ON DESIGN OF PUF-BASED RANDOM NUMBER GENERATORS

Mehdi Ayat[1], Reza Ebrahimi Atani[2], Sattar Mirzakuchaki[1]

[1]Department of Electrical Engineering, Iran University of Science and Technology, Tehran, Iran
Mehdiayat2006@gmail.com, m_kuchaki@guilan.ac.ir

[2]Department of Computer Engineering, The University of Guilan, P.O. Box 3756, Rasht, Iran
rebrahimi@guilan.ac.ir

ABSTRACT

In this paper we propose a new architecture Physical Random Functions (or Physical Unclonable Functions, PUFs) to create a candidate hardware random number generator. So far several random number generators based on ring oscillators were introduced but all of them have either security or stability problems. This paper presents a novel architecture which has solved both of these problems. This idea have a higher data complexity and also nonlinearity which secures the circuit against modeling attacks. The final architecture has also lower hardware complexity which make it suitable for lightweight random number generators.

KEYWORDS

Physical Unclonable Functions, Physical Cryptography, Random Number Generator,

1. INTRODUCTION

The need for random numbers in cryptographic processes is ubiquitous. Initialization vectors block padding, challenges, nonces, and, of course, keys are some of the cryptographic objects where a string of unpredictable bits is required. Often the same Random Number Generator (RNG) supplies bits for all of the above uses in a cryptographic system. Many of the bits generated by the RNG are transmitted in the clear and thus a passive attacker has sample opportunity to analyze the output of the RNG and can leverage any weaknesses found there [1]. RNGs can be separated into two general categories:

- Pseudo Random Number Generators (PRNGs).
- True Random Number Generators (TRNGs).

RNGs used for cryptographic processes must, therefore, be considered a critical part of the cryptographic system. A weakness or failure in the RNG can lead to a complete failure of the system.

One of the major techniques used for designing a RNG is PUFs. A PUF is a function that generates a set of responses while stimulated by a set of challenges. It is a physical function because the challenge-response relation is defined by complex properties of a physical material, such as the manufacturing variability of CMOS devices. Its unclonability is attributed to the fact that these properties cannot be controllably reproduced, making each device effectively unique.

DOI : 10.5121/ijnsa.2011.3303

A PUF must be Easy to evaluate which means the physical device must be capable of evaluating the function in a short amount of time. It should also be hard to characterize. Hence from a limited number of plausible physical measurements or queries of chosen Challenge-Response Pairs (CRP), an attacker who no longer has the device, and who can only use a limited amount of resources (time, money, raw material, etc...) can only extract a negligible amount of information about the response to a randomly chosen challenge. PUFs should be also prohibitively hard to copy (clone), emulate, simulate, or predict. The main goal of this paper is to investigate the PUF-based architectures for RNGs and compare their advantage and disadvantages. At the end of the paper a novel PUF-based architecture for RNGs will be presented. This architecture has a higher number of challenge responses compared to the conventional PUF-based architectures.

The remainder of the article is organized as follows. A brief background on PUFs and variation modelling is given in Section 2. Section 3 presents a survey of related literature. The novel architecture of PUF-based RNG is presented in section 4 and finally paper concludes in section 5.

2. PHYSICALLY UNCLONABLE FUNCTIONS AND VARIATION MODELLING

There is a wide consensus that intrinsic manufacturing variability of modern and pending deep submicron silicon is an excellent PUF implementation platform [5]. Silicon technologies form the basis for almost all computing platforms today, while it is not technologically possible to reproduce the inherent silicon variability. Security techniques that employ silicon PUFs have numerous important advantages over traditional cryptography-based security techniques including much better resiliency against physical attacks (e.g., radiation, reverse engineering) [21], the absence of covert channels (e.g., power, delay, electromagnetic measurements), and much lower time, speed, and power overheads. PUFs have been used for a variety of security applications ranging from ID creation and authentication, to hardware metering and remote enabling and disabling of integrated circuits [5].

By embedding PUFs into devices, the devices become unclonable. This makes them very useful for anti-counterfeiting applications. The challenge-response behaviour of a PUF changes drastically when it is damaged—for instance, by an attacker. Together with their unclonability property, this makes PUFs very interesting as a means for secure key storage. Instead of storing keys in digital form in the memory of a device, a key can be extracted from a PUF embedded in the device. Only when the key is required (e.g., for an authentication protocol), it is extracted from the PUF and deleted immediately when it is no longer needed. In this way, the key is only present in the device for a minimal amount of time and hence less vulnerable to physical attacks. Unfortunately, recent analysis has demonstrated that many of the current state-of-the-art PUF structures are susceptible to a variety of security attacks.

Silicon PUFs exploit manufacturing variability to generate a unique input/ output mapping for each IC. Delay-based silicon PUFs uses the delay variations of CMOS logic components to produce unique responses. The responses are generated by comparing the analog timing difference between two delay paths that must be equivalent by logic-level construction, but are different because of manufacturing variability. The delay-based structures use a digital component, arbiter that translates the analog timing difference into a digital value. An arbiter is a sequential component with two inputs and one output.

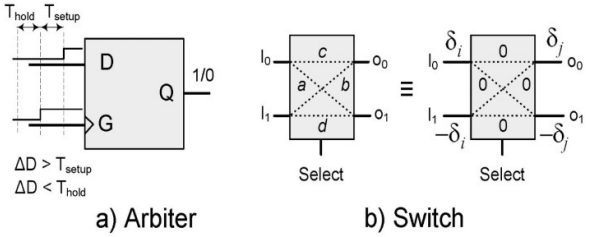

Figure 1. PUF fundamental building blocks.

The arbiter output is one if a rising edge signal arrives at its first input earlier by at least a threshold value compared to the signal arriving at the second input. The arbiter's output is zero otherwise. Figure 1(a) shows an arbiter implemented using an edge-triggered latch. If the time difference between the arriving signals are smaller than the setup and hold times of the latch, the arbiter may become metastable and not be able to produce an accurate and deterministic output.

Lee et al. [17] proposed a parallel delay-based PUF circuit shown in Figure 2. Generating one bit of output requires a signal to travel through two parallel paths with multiple segments that are connected by a series of 2-input/2-output switches.

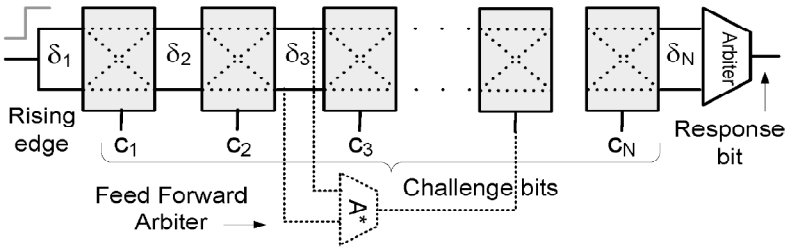

Figure 2. Parallel PUF structure [5].

As depicted in Figure 1(b), each switch is configured to be either a cross or a straight connector, based on its selector bit. The arbiter compares the signal arrival times at the end of parallel paths (i.e., at its inputs) to produce the corresponding response. The path segments are designed to have the same nominal delays, but their actual delays differ slightly due to manufacturing variability [5].

The difference between the top and bottom path delays on the segment n is denoted by δ_n on Figure 2. To ensure larger variations, one could insert additional delay elements on the path segments. The PUF challenges (inputs) are the selector bits of the switches. The output bit of the arbiter depends on the challenge bits and is permanent for each IC (for a range of operational conditions). Parallel PUF's liability to reverse engineering was previously addressed by introducing nonlinearities, such as feed forward (FF) arbiters, in the PUF structure [22]. Figure 2 also includes a FF arbiter (dashed line) that controls a switch selector. Unfortunately, our preliminary study shows even this structure can be reverse engineered using a combination of combinatorial and linear programming technique [4, 5].

The majority of the PUF designs are based on delay variation of logic and interconnect. The fundamental principle followed in these delay-based PUF is to compare a pair of structurally identical/symmetric circuit elements (composed of logic and interconnect), and measure any delay mismatch that is introduced by the manufacturing process variation, and not by the design. Arbiter PUF and Butterfly PUF are inherently difficult to implement on FPGA due to

the delay skew present between a pair of circuit elements that are required to be symmetric in these PUFs. This static skew is an order of magnitude higher than the delay variation due to random process variation.

The equation for delay d of a net N in a circuit is shown in equation 1, where d_S is the static delay as determined by the static timing analysis tools, and d_R is the random delay component due to process variation.

$$d_N = d_S + d_R \quad (1)$$

The delay difference between two nets, N_1 and N_2, in a circuit maybe be expressed as a sum of static delay difference Δd_S and random delay difference Δd_R [15] as shown in Equation 2.

$$\Delta d = d_{S1} - d_{S2} + d_{R1} - d_{R2} = \Delta d_S + \Delta d_R \quad (2)$$

A delay-based PUF circuit involves extraction and comparison of the random delay, d_R while minimizing Δd_S. In the ideal case for a delay based PUF, $\Delta d_S \rightarrow 0$ and the delay skew is purely a function of the random delay component. However, typically the output of a given PUF structure will be at least partially dependent on Δd_S, causing the output to be biased. Further, if $\Delta d_S > \Delta d_R$, the effect of random variation becomes insignificant, and the output of the PUF structure becomes static regardless of d_R. The effectiveness of the PUF depends on how much symmetry we can achieve between a particular pair of elements in order to minimize the effect of Δd_S. This symmetry requirement determines the implementation complexity of a PUF on FPGA. Figure (3) shows a sample FPGA symmetric routing for arbiter PUF implementation.

Figure 3. Arbiter PUF.

An Arbiter PUF, proposed by Lim et.al [23], is composed of two identically configured delay paths that are stimulated by an activating signal. The basic architecture of the arbiter PUF is shown in figure 4. The difference in the propagation delay of the signal in the two delay paths is measured by an edge triggered flip-flop known as the arbiter. Several PUF response bits can be generated by configuring the delay paths in multiple ways using the challenge inputs.

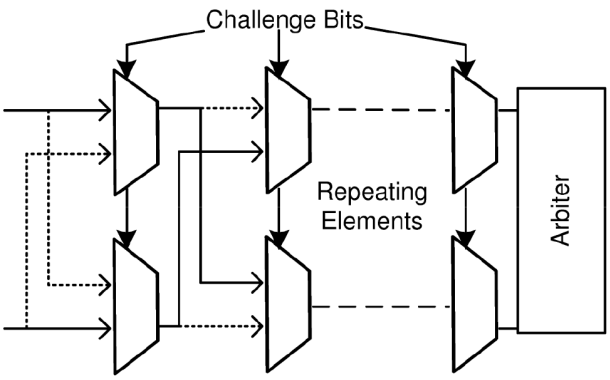

Figure 4. The Arbiter PUF structure.

The pairs of nets connected to the multiplexers (pairs shown with different patterns) need to be symmetric in order to minimize Δd_S. Symmetry requirements for Arbiter PUF architectures cannot be satisfied using available FPGA routing schemes, despite the apparent routing flexibility of FPGA devices. Using the best possible routing, the delay difference due to static variation routes is an order of magnitude higher than expected delay variation due to manufacturing variability. Yet an architecture without the mirror symmetry requirement, such a Ring Oscillator based PUF, can produce a working PUF. As can be seen in Figure (5) a Ring Oscillator PUF compares the frequency of two Ring Oscillators which have the same implementations and can produce an unpredicted output.

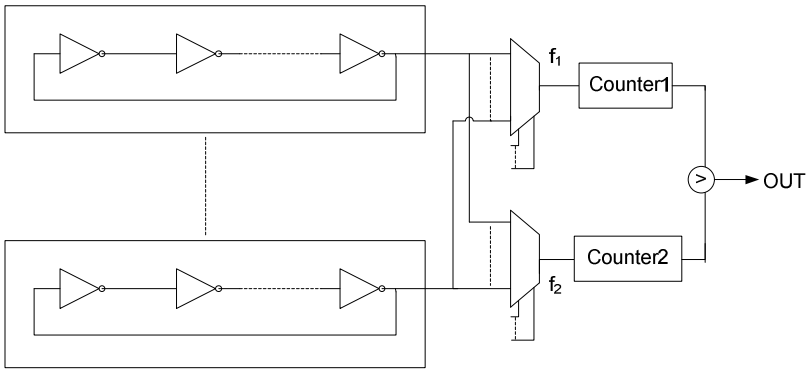

Figure 5. The Ring Oscillator PUF structure

One of the main disadvantages of Ring Oscillator PUFs is related to the shortage of their input vector length compared to arbiter PUFs. This is why the reachable state of the arbiter PUFs are much higher than Ring Oscillator PUFs. Table 1 shows a comparison of different aspects of the Ring Oscillator PUF with arbiter PUFs.

Table 1. Comparison of different properties of the Ring Oscillator PUF with Arbiter PUFs.

	Hardware Complexity	**Speed**	**Security**	**Implementation**	**No. Output States**
Arbiter-PUF	Low	High	Medium	Hard	High
RO-PUF	Low	Medium	High	Simple	Low

Now let us consider how many bits we can generate from the Ring Oscillator PUF. Each comparison of a pair of oscillators generates a bit. There are $\frac{N(N-1)}{2}$ distinct pairs given N ring oscillators. However, the entropy of this circuit, which corresponds to the number of independent bits that can be generated from the circuit, is clearly less than $\frac{N(N-1)}{2}$ because the bits obtained from pair-wise comparisons are correlated. Fortunately, it is possible to derive the maximum entropy of this circuit assuming pair-wise comparisons. There are N! different orderings of ring oscillators based on their frequencies. If the orderings are equally likely, the entropy will be $\log_2(\mathrm{N!})$ bits. Therefore the attacker can check the limited number of challenges and easily predict the modelling of the output structures. There are several strategies to attack a RO-PUF. As an example in [19] with a novel technique the operation complexity of predicting the output is decreased from $\mathrm{O}(N^2)$ to $\mathrm{O}(N.\log N)$ and with a limited number of challenge response pairs of $N_{CRP} \approx \frac{N(N-1)(1-2\varepsilon)}{2+\varepsilon(N-1)}$ can predict the output with a high accuracy. In a circuit of 1024 Ring Oscillators and by observing only 83941 challenge response pairs, they could predict the output with the accuracy of over 99%. This example shows the weakness of Ring Oscillator architecture against modelling attacks. Because of these vulnerabilities, some architectures were proposed [2, 12, 20] in order to improve the entropy of the circuits based of ring oscillators. In this paper first a survey of the related works done regarding ring oscillator based circuits will be presented and their disadvantages will be discussed then a novel circuit for ring oscillator will be presented which has a better entropy and higher number of challenge response pairs. This circuit has a nonlinear behaviour and higher security compared to the conventional ring oscillators. Because of its low hardware resources this circuit has a much higher performance as well.

3. RELATED WORKS:

The first improved ring oscillator architecture discussed here is IC-EK generator [2]. Block scheme of a whole IC-EK generator is shown in Figure 6. An ECC encoder generates the configuration code word which is translated by a code conversion circuit into the configuration circuit control vector. The oscillations counted consecutively by counter C_1 and C_2 will be compared to generate a response bit β_{ij} .

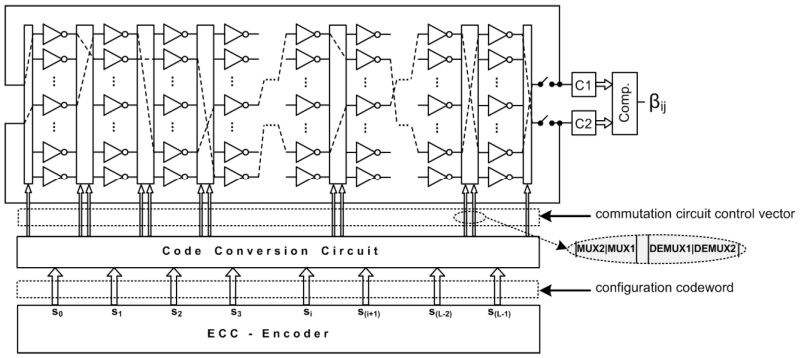

Figure 6. The Block Diagram of IC-EK generator.

Although this architecture has a higher number of challenge response compared to the conventional Ring-Oscillator PUF, there are a couple of major disadvantages. The first weakness of this circuit is the hardware overhead of the encoder circuit which is used

for the routing of the ring oscillators. This circuit also will remove some of the possible input challenges. But the worst problem happens when only one or two bits of the challenge changes. In this case there is a big gap in the frequencies, with having the old outputs and the equivalent frequency the new output can be easily predicted with probability of 0.5. Not that the prediction is easier if the number of inverters of each stage of Ring oscillator is low. But if the frequency difference is not large enough then the output is not stable. So this circuit have either security problem or stability problem.

There is another architecture used for RO-PUF which is called Configurable RO-PUF [12]. The simple architecture of a Configurable RO is shown in figure 7. In this circuit, instead of the conventional Ring Oscillators, configurable Ring oscillators are used. So with a slight change of the input bits the frequency of Ring oscillator will change and with a comparison the output is evaluated. The main disadvantage of this architecture is again the easily prediction of the output. In other words the new output can be easily predicted with probability of 0.5 if the old output is known. As an example suppose the old output is one and the frequency difference of the Ring oscillators is large, if one bit of the multiplexers select changes the probability to have another one in the output is higher than zero output.

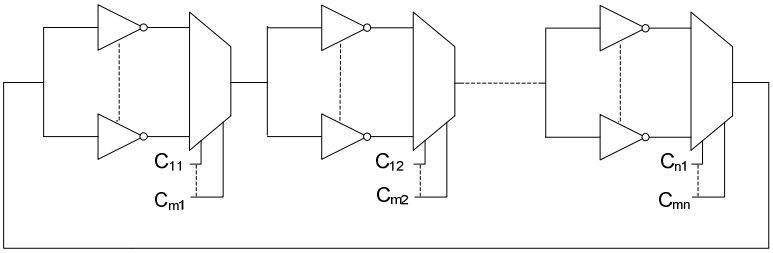

Figure 7. The Block Diagram of Configurable Ring Oscillator.

In other words, if the last two bits of the output is $o_1 o_2$, the above conditions happens then:

$$P(o_2 = 0|o_1 = 0) > P(o_2 = 0|o_1 = 1)\,, P(o_2 = 1|o_1 = 1) > P(o_2 = 1|o_1 = 0) \quad (3)$$

4. THE NOVEL ARCHITECTURE FOR PUF-BASED RNG

As we discussed in the section 4, there are several stability and security problems with the current ring oscillator architectures. In order to combat these issues, using the conventional architectures, a novel Ring oscillator circuit is proposed in this section. So in each stage of the Ring oscillator m inverter gates are employed.

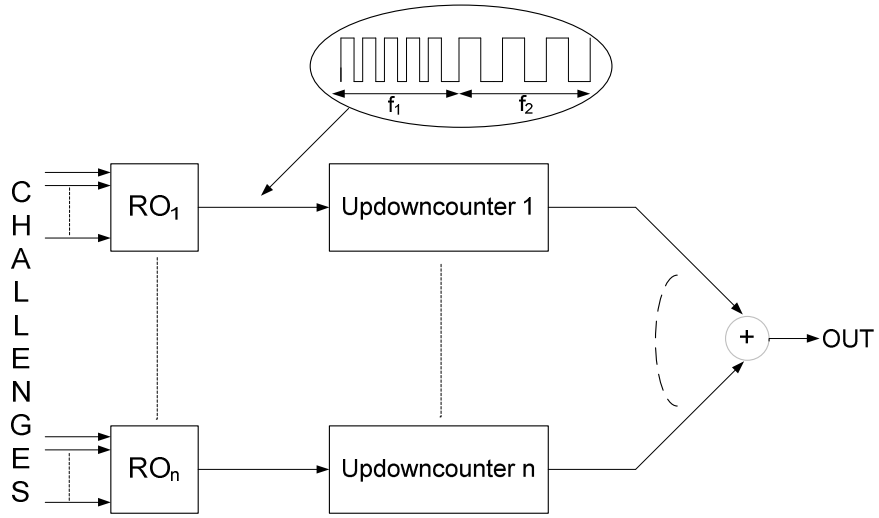

Figure 8. The Block Diagram of the Novel Ring Oscillator.

The circuit works in a way that the output of the Ring Oscillators is applied to a UP/Down counter. The counter first up count according to the output of the Ring oscillator (for an input challenge). Then the counter down counts and the results is compared to the original state of the counter. If the challenge of the ring oscillator is not changed or in other words the hamming distance of the input challenges are the same then the output of the ring oscillator will produce a zero in the up/down counter. In the implementation of each block, each inverter is implemented in only on LUT so we have a difference of the delays in the inverters and ring oscillators. For an n stage ring oscillator which has 2^m inverter gates, the delay of the circuit after one challenge input can be computed according to equation (4):

$$d(c_i) = \sum_{i=1}^{n} \sum_{j=1}^{2^m} d_{i,j} \prod_{k \in \Pi_k} (1 - c_{i,k}) \prod_{k \in \Lambda_k} c_{i,k} \tag{4}$$

Where $\Pi_k$the set of one is input challenges and Λ_k is the set of zero input challenges and we have$|\Pi_k| + |\Lambda_k| = \mathrm{m}$. Therefore according to this relation the output function of two challenges $c_a$and c_b can be calculated according to the equation ().

$$\mathrm{R_i}(c_a; c_b) = \begin{cases} 1 & d_i(c_a) - d_i(c_b) > th \\ 0 & d_i(c_a) - d_i(c_b) < -(th) \\ \mathrm{No-change} & \mathrm{o.w.} \end{cases} \tag{5}$$

In order to increase the stability of the circuit, the comparison is done by a threshold call th. So the effect of the oscillators which have a marginal frequency difference will be omitted and the stability of the circuit will be increased. The outputs of the Ring oscillators are Xored to produce the final output. In this architecture the current frequency of a Ring oscillator is compared to the last state frequency. This will decrease the speed of the architecture but on the other hand the required hardware for implementation is also decreased. For and m stage Ring oscillator there are $\frac{m(m-1)}{2}$ different pair comparisons and if we choose n ring oscillators for Xoring then we can

easily produce $(\frac{m(m-1)}{2})^n$ in dependent output bits. If the distribution of the inverter delays is Normal with a mean value of zero and variance of σ^2 $[d \sim N(0, \sigma^2)]$, since these delays have Identical independent distribution (i.i.d) their difference has also a normal distribution:

$$Prob\big(R_{i(new)}(c_a; c_b) = 1\big) = prob(\Delta d_i(c_a; c_b) > th) + prob(|\Delta d_i(c_a; c_b)| < th \mid R_{i(old)}(c_a; c_b) = 1)$$

Because of the usage of a threshold in this novel architecture the probability of prediction of the output by knowing the last output is still greater than 0.5. This weakness can easily lead to a modelling attack. If this threshold is picked with a small number the stability of the circuit will face problems and if the threshold is pick a large number then security is in danger. In order to solve this problem and increase the complexity and nonlinearity of the architecture all the ring oscillators are Xored first and the output like a scrambler will decide the next challenge. For this architecture two LFSRs with the length of 96 flip-flops are used the primitive update function of the LFSRs are chosen as equation (6,7).

$$f_1(x) = x^{96} + x^{94} + x^{49} + x^{47} + 1 \quad (6)$$

$$f_2(x) = x^{96} + x^7 + x^6 + x^4 + x^3 + x^2 + 1 \quad (7)$$

The final architecture of the novel ring oscillator is shown in the figure 9. This architecture is composed of 16 configurable classic ring oscillators each of which have 12 inverter gates and 3 multiplexer 4×1 . Therefore in total there are 96 input bit challenges which can be applied to the circuit. These input challenges are produced by output bits of two LFSRs. The last output will act as a scrambler and will choose which LFSR will be used for the input challenges.

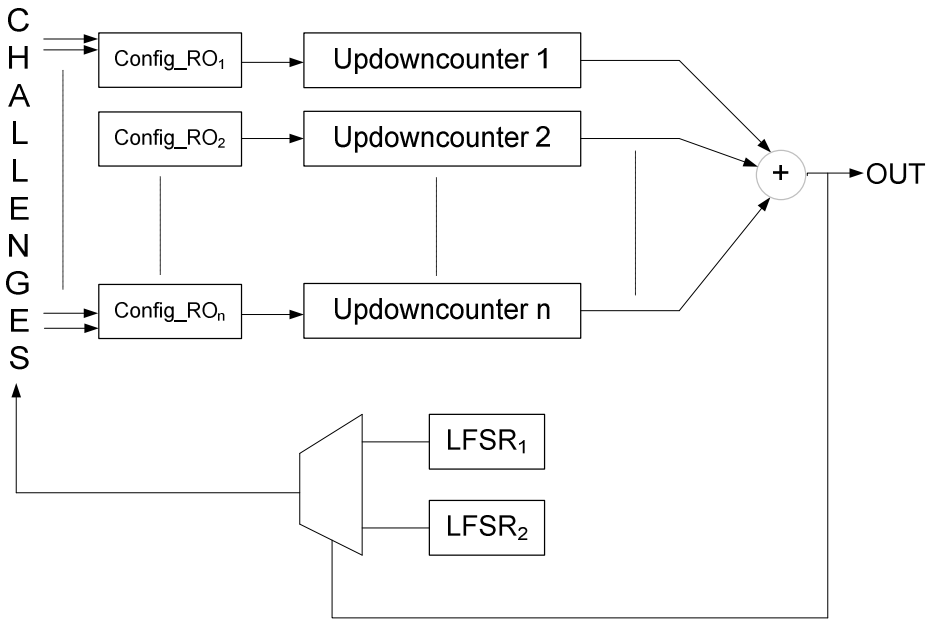

Figure 9. The Block Diagram of the Novel Ring Oscillator.

5. CONCLUSIONS

The paper addresses the security and stability issues in the design of PUF-based random number generators. The main security flaw in the design of ring oscillator PUFs is the modelling attacks and this will make them less likely to be used as random number generators. In this paper a novel architecture for ring oscillator PUFs is proposed. This architecture has solved both of security and stability problems of the classic ring oscillators. This idea has a higher data complexity and also nonlinearity. The final architecture has also lower hardware complexity which make it suitable for lightweight random number generators.

ACKNOWLEDGEMENTS

Mehdi Ayat wishes to thank the Iran Telecommunication Research Center (ITRC) for their financial support (www.itrc.ac.ir).

REFERENCES

[1] P. Kohlbrenner, K. Gaj, (2004) “An embedded true random number generator for FPGAs”, 12th international symposium on Field programmable gate arrays, FPGA '04, Pages 71--78.

[2] Lazich, Dejan E., Wuensche, Micaela, (2008) “Protection of Sensitive Security Parameters in Integrated Circuits”, Mathematical Methods in Computer Science, Lecture Notes in Computer Science, Volume 5393, Pages 157-178. (Fulltext)

[3] M. Majzoobi, F. Koushanfar, M. Potkonjak. “Lightweight Secure PUF.” IEEE/ACM International Conference on Computer Aided Design (ICCAD), 2008

[4] M. Majzoobi, F. Koushanfar, M. Potkonjak. “Testing Techniques for Hardware Security.” IEEE International Test Conference, 2008 (2)

[5] M. Majzoobi, F. Koushanfar, M. Potkonjak. “Techniques for Design and Implementation of Secure Reconfigurable PUFs.” ACM Transactions on Reconfigurable Technology and Systems, 2009

[6] F. Koushanfar, M. Majzoobi, M. Potkonjak. “Nonparametric Combinatorial Regression for Shape Constrained Modeling.”, IEEE Transactions on Signal Processing, 2008

[7] D. Shamsi, M. Majzoobi, F. Koushanfar, N. Kiyavash. "Multiple Statistical Validations for Sensor Networks Optimization.", innovations 2008

[8] M. Majzoobi, E.L. Dyer, A. Enably, F. Koushanfar. "Rapid PFGA Characterization Using Clock Synthesis and Signal Sparsity". IEEE International Test Conference (ITC), Austin, TX, November 2010.

[9] M. Majzoobi, A. Enably, F. Koushanfar. "FPGA Time-bounded Unclonable Authentication", Information Hiding Concerence (IH), 2010

[10] M. Majzoobi, F. Koushanfar, S. Devadas. "FPGA PUF using programmable delay lines.” IEEE international Workshop on Information Forensics and Security, 2010.

[11] R. Pappu, B. Recht, J. Taylor, N. Gershenfeld, “Physical One-Way Functions” SCIENCE, Vol. 297, Pages: 2026-2028, 2002. (02.PapEA.powf)

[12] A. Maiti, P. Schaumont, “Improving the quality of a Physical Unclonable Function using configurable Ring Oscillators”, International Conference on Field Programmable Logic and Applications (FPL’09), pages: 703 – 707, 2009. (5)

[13] G.E. Suh, C.W. O'Donnell, S. Ishan, D. Srinivas, “Design and implementation of the AEGIS single-chip secure processor using physical random functions” 32nd International Symposium on Computer Architecture (ISCA'05), Pages: 25 – 36, 2005. (007)

[14] J.H. Anderson, “A PUF design for secure FPGA-based embedded systems”, 15th Asia and South Pacific Design Automation Conference (ASP-DAC), pages: 1 – 6, 2010. (111)

[15] S. Morozov, A. Maiti, P. Schaumont, "An Analysis of Delay Based PUF Implementations on FPGA", Reconfigurable Computing: Architectures, Tools and Applications, Lecture Notes in Computer Science, 2010, Volume 5992/2010, Pages: 382-387, 2010. (fulltext3)

[16] P. S. Ravikanth, "*Physical One-Way Functions"*, *Ph.D*. thesis, Massachusetts Institute of Technology, March 2001. (Pappu-PhD-POWF-2001)

[17] J. Lee, L. Daihyun, B. Gassend, G. SUH, M. Van Dijk, S. Devadas. "A technique to build a secret key in integrated circuits for identification and authentication applications", Proceedings of the Symposium of VLSI Circuits, Pages: 176–179, 2004. (arbiter)

[18] G. Suh, S. Devadas, "Physical unclonable functions for device authentication and secret key generation", Proceedings of the Design Automation Conference (DAC), pages: 9–14, 2007. arbiter&RO_PUF

[19] U. R. Uhrmair, F. Sehnke, J. S. Olter, Gideon Dror, S. Devadas, J. Schmidhuber, "Modeling attacks on physical unclonable functions", Proceedings of the 17th ACM conference on Computer and communications security, 2010. (251_2)

[20] M. Yu and S. Devadas, "Recombination of Physical Unclonable Functions", *GOMACTech-10 Conference*, March 2010. (gomactech2010)

[21] R. ANDERSON, (2001) "*Security Engineering: A Guide to Building Dependable Distributed Systems*", John Wiley and Sons.

[22] B. Gassend, D. Clarke, M. Van Dijk, S. Devadas, "Silicon physical random functions. In Proceedings of the Conference on Computer and Communications Security (CCS), Pages: 148–160, 2002.

[23] D. Lim, J.W. Lee, B. Gassend, G.E. Suh, M.Van Dijk, S. Devadas, "Extracting secret keys from integrated circuits. IEEE Transactions on Very Large Scale Integration (VLSI) Systems (2005).