# Design and FPGA Implementation of Secure Key Management

Xiaoxun Li, Zhiqiang Gao, and Guoqiang Bai

***ABSTRACT*—Cryptographic device ensures the secure information exchange and private authentication even in the face of various attacks. In this paper, we focus on how to prevent physical attacks and present the PUF-based security secret-key generation architecture. We develop a novel Spread PUF architecture that is more secured and has higher performance than existing PUF architectures. Our thesis also gives applications of the Spread PUF for security secret-key management, unlike prior proposals which only present their designed PUF without showing how to apply it in practical security device. All the experiments are implemented on FPGAs.**

***Index Terms*—Secret-key, Physical attack, Physical Unclonable Function (PUF), FPGA.**

## I. INTRODUCTION

As cryptographic devices become ubiquitous, secure and trusted computation are escalated as the prior needs. Fundamental to almost any security question is how to protect the secret-key. However, recently developed invasive and noninvasive physical tampering methods such as micro-probing[1], FIB, reverse engineering[2], glitch attacks, and power analysis have made it possible to extract digitalized secret information from integrated circuit(IC) and access systems by using illegal copies of the secret information. For example, an adversary can remove a device's package and reconstruct the layout of the circuit using chemical and optical methods [2]. Even the data stored in EEPROM or FLASH can be revealed by sophisticated methods. So the traditional assumption that security device have been designed to provide security against an adversary who has only black-box access to the secret information of honest parties is removed [3]. It was observed that this problem is highly non-trivial and that in the most general setting no security can be guaranteed.

One solution is to build a sensor network which is easily destroyed under physical attacks. At the same time, this network is one part of the circuit. Without it the circuit can not complete its operation. These sensor nodes are vulnerable to little invasive, and even a tiny modification can propagate fake messages which bring down the entire network. Such attacks can be preventable if the message passing routines of each node which together form the tamper-evident environment.

The variations of process make the number of ion implantation, transistor width and length of wire the most unpredictable characters. On the high level, it reflected by the variation of threshold voltage and the delay of wire. These variations are beyond the control of the manufacturers. We present in this paper the secure secrets generator architecture which utilizes these process variations to construct Physical Unclonable Functions to veritably create and maintain secret-keys. The concept of PUFs have first been proposed in [4] and been developed in [5] [6] [7]. We propose a new PUF architecture- Spread PUF, which extract a secret-key from hidden timing or delay information rather than digital information. Compared with previous structure, it is more rigorous and it mix complex algorithms into PUF. We also give the implementation of security secret-key management on FPGA.

The remainder of the paper is organized as follows. In Section II we provide the previous work on PUF. We present the design of our PUF architecture in Section III. We perform experimental studies by using statistics in Section IV. Section V detailed designs how Spread PUF can be used for security secret-key management implementation. Finally, we conclude the paper in Section VI.

Manuscript received November 26, 2010. This work was supported in part by the National Natural Science Foundation of China.

X. X. Li is with the Institute of Microelectronics, Tsinghua University, Beijing, 100084,China (phone: +86-10-62792712; e-mail: LixiaoxunTU@gmail.com).

Z. Q. Gao is with the Institute of Microelectronics, Tsinghua University, Beijing, 100084,China (e-mail: Gaozhiqiang@mail.tsinghua.edu.cn).

G. Q. Bai is with the Institute of Microelectronics, Tsinghua University, Beijing, 100084,China (e-mail: Baigq@tsinghua.edu.cn).

## II. PRIVIOUS WORKS

In 2001, Pappu et al. [4], [8], introduced the concept of Physical Unclonable Functions (PUFs) or Physical Random Functions. The original construction of [8] is based on the response (scattering) obtained when shining a laser on a bubble-filled transparent epoxy wafer. Lim et al. [5] introduce arbiter based PUFs which use a differential structure and an arbiter to distinguish the difference in the delay between the paths. In [6], the authors introduced intrinsic PUFs for FPGAs based on the startup values of SRAM memories. An SRAM cell is a cross-coupled inverter circuit which maintains its state using positive feedback. During startup, a slight difference in the voltage on one of the floating inverters output is driven positively within the loop to force the SRAM to go to a 1 or a 0. In [7], Tuyls et al. present a coating PUF in which an IC is covered with a protective matrix coating, doped with random dielectric particles at random locations. The IC also has a top metal layer with an array of sensors to measure the local capacitance of the coating matrix that is used to characterize the IC.

## III. SPREAD PUF

In this section, we introduce a new PUF design based on architecture symmetry delay chains and latch to determine the physical unclonable secret-key. It is more complicated to model and can fast generate secret- key in one clock. We call this new structure a *Spread PUF*. Compared to the PUFs described in the previous section, the Spread PUF allows an easier implementation for practical application, an easier function evaluation on FPGAs, and higher reliability and speed.

### A. SPREAD PUF overview

A Spread PUF is a complicated delay network which can generate a series of binary bits. Fig. 1 depicts the Spread PUF schematic. It is composed by transposition units and latches which determine the faster of two signals. In this scheme, we excite the source with a positive pulse. At the first stage, the source separate to two directions. At the second stage, the above signal separate to two directions and the bottom do the same thing. The spread speed grows exponentially. At the n stage, the number of signal is $2^n$ .

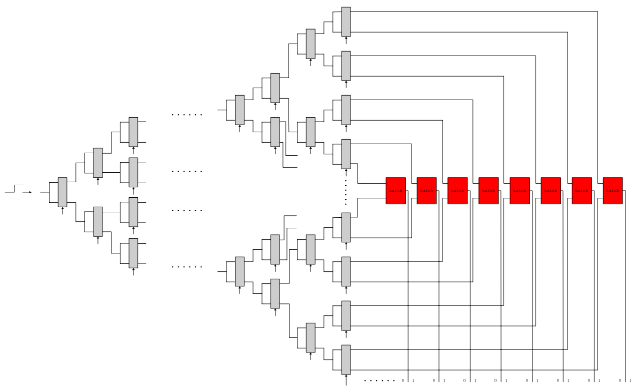


Fig.1. Structure of a SPREAD PUF

As the unpredictability of process variation, each two delay path pairs' arrival time is undetermined and the race against each other can be use for information storage. The latch determines which rising edge arrives first and sets its output to 0 or 1 depending on the winner. The structural symmetry signals are used to compare for they go through the most different path even though they across the same distance visually. N pairs of signals can determine n bits keys. At every section, there exist shift units which can change two directions which make paths different. The circuit with n stages takes $2^n - 1$ challenge bits as inputs to configure the delay paths and can determine $2^{n-1}$ bits key as output. There are $2^n - 1$ switches, and each value of challenge can change the configuration of the delay paths. Thus, the number of possible different configurations of the delay paths is $2^{2^n - 1}$ which is very huge if $n \geq 5$. The delay difference between the top and bottom paths is determined by the configuration of delay paths. At the output, we need to generate determined bits and we challenge the circuit with all possible combination. The solutions can vary obvious across ICs if the maximum delay variation in manufacturing is as great as possible. For the challenges, them should be imposed on the switches first of all and the source need to wait the configuration stable in that the delay difference between challenges could affect correct judgments. In addition, in order to maximize the path delay variation of PUF responses and prevent the simplification of model, the delay paths must be placed and routed symmetrically that can minimize the nominal delay difference between each two paths. Fig. 2 details a transposition scatter component. A unit scatters its input port i to the output ports (o0 and o1) with different configurations depending on the control bit (Ci); for Ci = 0, the paths go straight through, while for Ci=1 they are crossed. It is implemented with a pair of 2-to-1 multiplexers and delay cell.

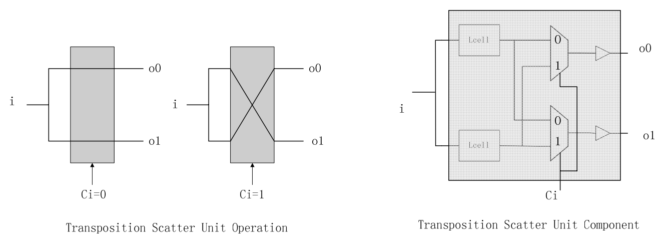
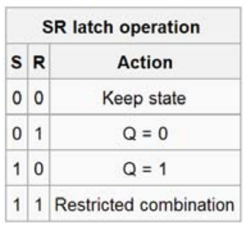

i
o0
o1
Ci=0
Ci=1
Transposition Scatter Unit Operation
Lcell
Ci
Transposition Scatter Unit Component

Fig.2. Operation and Component of Transposition Scatter

Regarding the parallel lines through each node as the axis of symmetry, the cells are placed and routed symmetrically. On an FPGA platform, implementing a PUF is a challenging task because we neither have the ability to exploit layout level design techniques, nor have the knowledge about the gate-level structure of an FPGA fabric. In this constrained platform, it is expected that we lose significant variation information upfront, due to the averaging effect of individual component-level variations over larger composite structures such as LUTs and other vendor-specific structures. However, as we extract the variation of process, the imperfect of structure can be ignored in the functional verification phase.

### B. The architecture of important units

The latch arbiter plays an important part in the PUF architecture. The scheme needs to exploits the precision delay of the path and the arbiter determines the winner for which to generate bits we need. A well-designed data latch with careful and precise simulation is necessary. Fig.3 is the logic gate level schematic of the latch. It is built by cross-coupling NOR gates with inverter drivers.



| SR latch operation | | |
|---|---|---|
| S | R | Action |
| 0 | 0 | Keep state |
| 0 | 1 | Q = 0 |
| 1 | 0 | Q = 1 |
| 1 | 1 | Restricted combination |

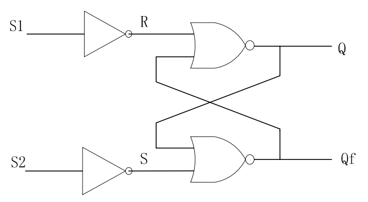


Fig.3. The Schematic of Arbiter Latch

When S1 and S2 both are zero, then the output of latch is $Q = \overline{Q} = 0$. Giving a challenge, the signal will go through two paths and arrive at the input of latch. If the above signal arrival first, S=0 and pull-up transistor turn to pull the output Q to high. Feedback make $\overline{Q} = 0$ whether the R=0 or 1. We also need to consider inappropriate circumstances in circuits when both signal arrival at the same time or the gap is very small. SR may go low simultaneously (i.e. a transition from restricted to keep). The output would lock at either 1 or 0 depending on the propagation time relations between the gates (a race condition). In certain implementations, it could also lead to longer ringing (damped oscillations) before the output settles, and thereby result in undetermined values (errors) in high-frequency digital circuits. Fig. 4 shows the operation of the latch as an arbiter in all possible delay differences. We define $\Delta t = t_{s2} - t_{s1}$ and the value of each graph is as follow: (1) 350 ps; (2) 200 ps; (3) 100 ps; (4) 0 ps; (5) -350 ps. We can see the threshold is not exactly at $\Delta t = 0$, but it is around 200ps. But it does not matter as two sides of the threshold can give the determined value and we just treat this point as center.

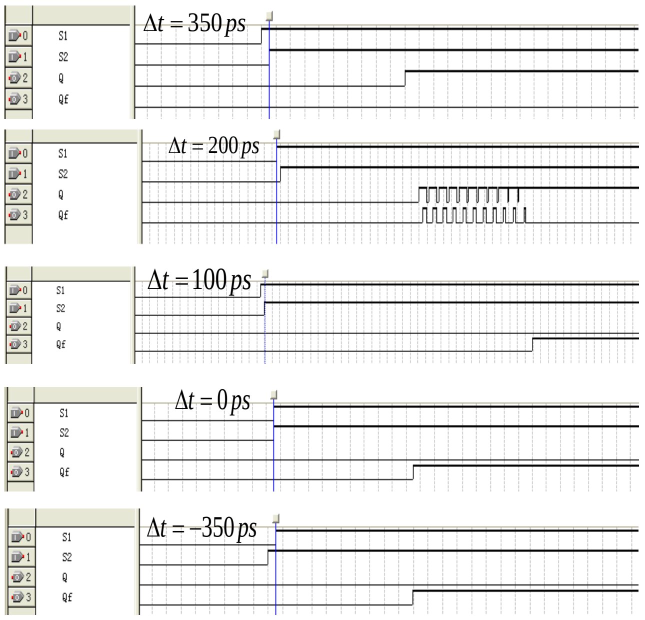


Fig. 4. The Simulation Results of Arbiter Latch

### C. Characteristics of Spread PUFs

PUF's output is unclonable and unpredictable for individual. In order to know the habits of spread PUF, we need to identify individual ICs. We treat PUF as a function f(x), and the secret- key is the dependent variable. We can construct the equation as $K = f(C)$. Solving the equation can tell us what configuration of the spread PUF could make it generates the secret-key we wanted. Obviously, the function is one direction function and it must have many solutions. We believe that the variation between each two different delay paths for big probability being different from each other with random configurations. And most of delay difference pair is sufficiently large compared to the noise effect. We also believe that the inter-chip variation between the outputs of PUFs for big probability being different from the same configuration. Then, by generating a sufficient number of solutions, we can distinguish different ICs with negligible probability of error.

Meanwhile, environmental variation, metastable, and aging could cause noise of PUF responses in measurements. To quantify the effect of noise, we define the bit errors percentage as the probability that how many a newly measured PUF output with all the reference solution apply on a given IC different from the corresponding reference output which we used to solve the function. To estimate the identification capability of the spread PUF scheme, this noise probability should be precisely measured by experiments. This will be shown in the next section.

In the spread PUF scheme, we devise that the delay path pair go through the same distance in theory and use a relative delay measurement which can reduce a significant amount of noise effect induced by environmental variation. Even if environmental variation changes the absolute values of two delays, the difference between the two delays is likely to be preserved.

Additionally, with the increasing of security and speed, the complexity and the area go up. A 256bits outputs need 9 stages. This measurement scheme takes only one cycle to finish secret key generation and we could make the cycle as the multiple of the system clock in order to satisfy the timing relations. But then we can not solve all the solutions of the function because the computation is $2^{256}-1$ and it will cost $1.835\times10^{61}$ years for a 200MHz speed. So we will use a hardware random number generator to try the solutions. The calculation stops when there is one configuration generate the secret-key we wanted. Then we can store the solution into the device to reconfigure the PUF when need secret-key.

## IV. Experiment of SPREAD PUF on FPGA

To examine characteristic of spread PUF, we solve all the solution when the stage is low and using random number generator to try the solution when the stage is high. The experiment using a FPGA implementation of an spread PUF may not satisfy the symmetrical architecture, but we can test inter-chip variation and environment variation conveniently. We present the experimental results of characteristics of spread PUFs such as inter-chip variation by using different FPGA boards or change HDL code and environmental noise over a practical range of environment variation. We also examine an aging effect that can potentially degrade identification capability after prolonged use.

*Inter-chip Variation:* Inter-chip variation of spread PUFs has been tested using 5 stages select delay path which generate 16 bits data. The solution space is $2^{31}$. With the system clock of 50MHz, we use one cycle to determine one configuration. This will takes 43 seconds for each PUF to solve all the solutions. There are five FPGA boards and each have 20 type modified HDL code so as to form 100 PUF architectures. Fig. 5 shows the distribution histogram of the number of solutions. The expected number is $2^{15}$ and the experiment results fall into 32000-33000 with great probability. No two numbers is identical, and the relation between two solutions has no obvious features. Since manufacturing variation consists of transistor to transistor and chip-to-chip, inter-chip variations then are classified to every board and assess internal units' variation. We compared the inter-chip variation within a single FPGA and across FPGAs. Fig. 6 shows the average, min, and max number of solution variation across these five FPGA boards.

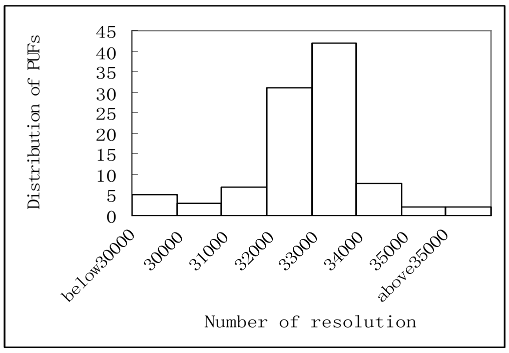


Fig.5 Distribution of the Number of Solutions

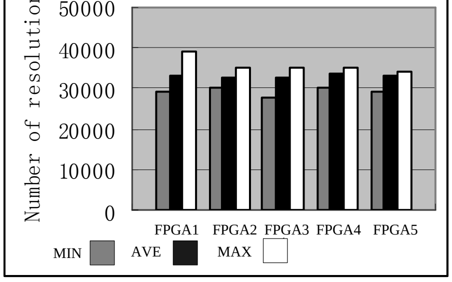


Fig.6 Distribution of the Number of Solutions across FPGAs

Fig.5. is the histogram of the solution distribution for all 100 PUF configurations. Fig.6. classify these PUFs into every FPGAs. The results show chaotic distribution no matter in signal FPGA or across FPGAs, even the amount of PUF pairs is not strongly support the judgment. And 98% numbers of solution close to expectations with probability.

*Noise and aging effect on PUF:* Environmental variations such as temperature and power supply voltage variations are the primary causes of noise in PUF responses. Even without environmental variations, a setup time violation for a latch or the small fluctuation of junction temperatures and internal voltages can cause measurement noise. And as the using of device, the variations raised from aging effect can also cause measurement errors. This time we increase the stage of delay distance and use random number generator to try the solution. For a fixed configuration, we gain the first 100 solutions and write these solutions into device to configure the PUF. Then we test the output of these PUF configurations under temperature variation and aging effect. Fig. 7 shows the amount of environmental noise introduced by temperature variations. The solutions are measured at 25℃. Even if the circumstance temperature increases to 45℃, the maximal variation of output key is 3.12%. This shows that the differential structure of spread PUF circuit reduces the environmental variations on delay path and the main effect comes from the variation of latch. Electro migration and hot-carrier effects cause the aging of wires and transistors in ICs. To consider the aging effect, Fig. 8 shows the result of a one-month-long aging test. We still calculated the percentage of outputs bit differences from the original secret key with 100 solution pairs which were generated at the start of the one-month period. The percentage varies slightly around 0.25%. We conclude that any significant performance degradation has not been observed in this one-month aging test under normal operating conditions. However, a longer term aging test in more severe environmental fluctuation must be performed to guarantee the reliability of the PUF scheme in practice. For industry use of the PUF scheme, the test range of environmental conditions should be enlarged to prove the reliability of operation. For applications where a secret key is generated, we require substantially higher reliability. This can be achieved using error correction, which is discussed in the next section.

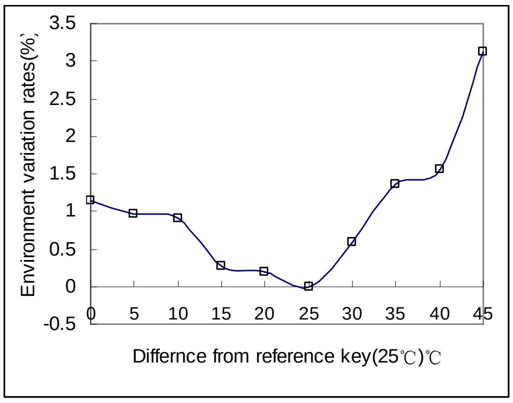


Fig.7 Environmental Noise introduced by Temperature Variations

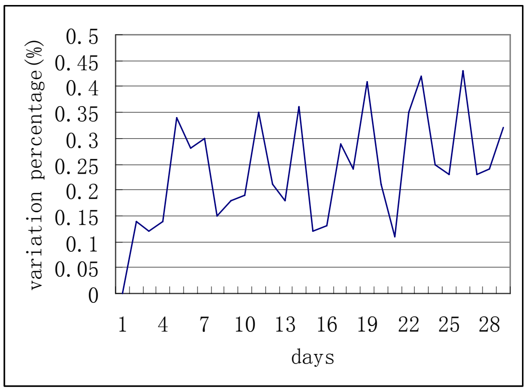


Fig.8 Aging Variation Test in 30 days

## V. Key Management Implementation

PUF responses can be considered as secrets because they are randomly determined by manufacturing variations, and difficult to predict without access the PUF. If the PUF architecture can generate the correct secret- key, the system can authenticate the information and deliver notice. Because of the measurement noise, PUF responses are likely to be slightly different on each evaluation, even for the exact same configuration. So we can not use PUF to generate secret key directly. There need some processes on the data. At first we provide the device with the original m-bits secret key K. A simple encryption is necessary. Then the processor computes a BCH Code syndrome for the encrypted key E (K). The BCH code is a popular error correcting code that is widely used for binary data. A syndrome is redundant information that allows a BCH decoder to correct errors on the PUF delay circuit output. The BCH (n, k, t, d) code can correct up to $(d-1)/2$ errors out of n bits with an (n- k)-bit syndrome (b = n - k). For example, we can use the BCH (255, 139, 15) code to reliably generate 139-bit data. As the experiment shown, the error code rate do not need to much. The encoded data is expected response of PUF and now we can construct the equation as follow:

$$E(K) \circ BS(E(K)) = PUF(C)$$

Supposing input m- bit secret-key has been encrypted by the hardware and E (K) is k bits. Then BCH (n, k, t) compute the syndrome and add it to the E (K). So the response of PUF is $E(K) \circ BS(E(K))$. The random number generator can generate random 2n-1 bits numbers. When a random number configuration can makes the spread PUF response $E(K) \circ BS(E(K))$, we get one of the solution C1. For higher security, we can solve the PUF as many as possible and random choose one solution.

After that we can write the solution C into memory which can be EEPROM or ROM. In the process of operation, the circuit use C to configure the spread PUT delay path. The latches determine the winner of fast and generate n bits data. This bit series may not accurately equate to $E(K) \circ BS(E(K))$. Then the BCH decoder's correction ability is useful and E (K) will be regained if the error within a reasonable range. Maybe the BCH's error correction fails only once in half a million tries or the PUF response failure are so big that the BCH code cannot correct all the errors. After all, the probability of a incorrect is negligible. Therefore, with a high probability, the processor can believe the secret- key after a decryption $E^{-1}$.

The returned secret-key is: $K = E^{-1}(BD(PUF(C1)))$

Fig.9 illustrates our secure secret- key management implementation. We implemented our secure processor on an FPGA to validate and evaluate our design. All processor components including the processing core and protection modules are written in VHDL RTL. Our current implementation runs at 50MHz on an Alter StratixIII FPGA with 256-MB off-chip SDRAM (PC100 DIMM).

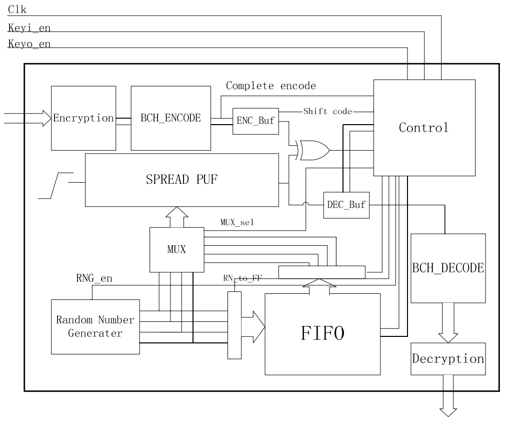


Fig.9 System Block Diagram of FPAG Implementation

## VI. CONCLUTION

We have described the security secret- key management architecture that can prevent physical attacks. A Spread PUF (Physical unclonable function) is used as the core of the processing architecture to reliably create, protect, and share secrets. The new proposed PUF provides more security and a few areas overhead. The integration of this PUF can run at a high speed and create precise secret-keys. The combination of the physical structure and cryptography is a new contribution which allows using mathematical model to describe the security of physical structure. Our design has been implemented on FPGAs, and we have shown that adoption BCH with low correction ability for verification of PUF output is reasonable.

## REFERENCES

[1] O. Kommerling, M. G. Kuhn, "Design principles for tamper-resistant smartcard processors," in Proc. USENIX Workshop Smartcard Technology, pp. 9–20, 1999.

[2] R. Torrance, D. James, "The State-of-the-Art in IC Reverse Engineering," Cryptographic Hardware and Embedded Systems — CHES 2009, pp. 363–381, Oct 2009.

[3] R. Gennaro, A. Lysyanskaya, T. Malkin, S. Micali and T. Rabin, "Algorithmic Tamper-Proof Security: Theoretical Foundations for Security against Hardware Tampering," Theory of Cryptography Conference (TCC), pp. 258-277, Feb 2004.

[4] R. Pappu, B. Recht, J. Taylor, and N. Gershen-Feld, "Physical one-way functions," *Science*, vol. 297, pp. 2026–2030, 2002.

[5] J.W. Lee, D. Lim, B. Gassend, G. E. Suh, M. vanDijk, and S. Devadas. "A technique to build a secret key in integrated circuits with identification and authentication applications", Proceedings of the IEEE VLSI Circuits Symposium, June 2004.

[6] J. Guajardo, S. Kumar, G. Schrijen, and P. Tuyls. "FPGA Intrinsic PUF and Their Use for IPProtection. Cryptographic Hardware and Embedded Systems", CHES 2007, pp.63–80, Oct 2007.

[7] P.Tuyls, G.-J. Schrijen, B.Skoric, J. van Geloven, N. Verhaegh, and R. Wolters. Read-Proof Hardware from Protective Coatings .Cryptographic Hardware and Embedded Systems — CHES 2006, pages 369–383, Oct 2006.

[8] R. S. Pappu. "Physical one-way functions." PhD thesis. Massachusetts Institute of Technology. Mar 2001.