that we discussed above would be performed at startup of the system when there is no meaningful data in the system. The random bits would be generated and then collected and stored on some non-volatile medium such as flash or disk. During normal operation, the bits can then be retrieved from non-volatile storage. Note that the DRAMs can generate a very large but still finite set of random bits. Thus, when that set of bits has been exhausted, the random number generation process must be performed again. This will require the DRAM memory to be temporarily saved to non-volatile storage, the random number generation process performed, and the DRAM memory restored, before normal operation can start again. The impact of this interruption can be reduced if the DRAM memory is banked and one bank is

Authorized licensed use limited to: ETH BIBLIOTHEK ZURICH. Downloaded on November 19,2020 at 13:01:18 UTC from IEEE Xplore. Restrictions apply.

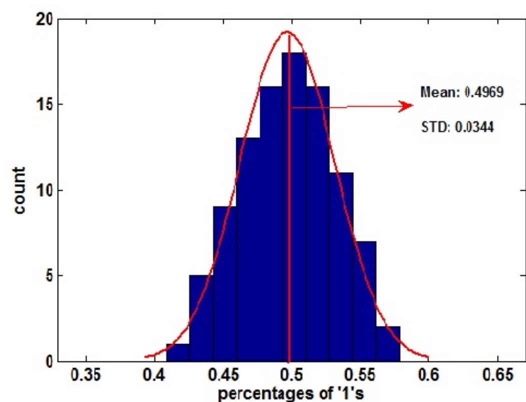

Figure 6: The histogram of the Uniqueness of the data at Fluctuation Region.

turned off at a time resulting a slight reduction in the amount of memory available during this regeneration process.

Since we have a finite set of data, when we regenerate the data, we might get the same set of bits again and that is a security vulnerability because if someone was able to break into the nonvolatile storage where the random bits are stored, they know the sequence of random bits even if we regenerate the bits. For these reasons, we did some experiments where we could generate two different sets of random bits from two different data points from the fluctuation region. Then compared these sets of random bits to see how different they are. Based on the results we got from Hamming Distance which is shown in Figure 6, we can always get new sets of random bits. For Figure 6, we selected 100 pairs of the measurements in the fluctuation region and got the Hamming Distance. The mean of the value and the standard deviation are 0.4969 and 0.0344, respectively.

VI. Conclusion and Future Works

We proposed a new method for generating hardware true random number generators and studied the potential of using remanence effect of the DRAM cells for constructing a high quality TRNG. Our new constructed TRNG has all the characteristics of a strong TRNG and the approach to build this new TRNG is completely novel and concise. The results from NIST Tests clearly shows that the remanence based DRAM TRNG is a promising candidate for cryptographic systems. We also investigated the high temperature effect on data remanence and randomness of the data. Our results shows that even at high temperature, the measurements pass the requirements for randomness. As future work, we would like to investigate DRAM remanence on different memory platforms. We also plan to consider different operating conditions (aging, temperature and voltage) and their impact on our results.

Acknowledgment

This work was supported in part by grants from Comcast Corp. and Honeywell Corp. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of Comcast or Honeywell.

References

[1] H. C. Van Tilborg and S. Jajodia, *Encyclopedia of cryptography and security*. Springer Science & Business Media, 2011.

[2] J. Chan, B. Sharma, J. Lv, G. Thomas, R. Thulasiram, and P. Thulasiraman, "True random number generator using gpus and histogram equalization techniques," in *High Performance Computing and Communications (HPCC), 2011 IEEE 13th International Conference on*, Sept 2011, pp. 161–170.

[3] P. R. Gallagher, "A guide to understanding data remanence in automated information systems," 1991.

[4] P. Gutmann, "Secure deletion of data from magnetic and solid-state memory," in *Proceedings of the 6th Conference on USENIX Security Symposium, Focusing on Applications of Cryptography - Volume 6*, ser. SSYM'96. Berkeley, CA, USA: USENIX Association, 1996, pp. 8–8. [Online]. Available: http://dl.acm.org/citation.cfm?id=1267569.1267577

[5] J. A. Halderman, S. D. Schoen, N. Heninger, W. Clarkson, W. Paul, J. A. Calandrino, A. J. Feldman, J. Appelbaum, and E. W. Felten, "Lest we remember: cold-boot attacks on encryption keys," *Communications of the ACM*, vol. 52, no. 5, pp. 91–98, 2009.

[6] H. Bruyninckx, F. Lafitte, and D. Van Heule, "Safe cryptographic random number generation using untrusted generators," in *Communications (ICC), 2014 IEEE International Conference on*, June 2014, pp. 731–736.

[7] P. Wieczorek, "An FPGA implementation of the resolve time-based true random number generator with quality control," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 61, no. 12, pp. 3450–3459, Dec 2014.

[8] T. Amaki, M. Hashimoto, and T. Onoye, "An oscillator-based true random number generator with process and temperature tolerance," in *Design Automation Conference (ASP-DAC), 2015 20th Asia and South Pacific*, Jan 2015, pp. 4–5.

[9] M. Bhargava, K. Sheikh, and K. Mai, "Robust true random number generator using hot-carrier injection balanced metastable sense amplifiers," in *Hardware Oriented Security and Trust (HOST), 2015 IEEE International Symposium on*, May 2015, pp. 7–13.

[10] M. T. Rahman, K. Xiao, D. Forte, X. Zhang, J. Shi, and M. Tehranipoor, "TI-TRNG: technology independent true random number generator," in *Proceedings of the 51st Annual Design Automation Conference*, ser. DAC '14. New York, NY, USA: ACM, 2014, pp. 179:1–179:6. [Online]. Available: http://doi.acm.org/10.1145/2593069.2593236

[11] M. Bhargava, K. Sheikh, and K. Mai, "Robust true random number generator using hot-carrier injection balanced metastable sense amplifiers," in *Hardware Oriented Security and Trust (HOST), 2015 IEEE International Symposium on*, May 2015, pp. 7–13.

[12] X. Zhang, L. Qi, Z. Tang, and Y. Zhang, "Portable true random number generator for personal encryption application based on smartphone camera," *Electronics Letters*, vol. 50, no. 24, pp. 1841–1843, 2014.

[13] N. Deak, T. Gyorfi, K. Marton, L. Vacariu, and O. Cret, "Highly efficient true random number generator in FPGA devices using phase-locked loops," in *Control Systems and Computer Science (CSCS), 2015 20th International Conference on*, May 2015, pp. 453–458.

[14] F. Tehranipoor, N. Karimian, K. Xiao, and J. Chandy, "DRAM based intrinsic physical unclonable functions for system level security," in *Proceedings of the 25th Edition on Great Lakes Symposium on VLSI*, ser. GLSVLSI '15. New York, NY, USA: ACM, 2015, pp. 15–20. [Online]. Available: http://doi.acm.org/10.1145/2742060.2742069

[15] N. Karimian, F. Tehranipoor, M. Rahman, S. Kelly, and D. Forte, "Genetic algorithm for hardware Trojan detection with ring oscillator network (RON)," in *Technologies for Homeland Security (HST), 2015 IEEE International Symposium on*, April 2015, pp. 1–6.

[16] A. Rahmati, M. Salajegheh, D. E. Holcomb, J. Sorber, W. P. Burleson, and K. Fu, "TARDIS: Time and remanence decay in SRAM to implement secure protocols on embedded devices without clocks," in *USENIX Security Symposium*, 2012, pp. 221–236.

[17] M. Gruhn and T. Muller, "On the practicability of cold boot attacks," in *Availability, Reliability and Security (ARES), 2013 Eighth International Conference on*, Sept 2013, pp. 390–397.

[18] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, and E. Barker, "A statistical test suite for random and pseudorandom number generators for cryptographic applications," DTIC Document, Tech. Rep., 2001.

Authorized licensed use limited to: ETH BIBLIOTHEK ZURICH. Downloaded on November 19,2020 at 13:01:18 UTC from IEEE Xplore. Restrictions apply.