FIRAT UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
TÜRKIYE

GENERATION AND REALIZATION OF TRUE RANDOM NUMBERS BASED ON PHYSICAL UNCLONABLE FUNCTIONS

Ali Yau Yusuf

Master's Thesis

DEPARTMENT OF SOFTWARE ENGINEERING

MARCH 2021

FIRAT UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
TÜRKIYE

Department of Software Engineering

Master's Thesis

GENERATION AND REALIZATION OF TRUE RANDOM NUMBERS BASED ON PHYSICAL UNCLONABLE FUNCTIONS

Author
Ali Yau Yusuf

Supervisor
Assoc. Prof. Seda Arslan TUNCER

MARCH 2021
ELAZIĞ

FIRAT UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

TÜRKIYE

Department of Software Engineering

Master's Thesis

Title:	Generation and Realization of True Random Numbers Based on Physical Unclonable Functions
Author:	Ali Yau Yusuf
Submission:	25.03.2020
Defense Date:	1.03.2021

THESIS APPROVAL

This thesis, which was prepared according to the thesis writing rules of the Graduate School of Natural and Applied Sciences, Fırat University, was evaluated by the committee members who have signed the following signatures and was unanimously approved after the defense exam made open to the academic audience.

Signature

Supervisor:	Assoc. Prof. Seda Arslan TUNCER Firat University, Faculty of Science	Approved
Chair:	Assoc. Prof. Fatih ÖZKAYNAK Firat University, Faculty of Science	Approved
Member:	Assoc. Prof. Erdinç AVAROĞLU Mersin University, Faculty of Science	Approved

This thesis was approved by the Administrative Board of the Graduate School on

..... / ..... / 20 .....

Signature

Doç. Dr. Kürşat Esat ALYAMAÇ
Director of the Graduate School

DECLARATION

I hereby declare that I wrote this Master's Thesis titled “Generation and Realization of True Random Numbers Based on Physical Unclonable Functions” in consistent with the thesis writing guide of the Graduate School of Natural and Applied Sciences, Fırat University. I also declare that all information in it is correct, that I acted according to scientific ethics in producing and presenting the findings, cited all the references I used, express all institutions or organizations or persons who supported the thesis financially. I have never used the data and information I provide here in order to get a degree in any way.

1 March 2021

Ali Yau YUSUF

PREFACE

The My profound gratitude and appreciation goes to Almighty Allah for everything. I want to thank my supervisor, thesis advisor, and guardian, **Assoc. Prof. Seda Arslan TUNCER,** who has taught me more than I could ever deserve. She has shown me what a good supervisor and a person should be. Her door was always open whenever I had a problem or had a question about my research, writing, or any other issue. She insisted and allowed this paper to be my work, but she guided me in the right direction whenever she thought I needed it. Without her guidance and persistence help this thesis would not have happened.

To all my friends, My Late Father, My Dear Mother, My Dear Wife, My Siblings, My Family back home, My New Families in Turkey and the entire Muslim Umma, May ALLAH (S.W.T) Bless Us All with Aljannatul Firdausi. Ameen ya Rabb.

Ali Yau YUSUF

ELAZIG 2021

TABLE OF CONTENTS

Sayfa

DECLARATION ii

PREFACE iii

ABSTRACT vii

ÖZET viii

LIST OF TABLES ix

1. INTRODUCTION 1

2. THEORETICAL CONCEPTS 3

2.1. Random Numbers 3

2.2. Classification of Random Number Generators 4

2.3. Pseudo-Random Number Generators 5

2.4. True Random Number Generators 6

2.5. The Fundamental difference between PRNGs and TRNGs 8

2.6. Comparison of PRNGs and TRNGs 8

2.7. Random Number Generation Methods 9

2.7.1. Middle Square Algorithm 9

2.7.2. Linear Congruential Method 10

2.7.3. Structure in the Generated Numbers 10

2.8.True Random Number Generation Methods 11

3. PHYSICAL UNCLONABLE FUNCTION ARCHITECTURE 13

3.1. Static random-access memory (SRAM) PUF 13

3.2. Butterfly PUF 13

3.3. Ring Oscillator PUF 16

3.4. Arbiter PUF 17

4. NIST STATISTICAL TEST SUITE 18

4.1. Parameter (Mono - Bit) Test (with 1 parameter, y) the idea of the test is to see if 0s and 1s are the same in the q - bit string. 19

4.1.1. Preface 19

4.1.2. Context data relating to randomness: 19

4.1.3. The Analysis Rely: 19

4.2. Parameter test within a pattern (with 2 parameters, b & y) 19
4.2.1. Preface 19
4.2.2. Context data relating to randomness: 19
4.2.3. The Analysis Rely 20
4.3. Runs Test (that involves 1 parameter, y) 20
4.3.1. Preface 20
4.3.2. Context data relating to randomness 20
4.3.3. The Analysis Rely 20
4.4. Test of Long Run 1s in a Pattern (includes 2 Parameters, b & y) 21
4.4.1. Preface 21
4.4.2. Context data relating to randomness 21
4.4.3. The Analysis Rely 22
4.5. Test 5: Binary Matrix Rank Test (including 2 parameters, b & y) 22
4.5.1. Preface 22
4.5.2. Context data relating to randomness 22
4.5.3. The Analysis Rely 22
4.6. Test 6: Discrete Fourier Transform (Spectral) Test (this Test includes 1 parameter, y) 23
4.6.1. Preface 23
4.6.2. Context data relating to randomness 23
4.6.3. The Analysis Rely 23
4.7. Test 7: Non-overlapping Template Matching Test (this test includes 2 parameters, b & y) 24
4.7.1. Preface 24
4.7.2. Context data relating to randomness 24
4.7.3. The Analysis Rely 24
4.8. Test 8: Overlapping Template Matching Test (this test includes 2 parameters, b & y) 24
4.8.1. Preface 24
4.8.2. Context data relating to randomness 25
4.8.3. The Analysis Rely 25
4.9. Test 9: Maurer's "Universal Statistical" Test (this test includes 1 parameter, x) 25
4.9.1. Preface 25
4.9.2. Context data relating to randomness 25
4.9.3. The Analysis Rely 26

4.10. Test 10: Linear Complexity Test (this test includes 2 parameters, b & y) 26
4.10.1. Preface 26
4.10.2. Context data relating to randomness 26
4.10.3. The Analysis Rely 26
4.11. Test 11: Serial Test (2 P-values each one with 2 parameters, b & y) 27
4.11.1. Preface 27
4.11.2. Context data relating to randomness 27
4.11.3. The Analysis Rely 27
4.12. Test 12: Approximate Entropy Test (1 P-rate with 2 parameters, b & y) 28
4.12.1. Preface 28
4.12.2. Context data relating to randomness 28
4.12.3. The Analysis Rely 28
4.13. Test 13: Cumulative Sums Test (2 P-rates each one with 1 parameter, y) 29
4.13.1. Preface 29
4.13.2. Context data relating to randomness 29
4.13.3. The Analysis Rely 29
4.14. Test 14: Random Excursions Test (8 P-values each one with 2 parameters b & y) 29
4.14.1. Preface 29
4.14.2. Context data relating to randomness 29
4.14.3. The Analysis Rely 30
4.15. Test 15: Random Excursions Variant Test 30
4.15.1. Preface 30
4.15.2. Context data relating to randomness 31
4.15.3. The Analysis Rely 31
5. RESULTS AND DISCUSSIONS 32
5.1. Puf Architecture and Its Implementation in Fpga Environment 32
5.2. Ring Oscillator Based Puf 32
6. CONCLUSION 36
REFERENCES 37
CURRICULUM VITAE

ABSTRACT

Generation and Realization of True Random Numbers Based on Physical Unclonable Functions

Ali Yau Yusuf

Master's Thesis

FIRAT UNIVERSITY
Graduate School of Natural and Applied Sciences

Software Engineering
March 2021, Pages: xi + 39

The need for random numbers and random number generators is increasing day by day. Random numbers are compiled, especially in computer science and such as computer compilers and cryptographic systems. Random numbers can be generated by using some numerical operations to an initial value, often called a kernel. We can also generate random numbers in many ways, such as using a specific algorithm, a mathematical formula, predetermined tables, or using natural physical events that do not have a deterministic character. The random number generation process can be reproducible when the same channel is used. Therefore, the output of the random number generator may not be truly random. Random number sequences are used in many fields since they consist of numbers that are statistically self-reliant of each other and have no correlation. Random number generators are also used in computer simulations, numerical analysis applications, statistical analysis, applications using the Monte Carlo method, and especially in encryption. For example, the reliability of cryptographic algorithms depends on the numbers generated by random number generators. If these numbers are statistically random, that is, if the recent cannot be determined by looking at the previous outputs, then the generator has excellent statistical properties. In other words, proper encryption requires the right Random Number Generator (RNG). It is possible to divide the RNGs into True-RNGs and Pseudo-RNGs. Depending on the purpose of the application, one of these two structures is preferred. While the actual TRNGs are based on the measurement of natural processes such as noise, the so-called TRNGs use deterministic methods such as numerical algorithms. While true RNGs are used in applications where security is essential before mentioned as encryption, the performance of so-called RNGs is sufficient to be used in computer simulations.

In this thesis, Generation and Realization of True Random Numbers Based on Physical Unclonable Functions is presented. Both PUF structures were performed in FPGA and the numbers produced were tested with the NIST test suite. Successful results were obtained from the tests.

Chapter 2 of this work shows the definition of random numbers and their classification, current usage areas, and the reviews about random numbers while Chapter 3 shows PUF architecture. In chapter 4 the National Institute of Standards and Technology (NIST-Test), while in chapter 5 PUF architectures and their implementation in the FPGA environment is discussed and a short evaluation in the conclusion part is given.

Keywords: Random Numbers, Random Numbers Generation, FPGA, PUF, NIST Test

ÖZET

Fiziksel Kolonlanamaz Fonksiyonlar Tabanlı Gerçek Rasgele Sayı Üretimi ve Gerçekleştirilmesi

Ali Yau Yusuf

Yüksek Lisans Tezi

FIRAT ÜNIVERSITESI
Fen Bilimleri Enstitüsü

Yazılım Mühendisliği Anabilim Dalı
Mart 2021, Sayfa: xi + 39

Rasgele sayılar ve rasgele sayı üreteçlerine olan ihtiyaç her geçen gün artmaktadır. Rastgele sayılar, özellikle bilgisayar bilimlerinde ve bilgisayar derleyicileri ve şifreleme sistemleri gibi derlenir. Rastgele sayılar, genellikle çekirdek olarak adlandırılan bir başlangıç değerine bazı sayısal işlemler kullanılarak üretilebilir. Ayrıca, belirli bir algoritma, matematiksel formül, önceden belirlenmiş tablolar kullanma veya deterministik bir karaktere sahip olmayan doğal fiziksel olayları kullanma gibi birçok şekilde rastgele sayılar üretebiliriz.

Aynı kanal kullanıldığında rastgele sayı oluşturma işlemi tekrarlanabilir. Bu nedenle, rasgele sayı üretecinin çıktısı gerçekten rasgele olmayabilir. Rastgele sayı dizileri, istatistiksel olarak birbirlerinden bağımsız olan ve hiçbir korelasyonu olmayan sayılardan oluştuğu için birçok alanda kullanılır. Rastgele sayı üreteçleri ayrıca bilgisayar simülasyonlarında, sayısal analiz uygulamalarında, istatistiksel analizlerde, Monte Carlo yöntemini kullanan uygulamalarda ve özellikle şifrelemede kullanılmaktadır. Örneğin, kriptografik algoritmaların güvenilirliği rastgele sayı üreteçleri tarafından üretilen sayılara bağlıdır. Bu sayılar istatistiksel olarak rasgele ise, yani son çıktılar önceki çıktılara bakarak belirlenemiyorsa, jeneratör mükemmel istatistiksel özelliklere sahiptir. Başka bir deyişle, uygun şifreleme için doğru Rasgele Sayı Üreticisi (RNG) gerekir. RNG'leri True-RNG'lere ve Pseudo-RNG'lere bölmek mümkündür. Uygulamanın amacına bağlı olarak, bu iki yapıdan biri tercih edilir. Gerçek TRNG'ler gürültü gibi doğal süreçlerin ölçülmesine dayanırken, TRNG'ler sayısal algoritmalar gibi deterministik yöntemler kullanırlar. Gerçek RNG'ler, daha önce şifreleme olarak anılan güvenliğin gerekli olduğu uygulamalarda kullanılırken, RNG'lerin performansının bilgisayar simülasyonlarında kullanılması yeterlidir.

Gerçek rasgele sayı üreteçleri olarak kullanılan bu özellikleri sağlayan fiziksel klonlanamayan fonksiyonlar, üretim sürecindeki kontrol edilemeyen varyasyonlar nedeniyle talaşa özgü ve benzersiz çıktılar üreten yapılardır. Bu tezde, Fiziksel Klonlanamayan Fonksiyonlara Dayalı Gerçek Rasgele Sayıların Üretilmesi ve Gerçekleştirilmesi sunulmaktadır. Her iki PUF yapısı da FPGA'da yapıldı ve üretilen sayılar NIST test takımı ile test edildi. Testlerden başarılı sonuçlar alınmıştır.

Bu çalışmanın 2. bölümünde rastgele sayıların tanımı ve sınıflandırılması, mevcut kullanım alanları ve rastgele sayılarla ilgili incelemeler, 3. Bölüm ise PUF mimarisini gösterir. Bölüm 4'te Ulusal Standartlar ve Teknoloji Enstitüsü (NIST-Testi), 5. bölümde PUF mimarileri ve bunların FPGA ortamında uygulanması tartışılmış ve sonuç kısmında kısa bir değerlendirme verilmiştir.

Anahtar Kelimeler: Rasgele Sayılar, Rasgele Sayılar Üretimi, FPGA, PUF, NIST Testi

LIST OF TABLES

Page

Table 2.1. Comparison of TRNG and PRNG based on applications 9
Table 2.2. Comparison of TRNG and PRNG based on characteristics 8
Table 4.1. The values of q, G, Q, and L 21
Table 4.2. Representative set points of G & L 21
Table 4.3. A dynamic look-up table 25
Table 4.4. A dynamic look-up table 30
Table 5.1. NIST test results of the implemented models are as in the table ……...35

LIST OF FIGURES

Page

Figure 2.1. Classification of Random Number Generators 5
Figure 2.2. Physical TRNG general design 11
Figure 3.1. Butterfly PUF structure 14
Figure 3.2. A Cross-coupled Bistable Circuit 15
Figure 3.3. Ring oscillator 16
Figure 3.4. Referee PUF 17
Figure 3.5. Feed Forward Referee PUF 17
Figure 5.1. PUF basic structure 32
Figure 5.2. Basic RO structure 33
Figure 5.3. RO based PUF structure 33
Figure 5.4. RO structure realized in FPGA 34
Figure 5.5. Part of the random numbers obtained with the RO-PUF structure 5.2. Referee PUF 34
Figure 5.6. Referee PUF structure 34
Figure 5.7. Realization of Referee PUF in FPGA and storing the produced numbers in memory 35
Figure 5.8. Some of the numbers produced by the referee PUF 35

SYMBOLS AND ABBREVIATIONS

Abbreviations

ASIC : Application-specific integrated circuit
CMOS : Complementary metal–oxide–semiconductor
DRBG : Deterministic Random Bit Generator
FPGA : Field Programmable Gate Array
FRO : Free-running oscillators
IoT : Internet of Things
LC : Linear Congruential
LGA : Lotteries and Gaming Authority
NIST : National Institute of Standards and Technology
PLL : Phase Locked Loop
PRNG : Pseudo-Random Number Generation
PUF : Physical Unclonable Function
RNG : Random Number Generation
RO : Ring oscillator
SRAM : Static random-access memory
TRNG : True Random Number Generation

1. INTRODUCTION

A random number is a number chosen somehow by accident from certain arrays provided to replicate the underlying distribution by choosing a broad range of these numbers. These numbers are also required to be autonomous in general, to ensure that no distinction was made between successive numbers. In several fields, random numbers are used, such as cryptography, computer and simulation of Monte Carlo, industrial checking and marking danger games, gambling, etc. Random numbers are hard to analyse since computers function in a codified form and they do not generate random numbers. Rather, they are built using a true random number generator (TRNG) that really functions by evaluating excellent and substantial methods [1].

The randomness of TRNG can be precisely, scientifically defined, and measured. The highly valuable random numbers are the realistically proven RNGs, which, at state of the art, seem to be possible only by exploiting the randomness inherent in specific quantum systems. On the other hand, the existing industry-standard determines the use of RNGs based on free-running oscillators (FRO) whose play-dimness is obtained from electronic sounds given in the logic circuits which cannot be maliciously random but serve more straightforward technological implementation. The FRO technique is currently used in 3rd and 4th generation FPGA and ASIC hardware, which cannot be suited for quantum RNGs. In this Thesis, we compare the weak and energetic aspects of both approaches. True random numbers and physical non-deterministic random number generators appear to have been of increasing importance. Random numbers are essential for cryptography (mathematical, stochastic and quantum), Monte Carlo calculations, mathematical computations, quantitative analysis, randomly generated algorithms, lottery, etc [1].

Currently, TRN is one of the most fundamentally engineered in cryptography as well as multiple programs for our everyday lives, portable communication systems, e-mail access, electronic transfers, digital payments, ATMs, e-banking, internet buying, and selling, point of sale, debit cards, wireless keys, general cybersecurity, distributed power grid security, etc. Without widespread loss in the remainder of the article, we will suppose that generators generate random bits. In entries at which likelihoods are crucial, the sources of randomness (maybe involved) have to be provably random, and then the entire structure of equations will collapse. In cryptography, where, because of the Kerkhoff principle, the public besides a few secrets (key or other information) known to just the addressee as well as the beneficiary knows both sections of protocols, it becomes apparent that the unknown party can never determine the secret, i.e., it has to be random [1].

The lottery is still another severe source of income where random numbers are essential. Due to the significant amount of cash engaged (approximately at USD 6 billion every year only online and only in the US, some nations have set specific criteria for random number generators for the use in online gambling and lottery machines and have also awarded certificates. Random Number

Generations, which does not adhere, with that kind of activity, may not have been lawfully used for gambling business purposes. These rules were forwarded to guarantee fair play by suppliers and to discourage gamers from manipulating the system by anticipating outcomes. For instance, the Lotteries and Gaming Authority (LGA) of Malta has established a list of criteria for RNGs set out in the Remote Gaming Regulations Act. Random number generators have been a long-standing activity of scientists and inventors [1, 2, 3, 4].

True random number generators are not deterministic but use true physical processes as a source of the noise. For random numbers to be used in computer science the random number generator must have a long repetition time and the generated numbers must be completely independent from one another [5].

PUFs that are specific to physical structure (PUF output) varies, therefore they produce true random numbers. Because of this feature, they are widely used in computer science. In the literature, there are Ring Oscillator-based PUF (Ring oscillator-PUF), Referee PUF (Arbiter PUF), and SRAM PUF, (Butterfly PUF), Latch PUF, Flip Flop PUF, and Anderson PUF structures [6, 7, 8, 9].

The success of tests with successful results from NIST facilities is higher than 96% in RO-PUF designs and it requires all RO to be mutually symmetrical. Besides, the number of RO pairs is limited to maintain the independence of the bits in the PUF response, and to overcome this disadvantage, a specific part of the RO pairs were used as PUF output [10]. This approach is more efficient than the classical approach and the design has been realized in FPGA [11]. The design of RO structures with different logic styles was developed in CRO PUF using these structures. More results that are reliable have been shown with CRO PUF developed with CMOS technology [12].

A new type of referee PUF has been introduced, which uses 4x4 key patterns instead of the traditional Referee PUF's 2x2 structure. Uniqueness, unclonability, unpredictability, and robustness parameters were examined and successful results were obtained [13]. Cryptographically secure so-called random number generator consisting of the (DRBG) and the Entropy module are designed. Static random-access memory and Pufs have been used as the source of entropy in the number generator. Latest technology was used to test the numbers produced, a reliable and Successful PUF architecture based on SR-Latch has been suggested many times and changes have been made [14, 15].

2. THEORETICAL CONCEPTS

2.1. Random Numbers

A random number can be defined, as a series of steps in which each step is likely to be the same, and the consecutive steps are entirely independent of each other. There is a need for random numbers in many fields of computer science; it also has an essential role in encryption algorithms. Random numbers are significant for the confidentiality and reliability of the encryption process. Keys and cryptographic protocols require random bit sources that no enemy can predict. Similarly, random numbers are needed when programming games or modelling random events during a simulation. These numbers are also expected to be random. It is, therefore, undesirable that generators always generate the same random number. Because in this case, the attacker who seized the key by attacking the system once does not have to waste time finding the key for the next attacks. Similarly, in cases where different numbers generations, but it is possible to estimate the numbers, it can also reduce the complexity of the system, and the system can be attacked in a shorter time than expected [16, 17].

Computers are machines that design certain things that will happen at every step without prejudice to randomness in terms of their design and structure. In this sense, it is challenging to generate random information on computers, and it is necessary to comply with specific conditions to obtain the desired random number sequences. The essential summary is required as follows [18].

- The repetition period of the random number generator must belong. The self-iteration period for a generator to produce n numbers must be very long. Random number generators provide figures based on a mathematical function. Therefore, the role can repeat itself over a period. Thus, the iteration period of the generated algorithm must be very long.
- The number of sequences obtained in the generation of n numbers of the random number generator must be successively independent of each other.
- The elements of n numbers obtained at time t in the random number generator should not show clustering in period's ti, ti + k.
- In the random number generator, the number must be reproducible and reproducible. If an algorithm runs for each period t, the generated sequences must be equal.
- The random number generator must not depend on the type of computer on which it runs on. It should be possible to produce under the general principle.
- Random number generators should be flexible enough to be easily fitted to the asymptotic distribution of any X variable in the system. Derived arrays should be able to switch to the intended format easily.
- There should be no dependence on the previous and next values of numbers in the series of numbers generated by random number generators.

The cryptogenic security systems that are using random numbers generated on both hardware and software basis concerns about safeguarding the passcode. Development of unregulated conditions of passwords used in computer protection systems outside the system reduces the reliability of the system. Therefore, if the keys are implemented in a chip (cryptographic system on chip), the system is more secure. It is, therefore, essential to implement random number generators with programmable hardware such as FPGAs that may be programmed repeatedly for application-specific applications. Because it is fast, inexpensive, easily programmable, it is also widely used in areas such as network, availability, multimedia, and control. Field Programmable Gate Arrays (FPGA) are ideal environments for developing random number generators because they provide great flexibility during design and are capable of parallel processing [16].

The pseudo random number generators (PRNGs) are commonly used to realize the Lagged Fibonacci generator, Linear Congruential Generator, Cellular Automata, and Linear Feedback Shift Generator. In addition to being realized separately, mixed PRNGs can be designed by using one or more of these structures. Hybrid PRNG design using two or more LFSR constructs was conducted in the FPGA environment performed Fibonacci and Galois compositions, which are two different models of LFSR with excellent and straightforward cryptographic properties, and in trials such as Diehard and FIPS, promising outcomes have been obtained. Linear-feedback shift register (LFSR) hybrid systems are modified as random number generators for the Shrinking and Alternating Phase Generator. In addition to PRNG studies relevant to TRNGs that must be carried out in hardware due to their structure, performance comparisons and review of these two designs are given. Three GRPS were performed using chaotic hash and mouse movements. In another study, random number generators have been developed by using mobility in market purchases. In an FPGA environment, TRNG was performed using Phase Locked Loop (PLL), chaotic system, and eye iris structures. The randomness of the numbers obtained because of all these implementations was accepted in the literature and proved by tests such as NIST and Diehard [16].

2.2. Classification of Random Number Generators

For cryptographic applications, the purpose of using RNGs is to produce random numbers (keys). Using cryptographic random numbers increases the encryption strength. Random number generators are used to create such numbers in computer science; they are classified, as shown in Figure 2.1. TRNGs with a more straightforward structure is sufficient and adequate for many applications, while any form of weakness or deficiency in the RNGs leads to system failure if we applied in cryptographic systems. Therefore, the need to generate adequate random numbers and increasing the strength adequately should be given due consideration and priority [17, 18].

Figure 2.1 below, shows the Classification of Random Number Generators.

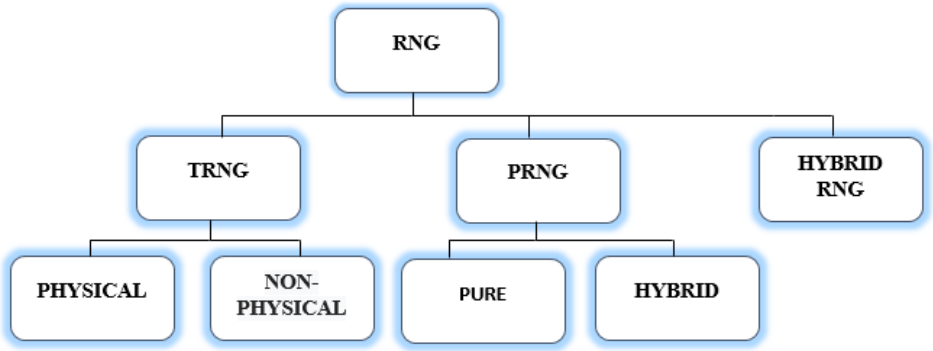

Figure 2.1. Classification of Random Number Generators [18].

Although PRNGs can be inserted into base production software and freezer, which is a sub-classification, according to Figure 2.1, PRNGs can only be found under Hardware, where physical events are used as the source of randomness in hardware-based random number generation. These sources of randomness can be summarized as follows:

- Time elapsed throughout the secretion of electrostatic within radioactive degradation
- Thermal noise from a resistor or diode element
- Parameter instability among independently operating oscillators
- Charging time of the semiconductor capacitor for a particular time
- Air turbulence on a hard disk
- Audio sound from a microphone or image from a camera in software-based Random number generators, the sources of randomness are deterministic and can be categorized as follows [20].
    - System clock parameter
    - Mouse gestures
    - Input / Output buffer content
    - User input values
    - Operating system related events, system load or network usage statistics

2.3. Pseudo-Random Number Generators

Systems that generate numbers with a deterministic algorithm implemented by finite state machines. Compared to TRNGs, it has some advantages such as easy realization and low-cost production. However, the algorithms used are deterministic, and therefore the outputs are not specifically random as desired. When the input algorithm is known, the next outputs can be estimated

by looking at the value at any given moment. This also restricts the use of encryption algorithms that require privacy [19, 21].

Besides, they are preferred where the randomness of lower statistical quality is sufficient, such as numerical analysis or physical process modelling. Even though the term' pseudo' asserts, pseudo-random numbers may not be random in just the sense you may anticipate, at least because you are used to dice rolls or lottery tickets. PRNGs are algorithms, which use algorithms or merely non-calculated tables to generate patterns of numbers that tend random. The linear congruential method is indeed a grand instant of PRNG. Much thought has gone into pseudo-random number theory, and advanced algorithms for extracting pseudo-random numbers have become so perfect that figures look completely unclear as though they were random [19, 21].

Lately, there have been two approaches to random number generation: algorithmic (pseudorandom) and physical (non-deterministic) methods. Pseudo-random number generators (PRNG) are well known in the field of art, and PRNG is nothing other than a mathematical formula, which generates a probabilistic periodic set of numbers, which is entirely decided by the origin point named the seed [21]. By default, these generators are not likely to be random. In action, PRNGs have a perfect balance between 0's and 1's (zero bias) as well as reliable long-range comparisons that compromise cryptographic stability and can appear as unintended errors in Monte Carlo simulations and data analysis. Although most modern PRNGs pass all proven statistical analysis, there are theories that some PRNGs are much better than others are. The reality is that every PRNG reveals its weakness in a particular specific application. However, PRNGs are always discovered to be the cause of incorrect stochastic simulations and calculations. Just like with cryptographic reasons, all significant PRNG families were cryptanalyzed. The advantages of PRNGs are their inexpensive, simplicity of execution and usability, particularly in a CPU-available atmosphere or something like a PC, but one has to be careful when it comes to using PR numbers for experiments cryptography and any use [22].

2.4. True Random Number Generators

According to Kirchhoff’s law, the definition of a random number generator proper for cryptography has to include that even though every information is identified as to the generator (schematics, algorithms, etc.); it also has to produce completely random and unpredictable bits. Unlike PRNGs, physical (true, hardware) random number generators derive randomness from physical processes that behave in a fundamentally non-deterministic way, making them better candidates for true random number generation. TRNGs are systems that generate numbers with an algorithm whose inputs are not deterministic and use the randomness of natural processes. Since the numbers in the output are random, they are widely used in encryption [23].

They can be implemented in two different techniques, hardware, and software-based. Hardware-based generators utilize the randomness of physical phenomena, such as the thermal noise of a semiconductor diode or a resistor, the phase noise of an oscillator; the time elapsed between the propagation of particles during radioactive decay. Since the signals generated by these processes may also be related to each other, the output can be subjected to a simple algorithm again to ensure complete randomness. Software-based generators are based on computer-based events such as the system clock, time between mouse movements, and access to the hard disk. Implementing software-based generators is more complicated and less reliable than hardware-based generators. For example, it was found that the data in which Netscape's random number generator is based on the time of day and the process number [24].

Random numbers are needed in many areas such as; Cryptography, Monte Carlo computational and simulation, industrial testing and labelling, hazard games, gambling, etc. It has been assumed that random numbers cannot be computed, because computers operate in a deterministic manner, they cannot produce random numbers. Instead, random numbers are best obtained by using a physical (true) random number generator (TRNG) that operates by measuring a well-controlled and specially prepared physical process [25].

The randomness of the TRNG can be precisely, scientifically defined, and measured. Highly valuable are the realistically proven RNGs, which, at state of the art, seem to be possible only by exploiting the randomness inherent in specific quantum systems. On the other hand, the existing industry-standard determines the use of RNGs based on free-running oscillators (FRO) whose play-dimness is obtained from electronic sounds given in logic circuits so that it cannot be solely demonstrated to be maliciously random but serve more straightforward technological implementation. The FRO technique is currently used in 3rd and 4th generation FPGA and ASIC hardware, which is not suited for quantum RNGs [26].

Subsequently, we address many explanations in when the use of true RNG is critical and how it can exponentially increase the security of cryptographic systems, address technical, and research issues that pattern the widely accepted method of TRNGs. True random numbers and physical non-deterministic random number generators (RNGs) appear to have been increasing importance. Random numbers are essential for cryptography (mathematical, stochastic and quantum), Monte Carlo calculations, mathematical computations, quantitative analysis, randomly generated algorithms, lottery, etc. Today, true random numbers which are most critically re-engineered in cryptography and its many programs to our daily lives: portable communication systems, e-mail access, electronic transfers, digital payments, ATMs, e-banking, internet buying, and selling, point of sale, debit cards, wireless keys, general cybersecurity, distributed power grid security, etc. Without a widespread loss in the remaining of the article, we will suppose that generators generate random bits [27].

In entries at which likelihoods are crucial, the sources of randomness (maybe involved) have to be provably random, and then the entire structure of equations will collapse. In cryptography, where, because of the Kerckhoff principle, the public besides a few secrets (key or other information) known to just the addressee as well as the beneficiary knows both sections of protocols, it becomes apparent that the unknown party can never determine the secret, i.e., it has to be random [27].

2.5. The Fundamental difference between PRNGs and TRNGs

It is easy to comprehend if we equate machine-generated random numbers with rolls of a die. Since PRNGs generate random numbers using statistical methods or semi-calculated entries, utilizing one relates to the one who rolls a die several times and publishes down the outcomes. You get the next one on the list every time you ask for a die roll. Mostly, the results appear random, yet they are predetermined. TRNGs operate by having a computer to roll a die or, perhaps generally, choose another mathematical theorem, which is easier to communicate to a device than a die [28].

PRNGs are productive, which means that they will always generate several numbers in a short time, and probabilistic, which means that a specified pattern of numbers could be replicated at such a future date even when the baseline in the pattern is recognized. Effectiveness is indeed a great feature if your application needs many numbers, and determinism is helpful if you have to playback the same pattern of numbers at a future stage. PRNGs are also commonly periodic, which indicates perhaps the sequence will eventually repeat itself. Although periodicity is hardly ever a desirable feature, modern PRNGs have a timeline that is so long that it can be overlooked for most practical purposes [29, 37].

Such characteristics render PRNGs ideal for applications where many numbers are needed and where it is helpful to replay the same sequence easily. Simulation and simulation programs are typical examples of such claims. PRNGs are not appropriate for applications where numbers must be completely random, such as data encryption and gambling. It must be remembered that although there are successful PRNG algorithms, they are not used, but it is simple to get future problems still [34].

2.6. Comparison of PRNGs and TRNGs

Some features make TRNGs appropriate for the collection of tasks that PRNGs are not adapted for, like data encryption, gaming, and gambling. Alternatively, the low performance and un-deterministic aspect of the TRNGs render them less appropriate for simulation and modelling applications that often necessitate further data than can be generated with the TRNG.

Table 2.1. below shows the overview of which claims are made according to which generator type suit is the best for the given applications.

Table 2.1. Comparison of TRNG and PRNG based on applications [34].

Application	Most Suitable Generator
Lotteries and Draws	True Random Number Generator
Games and Gambling	True Random Number Generator
Random Sampling (e.g., drug screening)	True Random Number Generator
Simulation and Modelling	Pseudo-Random Number Generator
Security (e.g., generation of data encryption keys)	True Random Number Generator
The Arts	It depends

Table 2.2 below, explains Comparison of TRNG and PRNG based on Characteristics [34].

Table 2.2. Comparison of TRNG and PRNG based on Characteristics [34].

Characteristic	Pseudo-Random Number Generators	True Random Number Generators
Efficiency	Excellent	Poor
Deterministic	Determinism	Nondeterministic
Periodicity	Periodic	Aperiodic

2.7. Random Number Generation Methods

A random number is defined as a series of digits, whether its figures have the same probability of occurrence and whether successive digits are independent of each other. It does a mathematical method if random numbers are generated without relying on a random number table. Once the initial value of this random number is determined, the specified algorithm can generate other random numbers. Random numbers, which can be created without using the previously made random number sequence, and which have a short or long period up to the repetition of a number in the chain, are considered pseudo-random numbers. The so-called random numbers are true. We prefer beef to true random numbers. Random numbers can be prevented again for control purposes, and so-called random numbers can take this requirement [30, 31,32, 33,34].

There are several methods used in random number generation. Some of these are:

- Middle Square Algorithm
- Linear Congruential Method

2.7.1. Middle Square Algorithm

Von Neumann introduced the middle square approach for producing pseudorandom numbers in 1949.

4373-19123129-1231

The steps are easy: take a seed count 4 digits long, square it, and subtract the middle 4 digits, which will be the next seed. For instance: A random start value is selected (if the random number is 4-digit, a 4-digit number must be chosen) It seems spontaneous enough, however, is it? A successful pseudorandom generator should not rely very much on the seed. It should also have a fixed length of time, that is, to reach any number in its range before it starts to replicate itself. The transformation chart would have been just one big cycle. Therefore, the middle square generator has both quite poorly: based on the seed, and you can get into a period very quickly. For instance, using a 2-digit seed, we can 50 – 2500 - 50. Some other instances could be 60 or 24 – 576 – 57 – 3249 - 24 [32].

2.7.2. Linear Congruential Method

D. H. Lehmer in 1948 suggested a linear congruential generator as nothing more than an origin for random numbers. Throughout this generator, every number decides its replacement. The form of the generator is [33].

$$X_{i+1} = (aX_{i+c}) \bmod m, \text{with } 0 \le X_{i \le m}$$

M is called modulus. $X0$, a, and c are known as the seed, multiplier, as well as the increment. For instance, perceive $m = 31, a = 7, c = 0$ and start with $X0 = 19$. The next numbers in the series are.

$$9,1,7,18,2,14,2,14,5,28,10,8,25,20,16,19$$

Therefore, of course, the sequence is starting to repeat at this point. If we were to take $a = 3$ instead of $a = 7$, we would have had.

$$26,16,17,20,25,13,8,24,10,30,28,22,4,12,5,15,$$
$$14,11,2,6,18,23,7,21,1,3,9,27,19$$

From the simplified instance described above, we may conclude that modulus, multiplier, and increment play a significant role in the length of time of the linear congruential generator.

2.7.3. Structure in the Generated Numbers

The idea under the random number generated would be that the numbers ought to be random. This implies that perhaps the numbers must tend to become distributional autonomous of each other; that is, the serial correlations should be small. Too bad the construction of the series is (that is, how much this condition allows the production to seem non-random) depends on the lattice structure [34].

Look at the output of the generator with $m = 31$ and $a = 3$ beginning at $x0 = 19$. Successive pairs of stories.

$$(27,19),(19,26),(26,16)$$

The consecutive pairs of random numbers are only on three rows. The numbers generated are still not random. Instead, they have to be linked. Out of a production standpoint, we can infer that a generator with a limited number of lines does not well fill the area and has a weak lattice design.

McLaren and Mars glia (1965) suggest that perhaps the data flow of a linear congruential random number generator must be shunted using another generator to permute subsequence from either the first generator. In this way, the time can be expanded, and the shuffling of the production can break down the lousy lattice structure [34].

2.8. True Random Number Generation Methods

The earliest methods for true generating random numbers, such as dice, coin flipping and roulette wheels, are still used today, mainly in games and gambling as they tend to be too slow for most applications in statistics and cryptography. A physical random number generator can be based on an essentially random atomic or subatomic physical phenomenon whose unpredictability can be traced to the laws of quantum mechanics. Figure 2.2 below, shows the physical True Random Number Generators overall methods structure [34].

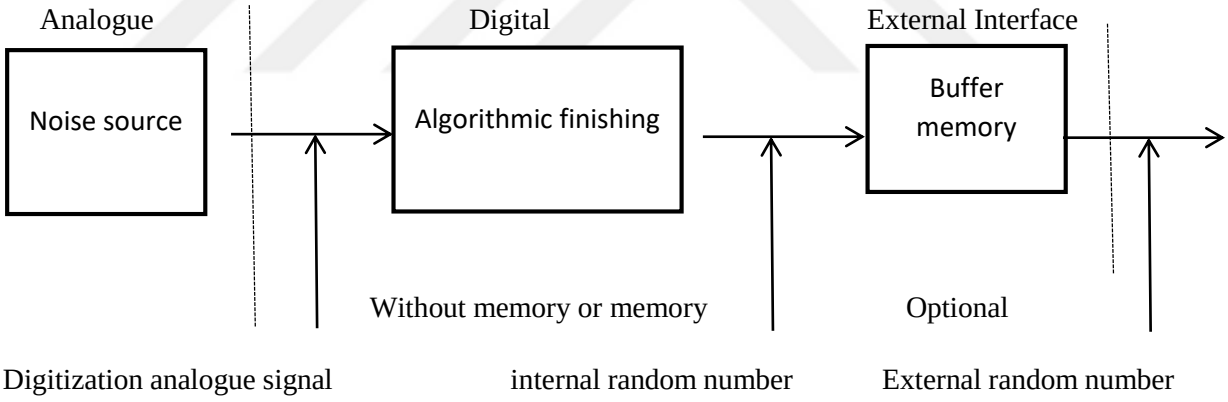

Figure 2.2. Physical TRNG general method structure [34].

In figure 2.2 above, the digitized analogue signals to the digitized values or DAS for short, it is called as a random. Reduce potential weaknesses of DAS random numbers Algorithmic finishing is applied with paste. However, during this application, the output bit rate should be noted that it decreases the working speed. Strong noises resources do not need algorithmic post processing. Most of the algorithmic finishing applications used are Von Neumann, XOR and Hash functions. For example, Von Neumann application, 11 and 00-bit sequences in the sequence are discarded. Other 01 (0) and 10 (1) what is the first start bit of the bit sequences is taken [34].

Many TRNG designs have been proposed in the literature [7, 34, 35]. These designs can vary significantly according to entropy sources and production techniques. Every design has its strengths

and weaknesses. Some of these features are performance related and others are related to security and robustness. The features expected from a TRNG can be summarized as follows.

- From a practical point of view, TRNGs are generally created using inexpensive digital circuits available. In the literature, it is desired to perform TRNG application using only digital design techniques. With these digital circuits, true numbers can be easily generated and GROSS applications are possible in popular adjustable platforms (FPGA etc.).
- The production mechanism (sampler) should be simple (easy to analyse), protected, and exemplify the source of entropy. In other words, the unpredictability of TRNG should not be based on the complexity of the production mechanism, but only on the unpredictable source of entropy.
- Efficient and efficient design should be provided with energy consumption, high output speed. The use of amplifiers and other analogue components should be avoided, if possible. Analogue components consume a lot of energy and are difficult to analyse.
- It should be simple enough to ensure a good analysis of the design. Diehard and NIST test packages are mostly used to confirm the output of TRNG.

TRNG Designs in the literature typical implementation consists of three parts. First, a physical random noise source. Second, the digitizer gathers randomness from the entropy source and generates a binary bit sequence and the third one is, finishing improves the statistical properties of the random bit stream by reducing correlation and compressing data. Many TRNG applications given below are in the literature [36, 37, 38, 39].

- Bagini and Bucci Design
- Intel TRNG Design
- The Fischer – Drutarovsk´y Design
- Tkacik TRNG Design
- Epstein TRNG Design
- Kohl Brenner TRNG Design
- Figaro TRNG Design
- Sunar TRNG Design
- Vasyltsov TRNG Design

3. PHYSICAL UNCLONABLE FUNCTION ARCHITECTURE

3.1. Static random-access memory (SRAM) PUF

A Physical Unclonable Function (PUF) is an intangible physical entity in the given process. PUFs use digital logic differences which arise spontaneously during the manufacture of semiconductors, providing strange electrical properties to that transistor and hence a distinct personality. Intrinsic-ID designed SRAM PUF to separate devices like microcontrollers from each other due to the actions of normal SRAM memory that is accessible on every digital chip. Any SRAM cell, resulting from random variations in the operating voltage, has its desired condition each time the SRAM is operated. This randomness is reflected as in "unreferenced" SRAM memory start-up values. Thus, a special and random sequence of 0's and 1's is generated by SRAM and this pattern is like the fingerprint of a chip since it is unique to a specific SRAM and indeed a specific chip [40, 41, 42].

SRAM PUF output is a noisy fingerprint so it needs more validation to convert it into a stable and strong key vault. It is necessary to recreate a similar cryptographic key during each period and in certain environmental situations [43, 44].

In contrast to conventional key storage in non-volatile memory, this method of deriving a key from the SRAM features has strong security strengths. Because the key is not indefinitely retained, If the system is not working (no keys at rest), it is not available and can thus not be identified by an intruder who opens the system and exploits the action's potential [44].

Intrinsic ID’s SRAM PUF technology has been tested under extreme conditions of temperature, aging, and usage and has been fine-tuned over a decade to ensure that it works reliably over time in every imaginable scenario for all major IC technologies it operates consistently over time with all big IC innovations in any possible situation.

3.2. Butterfly PUF

For several FPGA IP suppliers, the security of hardware models is the most significant prerequisite. IPs that can be utilized in SRAM FPGAs seem to be more sensitive to threats because the processing bit stream must be stored externally to the chip on non-volatile memory (NVM). Towards this point, developers of FPGA also suggested multiple strategies depending upon the bit stream cryptographic algorithms. Nevertheless, as the password (in the nature of the password vault) is available on the device during such a threat, these conventional encryption methods based on NVM/fuses are susceptible to intrusive threats. Nevertheless, as the password (in the nature of the password vault) is available on the device during such a threat, these conventional encryption methods based on NVM/fuses are susceptible to intrusive threats. Keys extracted via PUF are

unstable yet will be retrieved once appropriate. They do not have to be saved in NVM and when a chip is turned off with no keys and an intruder can be detected [44].

The initial experiment of PUF on the FPGA was suggested in 2007 [44] and was SRAM PUF, which uses invalid SRAM start-up values. Uninitialized SRAM memory is not provided by most FPGAs. In certain SRAM FPGAs, during initialization, memory is forcefully set to a consistent point. An often-promising substitute so far is the Butterfly PUF, recommended in 2008 [44], that, even without array list SRAM can be executed on a wide class of FPGAs and powerful encryption techniques can be extracted.

Figure 3.1 below, consists of two latches whose outputs are cross-coupled, which is taking the Butterfly PUF circuit to an unpredictable running stage (as both latches have opposite signals on their inputs and outputs) [44].

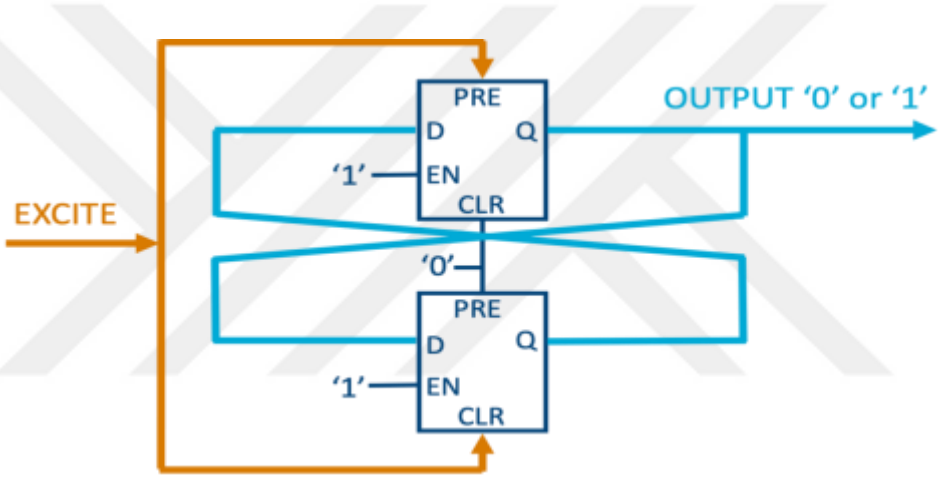

Figure 3.1. Butterfly PUF structure [44].

The definition of Butterfly PUF is based on the principle of developing mechanisms within the FPGA framework that during the start-up process acts accordingly to an SRAM cell. Butterfly PUF cell is a reversible, cross-coupled circuits that can be taken to an awkward place until it moves within one of the two potentially secure conditions [44].

The interesting trigger is adjusted to high to initiate the PUF operation, thereby taking the Butterfly PUF circuit to an unpredictable running stage (as both latches have opposite signals on their inputs and outputs). The thrilling signal is set to low after several time steps. This initiates the PUF circuit phase to reach any one of the two potentially reliable levels 0 and 1, on the output signal. The reliable phase relies on limited delay variations between the linking cables that are built on the FPGA matrix using convex routes. Therefore, such minor differences are based more on the integrated circuit's unique features and differs on the FPGA from device to device and location. Interestingly, for a broad temperature spectrum, the phase shift continues to be the same over time

with the same FPGA, latch locations, and routing services. Since it does not include these properties, an intruder cannot extract these stable states from the bit stream [44, 45].

The Butterfly circuit is therefore an interface inside the FPGA matrix of a PUF loop in which the characteristics rely upon the inherent visual effects of the embedded system if used for recognition criteria Intrinsic-ID has established products specifically designed to satisfy safety and reliability needs of defense contractors and government agencies for FPGAs.

Figure 3.2 below, shows that there are two distinct constant power spots for such bridge circuits (to store the bit value) and an unstable operating point [44].

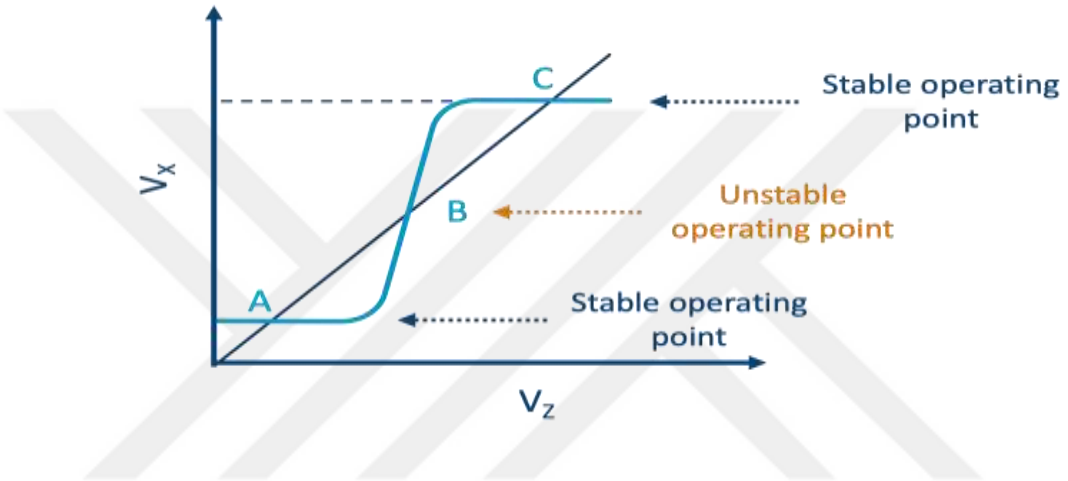

Figure 3.2. A Cross-coupled Bistable Circuit [44].

Bridge loop is a basic component used in integrated devices, like latches, flip-flops, and SRAM memories, for all types of memory aspects It is planned in such a manner that a positive reinforcement loop is given to store the pixel values necessary inside the loop. Despite the minor variations in the aspects used to design a circuit, the circuit resolves instantly in one of the two solid levels (latches in the case of the butterfly PUF). Experts use this to construct a PUF at which loop is at an unpredictable angular positioning and left to accomplish one of the two constant power points. The loop passes to a favoured steady position, with a strong likelihood. This activity is due to the tiny range of the sample of wire disruptions and bridge aspect (here latch) voltage transmission. It is interesting to emphasize that such circuits are built as diagonally as potential, and all differences are due to randomness in the circuit, which is over and above the designer's control [45].

3.3. Ring Oscillator PUF

Originally suggested by Gassendi [46, 47], RO-PUF is primarily based on time gaps throughout founder. In these devices, certain loops cause definite volume, and the result is accomplished by comparing the cyclic parameters of loops that have the same number of parts. If researchers contact the two RO1 and RO2 circuits, for which wavelengths are to be contrasted, when RO1 is fast, a value of'1' is obtained and a value of'0' is obtained when RO2 is fast. Because the PUF has to generate a lot of output signals in fact, the ROs to be contrasted are also picked via multiplexer from the group. The contrast of signals in such Processes are usually achieved with counters. For the said time, the results of the measures that measure the fluctuations at the two-transducer results are measured and the outcome achieved. Like many PUF systems, RO-PUFs are quite susceptible to atmospheric influences. Variations in distribution voltage and current may decrease the quick modulation fluctuating next to each other at speeds, and the slow one remains fast and there could be defective bit output. It is really a tool that has to generate results by sorting out over two ROs and choosing the signals are the distance measured from others to eradicate this problem consequently, because it is not quite effective in spite of volume, it is never considered as a definite alternative [47].

Figure 3.3 below shows an instance of the Ring oscillator circuit structure [47].

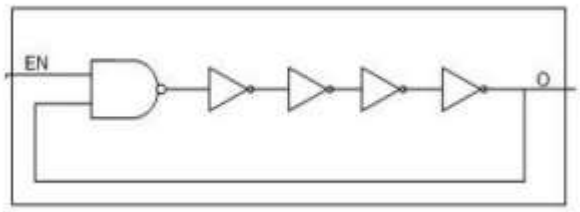

Figure 3.3. Ring oscillator [47].

RO Based PUF Application Electromagnetic PUF architecture is not favoured since, relative to Silicone PUFs are hard to adhere to and use and are therefore not ideal for FPGA. For instance, as the referee PUF, architectures are often the same as the connectivity, the FPGA execution of the RO-based PUF circuit is already addressed, and it can be constructed as simple with a completely different placement methodology and hence should not be used in FPGA. The two separate iterations of the architecture suggested by Gassendi [48] have been introduced on the Xilinx 3S5000 circuit throughout this case. RO configuration comprising of a Logic gate and four substations are used for all models. Fluctuation with Combinational logic cells is made possible in this design, thereby stopping untouched RO circuits from wasting excess power and influencing other operating ROs. RO is constructed as a macro and to build the PUF architecture, it is important to position a significant volume of this RO design and both architectures are supposed to be like that.

The results of two RO's are linked to a reference for both RO-PUF models and one-bit PUF flow rate is created by checking at whether any of the measures initially hit a fixed value. During the first edition, 2n ROs are used for n bit PUF results and each RO is only run once. N + 1 RO is used in the later iteration to produce n bit of results and each RO is contrasted on both ends to the RO. Both ROs have been used twice in this edition tend to result in a more space-efficient PUF configuration.

3.4. Arbiter PUF

Arbiter PUF The first referee type PUF structure is presented by Lim [49, 50, 51, 52]. In this structure, a certain number of delay elements are connected consecutively to form two parallel and identical lines and a signal is applied to two lines simultaneously. A referee circuit at the end of the lines decides which line is faster and produces a one-bit PUF output as shown in Figure 3.4. When n multiplexers are used as the delay element, 2 n different paths can be created depending on the selection sign to be applied, and multiple PUF outputs can be produced with the same circuit. The weakest part of these structures is that if the same circuit is used by applying different selection signals, it can be modeled and when different signals are applied, the outputs it can produce become predictable, thus losing the PUF feature. Against this type of attack, the structure called "feed-forward referee" it has been proposed in Figure 3.5 [53]. Some delay elements are bypassed in the feed-forward arbitrator circuit. In this way, modeling of the system is made difficult with the measurements to be made and PUF is stronger against attacks [53].

Figures 3.4 and 3.5 below show raferee physical unclonable function and Feed Forward Referee PUF respectively [53].

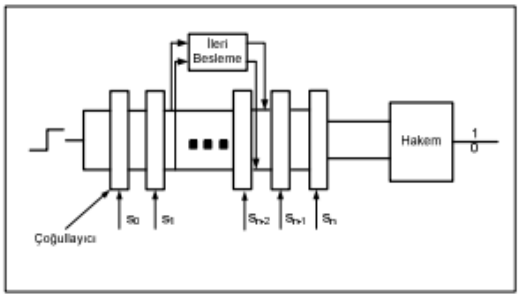

Figure 3.4. Referee PUF [53].

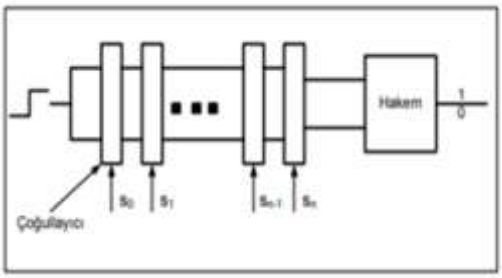

Figure 3.5. Feed Forward Referee PUF [53].

4. NIST STATISTICAL TEST SUITE

Randomness is a potential attribute that is used to define and perform the properties of a random sequence in terms of probability. In order to eliminate erroneous test results, the findings of mathematical analysis can be viewed with caution and random coefficient should be chosen for each test to assess the approval or dismissal of the hypothesis calculated and referred to as the null hypothesis. Null hypothesis contrast distribution is tested through statistical methods. There is a critical value determined by the contrast package. The meaning is compared with the critical importance of the appraisal statistics. When the statistical importance of the sample exceeds the critical value, it rejects the unpredictability of the zero hypothesis. Alternatively, they will consider the null assumption. The Nist test suit is one of the strategies constructed to measure numbers at random in the literature [30, 31, 32, 42].

The critical values used in the tests are α and P-rate. The degree of significance, choosing α as 0.01 indicates that the randomness of the numbers to be tested has 99% confidence. P-rate has excellent randomness if the measure of randomness equals 1. A p - rate equal to 0 indicates that the numbers are not random. The established analytical works related to multiple model study of bit sequences are the methodological basis for evaluating the difference under the assumption of randomness. Based on broad theoretical considerations, 15 tests are being grouped into four categories, including parameter tests (tests: 1-4), repetitive trends tests (tests: 5 and 6), corresponding trend tests (tests: 7-12) and non - stationary tests (tests: 13-15 [30, 31, 32, 42].

1. Frequency (Monobit) Test
2. Frequency Test within a Block
3. Runs Test
4. Test of Longest Runs of 1s in a Block
5. Binary Matrix Rank Test
6. Discrete Fourier Transform (Spectral) Test
7. Non - overlapping Template Matching Test
8. Overlapping Template Matching Test
9. Maurer's "Universal Statistical" Test
10. Linear Complexity Test
11. Serial Test
12. Approximate Entropy Test
13. Cumulative Sums (Cusum) Test
14. Random Excursions Test
15. Random Excursions Variant Test

4.1. Parameter (Mono - Bit) Test (with 1 parameter, y) the idea of the test is to see if 0s and 1s are the same in the q - bit string.

4.1.1. Preface

1. In this test, it is expected to see if the parameter 1 and 0 are approximately equal beyond the whole $q-bit$ pattern, i.e., whether the ratio of every of 1 and 0 is near to $1/2$
2. It is required to see how their variations collapse beyond the randomness limit where the number of 0s and 1s is not equal.

4.1.2. Context data relating to randomness:

1. This measure is derived from the principle of restrictions for random estimates.
2. The fundamental De Moivre-Laplace the derivation showed that the numeric total array is standardized by √q the standard normal allocation for many studies is closely approximated.

4.1.3. The Analysis Rely:

1. Accordingly, every bit of 0 or 1 in the sequence given by -1 and 1 represents in this new value of position where yi = εi - 1 tries the mathematical relationship.
2. Each sum of yi is Sn and Sobs = |Sq| Represents $\div \sqrt{q}.y = \text{Sobs} \div \sqrt{2}$.
3. P - rate = 1-érf (y).

4.2. Parameter test within a pattern (with 2 parameters, b & y)

4.2.1. Preface

1. It is noted that the test passes even though the initial half of the n bit array is loaded with 1 and the second half with 0, the test passes even though the sequence is not quite random.
2. Throughout this test, parameter 1 and 0 intended to be spread uniformly over the whole range of q bits

4.2.2. Context data relating to randomness:

1. The q- bit string is split into q patterns that do not overlap either of the g – bit; where q = [q/g] bases with fewer bits than g (q+1) the pattern is rejected.
2. If the ratio of 1s in each pattern is almost ½, the q-bit string can be randomly named.
3. The speed of 1 second in each pattern is shown by,

   $\prod i = (\sum \varepsilon (\text{i - 1}) \text{G} + \text{z}) / \text{G}$, because the sum of z varies from 1 to G.

where it varies from 1 to q.

4. It is denoted by chi square (χ^2),

$$\chi^2 = 2 = 4G \sum(\prod i - ½)\, 2, i = 1 \text{ to } Q.$$

4.2.3. The Analysis Rely

1. In every pattern, $\prod i$ as indicated earlier, it is determined for i ranging from 1 to Q.
2. It is indicated that (χ^2), is calculated.
3. Q is also in a degree of autonomy.
4. $b = ½\,(Q) \text{ and } y = ½\,( )$
5. $P - rate = 1 - \Gamma(b, y) / \Gamma(b, \infty)$.

4.3. Runs Test (that involves 1 parameter, y)

4.3.1. Preface

1. The steps in the range of h signify h of the same bits, bounded by bits with the inverse quality.
2. It is intended to be seen from this experiment whether the functional rhythm of 1s and 0s in varying lengths are around random boundaries.

4.3.2. Context data relating to randomness

1. $\varepsilon i = \text{bit}$ Value at position i of the bit string n; $\Pi = (\sum \varepsilon i) / n$ is for i ranging from $\sum 1$ to n.
2. A control parameter (τ) $\tau = 2/\sqrt{n}$ can be defined as $\tau = 2/\sqrt{q}$.
3. If $|\Pi - ½|1\tau$ is not mandatory to operate the current analysis because first analysis will fail for the pattern when $|\Pi - ½| < \tau$, Run test is executed.

4.3.3. The Analysis Rely

1. $y = |Vq\,(ert) - 2q\,\Pi(1 - \Pi)| / [2\sqrt{(2q)}\,\Pi\,(1 - \Pi)]$
2. P - rate = 1-érf (y).

4.4. Test of Long Run 1s in a Pattern (includes 2 Parameters, b & y)

4.4.1. Preface

1. In this test, it is anticipated to be see whether the trend of the extensive operation of 1s of different extent seen through the sequences is compliant with what is predicted for a random trend. $x = |Vn(obs) - 2n\Pi(1-\Pi)| / [2\sqrt{(2n)}\,\Pi(1-\Pi)]$
2. To run this test, the q-bit string , is split into blocks of g-bit each q = not overlapping, and $Q = \text{base}[q / G]$ as done for Test 2 is ignored. Tables 4.1 below, shows the values of G, Q and L assumed as indicated [30].

Table 4.1. shows the values of G, Q, and L [30].

MIN. Q	G	MAX.Q	L
128	8	16	3
6272	128	49	5
7,50,000	10,000	75	6

4.4.2. Context data relating to randomness

1. With all patterns in mind, vi represents the total of all parameters of the extensive runs of the specific 1s in every pattern.
2. For the sake of the calculation, vi is divided into two as L + 1, I range from 0 to L an empirical relationship between Q, G, N, and L given. Table 4.2 below, represents the probability of four classes of sets of points representing G and L [30]

Table 4.2. Representative set points of G & L [30].

	4 classes	**Probabilities**
G = 8 & L = 3	$V_0 \leq$ '1', one 1s or no 1s	0.2148
	V_1 = '11' (2 ones)	0.3672
	V_2 = '111' (3 ones)	0.2305
	$V_3 \geq$ '1111' (4 ones or above)	0.1875

4.4.3. The Analysis Rely

1. The q-bit chain is separate in to Q patterns.
2. In any pattern, the extensive operation of 1s is noted and the suitable categories (vi) have increased. Consequently, all categories were completed correctly at the end of the nth pattern.
3. The observations of ($\chi 2$) are offered by $\chi 2 = \Sigma$ (VI - Qπi) 2 / Qπi, at which I differ between 0 and L.
4. b = ½ (L) and y = ½ ($\chi 2$)
5. P-rate= 1 - Γ (b, y) / Γ (b, ∞)

4.5. Test 5: Binary Matrix Rank Test (including 2 parameters, b & y)

4.5.1. Preface

1. It is anticipated to see whether there are recurrent trends in the q-bit string throughout the whole array. The Q-bit sequence is broken into different Q patterns simultaneously, and attempts are made to see if the policy is based on each pattern's defined subsets for every category.

4.5.2. Context data relating to randomness

1. There were many theoretical studies on the first square binary order matrix (G = 10) or above. The analysis reveals that the order (PG) for the square discrete vector pattern of degree G, G-1 and G-2 is 0,

   PG = Π [1 - 2-j], for j ranging from 1 to ∞

   = 0.2888

   = 0.5 * 0.75 * 0.875 * 0.9375 * 0.9843755 * 0.9921875 * 0.996009375 * PG-1 = 14.00

   = 0.5776 PGM-2 = (4/9) PG = 0.1284

   Moreover, all other possibilities are very small ($\leq$ 0.005).
2. PG-2, PG-3, etc. Given very low probability values, it is assumed that G-2 and lower order matrices can be combined with PG-2, and PG-2 will take a value.
3. Considering the possibility, since there are 3 classes, the degree of freedom (K) will be 2.

4.5.3. The Analysis Rely

1. The determinants of all the sub-matrices of order 32 are determined, and non-zero ones are counted.

4.6. Test 6: Discrete Fourier Transform (Spectral) Test (this Test includes 1 parameter, y)

4.6.1. Preface

1. In this test, it is intended whether the q-bit string has periodic properties with periodic characteristics throughout its overall range, recognizing trends, which repeat similar to one another.
2. Giving that the randomness, the peak height threshold value (T) can be found. The series will be arbitrarily called if no more than 5 percent of maximum lengths are more than T.

4.6.2. Context data relating to randomness

1. DFT creates a variety of dynamic parameters with various factors describing repetitive elements and the (DFT) portion of jth bit is supplied by the Fj, for k = 1, 2. Fj = Σ [xk exp (j * 2πi (k-1) / q)], n and xk are kth bits.
2. The data point for the maximum value (T) is determined using the correlation T = √ [(log (1 / 0.05)) * q].
3. Due to the consistency of the actual transition, only j variables from 0 (q / 2 - 1) are considered instead of q.

4.6.3. The Analysis Rely

1. Throughout the q-bit chain, every bit of 0 and 1 is defined by -1 and 1 precisely with each connection. Yi =2εi -1, and Yi reflect the current variable of the bit εi at the ith location. (i is not the same as 0 to q).
2. T has been evaluated via relationship as described earlier.
3. Q0 = Potential predicted (95 percent) number of levels underneath the randomness principle = 0.95q/2.
4. The variance (G) for Fj is determined for j = 0 to (q/2-1) regarding any expression specified earlier.
5. Q1 = Number of spikes in M < T.
6. d = (Q1-Q0)/√ [q (0.95) (0.05) / 2].
7. y= | d | / √ 2.
8. P - rate = 1 – erf (y).

4.7. Test 7: Non-overlapping Template Matching Test (this test includes 2 parameters, b & y)

4.7.1. Preface

1. The template match is detected as over lapping, it searches the happening of the predetermined bit array that investigates whether numbers of such occurrences are within the statistical limitations of the sequence below random assumption.
2. A g-bit window is considered to be looking for a specific m-bit chain. If no chain is found, the window shifts to a single-bit location, and if Chan is found the window resets to the bit next to the pattern with the window.
3. This test distinguishes generators that create all these non-periodic trend manifestations (aperiodic).

4.7.2. Context data relating to randomness

1. The original description hypothesis is believed to be valid for random fragments.
2. Mean (μ) and deviation ($\sigma 2$) are computed because of the rough confidence interval provided via, $\mu = (G-g+1)/2g$ and $\sigma 2 = G\,[1/2g – (2g-1)/\ 22g]$, thus, g seems to be a linear combination of an occurring G times of the non-periodic trend.

4.7.3. The Analysis Rely

1. $\chi 2 = \Sigma\ (Wj – \mu)2 / \sigma 2$, for j =1 to Q
2. b = ½ (Q), y = ½ ($\chi 2$)
3. P-rate= 1 – Γ (b, y) / Γ (b, ∞)

4.8. Test 8: Overlapping Template Matching Test (this test includes 2 parameters, b & y)

4.8.1. Preface

1. The template match is detected overlapping; therefore, it searches for happenings of the predetermined bit array and investigates if the number of certain instances is, under the presumption of randomness, toward a sequence.
2. To check for a particular g-bit trend, a g-bit variable is brought on considering if there is a pattern or not, the gap always moves a bit.
3. Poisson asymptotic distribution assumed to be monitored for this test.
4. The q-bit array is shared into Q patterns of each G-bit such that Q = [q / G].

4.8.2. Context data relating to randomness

1. Many theoretical studies have been made regarding overlapping template matching. Λ and η values for theoretical calculation probabilities (πi) corresponding to classes Qi,
   $\lambda = (G-g + 1) / 2g$, $\eta = \lambda / 2$
   g is the defined duration of aperiodic model then G is the bit size of each pattern.
2. within the hypothesis of unpredictability conceptual explanatory variables in the normative literature are valid as follows,
   $\pi0 = 0.324652$, $\pi1 = 0.182617$, $\pi2 = 0.142670$, $\pi3 = 0.106645$,
   $\pi4 = 0.077147$, $\pi5 = 0.166269$.

4.8.3. The Analysis Rely

1. $\chi2$ is equal to $\Sigma (\nu i - Q\pi i)\ 2 / Q\pi i$, for i = 0 to 5.
2. a = ½ (K) and x = ½ ($\chi2$)
3. P - rate = 1 – ┌ (b, y) / ┌ (b, ∞).

4.9. Test 9: Maurer's "Universal Statistical" Test (this test includes 1 parameter, x)

4.9.1. Preface

1. The assessment depends on the analysis of the interval between M bit map trends in M bit pattern numbers. The sum of log 2 distances between M-bit mapping trends are required in statistical propagation.

4.9.2. Context data relating to randomness

Table 4.3 below, shows the dynamic look-up table of Maurer's "Universal Statistical" Test [30].

Table 4.3. A dynamic look-up table [30].

L	Expected value $E(f_n)$	Variance	L	Expected value $E(f_n)$	Variance
6	5.2177052	2.954	12	11.168765	3.401
7	6.1962507	3.125	13	12.168070	3.410
8	7.1836656	3.238	14	13.167693	3.416
9	8.1764248	3.311	15	14.167488	3.419
10	9.1723243	3.356	16	15.167379	3.421
11	10.170032	3.384			

4.9.3. The Analysis Rely

1. $x = | fq - E(fq) | / \sqrt{2}\sigma$
2. P-rate = 1 - erf (y).

4.10. Test 10: Linear Complexity Test (this test includes 2 parameters, b & y)

4.10.1. Preface

1. Generally, a large bit of array is derived via an LFSR (Linear Feedback Shift Record).
2. Random can be called a bit stream wherein a larger LFSR is received, whereas the smaller LFSR reveals how it is not random.
3. In order to achieve an LFSR, the Berlekamp-Massey Algorithm was implemented.
4. The (LCT) looks for the LFSR duration and decides if the bit source wherein the LFSR is acquired is random.

4.10.2. Context data relating to randomness

1. The large q-bit structure is split into Q patterns, each with a G-bit.
2. Assuming randomness, the mid - gap of the LFSR (μ) of G-bit string is provided as shown below,
   $\mu = (G/2) + (9 + (-1)\ G+1)/36 – [(G/3) + (2/9)]/2G$
3. (Ti) of a LFSR of length (Qi) is provided as indicated below,
   $Ti = (-1)\ G\ (Li - \mu) + 2/9$

4.10.3. The Analysis Rely

1. If there were no groups, T is in a group. The creation of seven groups offers an additional six group options to T - hence the level of independence is six.
2. Chi-square statistics ($\chi 2$), $\chi 2 = \Sigma\ [(\nu i - Q\ \pi i)\ 2 / Q\pi i]$ for i = 0 to K
3. b = ½ (K) and y = ½ ($\chi 2$)
4. P-rate= 1 - Γ (b, y) / Γ (b, ∞)

4.11. Test 11: Serial Test (2 P-values each one with 2 parameters, b & y)

4.11.1. Preface

1. Every g-bit pattern has the possibility to occur like other g-bit patterns in such a long q-bit random chain
2. As much as someone will predict from a random set, the number of g-bit overlapping patterns is nearly the same.
3. g periods are predicted to appear in any of the g-bit patterns in the q-bit series, where g = q / 2g.

4.11.2. Context data relating to randomness

1. 2g2 = Σ (νi - Ag P)2 / A, addition is between i = 0 to (2 gg - 1). = (1 / Ag) Σ νi 2 - q
2. In the present case, chi-square statistics (χ2), Δψg2 = ψg2 - ψg-12 Δψg-12 = ψg-12 - ψg-22 Δ2ψg2 = Δψg2 - Δψg-12 = ψg2 - 2ψg-12 + ψg- 22
3. Here, Δψg2 is the distribution of K1 = 2m-1 degree of independence and ψg2 is another distribution of K2 = 2g-2 degree of independence.
4. Two chi-squares (χ2) sharing combined with dual rate of independence lead to dual P-rates.
5. The rate of g is always leaser as well as the fold [log2 (q)] -2.
6. The Series Test appears as q = 1 as the parameter test (Test 1).

4.11.3. The Analysis Rely

1. For m-bit pattern νi is counted for i = 0 to (2g – 1); ψg2 is computed with A g= q/2g.
2. For (g-1)-bit pattern νi is counted for i = 0 to (2g-1 – 1); ψg-12 is computed with A g-1 = q/2g-1.
3. For (g-2)-bit pattern νi is counted for i = 0 to (2g-2 – 1); ψg-22 is computed with A g-2 = q/2g-2.
4. On ψg2 and ψg-12, Δψg2 is computed and based ψg2, ψg-12 and ψg-22, Δ2ψg2 is computed.
5. Considering a1 = ½ (K1) and y1 = ½ (Δψg2),
   P-value1 = 1 – ┌ (b1, y1) / ┌ (a1, ∞)
6. Considering b = ½ (K2) and y2 = ½ (Δ2ψg2),
7. P-value2 = 1 – ┌ (b2, y2) / ┌ (b2, ∞)

4.12. Test 12: Approximate Entropy Test (1 P-rate with 2 parameters, b & y)

4.12.1. Preface

1. Entropy is a randomness test that is reliant on reoccurring themes as stability increases and randomness increases.
2. The q-bit string entropy is evaluated by matching the overlap pattern attribute of all potential m-bit trends with that of the (g + 1) bit patterns. Compared to the anticipated outcome of a random array, the correlation between the frequencies of the m-bit and (g + 1) -bit patterns are called the estimated entropy ApEq (g).

4.12.2. Context data relating to randomness

1. Bits (g-1) extracted from the start of the array are applied to the end of a specific q-bit segment.
2. Let vi reflect the alternating variable figures of a defined m-bit trend varying from 0 to 2g then i represents the decimal value of a given g-bit pattern.
3. Cig = νi / n, πi = Cig and Φg = Σ (πi lq πi) for i = 0 to (2g -1)
4. In order to count the (g + 1) bit map patterns, the original q bits are applied at the end of the provided n bit array.
5. Likewise, vi reflects the alternating variable counts of the defined trend (g + 1) -bit varying from 0 to 2g + 1, while I shows the decimal value of the provided trend (g + 1) -bit.
6. Cig + 1 = νi / n, πi = Cig + 1 and Φg + 1 = Σ (πi lq πi) for i = 0 to (2g + 1 -1)
7. K = Degree of independence = 2g
8. χ2 = 2 q [lq 2 - ApEq (g)]

4.12.3. The Analysis Rely

1. To count the g-bit matching patterns, νi is counted overlapping along the attached (q + g-1) bit sequence for all possible m-bit patterns, ranging from 0 to (2g-1).
2. Cig, πi, and Φg values are computed relying upon the 2g vi forms.
3. Again, to count (g + 1) -bit mapping patterns, νi is counted overlapping along the attached (q + g) -bit sequence for all possible (g + 1) -bit patterns, i changes to 0 (2g + - 1).
4. Cig + 1, πi and Φg + 1 values are calculated according to 2g + 1 type νi.
5. χ2 = 2q [lq 2 - ApEq (g)] where ApEq (g) = Φg - Φg + 1
6. Now b = ½ (K) and y = ½ (χ 2).
7. P-rate= 1 - ┌ (b, y) / ┌ (b, ∞)

4.13. Test 13: Cumulative Sums Test (2 P-rates each one with 1 parameter, y)

4.13.1. Preface

1. The aggregate total test examines if 1s or 0s appear in massive quantities throughout the entire series at the initial or afterward levels of 1s and 0s.

4.13.2. Context data relating to randomness

1. Even though the variation in aggregate sums is being examined, the P-rate is determined according to the usual transmission form (Φ) provided by,
   $\Phi(z) = (1 / \sqrt{(2\pi)}) \int \exp(-u2 / 2)$, While the smaller and higher limit stands as $-\infty$ and z simultaneously.

4.13.3. The Analysis Rely

1. Throughout the whole q-bit process, 0's are set to -1 as adapted in Test 1 (-1, + 1) the whole total of the Yi phases of the pattern are acquired as Si = Si-1 + Yi = 1 to q, while S0 = 0.
2. After the normal distribution function, two P rates are calculated, one corresponding to the P cumulative totals.

4.14. Test 14: Random Excursions Test (8 P-values each one with 2 parameters b & y)

4.14.1. Preface

1. The Random Trips Test aims in investigating if the number of visits to a given total state in a cycle falls into a stage assumed from the random pattern.
2. Eight states are examined, for example, visits to +4, -3, -2, -1 and +1, +2, +3, +4 to +4 states are collected during visits to +4 states and visits to 4 states - Visits under the age of 4 are collected.

4.14.2. Context data relating to randomness

Table 4.4 below, shows a dynamic look-up table which shows P - rate calculated for each (S) in random excursions test [30].

Table 4.4. A dynamic look-up table [30].

	$\Pi_0(S)$	$\Pi_1(S)$	$\Pi_2(S)$	$\Pi_3(S)$	$\Pi_4(S)$	$\Pi_5(S)$
S= ± 1	0.5000	0.2500	0.1250	0.0625	0.0312	0.0312
S= ± 2	0.7500	0.0625	0.0469	0.0352	0.0264	0.0791
S= ± 3	0.8333	0.0278	0.0231	0.0193	0.0161	0.0804
S= ± 4	0.8750	0.0156	0.0137	0.0120	0.0105	0.0733
S= ± 5	0.9000	0.0100	0.0090	0.0081	0.0073	0.0656
S= ± 6	0.9167	0.0069	0.0064	0.0058	0.0053	0.0588
S= ± 7	0.9286	0.0051	0.0047	0.0044	0.0041	0.0531

4.14.3. The Analysis Rely

1. Via the complete series of n-bits, 0's are set to -1 as executed in Test 1 (-1, +1) cumulative sums of the Yi steps of the sequence are obtained as Si = Si-1 + Yi i = 1 to q and S0 = 0 as well as Sq + 1 = 0
2. If the aggregate totals reach 0 (J-1) levels, J is called, the number of loops that is taken under consideration is the zero-crossing point in Sq + 1.
3. If J is very tiny, the pattern is not random. If J < 500, 10 Lac bits, strings are not considered random.
4. νk (s) = Parameter of the k-times of the visit to the x state during the J trips. For the sake of the calculation, the sum for νk (s) = Σ νk j (s) is taken for j = 1 to J. If the number of visits to situations during the jth trip is exactly k, then νk j (s) = 1 another νk j (s) = 0.
5. Counting data for νk (s) with k 5 is put in ν5 (s).
6. Chi-square statistics for every situation,

   $\chi 2 = \Sigma [\nu k (s) - J \pi k (s)] 2 / J \pi k (s)$, K = 0 to 5.4.
7. b = ½ (K) and y = ½ (χ2)
8. P-rate= 1 - ┌ (b, y) / ┌ (b, ∞)

4.15. Test 15: Random Excursions Variant Test

4.15.1. Preface

1. Random Excursions Variant Test looks for the number of visits to a specific system in the estimated random walk totals through the whole series of bits and calculates variations in the estimated number of random walking visits.
2. 18 states, e.g., s = –9, –8, –7, –6, –5, –4, –3, –2, –1, +1, +2, +3, +4, +5, +6, +7, +8, +9 are considered

4.15.2. Context data relating to randomness

1. Statistic Variation (σ) = 2 (2 | s | - 1) in respect of random walk for a visit to different states.

4.15.3. The Analysis Rely

1. Throughout the whole q-bit process, 0's are set to -1 as adapted in Test 1 (-1, +1) complete totals of the Yi levels of the pattern are found as Si = Si-1. + Xi i = 1 to q and S0 = 0 as well as Sq + 1 = 0

2. 2. If the aggregate totals reach 0 (J-1) levels, J is called, the number of loops that is taken under consideration is the zero-crossing point in Sq + 1

3. ξ (s) is defined as the total number of visits in all J loops of a state.

4. $y = |\xi(s) - J| / \sqrt{(2 J \sigma)}$.

5. $P\,rate = 1 - erf(y)$.

5. RESULTS AND DISCUSSIONS

5.1. Puf Architecture and Its Implementation in Fpga Environment

PUF is a function that is stimulated by a challenge set that produces a response set (Response Set). As shown in Figure 5.1, the challenge-response relationship differs due to the chemical and physical properties of the equipment. PUF structures have four main features. These are uniqueness, unclonability, unpredictability, and robustness. Uniqueness is that if the PUF circuit design is realized in different integrations, the responses (Response) are different according to different challenge values. This value is expected to be around 50%. This value indicates that the response set obtained from two different PUF designs will be different. Unclonability indicates that two PUF circuits will be difficult and time-consuming to achieve the same response set. Unpredictable is that the responses of a PUF to a challenge are unpredictable even if the environmental differences, structure, and layout of the PUF are known. Robustness means that PUF is likely to give similar answers when PUF is queried multiple times with the same challenges.

Figure 5.1 below shows the basic Physical Unclonable Function structure

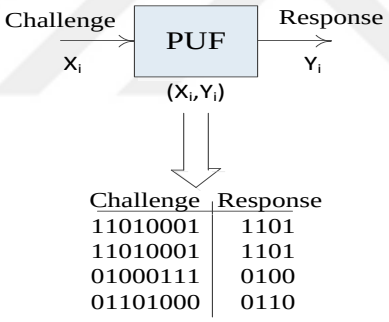

Figure 5.1. PUF basic structure

5.2. Ring Oscillator Based Puf

Ring oscillators consist of NOT-connected and interconnected door elements in their structures. The signs applied to the door elements entrance are obtained with delay in the door element outputs NOT. This delay causes continuous oscillation in the ring oscillator. Since the chemical properties of each door element are different, oscillation has a random feature.

Figure 5.2 below, shows the basic Ring Oscillator structure, when the flood bit is 0 to select the multiplexer inputs, the logic 1 applied to the Data0 input will be logic 0 at the output of the NOT gates in case of flood input 1, the oscillator will oscillate and it will be 1-0-1 at the output'

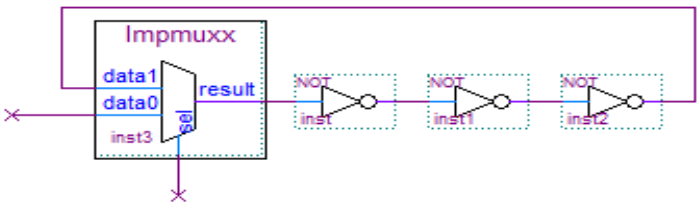

Figure 5.2. Basic RO structure

The Figure 5.3 below shows the RO-based Physical Unclonable Function structure:

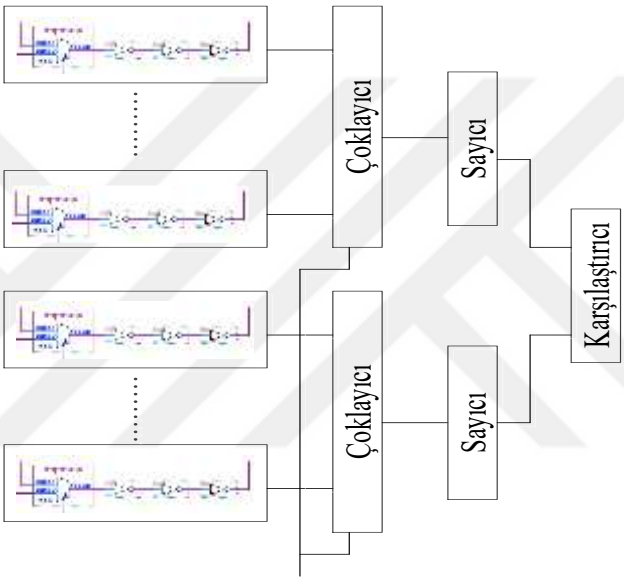

Figure 5.3. RO based PUF structure

Figure 5.3 above, shows the RO-based PUF structure, multiple ROs are used in the RO-PUF structure, each containing an equal number of elements. RO oscillations are selected through multiplexers. Random number generation is performed by comparing the oscillation parameter obtained from two different ROs. Counter circuits are used for parameter comparisons. The oscillation parameter of the RO groups differs from each other. For the signals to be given to the counter input to be variable, RO groups are provided with the signals given to the multiplexer selection input.

Figure 5.4 below, shows some of the random numbers obtained from the comparator output of ring oscillator structure that is realized in FPGA.

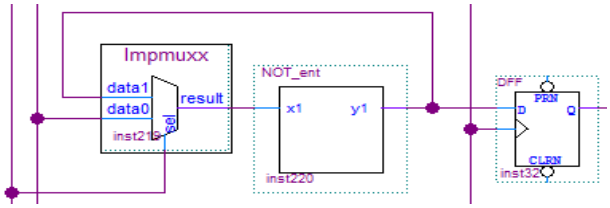

Figure 5.4. RO structure realized in FPGA

In figure 5.5 below, the Part of the random numbers obtained with the Ring Oscillator Physical Unclonable Function structure realized in FPGA is seen.

Figure 5.5. Part of the random numbers obtained with the RO-PUF structure

Number generation is made by transferring a signal over two circuits by using delay elements such as multiplexers. In figure 5.6 below, it is seen that the circuits called referees are used to determine which circuit of the signal comes first.

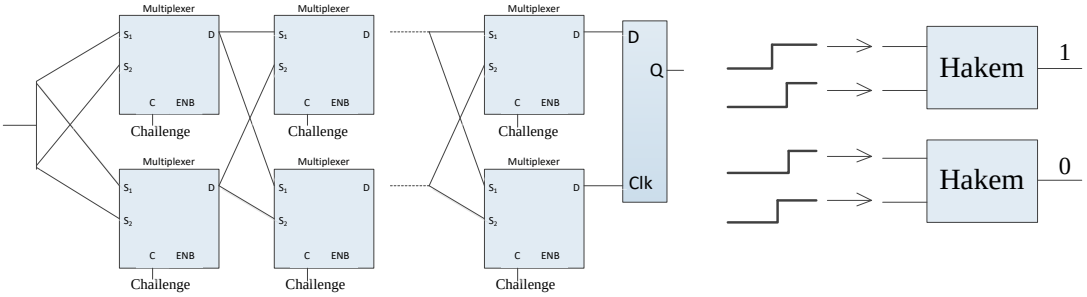

Figure 5.6. Referee PUF structure

N delay elements are used as the multiplexer (MUX) delay element. The referee PUF output generates a number based on the signal given to the challenge select input.

Figure 5.7 below, shows the Referee Physical Unclonable Function structure performed in the FPGA environment and storing the generated numbers into memory.

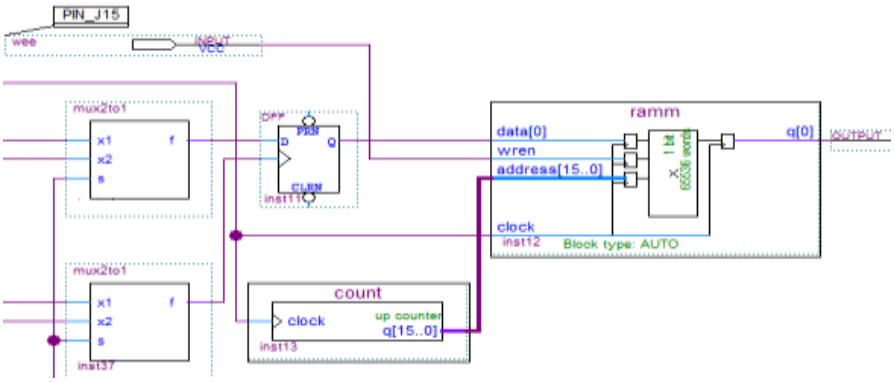

Figure 5.7. Realization of Referee PUF in FPGA and storing the produced numbers in memory

Figure 5.8 below, shows some part of the random numbers stored in the memory unit from the realization of Referee Physica Unclonable Function in FPGA and storing the produced numbers in memory.

Figure 5.8. Some of the numbers produced by the referee PUF

NIST test results of the models performed are shown in Table 5.1. The random numbers obtained with the implemented PUF architectures are successful in all tests of NIST SP 800-22.

Table 5.1. NIST test results of the implemented models are as in the table.

Test	**Referee PUF**
Frequency Monobit	0,949
Frequency test Within a Blok	0,931
Runs	0,671
Longest Run of Ones in a Blok	0,695
Binary Matrix Rank	0,621
Discrete Fourier Transform	0,207
Non-Overlapping Template Matching	0,025
Overlapping Template Matching	0,671
Universal	0,744
Linear Complexity	0,256
Serial	0,912 0,672
Approximate Entropy	0,011
Cumulative Sums	0,801

6. CONCLUSION

The so-called pseudo random number generators are deterministic and easily generate number sequences with a certain algorithm using an initial (seed) value. True random number generators are not deterministic but use true physical processes as a source of noise. In order for random numbers to be used in computer science, the random number generator must have a long repetition time and the generated numbers must be independent from each other. In the literature, there are also hybrid random number generating structures in which Pseudo and True random number generators are used together. Pseudo-random numbers are both software- and hardware-based, and true random numbers are hardware-based. One of the true random number generation structures developed based on hardware is Physical Unclonable Function (PUF). PUF output varies because PUFs produce outputs specific to the physical structure. Because of this feature, they are widely used in computer science.

In this Thesis, literature review about TRNG has been done. TRNG based PUF structures have been studied. RO-PUF and Arbiter PUF structures have also been carried out and tests have been carried out in the FPGA environment. The random numbers obtained were subjected to the NIST test and successful results were obtained.

REFERENCES

[1] M. Stipˇ, Open Problems in Mathematics and Computational Science, no. November 2014. 2014.

[2] T. McGrath, I. E. Bagci, Z. M. Wang, U. Roedig, and R. J. Young, "A PUF taxonomy," Appl. Phys. Rev., vol. 6, no. 1, 2019.

[3] C. Herder, M. D. Yu, F. Koushanfar, and S. Devadas, "Physical unclonable functions and applications: A tutorial," Proc. IEEE, vol. 102, no. 8, pp. 1126–1141, 2014.

[4] R. Maes and I. Verbauwhede, "Physically unclonable functions: A study on the state of the art and future research directions," Inf. Secur. Cryptogr., no. 9783642143120, pp. 3–37, 2010.

[5] A. Sadr, "Physical Unclonable Function (PUF) Based Random Number Generator," Adv. Comput. An Int. J., vol. 3, no. 2, pp. 139–145, 2012.

[6] M. C. Integration et al., "Physical unclonable functions and applications: A tutorial," Proc. IEEE, vol. 102, no. 8, pp. 1126–1141, 2019.

[7] V. Van Der Leest, E. Van Der Sluis, G. J. Schrijen, P. Tuyls, and H. Handschuh, "Efficient implementation of true random number generator based on SRAM PUFs," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 6805 LNCS, pp. 300–318, 2012.

[8] A. Maiti, R. Nagesh, A. Reddy, and P. Schaumont, "Physical unclonable function and true random number generator: A compact and scalable implementation," Proc. ACM Gt. Lakes Symp. VLSI, GLSVLSI, pp. 425–428, 2009.

[9] S. Arslan Tuncer and T. Kaya, "True Random Number Generation from Bioelectrical and Physical Signals," Comput. Math. Methods Med., vol. 2018, 2018.

[10] M. Skorski, "True random number generators secure in a changing environment: Improved security bounds," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 8939, pp. 590–602, 2015.

[11] Microchip Technology, "PIC32 Section 49 . Crypto Engine and Random Number Generator (RNG)."

[12] 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," Proc. - 25th IEEE Int. Symp. High Perform. Comput. Archit. HPCA 2019, pp. 582–595, 2019.

[13] D. DiCarlo, "Random Number Generation: Types and Techniques," Sr. Honor. Theses, 2012.

[14] J. L. Brady, "Limitations of a True Random Number Generator in a Field."

[15] C. Böhm and M. Hofer, Physical unclonable functions in theory and practice, vol. 9781461450. 2013.

[16] E. Avaroğlu, "LFSR Soru Girdisi İle PUF Tasarımının Gerçeklenmesi," Fırat Üniversitesi Mühendislik Bilim. Derg., vol. 29, no. 2, pp. 15–21, 2017.

[17] Y. Genç and S. A. Tuncer, "Araştırma Makalesi / Research Article İnsan Hareketleri Tabanlı Gerçek Rasgele Sayı Üretimi True Random Number Generation Based on Human Movements," vol. 8, no. 1, pp. 261–269, 2019.

[18] E. Erkek, "SAYI ÜRETİLMESİ VE FPGA ORTAMINDA GERÇEKLEŞTİRİLMESİ," 2015.

[19] K. MORI and A. MATANO, "Pseudo Random Number Generation," J. Japan Ind. Manag. Assoc., vol. 25, no. 2, pp. 103–107, 1974.

[20] T. Android et al., "Application Sandbox," pp. 5–7, 2020.

[21] Android Developers, "Android Open Source Project," YouTube, no. November 2011, pp. 1–4, 2008.

[22] "10 examples of mobile malware," p. 2020, 2020.

[23] D. Boot, "File-based encryption," pp. 1–2, 2020.

[24] A. Signing, S. Architecture, and F. Access, "Application Signing," vol. 2, pp. 10–12, 2020.

[25] I. I. Security et al., "6RQ LúOHPLQ * HUoHN 5DVJHOH 6D \ Õ hUHWHoOHUL h] HULQGHNL (WNLVLQLQ øQFHOHQPHVL," vol. 9, pp. 290–294, 2013.

[26] A. Sciences, “Statistical Testing of Cryptographic Randomness,” İstatistikçiler Derg. İstatistik Aktüerya J. Stat. Stat. Actuar. Sci. IDIA, vol. 9, no. 1, pp. 1–11, 2016.

[27] A. Yayık and Y. Kutlu, “Sözde Rastsal Say ı Üretecinin Yapay Sinir A ğ lar ı ile Güçlendirilmesi,” Signal Process Commun. Conferance, no. January 2013, 2013.

[28] P. L’Ecuyer, “Testing random number generators,” pp. 305–313, 1992.

[29] W. Schindler, “Efficient online tests for true random number generators,” Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 2162, pp. 103–117, 2001.

[30] J. Zaman and R. Ghosh, “A Review Study of NIST Statistical Test Suite: Development of an indigenous Computer Package,” arXiv Prepr. arXiv1208.5740, 2012.

[31] E. E. Engineering, “The study of the reasons for not giving the results of NIST Tests of Random Walk , Random Walk Variable and Lempel Ziv Rasgele Yürüyüş , Rasgele Yürüyüş Değişken ve Lempel Ziv NIST test sonuçlarının verilmeme sebeplerinin incelenmesi Pseudo random number ,” vol. 10, no. 1, pp. 1–8, 2015.

[32] R. A. Bryant et al., The NIST 3 Megawatt Quantitative Heat Release Rate Facility-Description and Procedures. US Department of Commerce, National Institute of Standards and Technology, 2004.

[33] W. Issues and W. Issues, “General metrics 35,371,” pp. 11–12, 2019.

[34] S. P. Karanam, “Tiny True Random Number Generator,” Design, p. 103, 2009.

[35] U. Random, N. Generation, and L. C. Generators, “Statistics / Numerical Methods / Random Number Generation Bays-Durham Shuffling of Uniform Deviates,” pp. 8–12, 2019.

[36] M. C. Integration, M. Pharr, G. Humphreys, B. Rendering, and T. Edition, “Pseudo-Random Number Generator Learn more about Pseudo-Random Number Gen- erator What is This Stuff Called Probability ?,” 2017.

[37] R. Numbers, “Introduction to Randomness and Random Numbers Pseudo-Random Number Generators (PRNGs) True Random Number Generators (TRNGs) Comparison of PRNGs and TRNGs,” Online, pp. 2008–2011, 2010.

[38] V. Bagini and M. Bucci, “A design of reliable true random number generator for cryptographic applications,” Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 1717, no. August, pp. 204–218, 1999.

[39] C. L. Pérez, “Correlation evaluation between the randomness tests of the Toolbox Battery . Evaluacion de Correlacion entre los tests de la bateria TOOLBOX . EVALUACIÓN DE CORRELACIÓN ENTRE LOS TEST DE LA MRC TOOLBOX Introducción,” no. November, 2018.

[40] L. Gong, J. Zhang, H. Liu, L. Sang, and Y. Wang, “True Random Number Generators Using Electrical Noise,” IEEE Access, vol. 7, pp. 125796–125805, 2019.

[41] T. Middle, S. Method, G. Random, and S. Viii, “Harder , Better , Faster , Stronger The Middle Square Method (Generating Random Sequences VIII),” pp. 2017–2019, 2019.

[42] J. K. M. Sadique, U. Z. Zaman, and R. Ghosh, “NIST Statistical Test Suite for Crypto System The following 15 tests are mentioned in NIST statistical test suite,” pp. 1–20, 2011.

[43] J. Guajardo, S. S. Kumar, G.-J. Schrijen, and P. Tuyls. FPGA Intrinsic PUFs and Their Use for IP Protection. In P. Paillier and I. Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems — CHES 2007*, volume 4727 of *LNCS*, pages 63–80. Springer, September 10-13, 2007.

[44] Kumar, S.S., Guajardo, J., Maes, R., Schrijen, G.J., Tuyls, P.: The Butterfly PUF: Protecting IP on every FPGA. In: IEEE International Workshop on Hardware-Oriented Security and Trust, HOST (2008)

[45] R. S. Pappu, “Physical one-way functions,” Ph.D. dissertation, Massachusetts Institute of Technology, 2001.

[46] R. S. Pappu, B. Recht, J. Taylor, N. Gershenfeld, “Physical one-way functions”, Science, vol. 297, no. 6, pp. 2026-2030, 2002.

[47] B. Gassendi, D. Clarke, M. van Dijk, and S. Devadas, “Silicon physical random functions,” in ACM Conference on Computer and Communications Security CCS, pp. 148–160, 2002.

[48] B. Gassendi, D. Clarke, M. Dijk, and S. Devadas, “Controlled physical random functions,” in 18th Annual Computer Security Applications Conference (ACSAC), 2002.

[49] G. E. Suh, E. Devadas, “Physical Unclonable Functions for Device Authentication and Secret Key Generation”, ACM DAC, 2007.

[50] D. Lim, J.W. Lee, B. Gassendi, G. E. Suh, M. Van Dijk, S. Devadas, “Extracting secret keys from integrated circuits” IEEE Transactions on VLSI Systems, 2005.

[51] B. Gassendi, D. Clarke, M. Van Dijk, S. Devadas, D. Lim, “Identification and Authentication of Integrated Circuits”, Concurrency and Computation: Practice & Experience. Vol. 16, no. 11, pp. 1077-1098. Sept. 2004.

[52] B. Gassendi, D. Clarke, M. Van Dijk, S. Devadas, “DelayBased Circuit Authentication and Applications", ACM Symposium on Applied Computing, 2003.

[53] J. W. Lee, D. Lim, B. Gassendi, G. E. Suh, M. Dijk, S. Devadas, “A Technique to Build a Secret Key in Integrated Circuits for Identification and Authentication Applications”. Symposium On VLSI Circuits Digest of Technical Papers, 2004

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

Computer Programs that I Know:

- Adobe Master Collection (Dreamweaver, Flash, Photoshop, Fireworks, InDesign. etc.)
- Microsoft Office (Word, PowerPoint, Excel) “Professional”

COMPUTER PROGRAMING LANGUAGES THAT I KNOW:

- HTML, CSS, PHP, ACTIONSCRIPT3, AIR for ANDROID, C#, Mat Lab

EDUCATION

- [redacted]

ACADEMIC ACTIVITIES

- Android Application – Educational Software
- Educational Software (Physics Lesson)
- Job information system application for graduates
- Web Design
- Developing Educational Software Programming
- Android Application Software Programming