Techniques for Design and Implementation of Secure Reconfigurable PUFs

MEHRDAD MAJZOOBI and FARINAZ KOUSHANFAR
Rice University
and
MIODRAG POTKONJAK
University of California, Los Angeles

Physically unclonable functions (PUFs) provide a basis for many security and digital rights management protocols. PUF-based security approaches have numerous comparative strengths with respect to traditional cryptography-based techniques, including resilience against physical and side channel attacks and suitability for lightweight protocols. However, classical delay-based PUF structures have a number of drawbacks including susceptibility to guessing, reverse engineering, and emulation attacks, as well as sensitivity to operational and environmental variations.

To address these limitations, we have developed a new set of techniques for FPGA-based PUF design and implementation. We demonstrate how reconfigurability can be exploited to eliminate the stated PUF limitations. We also show how FPGA-based PUFs can be used for privacy protection. Furthermore, reconfigurability enables the introduction of new techniques for PUF testing. The effectiveness of all the proposed techniques is validated using extensive implementations, simulations, and statistical analysis.

5

Authors' addresses: M. Majzoobi and F. Koushanfar, Department of Electrical and Computer Engineering, Rice University, Houston, TX 77005; email: Mm7@rice.edu; M. Potkonjak, Department of Computer Science, University of California Los Angeles, Los Angeles, CA 90095.

1. INTRODUCTION

Physically unclonable functions (PUFs) are physical systems with well-defined and stable mapping from a set of inputs (*challenges*) to a set of outputs (*responses*). Mapping is such that the owner of the system can rapidly obtain the output for any specified input but there is small probability of obtaining the output in any reasonable time by other parties [Pappu et al. 2002]. PUFs should be also prohibitively hard to copy (clone), emulate, simulate, or predict.

There is a wide consensus that intrinsic manufacturing variability of modern and pending deep submicron silicon is an excellent PUF implementation platform [Gassend et al. 2002b, 2002a; Trimberger 2007; Koushanfar and Potkonjak 2007; Bolotnyy and Robins 2007]. 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) [Anderson 2001; Hoeneisen and Mead 1972], the absence of covert channels (e.g., power, delay, electromagnetic measurements), and much lower time, speed, and power overheads [Gassend et al. 2002b, 2002a; Suh and Devadas 2007]. PUFs have been used for a variety of security applications ranging from ID creation and authentication [Gassend et al. 2002b, 2002a; Trimberger 2007; Koushanfar and Potkonjak 2007] to hardware metering and remote enabling and disabling of integrated circuits [Gassend et al. 2002b, 2002a; Trimberger 2007; Koushanfar and Potkonjak 2007; Alkabani et al. 2007].

Our research has two conceptual sources: (i) natural PUF evolution through vulnerability analysis; and (ii) quest to identify and create the best ways to leverage reconfigurability to improve PUF's security and operational properties. Unfortunately, recent analysis have demonstrated that many of the current state-of-the-art PUF structures are susceptible to a variety of security attacks. Our objective is to design and analyze reconfigurable robust PUFs that are resilient against different types of attacks.

Our analysis considers four types of PUF security attacks:(i) reverse engineering; (ii) emulation and statistical modeling; (iii) replay (man-in-the-middle); and (iv) reconfigurability-specific vulnerabilities. Reverse engineering aims at extracting the delay parameters of each delay element. The goal of emulation attack is to efficiently compress and store the PUF challenge/responses. Statistical attacks predict the value of the PUF outputs by exploiting the correlation among them and/or between the outputs and inputs. Replay attack looks for repeated challenges. This attack is in particular dangerous for PUF-based digital rights management protocols. A related attack, to a certain level, is the one where PUFs are fabricated in such a way that their replication is easy for a specific level of manufacturing variability. Finally, reconfigurability attacks aim to leverage the properties of reconfigurable implementation platform to compromise the security of the PUF. Our goal is to create reconfigurable PUF structures and the accompanying test procedures that ensure resiliency against all the stated attacks.
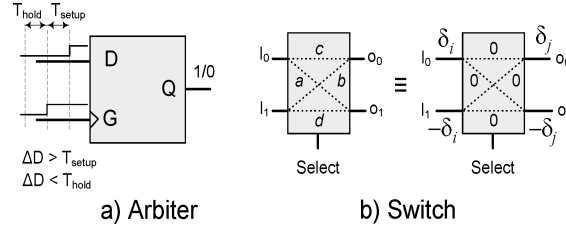
Fig. 1. PUF fundamental building blocks.

The starting point for our research is a new, generic, modular, and easy to parameterize PUF structure. The structure includes modules for combination of individual challenge bits, different configuration schemes of delay elements, and combinations of a subsets of the outputs using combinational circuitry to defend against the stated attacks. We show how reconfigurability can be employed to strengthen each of these defense mechanisms, to enable delay characterization, and to create notions of one-time PUFs.

The remainder of the article is organized as follows. A brief background on PUFs and variation modeling is given in Section 2. Section 3 presents a survey of related literature. We analyze vulnerabilities of PUFs and countermeasures in Section 4. Next in Section 5, we present a testing and characterization mechanism that is used for defining response error probability, PUF diagnosis, and compression of the challenge-responses. An approach to designing a secure and reliable PUF that overcomes the current vulnerabilities is presented in Section 6. In Section 7, a secure FPGA-based authentication system is presented. We demonstrate the effectiveness of the proposed concept and methods by extensive simulation and implementation in Sections 8 and 9 respectively.

2. PRELIMINARIES

2.1 Silicon Physically Unclonable Functions

Silicon PUFs exploit manufacturing variability to generate a unique input/output mapping for each IC. Delay-based silicon PUFs use 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. 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.
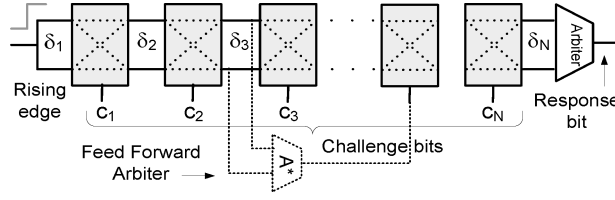
Fig. 2. Parallel PUF structure. The feed forward arbiter (shown in the dashed line) is used to introduce nonlinearity.

Lee et al. [2004] 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. 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. 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 feedforward (FF) arbiters, in the PUF structure [Gassend et al. 2002b]. 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 [Majzoobi et al. 2008b].

3. RELATED WORK

There is a wide and diverse body of literature related to the research presented in this article including reconfigurable computing, secure and trustable computing systems, physically unclonable functions (PUFs), techniques for hardware intellectual property protection, manufacturing variability (MV), and computer-aided techniques for addressing MV. We restrict our attention only on the most directly related research and development results. There are four major conceptual starting points for our research: (i) MV-based unique identifiers (IDs); (ii) security and reconfigurability (FPGAs); (iii) hardware security attacks; and (iv) integrated circuits (ICs) characterization.

Inevitable manufacturing variability, mainly due to dopants fluctuations, has been recognized as one of fundamental physical and technological CMOS scaling barriers in the early and mid-seventies [Hoeneisen and Mead 1972; Keyes 1975; Mead 1994]. In the late nineties, it again received a great deal of attention, since the first experimental studies demonstrate the validity of

early predictions [Asenov 1998]. Inspired by these studies and developments, Lofstrom at SiidTech in Portland and his collaborators where first to propose intrinsic silicon MV for ID extractions [Lofstrom et al. 2000]. Soon, several works from other groups followed [Maeda et al. 2003; Su et al. 2007]. Also, MV has been used as a basis for creation of high-quality random number generators [Sunar et al. 2007]. Extraction of unique gate-level features from the legacy designs for using as IDs was proposed in Alkabani et al. [2008b].

IC IDs are completely static features that provide excellent accounting mechanisms, but essentially have no security features. A great conceptual step forward was achieved by [Pappu et al. 2002] who introduced the notion of PUF. Their initial targeted PUF platform was an optical coherence system. A significant practical step to enable instantaneous and widespread application of PUF concept was proposal of Devadas et al. who leverage silicon MV for this task [Gassend et al. 2002b, 2003, 2004; Lee et al. 2004; Suh and Devadas 2007]. In addition, they developed a set of PUF architectures and a suite of PUF-based security protocols. These works motivated several silicon PUFs that use various mechanisms to extract a secret [Ozturk et al. 2008; Jie and Lach 2008; Tuyls et al. 2006]. Recently, by exploring the relationships between PUF-based IDs and functionality of the pertinent IC, researchers were able to create a comprehensive and powerful system of digital rights management protocols, including remote IC enabling and disabling and passive hardware metering [Alkabani et al. 2007; Alkabani and Koushanfar 2007]. Interestingly, the application domain of PUFs is much larger; They can be powerful candidates for creation of a new generation of security and cryptographical protocols that are intrinsically more resilient against physical and side channel attacks [Koushanfar and Potkonjak 2007]. This wide range of PUF applications has one ramification: significantly more stringent operation and security requirements. There are also conceptually sharply different mechanisms, one that use small scale reconfigurability, to associate unique IDs to each IC of a specific design [Koushanfar et al. 2001; Koushanfar and Qu 2001].

Unfortunately, the current generation of MV-based PUFs often is subject to significant security vulnerabilities. Recently, we have demonstrated surprisingly simple ways to reverse engineer and even emulate several PUF classes as well as their susceptibility to other types of attacks including (statistical) guessing and induced instability [Majzoobi et al. 2008b]. Our primary research objective in this article is to demonstrate that reconfigurability may serve as a principal component of techniques for PUF fortification against vulnerabilities.

Field-programmable gate arrays are by far the most popular and practical reconfigurable computing platform [Kuon et al. 2008]. The impact and techniques to address MV in FPGA recently attracted a great deal of attention [Drimer 2007; Fry and Langhammer 2005; Brizek et al. 2005; Eisenbarth et al. 2007; Glas et al. 2008; Guajardo et al. 2007; Kumar et al. 2008]. Several class of PUFs for static ID creation including SRAM and Butterfly PUFs were introduced and implemented on FPGAs in Philips Research Lab in Europe [Kumar et al. 2008; Guajardo et al. 2007]. Important conceptual and positional FPGA security references include [Wollinger et al. 2004; Trimberger 2007; Guneysu et al. 2007]. An excellent collection of security and intellectual property

protection papers can be accessed.[1] There are also more recent papers include [Drimer 2007; Fry and Langhammer 2005; Brizek et al. 2005; Eisenbarth et al. 2007; Glas et al. 2008].

Silicon manufacturing is a widely studied topic in many areas of computer-aided design. A recent excellent survey on CMOS MV is Bernstein et al. [2006]. There are two set of techniques for gate level characterization. The first one employs direct wafer microscopic measurements [Friedberg et al. 2005]. The other set of techniques use nondestructive indirect power and delay measurements and sophisticated techniques for solving systems of overconstrainted system of linear equation in presence of noisy data [Alkabani et al. 2008a; Shamsi et al. 2008].

Our primary research objective in this article and our earlier conference manuscript [Majzoobi et al. 2008a] is to demonstrate that reconfigurability may serve as a principal component of techniques for PUF fortification against vulnerabilities.

4. VULNERABILITIES AND SAFEGUARDS

In this section, we discuss the potential vulnerabilities of PUFs. Next, we outline possible countermeasures and safeguards.

4.1 Vulnerabilities

The PUF vulnerabilities are discussed by presenting attacks. The possible attacks are as follows:

4.1.1 *Reverse Engineering*. The reverse engineering attacks aim at estimating component-wise characteristics of the system (e.g., gate delays), so that the adversary could either clone the system or develop a software counterfeit for the PUF. Since cloning the PUFs is technologically infeasible, the attacker's objective is focused on soft-modeling the structure's behavior. In an effective reverse engineering attempt, the adversary models the system of N components in polynomial time with respect to N. This is because by linearly increasing N, one can easily provide countermeasures against the reverse-engineering attacks that have an exponential complexity. Reverse engineering of delay-based PUF has been previously studied [Gassend et al. 2004; Majzoobi et al. 2008b; Ozturk et al. 2008].

Let us briefly show the reverse-engineering attack on the delay based PUF shown in Figure 2 (ignoring the added FF arbiter). Figure 1(b) shows that each switch can be represented by four delays; $d_{i,j}$, $i, j = 0, 1$, where i/j denote the switch input/ output port indices respectively. However, it can be easily deduced that this model contains significant redundancy and the only important parameter in defining a switch's effect is the delay difference between its following top and bottom path segments. One can eliminate the redundancy and combine the series switches by lumping their delays to abstract thr representation of each switch using only one parameter shown as δs in Figure 1(b).

---

[1]http://www.cl.cam.ac.uk/~sd410/fpgasec/

Challenge	R
10**100111**	1
11**100111**	1
00**100111**	1
01**100111**	1
XX**100111**	1

Fig. 3. An example of emulation attack.

We refer to δ as the (differential) path segment delay. Thus, a linear PUF with N switches can be fully abstracted using $N + 1$ parameters.

Since the PUF structure is linear and the delays are additive, we can superimpose the effect of each path segment delay at the arbiter's input to find the total delay difference between the top and bottom paths. Note that each path segment's delay—depending on the odd or even number of switchings on its path—can appear on either the top path or the bottom path. The contribution of the segment to the total delay sum will be either positive or negative accordingly. Let c_i denote the i-th challenge bit in the challenge vector $\bar{c} \in \mathbb{B}^N$, $\mathbb{B} = \{0, 1\}$. The result of applying this challenge vector to the PUF in Figure 2 can be formally written as

$$\sum_{i=1}^{N}(-1)^{\rho_i^N}\delta_i + \delta_{N+1} \underset{r=1}{\overset{r=0}{\lessgtr}} 0, \tag{1}$$

where a transformation **T** defined as

$$\rho_i^j = \bigoplus_{x=i,i+1,\ldots,j} c_x = c_i \oplus c_{i+1} \oplus \ldots \oplus c_j \tag{2}$$

maps the challenges to ρs ($i \leq j$). The symbols $\bigoplus$ and $\oplus$ denote the parity generation and exclusive-or operations respectively. The inequality's direction is determined by the PUF response, r, for the given challenge vector. Each challenge-response pair (CRP) forms an inequality. By collecting enough CRPs, one can form and solve a system of linear inequalities to estimate the unknown δs. Note that other PUF structures, even the nonlinear ones, may also be susceptible to reverse engineering by using a polynomial order of measurements, and linear/nonlinear modeling.

4.1.2 *Emulation.* The goal of this attack is to emulate the PUF by effectively storing the CRPs in a memory. If the number of CRPs grows exponentially with respect to the number of inputs, the required memory would be very large, making the full CRP storage infeasible. Instead, the attackers attempt at exploiting the predictability of the CRPs (lack of randomness) to achieve high degrees of compressibility and reduce the storage demands drastically. For example, if a group of challenges that differ only in their first two bits produce the same response, there is no need that the first two bits are stored (Figure 3). A closely relevant attack would be to guess the responses to a given challenge with high probability by performing statistical analysis on the PUF responses. The data from the statistical analysis could also be used to efferently emulate the PUF.

4.1.3 *Man-in-the-Middle.* During the authentication process, CRPs stored in the database on a server are compared with those obtained from the PUF. In case there are a limited number of CRPs stored for each PUF in the database, the adversary can impersonate the PUF, if he can build a copy of the data base content. The man-in-the-middle attack involves eavesdropping the communication between the PUF and authentication server and recording the responses to the attempted challenges to later impersonate [Pfleeger and Pfleeger 1997].

4.1.4 *Reconfiguration.* While FPGA provides a versatile platform for implementing the PUF, the possibility of reconfiguring the FPGA by an unauthorized party poses a threat. For instance, if an adversary knows how to read the configuration bitstream and configure the FPGA, then he can gain full knowledge of the circuit structure. The attacker may reconfigure the FPGA to remove the nonlinearities or other added transformation circuitry at the input or output to facilitate reverse engineering by modeling the delays of the linear parts.

4.2 Countermeasures and Safeguards

The attacks presented in Section 4.1 can be alleviated by taking a number of safeguards. In Section 4.2.1 we review the countermeasures proposed earlier. In Section 4.2.2 we introduce new safeguards that are more comprehensive than the presently available approaches.

4.2.1 *State of the Art.* To protect PUFs against reverse engineering and emulation two lines of methods were mainly used: (i) introduction of nonlinearities, and (ii) challenge-response hashing [Lee et al. 2004; Gassend et al. 2002a]. The proposed nonlinearity-based methods are typically of two types: (a) feed forwarding and (b) MAX (MIN) operations.

A feedforward nonlinearity is introduced by inserting an internal arbiter that compare the signal delays at a certain point in the circuit [Gassend et al. 2002b]. The internal arbiter then forwards the arbitration results to a switch that is located ahead on the delay path. Figure 2 shows the feedforward arbiter in dashed lines. The feedforward arbiter introduces nonlinear behavior that complicates the reverse engineering process. However, our studies reveal that nonlinear PUF structures with a small number of feedforward arbiters are still prone to reverse engineering attacks [Majzoobi et al. 2008b]. To safeguard against this attack, multiple interleaved feedforward arbiters must be used. A major drawback of this protection method is that adding the nonlinearities skews the bit probabilities. Thus, the resulting nonlinear PUF is more vulnerable to statistical modeling and emulation attacks. The added internal arbiters also increase the circuit response's instability.

The use of MAX (MIN) functions was first proposed by Lee et al. [2004]. MAX (MIN) operations are carried out on delay values by using AND (OR) logic gates in the PUF structure. If two rising edge signals with delays d_1 and d_2 arrive at a two-input AND (OR) logic, then the delay of the output signal is $d_o = \text{MAX}(d_1, d_2)$. AND (OR) logics are inserted in the parallel PUF circuit and are connected to the bottom and top paths in between or at the end of the

Table I.   Latency and Area of Common Hash Functions

Algorithm	Chip area	Clock cycles
SHA-256	10,868	1,128
SHA1	8,120	1,274
MD5	8,400	612
MD4	7,350	456

structure. As we will show in Section 8.1, addition of this type of nonlinearity also renders the circuit more prone to emulation attacks.

As another countermeasure, the use of cryptographic hash function was proposed by Gassend et al. [2002a]. Hashing is performed on both challenges and responses of the PUF. To estimate the PUF model parameters, the adversary needs direct responses of the PUF arbiters for known challenges. However the use of a one-way *output hash function* inserted immediately after the arbiters will make the responses obfuscated and obscure. To discover the response, one needs to invert a one-way function which is known to be a hard problem. This process should also be repeated until sufficient number of responses are collected. An *input hash function* is attached to the PUF challenges to prevent bit-level control of the challenges. Due to the confusion and diffusion properties of hash functions, the final system is safe against emulation attacks. A drawback of the hash functions is that they incur significant hardware area and power overheads. Besides, the PUF needs to evaluate multiple clock cycles to prepare a standard size input message block to deliver to the hash function (MD5 can accept variable input message size). The input and output hash evaluations by themselves take many clock cycles, imposing a large overall latency on the system. Table I shows latency (in cycles) and area (in gate equivalents) of commonly used hash functions [Feldhofer and Rechberger 2006].

4.2.2 *New safeguarding methods.*   Figure 4 shows the global flow of the new PUF analysis and safeguarding approach. The steps are shown from left to right on the figure. We start by introducing a novel PUF architecture in Section 6. The objective for the new architecture is to be resilient against the attacks introduced in Section 4. The new architecture applies transformations to both challenges and responses by adding combinational logic to the input and output of the circuit. Next, we analyze the security of the proposed architecture against the outlined attacks. Lastly, countermeasures are added for safeguarding the architecture.

The objective of the new architecture is to be robust against three groups of vulnerabilities: (1) sensitivity to operational conditions, systematic, and temporal variations; (2) susceptibility to attacks, including reverse-engineering, statistical modeling, emulation, and man-in-the-middle attacks; and (3) reconfiguration-based attacks. Note that the robustness to changes in operational conditions and systematic/temporal variations ensures the stability of the resulting PUF responses.

To ensure robustness and resiliency, the proposed PUF structure utilizes a combination of the following five countermeasures: (i) introduction of techniques for determining the response error probability for each response bit; (ii) one-time CRP usage, providing a one-time pad security; (iii) using time
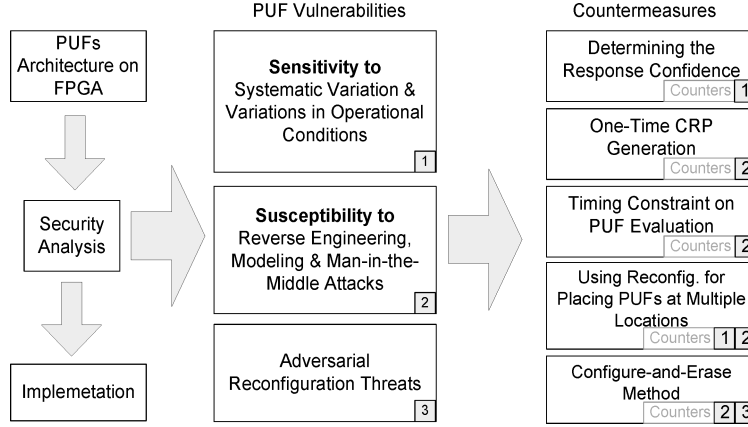
Fig. 4.   The overall flow of our approach. New safeguards are introduced to overcome the attack vulnerabilities.
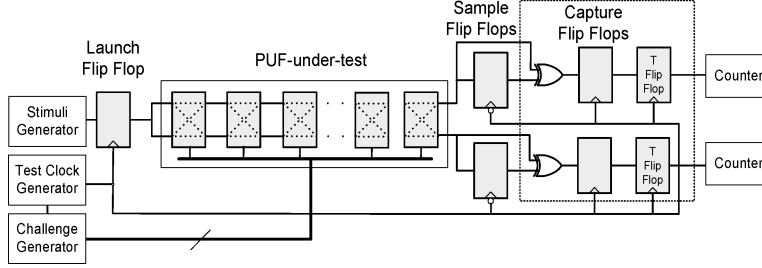

Fig. 5.   Delay characterization circuitry

constrained response evaluation; (iv) placement of PUFs at multiple locations on the FPGA; and (v) creation of the configure-and-erase method.

5. SECURE AND RELIABLE FPGA-BASED PUFS

In this section, we first present a mechanism to characterize the PUF components for a linear structure. There are at least three objectives for PUF characterization: (i) The measured delays and parameters can be used to achieve a higher robustness against variations in operational conditions and environment. This is accomplished by estimating the detection error probability for a given challenge. (ii) Switch delay values fully describe the PUF behavior and could be stored instead of challenges and responses. (iii) The delay values can be used to perform diagnosis, calibration and structural modifications for better performance. A similar test circuit to the one used in this article was suggested by Wong et al. [2007] as a BIST structure to estimate the delays of any combinatorial logic on FGPAs.

The suggested delay characterization circuit (Figure 5) consists of three flip-flops: *launch*, *sample*, and *capture*. A transition is invoked by the launch flip-flop at the combinatorial circuit under test (CUT) input. The output of the CUT is sampled t seconds later. The sampled value is compared to the real
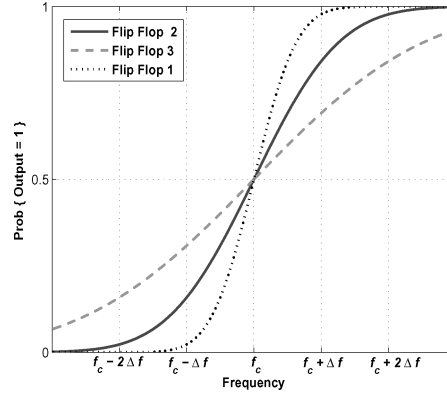
Fig. 6. Arbiter characteristics.

value by an XOR logic and the result is recorded by the capture flip-flop. If t is smaller (larger) than the CUT delay, then the sampling occurs before (after) the transition appears at the output, and thus the sampled value would be different from (same as) the input test signal. Note that the sample FF has certain setup and hold times which make the FF unable to sense smaller delay differences. Violation of setup and hold times places the FF into a metastable state and causes nondeterministic outputs. If t is swept by varying the clock frequency from f_l to f_u ($f_l < 1/t < f_u$) with steps of Δf, at some point the speed of CUT would be almost equal to the clocking speed. By counting the number of times the capture flip flop records a sampling error and then by forming a histogram, it is possible to accurately find the CUT delay.

Figure 6 depicts the probability that the sample FF outputs 1 versus the clock frequency. The symbol f_c marks the frequency at which the sample FF produces totally random outputs; $1/f_c$ is in fact equal to the CUT delay. In other words the clock edge and the signal edge coming from the CUT arrive at FF at the same time. The transition slope in Figure 6 implies the speed of the sample flip-flop. For example, flip-flop 1 has smaller setup/hold times than FFs 2 and 3.

The combinational test circuit used here is a PUF. The procedure is repeated N times for different challenge configurations. Then, a system of linear equations is solved to find each switch delay. To save time and efforts, the range of the scanned frequency can be adaptively adjusted in each iteration to scan a smaller window around the target frequency. Also instead of linearly sweeping the frequency to spot the transition point, a binary search algorithm can be used. If the frequency sweep range is partitioned into C_F steps, then the binary search would find the transition point in $log(C_F)$ steps [Corman et al. 2001]. Most advanced FPGAs, such as Xilinx Virtex family, provide Digital Clock Management (DCM) and Phase Locked Loops (PLL) blocks which enable building clock synthesizer with on-chip resources. This is useful if a standalone built-in system needs to be designed. In Section 9, we will implement the characterization circuit on Xilinx Virtex 5 FPGAs and present the measured results.

After solving Equation (3) to find the switch delays ($\Delta = \{\delta_1, \delta_2, ..., \delta_{N+1}\}$) and measuring the probabilistic characteristic of the sample flip-flops $g(.)$ (see Figure 6), the FPGA is reconfigured so that the test circuitry is removed. One of the sample flip-flops will be used as the PUF arbiter. Since the arbiter response only depends on the input delay differences rather than the absolute values, the flip-flop characteristic, $g(.)$ is transformed to represent the arbiter characteristic: $h(x) = g(\frac{f_c}{x \times f_c + 1})$. The estimated values for Δ and $h(.)$ completely characterize the PUF and are stored in a database to be later used for identification and authentication purposes.

5.1 Response Error Probability

Small delay differences at the arbiter inputs can cause metastability and inaccuracy of the response. Metastable arbiters are extremely sensitive to changes in operational conditions such as temperature variations and electromagnetic noise. Error correcting codes with syndrome decoding have been proposed to correct for such errors [Gassend et al. 2002a]. However, since the syndrome is public information, it can reveal some information about the responses and undermine the security of the PUF. In addition, correcting multiple bits requires a complicated decoding circuitry with large latency and hardware overheads.

To determine to what extent the responses are affected by metastability of arbiters, we propose a method that assigns a level of confidence to each response using the parameters obtained during the characterization step.

Let us represent the challenge vector by $\mathbf{C} = \{c_1, c_2, ..., c_N\}$. We define d as the delay difference (the top path delay minus the bottom path delay) in response to $\mathbf{C}$. Now, d can be written as

$$d = \sum_{i=1}^{N}(-1)^{\rho_i}\delta_i + \delta_{N+1} = [\mathbf{P} \ 1].\Delta^{-1}, \tag{3}$$

where $\mathbf{P} = \{(-1)^{\rho_1}, (-1)^{\rho_2}, ..., (-1)^{\rho_N}\}$ and $\rho_i = c_i \oplus c_{i+1} \oplus ... \oplus c_N$. The goal is to estimate the probability of false negative detection error, i.e., $Prob(H_1 \mid H_0)$ for a given $\mathbf{C}$ where the hypotheses H_1 and H_0 are defined in Equation (4):

$$H_0 : PUF = PUF' \tag{4}$$
$$H_1 : PUF \neq PUF'.$$

In fact, for a given delay difference d caused by the challenge $\mathbf{C}$, $h(d)$ (or $1 - h(d)$) is the probability that the arbiter produces a zero (or one) output while the delay difference at its inputs is greater (smaller) than zero, and $h(.)$ is the arbiter characteristic obtained by the test circuit explained in Section 5. We define the probability of false negative error (P_{error}) as the probability that at least one of the PUF responses to K challenges has an error; therefore:

$$P_{error} = Prob(H_1 \mid H_0) = \prod_{i=1}^{K}[\alpha_i(1 - h(d_i)) + (1 - \alpha_i)h(d_i)] \tag{5}$$

$$\alpha_i = \begin{cases} 1, & d_i \geq 0 \\ 0, & d_i < 0. \end{cases} \tag{6}$$

In writing Equation (5), we assume that the delays values caused by the K challenges are independent. Using this method, the delays resulting from a number of randomly selected challenges can be calculated by Equation (3) (assuming the switch delays are available from the characterization step). Then, the probability of false negative error is estimated using Equation (5). To ensure robustness against arbiter metastabilities, the responses with high estimated probability of error must be ignored.

5.2 Challenge-Response Compression

The characterization scheme allows an effective way to compress the challenge response pairs. A PUF with N switches in fact performs a transformation from N real numbers to 2^N binary numbers of length $N+1$. Therefore, by measuring the N parameters (that are in fact the path segment delay differences), one can fully describe the challenge-response space. In this way a huge reduction in database storage requirements can be achieved. Also it enables one-time pad encryption for large N values (e.g., $N > 128$). The idea of compressing the CRPs by collecting responses and performing reverse engineering is suggested in Ozturk et al. [2008]. However in this method, arbiter errors can cause large errors in estimating the switch delays [Majzoobi et al. 2008b]. We attempt to directly measuring the delays of the PUF before the arbiter. Our method also allows for arbiter characterization and stores the characteristic as part of the PUF parameters.

5.3 Diagnosis

The data obtained in the test phase can be used to diagnose and analyze the PUF. Small variations in delays, long setup/hold times for arbiters, large bias caused by systematic effects, or nonsymmetric routing may diminish the PUF's performance. Adding extra delay elements or switch to the PUF increases the total delay variation. Rerouting the connections and/or relocating the PUF can be utilized to overcome the delay bias. Also noisy flip-flops or those with large setup/hold times should be avoided being uses as arbiters.

6. SECURE PUF ARCHITECTURE

In this section, we introduce a secure and robust PUF structure. The proposed PUF as shown in Figure 7 consists of the four fundamental building blocks: (i) input (logic) network, (ii) output logic network, (iii) wire interconnect network, and (iv) parallel PUFs. After testing and characterizing each linear PUF in the parallel structure, the FPGA is reconfigured to integrate and attach these blocks to the core parallel PUFs.

6.1 Input Network

We design the input network attached to the parallel PUF (see the dashed box in Figure 7) to satisfy Strict Avalanche Criterion (SAC) for a parallel PUF circuit. A function is said to satisfy SAC if, whenever a single input bit is complemented, each of the output bits changes with a probability of one half. In Section 6.3 we will show how to bind multiple rows of the resulting structure to
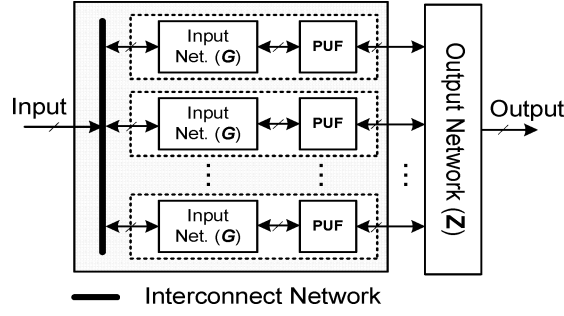
Fig. 7. The general architecture of the proposed Secure PUF.

construct an N input, Q output PUF structure that satisfies SAC. When introducing the output network, we demonstrate that the SAC property is required to achieve the maximum security. Before discussing the design steps of the input network, we first discuss the input/output characteristics of the parallel PUF.

As stated in Section 4, the PUF behavior can be represented by Equation (1). Let us assume the differential delay values (δ) in Equation (1) are independent and identically distributed. For simplification and without loss of generality, we assume the random variables have Gaussian distributions with zero mean, that is, $\delta_i \sim \mathcal{N}(0, \sigma^2)$. Our goal is to find the probability that the PUF output flips given that a challenge bit in the PUF input is flipped, that is, $Prob\{\sim O | \sim c_k\}$. Any change in the sign of the summation relates to a change in the output (response) bit value. Whenever a challenge bit value flips, some of the terms in Equation (1) change their sign (as a result of a change in the corresponding ρ values). Let us denote the set containing the indices of ρ's that flip (do not flip) as result of a flip in the k-th challenge bit by Γ_k (Λ_k). Note that Γ and Λ partition the index set, $\Omega = \{1, 2, ..., N\}$, where N is the total number of switches.

$$
\begin{aligned}
V_k &= \sum_{i \in \Gamma_k} (-1)^{\rho_i} \delta_i \\
W_k &= \sum_{j \in \Lambda_k} (-1)^{\rho_j} \delta_j + \delta_{N+1}.
\end{aligned}
\tag{7}
$$

If the absolute value of the sum of the terms whose indices are in Γ_k (i.e., $|V_k|$ in Equation (7)) is greater than the absolute value of the sum of terms whose indices are in Λ_k (i.e., $|W_k|$ in Equation (7)), then the response bit flips whenever c_k flips.

We define a new random variable X_k which has a value one if the output flips and zero otherwise, that is,

$$
Prob\{X_k = 1\} = Prob\{\sim O | \sim c_k\}
\tag{8}
$$

then,

$$
X_k = \begin{cases} 1, & |V_k| > |W_k| \\ 0, & Otherwise. \end{cases}
\tag{9}
$$

It is desired that $E\{X_k\} = 0.5$ for $k = 1, 2, ..., N$. The expectation is over all PUF realizations. Recalling that the sum of Gaussian random variables forms a new Gaussian, if $U_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$ and $U = \sum U_i$, then $U \sim \mathcal{N}(\sum \mu_i, \sum \sigma_i^2)$. Therefore, V and W can be viewed as realizations of Gaussian variables given by

$$
\begin{aligned}
W_k &\sim \mathcal{N}(0, (|\Lambda_k|+1) \times \sigma^2) \\
V_k &\sim \mathcal{N}(0, |\Gamma_k| \times \sigma^2).
\end{aligned}
\tag{10}
$$

where $|.|$ denotes the set cardinality. If $|\Gamma_k| = |\Lambda_k| + 1$, then V and W will be identical and independent Gaussian random variables. Also $|\Gamma_k| + |\Lambda_k| = N$. Therefore, if

$$
|\Gamma_k| = \frac{N+1}{2},
\tag{11}
$$

then $E\{X_k\} = 0.5$. Thus, if $\frac{N+1}{2}$ (almost half) of ρ's in Equation (1) flip as a result of a flip in k-th challenge bit (c_k), then the output of the PUF would flip with a probability 0.5. The result is in accordance with our initial intuitive observation.

We now verify if this property holds in the parallel PUF structure. The ρ's in Equation (1) are related to the challenges by the transformation T defined in Equation (2); that is, $\mathbf{P} = T(\mathbf{C})$. It can be seen that a flip in c_k causes a flip in ρ_j, where $j \leq k$. Thus, $|\Gamma_k| = k$. For example, if a flip in c_N happens, all of the ρ's flip as a result. Hence, Equation (11) is not satisfied for the parallel PUF structure. We define a transformation $G(.)$ on challenges that combined with T meets the criterion set by Equation (11).

Objective: Find $G(.)$ so that $\mathbf{P} = T(G(\mathbf{C}))$ satisfies $|\Gamma_k| = \frac{N+1}{2}$ for all k.

Method: (1) Before finding the proper transformation, we make an observation. Consider two challenge bits, c_k and $c_{k+\frac{N+1}{2}}$ in the parallel PUF that flip in succession. The first flip (c_k) causes ρ_j for all $j \leq k$ to flip, and the second challenge flip ($c_{k+\frac{N+1}{2}}$) causes ρ_j for all $j \leq k + \frac{N+1}{2}$ to flip. The ρ's that flip twice return to the first value and do not flip in effect. Therefore, flipping c_k and $c_{k+\frac{N+1}{2}}$ at the same time causes ρ_j for all $k < j \leq k + \frac{N+1}{2}$ to flip, hence satisfies the criterion set by Equation (11). The observation implies a constraint on the challenges. Whenever a challenge bit flips, another challenge bit at $\frac{N+1}{2}$ selectors apart must flip as well to guarantee SAC. Note that N must be an odd integer to yield integer challenge indices. However, it is easy to prove that it is infeasible to impose the derived constraint on the PUF challenges, although high quality approximations can be made. We design an input network that transforms the input challenges of the PUF and imposes constraints on the toPUF challenges.

(2) Find a transformation $\mathbf{C} = G(\mathbf{D})$, $G : \{0, 1\}^M \rightarrow \{0, 1\}^N$, so that any bit flip in the input causes two output bits at *approximately* $\frac{N+1}{2}$ locations apart to flip. We use two approximations to realize the proposed concept. One approximation can be implemented using a wire-only network, while the other one can be done using XOR logic (Figure 8).
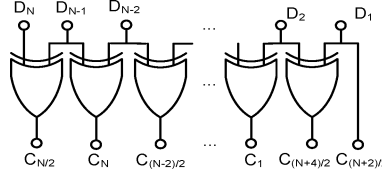
Fig. 8.   The input network realization using XOR logic.

(i) Wire-only network: for N an even integer and $M = \frac{N}{2}$, G performs the following transformation:

$$G : \quad c_i = c_{i+\frac{N}{2}} = d_i, \quad for \ i = 1, 2, ..., \frac{N}{2}. \tag{12}$$

The wire network connects the challenge bits of the PUF that are located $\frac{N}{2}$ apart. However, the input dimension of the wire network is half of the PUF. Hence, one needs to use twice the number of switches to achieve the same number of PUF inputs.

(ii) XOR network: for an even integer N and $M = N$, we define the transformation G as follows:

$$
\begin{aligned}
c_{\frac{N+i+1}{2}} &= d_i, \quad for \ i = 1 \\
c_{\frac{i+1}{2}} &= d_i \oplus d_{i+1}, \quad for \ i = 1, 3, 5, ..., N-1 \\
c_{\frac{N+i+2}{2}} &= d_i \oplus d_{i+1}, \quad for \ i = 2, 4, 6, ..., N-2
\end{aligned}
\tag{13}
$$

Unlike the wire-only network, the XOR network achieves a one-to-one mapping. However, an adversary with full knowledge of the circuit structure can apply the inverse transformation to make the input network ineffective. We alleviate this issue later by introducing a wire interconnecting scheme that physically binds the inputs of multiple PUF rows.

In addition to the expectation of X_k being equal to 0.5, it is desired that the X_k has as small variance as possible. Smaller variation guarantees that a larger number of PUFs satisfy the SAC property. The variance of X_k is related to the variances of W and V in Equation (9), that are themselves related to the number of switches and the variance of δ. The variance of δ is determined by the technology and the amount of process variations. Therefore, one can achieve a smaller variance for X_k by adding to the number of switches or incorporating multiple rows of the same structure as explained in Section 6.2.

6.2 Output Network

We introduce an XOR-based output network structure (see Figure 7) which achieves (i) fortification against reverse engineering attacks, and (ii) smaller deviation from the SAC on each PUF by combining multiple rows of parallel PUFs with the transformed challenges.

The output network performs a mapping denoted by $Z(.)$ from the PUF arbiter responses, $\mathbf{R}$, to the output, $\mathbf{O}$. The mapping is defined as $\mathbf{O} = Z(\mathbf{R})$, $Z : \{0, 1\}^Q \rightarrow \{0, 1\}^{Q'}$, $Q' < Q$, and

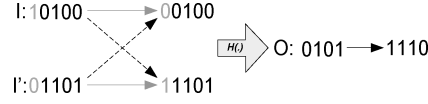$$o_j = \bigoplus_{i=1,...,x} r_{(j+s+i) \ mod \ Q} \quad for \ j = 1, 2, ..., Q' \tag{14}$$

Fig. 9. An example of output network for $Q = 5$, $Q' = 4$, $x = 4$, and $s = 1$.

where $\oplus$ denotes the parity generator function and s indicates shifting step. The transformation calculates the parity value for sets of x adjacent PUF arbiter responses where each set starting point is circularly shifted by s bits with respect to each other. The transformation can be parameterized by s (the shifting step) and x (the parity input size). We will discuss later how these parameters govern a trade-off among security, overhead, and randomness properties.

(i) The proposed transformation can hinder the efforts to reverse engineer the PUF in the following way. As stated in Section 4, to reverse engineer a linear PUF structure and estimate the delays of switches, the adversary needs to collect a set of challenge-responses from the PUF and solve a system of inequalities.

Suppose that the responses of Q parallel PUFs are mapped to a Q'-bit output by the transformation $Z(.)$. There are $2^{Q-Q'}$ possible inputs that map to a given output. Therefore, the adversary is faced with solving an ambiguity to discover the real PUF response. The number of assumptions grows exponentially, if he/she is not able to reject some of them at each step. It can be shown that the problem has exponential complexity of order $O(2^{(Q-Q')^{N_C}})$ with respect to the number of ambiguities $2^{(Q-Q')}$ and the number of CRPs needed to estimates the PUF switch delays N_C. Majzoobi et al. [2008b] show that to reverse engineer a linear parallel PUF having 64 switches with accuracy of over 98%, a minimum number of 2000 CRPs ($N_C = 2000$) are required. Then for $Q - Q' = 1$, the complexity of reverse engineering the secure PUF would be $O(2^{2000})$. Nevertheless, if the attacker can control (the transition of) the PUF arbiter responses, then it would be possible to reduce the number of assumptions by performing a differential attack. Let us illustrate the problem using an example. Consider a trivial case of Z where $Q = 5$, $Q' = 4$, and every four adjacent response bits are XOR-ed to produce the output, i.e., $s = 1$, $x = 4$ (see Figure 9).

Now imagine a transition in the output occurs from 0101 to 1110. As shown in Figure 10, there are four possible transition hypotheses about the Z inputs depicted by arrows. If we know only the first bit (or only one bit) of the input has caused such transition in the output then we can reject two hypotheses shown in dashed arrows. Also by associating probabilities with transitions and ranking assumptions accordingly, one can guess the PUF responses.

Thus, $Z(.)$ by itself does not guarantee significant resiliency against reverse engineering. To achieve a high level of resiliency, it is required that the PUF response bits (or Z inputs) could not be deterministically controllable. The Maximum resiliency is obtained if PUF response bits flip with a probability of 0.5 which is equivalent to SAC. We will use an interconnect network that connects rows of parallel PUFs, to design a Q-output PUF with SAC property.
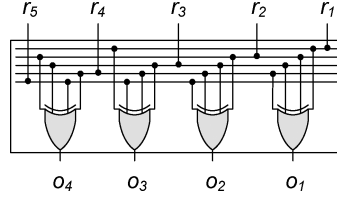
Fig. 10.   Example: the difficulty of inverting the output network.

(ii) The mixing property of XOR logic or in general the parity generator function also fortifies the PUF against emulation and statistical guessing attacks even in presence of outrageously large PUF switch (element) delays. For larger values of x in Equation (14), higher number of PUF rows and responses are mixed and smaller deviation from the transition probability of 0.5 (i.e., $\text{var}(X_k)$) is achieved.

6.3 Interconnect Network

In Section 6.1 we designed an input network that satisfied SAC for a single row PUF with one bit output. We design a PUF structure that consists of multiple rows of parallel PUFs and maps N challenge bits to Q response bits. The PUF is designed to satisfy SAC. The PUF is built upon an interconnect network that connects the challenge bits of rows of parallel PUFs (See the leftmost solid box in Figure 7). In order to satisfy the SAC, it is required and sufficient that one challenge bit on each row is connected to another challenge bit on a different row. A challenge bit is broadcasted to all PUFs, and since each PUF output flips with a probability of 0.5, the SAC is met. The interconnection can be expressed formally as follows:

$$c_i^m = c_j^{m+1} \quad for \ \ i, j \in \Omega, \ \ m = 1, 2, ..., Q - 1, \tag{15}$$

where c_i^m is the $i - th$ challenge bit in the m-th row, $\Omega = \{1, 2, ..., N\}$, $j = g_m(i)$ and $g_m: \Omega \to \Omega$ is a one-to-one permutation function. In Section 6.1, we mentioned that the XOR input network can be bypassed by applying the inverse transformation. If the inputs of PUF rows are connected in parallel (with no permutation), that is, $i = j$, by applying the inverse transformation (G^{-1}) all of the input networks are bypassed and thus, ineffective. By imposing a constraint on g_m to be nonidentity for all m's the attacker can *fully* bypass only one input network and the other input networks can only be partially bypassed. Figure 11 depicts an m-bit circular shift interconnecting scheme, that is, $j = g_m(i) = (i + m - 1) \bmod Q$.

7. APPLICATIONS

There are many possible applications for the introduced secure PUF, as pointed out in previous research [Gassend et al. 2002a; 2002b; Trimberger 2007; Koushanfar and Potkonjak 2007; Alkabani et al. 2007]. In this section, we present applications that leverage reconfigurable platforms to provide system security.
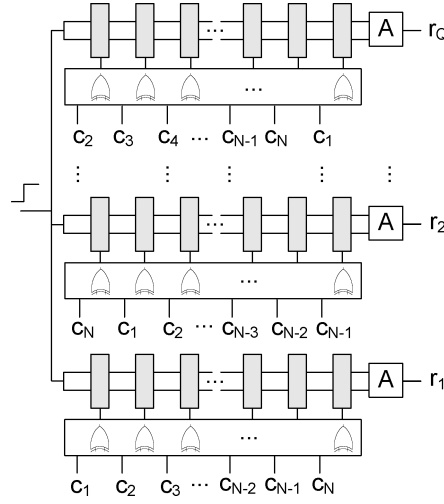
Fig. 11. An m-bit circular shift interconnecting scheme that connects Q rows of parallel PUFs with transformed challenges.

Path Segment Delays (Δ)	Arbiter Characteristic $h(.), ST, HT,...$	Location Parameters LUT(X,Y)	Configuration Bitstream	...

Fig. 12. An example of a database entry.

7.1 Configure-and-Erase Method

We introduce an FPGA-based authentication method for smart cards which limits the user's knowledge about the PUF circuit structure and its location on FPGA. In this method, the FPGA owner is identified by the unique manufacturer variability (MV) on his/her FPGA. The permanent placement of the PUF circuit, as in ASIC technology, would give the adversary unlimited access to the PUF inputs/output and thus make the PUF vulnerable to reverse engineering and emulation attacks. In the proposed method, the user is furnished with a blank FPGA and a Personal Identification Number (PIN). Before the FPGA is given to the user, the PUF is characterized using the methods described in Section 5. Thus, the switch delays and arbiter parameters are derived and stored. PUFs with different lengths and on various locations can be implemented and characterized. Therefore, each database entry could consist of multiple fields such as location attributes, circuit structure parameters (length), switch delay values, and arbiter parameters (see Figure 12). Because of the linear number of components, the space required to store all this data is still smaller than saving many challnge/response pairs.

When the user presents the FPGA and his PIN to the authentication system the following steps are performed: (i) The system retrieves the database entry associated with the provided PIN. (ii) Then, the FPGA is configured according to the database entry field values. The PUF will be placed on the
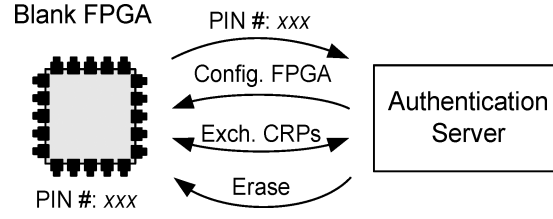
Fig. 13.   Configure-and-erase scheme steps.

location specified in the database and the input/output networks will be added to it. The configuration bitstream could be stored in each database entry along with other parameters or instead it could be generated online according to the circuit parameters and location attributes stored in the database. The latter method can drastically reduce the storage requirements, although it introduces a latency in generating the bitstream. Note that the number of feasible locations for placing the PUF is merely dependent on the size of the FPGA. (iii) At the third step, the binary challenges are sent and tried on the configured PUF; the responses are retrieved. Meanwhile, the database derives responses from the stored PUF parameters for the given challenge. Probability of error for each response is also calculated. The derived responses with lower error probabilities are compared with the received responses for authentication. (iv) After authentication is performed, the FPGA content is erased and the FPGA is returned to the user. The steps are shown in Figure 13.

The proposed system can be subject to the a number of attacks. First, by tapping the communication link between the FPGA and the authentication server and reading the configuration bitstream, the adversary might be able to discover the PUF structure and its location on the FPGA and later attempt to configure the FGPA and model the PUF. There are two ways to eliminate such threats. One way is to physically secure communication link which is not feasible in all applications. The other way is to encrypt the bitstream and make the FPGA non-reconfigurable by an unauthorized party. A timed authentication method, which is explained next, can be also used to hinder the efforts to perform PUF modeling attacks. The use of one-time PUFs (where each CRP is tried only once) would protect the PUF against man-in-the-middle attacks.

7.2 Timed Authentication

Swift evaluation of the arbiter-based PUF is a unique feature that can be used to safeguard the authentication process. Unlike ring oscillator PUFs, the arbiter-based PUFs can produce a response to a given challenge in a single clock cycle. The clock frequency is however limited by the delay of PUF structure itself. Attempts to build a software counterfeit of the PUF by either emulating the responses or reverse engineering can be encumbered by imposing a tight timing constraint on PUF evaluation. The method could be realized by time stamping the responses using either the embedded system clock or the authenticating server clock. The former requires a tamper resistant clock since otherwise an adversary with high-speed clocking resources at his
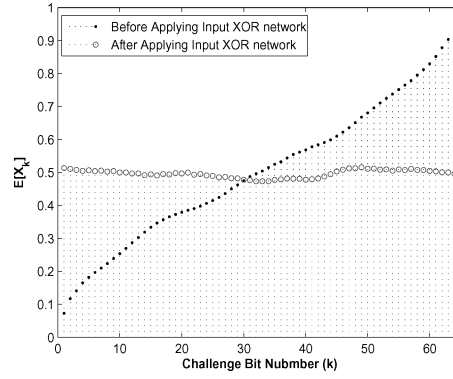
Fig. 14. Probability of the response flip given a flip in the k-th challenge bit for a single row parallel PUF. The dot (circular) markers show the probability before (after) transforming the challenges.

disposal might be able to reduce the system clock frequency and calculate the PUF response more quickly. Using the authentication server clock removes concern about clock tampering, however it would be limited to applications where the PUF and the server can communicate through a high-speed channel whose latency is not much higher than the PUF evaluation time.

8. SIMULATION RESULTS

In this section, we demonstrate the efficacy of the proposed concepts and methods. We thoroughly examine the secure PUF architecture that uses the proposed input/output network and the interconnecting method explained in Section 6. In the next section, we present delay measurement results obtained from the test platform.

8.1 Secure PUF Architecture

In this section we thoroughly examine the performance attributes of the PUF structure presented in Section 6. In the following experiments, we model each switch with four delays—two for straight connection and two for cross connection links. We assume that the delay components are samples from independent identical Gaussian distributions with $\mu = 0.5ns$ and $\sigma = 4ps$. The mean and variance conform with the 65nm technology [Sedcole and Cheung 2006].

For a single row parallel PUF circuit with 64 switches, we simulated the probability of output transition conditioned on each challenge bit transitions. In this experiment, we apply to the PUF 100 random 64-bit challenge vector pairs that differ only in the $i - th$ bit, where i = 1,...,64, and record the percentage of times the output transitions. We repeat this experiment for 50 PUF circuit realizations and find the expectation. Figure 14 shows the value of $E[X_k]$ before and after applying the input XOR transformation (defined in Equation (13)) on the PUF challenges. The figure shows that the probability of output flip conditioned on the k-th challenge bit before input transformation increases monotonically from less than 0.1 to over 0.9, where $k = 1,2,...,64$.
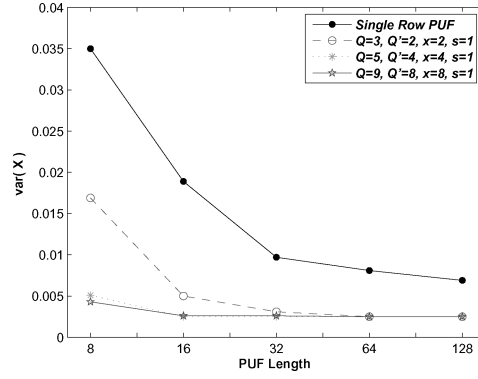
Fig. 15. Deviation of transitional probabilities of individual PUF instances from the SAC.

This can be intuitively viewed as the cumulative effect of switch delays in the parallel PUF circuit structure. Note that after applying the XOR transformation on the PUF challenges, the output flips with a probability close to 0.5 for a flip in input bits, which per se satisfies the SAC. A smaller deviation from the transition probability of 0.5 is desired for each individual PUF circuit realization. There are two ways to reduce such deviation: (i) by using more switches in the parallel PUF circuit (increasing N); (ii) by mixing the outputs of larger number of parallel PUF circuits (increasing x in Equation (14)). The black solid line in Figure 15 indicates how the variance ($var(X_k)$) decreases as the number of switches in a single row parallel PUF increases from 8 to 128. For a fixed number of switches in a row, the variance rapidly drops as 2, 4, and 8 adjacent outputs of rows of parallel PUFs ($x = 2,4,8$) are mixed (by an x-input parity generator function). Note that the challenges of the parallel PUFs arranged in rows are connected by the interconnection network presented earlier.

We now investigate the security of the proposed PUF against emulation attacks. We devised an algorithm that randomly selects a challenge vector within which it searches for the largest number of bits that can be represented with don't-cares. We set an upper bound on the number of search efforts for each challenge and use the knowledge of the statistical PUF characteristics to expedite the search. For example, we know that the left most challenges have a lower impact in determining the parallel PUF output (same scenario for FF PUF), thus they can be represented with don't-cares with higher probability.

Figure 16 shows the amount of compression achieved for a single row parallel PUF, FF PUF, and Secure PUF; $(Q,Q',x,s) = (9,8,8,1)$ and 64 switches in each row ($N = 64$). The feed forward arbiter in the FF PUF compares the delays at switch $20th$ and feeds the result to the selector of the 40th switch. The emulation attack is performed on 10 PUFs of each type. The ten values are shown in box plots in Figure 16. Note that compressibility of the challenge response pairs of the secure PUF structure is four and five orders of magnitude smaller than a single row parallel and FF PUF respectively. The smaller level of compressibility corresponds to lower predictability of the responses.
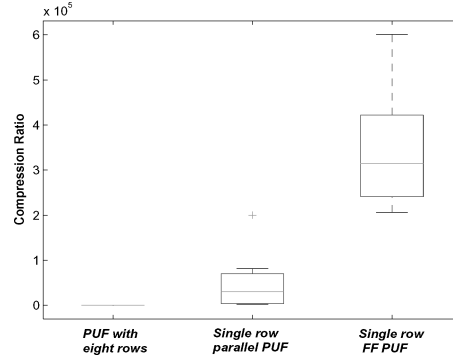
Fig. 16. The amount of CRP compressibility for parallel, FF and the proposed secure PUFs. Each box represents the results of 10 experiments on different PUF realizations.
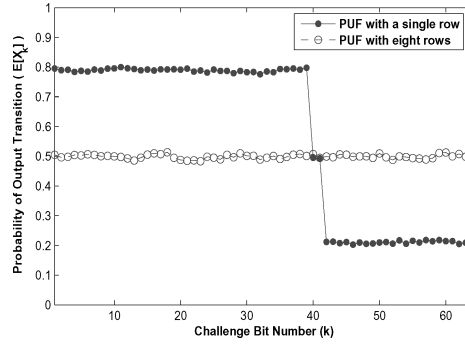

Fig. 17. Sensitivity of PUF transitional behavior to outlier switch delays.

We also examined PUF sensitivity to very large switch delays. In general, delay outliers cause high predictability, high compressibility of CRPs, and facilitate building of statistical models. We studied the sensitivity of secure PUF and the single row parallel PUF structures to outliers. A fault is injected as an outlying delay of $5ns$ (10 times larger than mean delay) into the $20th$ switch—of the first row for the secure PUF. Figure 17 shows the expected probability of output transition for both single row parallel PUF and the Secure PUF with the parameters (Q,Q',x,s) = (9,8,8,1). The expectation is taken over 50 PUF realizations. For the parallel PUF with one row, the transition probability is highly distorted; flipping inputs 1 (42) to 39 (64) (does not) flips the output with a probability of 0.8. Such divergence from SAC leads to high predictability of PUF responses and facilitates emulation and statistical modeling attacks. However, as it can be seen in Figure 17, the transition probability of output (any of the eight PUF output bits) does not change because of the mixing introduced by the output network. In addition, if the PUF responses in some of the rows do not show significant changes and variations due to arbiter failure, arbiter insensitivity, or large delay biases, the effect would not be transparent at the output.
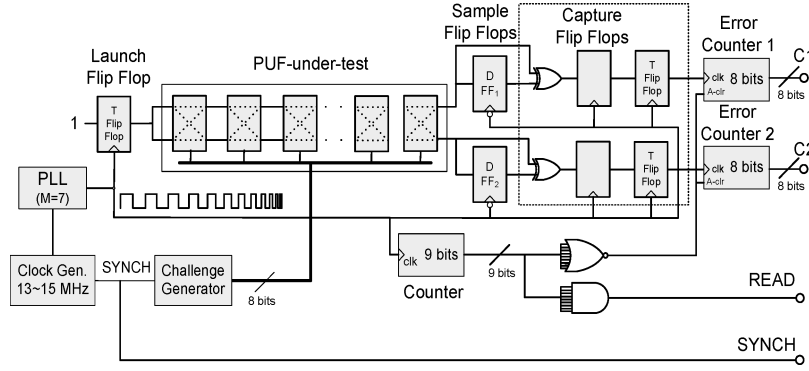
Fig. 18.   The detailed structure of the delay characterization test circuit. The clock frequency is consciously swept using an external clock generator and the frequency range is shifted up by the internal FPGA PLL.

9. IMPLEMENTATION RESULTS

We present the measurement and characterization results obtained by implementing the test circuit described in Section 5 on Xilinx Virtex 5 FPGAs. Figure 18 shows the details of the implemented circuit structure. The circuit benefits an external clocking source which sweeps the clock frequency continuously from 13 to 15 MHz. The frequency is swept every 65 milliseconds. The clock generator's output is then connected to a PLL inside the FPGA which multiplies the input frequency by 7, shifting the frequencies up to the 91–105 MHz interval. The PUF under test is triggered by a toggle flip flop which alternately produces a falling and rising edge signal. A 9-bit counter driven by the system clock resets the error counter values and issues a read signal every 512 clock cycles. The errors are counted by 8-bit counters.

The linear PUF structure was used for testing and characterization. The PUF consists of 8 switches with 6 delay elements in between. The delay elements are implemented by a series of 6 NOT gates. Each NOT gate is realized by a separate LUT. To balance the path delays, the PUF was manually placed and routed using the FPGA Editor in Xilinx ISE design tool. Figure 19 shows the PUF after placement on the FPGA floor plan. Eight different challenge values with corresponding decimal values of 0, 1, 3, 7, 15, 31, 63, 127 are tried at each sweep. With the delays measured by applying the challenges, one can find each path segment delay (δ) by solving a system of linear equations. The error counter values are read using a Tektronix LA714 logic analyzer when the READ signal goes high. The circuit was implemented on five XC5VLX50 and three XC5VLX110 Xilinx Vertex 5 chips.

Figure 20 shows the relationship between the clock period and the PUF output sampling failure rate. At clock periods above 11.45ns (i.e., Region 1), the PUF output is sampled and captured successfully, making Region 1 a fault-free region. As the clock period decreases, sampling errors begin to appear (Region 2). The failure rates reach a plateau of 0.5 (Region 3). In this region, the sample flip-flop always fails to properly sample the rising edge transitions
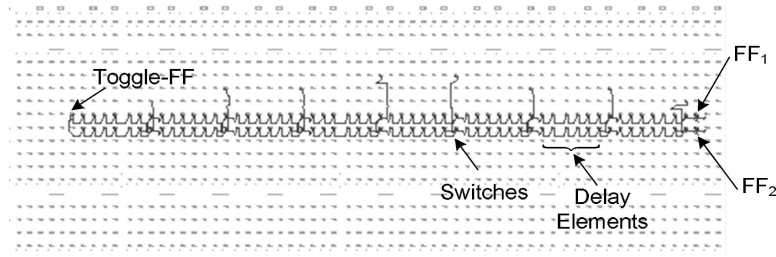
Fig. 19. The PUF circuit on the FPGA floor plan after manual placement. A screen shot from FPGA editor tool in the Xilinx ISE software.
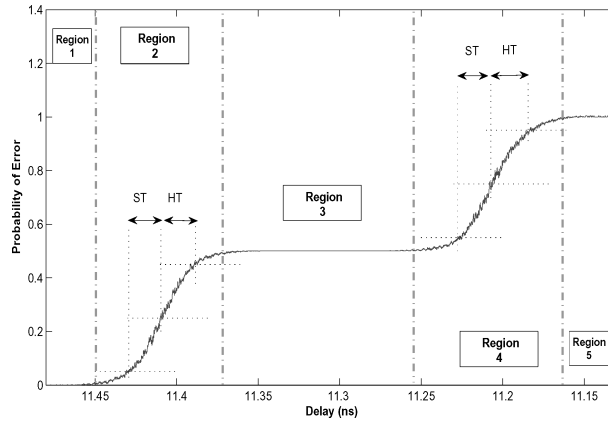

Fig. 20. Measured arbiter characteristic.

but it can capture the falling edge transitions successfully. This is due to the fact that the delays for positive and negative transitions through the PUF are different. Since half of the transitions are positive and the other half are negative, the failure rate would be 50%. If the clock period is further decreased, errors would appear for the falling edge signals too (Region 4). Finally in Region 5, all of the sample values would be erroneous. The curves in Regions 2 and 4 are in fact the flip flop characteristics. The flip flop setup and hold times are indicated by markers in Figure 20. The 10% and 90% values for the ST and HT times are chosen respectively. The points where the failure rates are equal to 0.25 and 0.75 virtually correspond to the cases where the clock period is equal to the circuit delay for rising and falling transitions respectively. Also note the fineness of delay resolutions at which the measurements are carried out.

We begin with characterizing the arbiter. To efficiently represent the arbiter, a parametric model could be fit to the arbiter characteristic and the pertinent parameters are estimated and stored in the database. The arbiter's non-deterministic behavior in presence of small input delay differences is influenced by circuit noise and many other surrounding effects. Due to central limit theorem we argue that the arbiter output can be represented by Gaussian
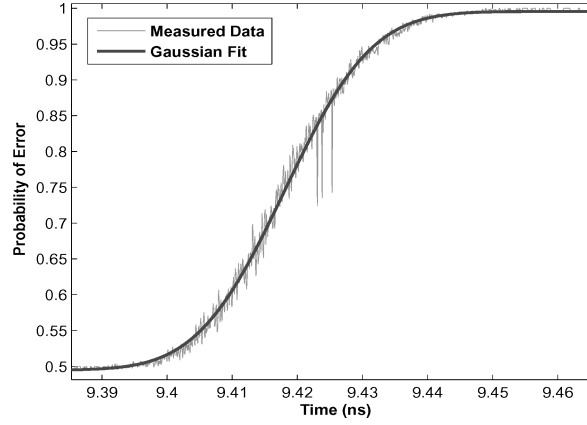
Fig. 21.   Measured arbiter characteristic in region 4 and the corresponding Gaussian fit.

cumulative function (CDF). We fit the Gaussian CDF to the measured arbiter characteristic in least square sense and estimate its mean and standard deviation. Figure 21 shows a section of the measured arbiter characteristic along with the Gaussian fit. The standard deviation (σ) of the fit determines the speed of the arbiter and the mean value corresponds to the PUF delay. Note that the σ (arbiter speed) can be different for rising edge and falling edge signals. Therefore, the arbiter can be effectively represented by two parameters, that is, σ_{rise} and σ_{fall}. Setup time and hold times are functions of σ.

We used the characterization circuit to measure the top and bottom path delays of the PUF. The delays are measured for eight different challenge values. Figures 22(a) and (b) respectively show the path delays for falling edge and rising edge signals propagating through the PUFs implemented on XC5VLX50 chips. Figures 22(c) and (d) show the same data but for XC5VLX110 chips. Each plot in these figures correspond to one chip. There are two sets of data on each plot distinguished by circle and dot markers. The circle marker represents the top path delay while the dot marker refers to the bottom path delay. The top and bottom path delays are shown for the given challenge values on the x-axis. As the measurements suggest, the path delays are in some case correlated among the chips (e.g., see how the delays on the second and third plots in Figure 22(d) follow the same trend).

We next estimated the σ_{rise} and σ_{fall} of the flip-flops from the measurement data. With two sample flip-flops per circuit and a total of 5 circuits on XC5VLX50 chips and 3 circuits on XC5VLX110 chips, there were 16 flip-flops to characterize. The measurements were repeated 8 times for each flip-flop. The box plots in Figures 23(a) and (b) show the estimated σ_{fall} and σ_{rise} respectively for the flip-flops on XC5VLX50. Figures 23(c) and (d) shows the same result for the flip-flops on three XC5VLX110 chips. The adjacent flip-flops in the figures (i.e., FF1 and FF2) come from the same circuit and are physically close to each other in the circuit. FF1 and FF2 reflect similar speed characteristics. The flip-flops on the third XC5VLX50 chip is the fastest among the other XC5VLX50 chips. Also the flip-flop on the first XC5VLX110 chip is the fastest.
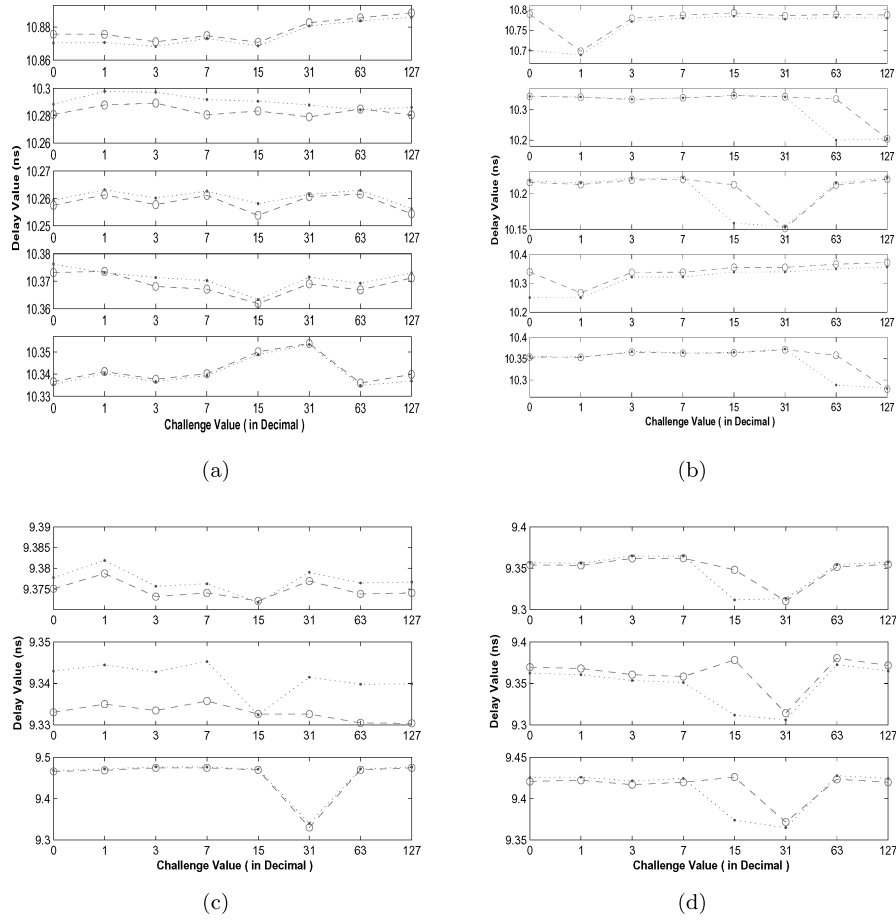
Fig. 22. The top and bottom path delays of PUFs on XC5VLX50 chips for (a) rising edge transition, and (b) falling edge transition; The top and bottom path delays of PUFs on XC5VLX110 chips for (c) rising edge transition, and (d) falling edge transition.

The narrowness of the boxes confirms the accuracy of the measurements and parameter estimations.

The measurements can also help locate faulty or unstable flip-flops. For example, the test results on one of the the flip-flops as shown in Figure 24 demonstrates a noisy glitch marked by the black circle. The glitch keeps repeating at every round of measurement. Thus, when implementing PUFs, the designer should avoid noisy arbiters and those with large setup and hold times. Even if the noisy flip-flop (like the one shown in Figure 24) has to used as an arbiter, then those challenges that cause a delay difference coinciding with the glitch must be avoided.

The amount of delay variability in future FPGA technology follows an upward trend. Sedcole and Cheung [2006] predict that with a slightly rapid increase in stochastic variability (2% every three years), stochastic variation
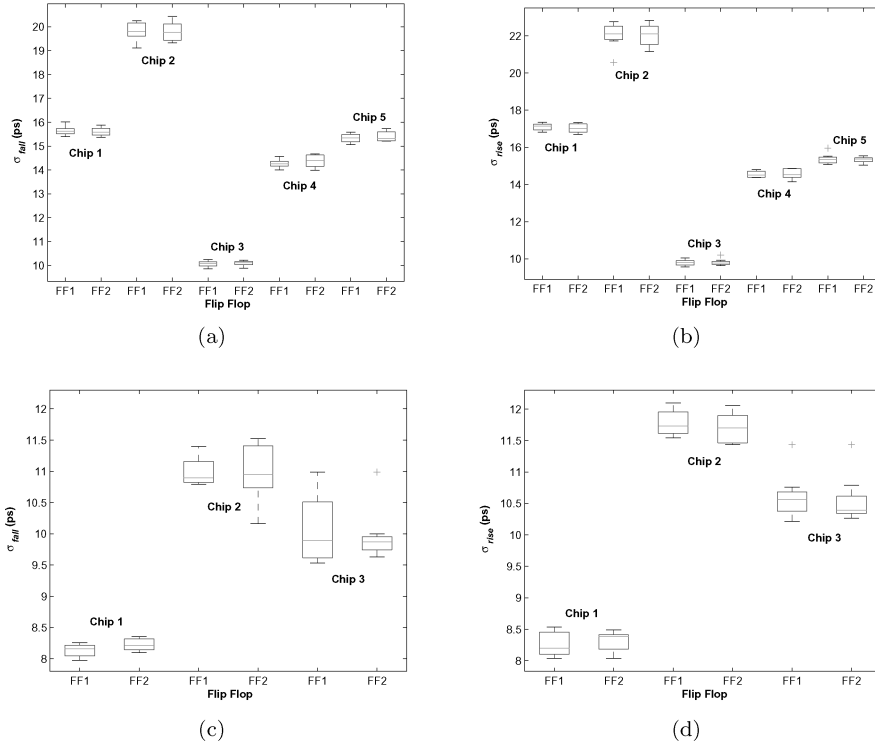
Fig. 23.   The sample flip-flop speed σ of PUF circuits on XC5VLX50 chips for (a) rising edge transition (σ_{rise}), and (b) falling edge transition (σ_{fall}); The sample flip-flop speed σ of PUF circuits on XC5VLX110 chips for (c) rising edge transition (σ_{rise}), and (d) falling edge transition (σ_{fall}).
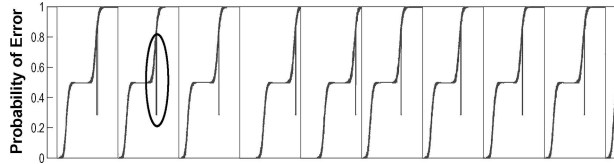

Fig. 24.   Faulty flip-flop behavior.

will amount to 11.5% in 22nm technology node while systematic variations keeps decreasing. Table II shows the amount of variability in the current and pending FPGA technologies [Sedcole and Cheung 2006].

We estimated the amount of observed variability on 65nm Virtex 5 FPGA family from the measurement data. Assuming the additive delay model for the PUF, the total path delay (Z) can be written as sum of independent and identically distributed random variables (X_i), that is, $Z = X_1 + X_2 + \ldots + X_n$ where n is the total number of delay elements in the PUF structure, $X_i \sim N(\mu, \sigma)$ and $Z \sim N(\mu_Z, \sigma_Z)$. Then

$$\sigma_Z / \mu_Z = (1/\sqrt{n}) \times (\sigma/\mu). \tag{16}$$

Table II. Delay Variability in Current and Future FPGA Technologies

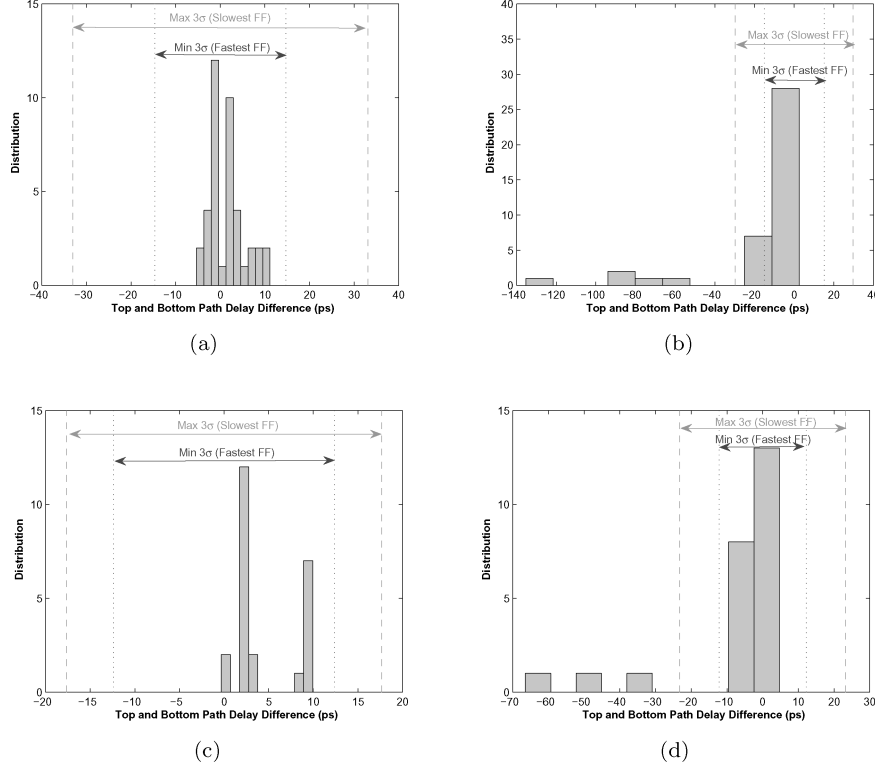| Production year	2004	2007	2010	2016
Node	90nm	65nm	45nm	22nm
Stochastic var. (3σ)	3.3%	5.5%	7.5%	11.5%

(a)

(b)

(c)

(d)

Fig. 25. The distribution of delay difference between the top and bottom paths. The distribution is calculated for 8 input challenges, 5 XC5VLX50 chips, and 3 XC5VLX110 chips. Figures (a) and (b) show the distribution for rising and falling transition delays on XC5VLX50 chips. Similarly, Figures (c) and (d) show the distribution for rising and falling transition delays on XC5VLX110 chips. The vertical lines show the 3σ detection edges of slow and fast arbiters (FF).

The mean and variance of Z were estimated as $\mu_Z = 10.41$ns and $\sigma_Z = 0.068$ns using the delay measurements obtained from XC5VLX50 chips. Since the PUF uses 8 switches and 6 delay elements in between then the total length of the PUF would be n = 56. Therefore from Equation (16), the variability is derived as $(3 \times \sigma)/\mu \simeq 5\%$. The amount of variability conforms with the estimated values shown on Table II. The measurement results show that even though variability exists in the structure, the amount of variability in presence of the arbiter imperfections and measurement biases is not enough to make a robust and yet random response on all of our test chips. In what follows, we present a number of interesting observations from our implementation experiments.

Figure 25 shows the distribution of the delay differences between the top and bottom paths for XC5VLX50 test chips. Figures 25(a) and (b) show the

distribution for rising and falling edge transition delays. Each distribution contains 40 points; 5 chips and 8 challenges per chip. The flip-flop 3σ setup/hold-times are depicted on the same figure by vertical lines for the fastest and slowest flip-flops on the chips.

It can be seen that in Figure 25(a), on our test chips, the delay differences are smaller than 3σ setup/hold times of the arbiter, resulting in metastable responses. In the falling edge transition case shown in Figure 25(b), 87.5% and 70% of the points fall inside the metastable region of the slow and fast flip-flops respectively. Figures 25(c) and (d) show the same results for XC5VLX110 chips. For the falling and rising edge transition cases (Figure 25(d) and (c)) 87.5% and 100% of the points fall inside the metastable region of both slow and fast flip-flops. As it can be seen, because of the lack of calibration, the distributions are skewed toward positive and negative values.

Also, it is important to note that a delay difference between the top and bottom paths after the last switch and before the arbiter may cause a bias in the responses toward zero or one since this is the only path that does not switch. As shown on Figure 19, the sample flip-flops (FF1 and FF2) are symmetrically placed on the top and bottom paths to minimize the delay difference in the measurements. To implement the PUF, the two sample FFs must be replaced by a single FF whose inputs could potentially follow asymmetric routes introducing bias in the PUF responses. As the measurement results suggest, any delay difference on the path segment between the last switch and the arbiter in the order of 100 picoseconds can be deadly and force the responses completely into zero or one by moving the mean of the distribution. Thus, careful calibration and compensation of this bias is crucial for obtaining robust results. This could be achieved by insertion of extra delay elements or trying different routes on either of the top or bottom path.

We have also tried changing the placement strategies and the PUF lengths (64 switches, 128 switches) on the FPGAs. The small delay variability can partially be compensated by including more delay elements. As Equation (16) implies, σ_Z increases as the square root of the number of delay elements (n) assuming that μ_Z increases linearly n. Assuming the amount of variability on 22nm FPGA with same nominal delays, the total delay variation of the PUF (n=56) would be $3\sigma_Z$ = 160 ps which is more than twice the measured 3σ of the slowest arbiter.

Successful and systematic implementation of PUFs requires a larger variability (that is technologically inevitable with the current trends), details of the variability information about the FPGA fabric and switches, as well as development of tools for automatic timing-aware PUF placement and routing.

10. CONCLUSION

In this article, we developed new techniques for the design and implementation of FPGA-based PUFs. The PUF vulnerabilities to various types of potential attacks were discussed. We demonstrated how reconfigurability can be exploited to ensure that PUFs are resilient against the potential attacks and are robust to unpredictable operational conditions. A PUF testing and characterization

mechanism enabled by reconfigurability feature provides tools for diagnosis, CRP compression, and determining the level of confidence in responses. A unique input/output logic network along with an interconnecting approach is introduced to encumber attempts at reverse engineering or modeling the PUF. The proposed building blocks are added to the PUF after the characterization step. We have shown two new applications where FPGA-based PUFs can be used for security and privacy protection. The effectiveness of all the proposed claims is validated using extensive implementations, simulations, and statistical analysis.

REFERENCES

ALKABANI, Y. M. AND KOUSHANFAR, F. 2007. Active hardware metering for intellectual property protection and security. In *Proceedings of the USENIX Security Symposium*. 1–16.

ALKABANI, Y., KOUSHANFAR, F., AND POTKONJAK, M. 2007. Remote activation of ICs for piracy prevention and digital right management. In *Proceedings of the International Conference on Computer Aided Design (*ICCAD*)*. 674–677.

ALKABANI, Y., MASSEY, T., KOUSHANFAR, F., AND POTKONJAK, M. 2008. Input vector control for post-silicon leakage current minimization in the presence of manufacturing variability. In *Proceedings of the Design Automation Conference (*DAC*)*. 606–609.

ALKABANI, Y. M., KOUSHANFAR, F. N., AND KIYAVASH, M. P. 2008. Trusted integrated circuits: A nondestructive hidden characteristics extraction approach. In *Proceedings of the Conference on Information Hiding*.

ANDERSON, R. 2001. *Security Engineering: A Guide to Building Dependable Distributed Systems*. John Wiley and Sons.

ASENOV, A. 1998. Random dopant induced threshold voltage lowering and fluctuations in sub-0.1 μm MOSFET's: A 3-d atomistic simulation study. *IEEE Trans. Electron. Dev. 45*, 12, 2505–2513.

BERNSTEIN, K., FRANK, D., GATTIKER, A., HAENSCH, W., JI, B., NASSIF, S., NOWAK, E., PEARSON, D., AND ROHRER, N. 2006. High-performance CMOS variability in the 65-nm regime and beyond. *IBM J. Resear. Dev. 50*, 4/5, 433–450.

BOLOTNYY, L. AND ROBINS, G. 2007. Physically unclonable function-based security and privacy in RFID systems. In *Proceedings of the International Conference on Pervasive Computing and Communications*. 211–220.

BRIZEK, J., KHAN, M., SEIFERT, J. P., AND WHEELER, D. 2005. A platform-level trust-architecture for hand-held devices. In *Proceedings of the Workshop on Scalable Trusted Computing*. 19–20.

CORMAN, T., LEISERSON, C., RIVEST, R., AND STEIN, C. 2001. *Introduction to Algorithms*. MIT Press.

DRIMER, S. 2007. Authentication of FPGA bitstreams: Why and how. In *Proceedings of the Conference on Reconfigurable Computing: Architectures, Tools and Applications*. Lecture Notes in Computer Science, vol. 4419, 73–84.

EISENBARTH, T., GÜNEYSU, T., PAAR, C., SADEGHI, A., SCHELLEKENS, D., AND WOLF, M. 2007. Reconfigurable trusted computing in hardware. In *Proceedings of the Workshop on Scalable Trusted Computing*. 15–20.

FELDHOFER, M. AND RECHBERGER, C. 2006. A case against currently used hash functions in RFID protocols. In *Proceedings of the Workshop on RFID Security*. 372–381.

FRIEDBERG, P., CAO, Y., CAIN, J., WANG, R., RABAEY, J., AND SPANOS, C. 2005. Modeling within-die spatial correlation effects for process-design co-optimization. In *Proceedings of the International Symposium on Quality of Electronic Design (ISQED)*, 516–521.

FRY, J. AND LANGHAMMER, M. 2005. RSA and public key cryptography in FPGAs. Tech. rep., *Altera Corporation*.

GASSEND, B., CLARKE, D., VAN DIJK, M., AND DEVADAS, S. 2002a. Controlled physical random functions. In *Proceedings of the Computer Security Applications Conference (ACSAC)*. 149–160.

GASSEND, B., CLARKE, D., VAN DIJK, M., AND DEVADAS, S. 2002b. Silicon physical random functions. In *Proceedings of the Conference on Computer and Communications Security (CCS)*. 148–160.

GASSEND, B., CLARKE, D., VAN DIJK, M., AND DEVADAS, S. 2003. Delay-based circuit authentication and applications. In *Proceedings of the Symposium on Applied Computing*. 294–301.

GASSEND, B., LIM, D., CLARKE, D., VAN DIJK, M., AND DEVADAS, S. 2004. Identification and authentication of integrated circuits. *Concurrency and Computation: Practice and Experience. John Wiley & Sons 16*, 11, 1077–1098.

GLAS, B., KLIMM, A., SANDER, O., MULLER-GLASER, K., AND BECKER, J. 2008. A system architecture for reconfigurable trusted platforms. In *Proceedings of the Conference on Design, Automation and Test in Europe (DATE)*. 541–544.

GUAJARDO, J., KUMAR, S., SCHRIJEN, G., AND TUYLS, P. 2007. FPGA intrinsic PUFs and their use for IP protection. In *Proceedings of the Workshop on Cryptographic Hardware and Embedded Systems (CHES)*. 63–80.

GUNEYSU, T., MOLLER, B., AND PAAR, C. 2007. Dynamic intellectual property protection for reconfigurable devices. In *Proceedings of the International Conference on Field-Programmable Technology (ICFPT)*, 169–176.

HOENEISEN, B. AND MEAD, C. A. 1972. Fundamental limitations in microelectronics I-MOS technology. *Solid-State Electron. 15*, 7, 819–829.

JIE, L. AND LACH, J. 2008. At-speed delay characterization for IC authentication and trojan horse detection. In *Proceedings of the International Workshop on Hardware-Oriented Security and Trust(HOST)*. 8–14.

KEYES, R. 1975. Physical limits in digital electronics. *Proc. IEEE 63*, 5, 740–767.

KOUSHANFAR, F. AND POTKONJAK, M. 2007. CAD-based security, cryptography, and digital rights management. In *Proceedings of the Design Automation Conference (DAC)*.

KOUSHANFAR, F. AND QU, G. 2001. Hardware metering. In *Proceedings of the Design Automation Conference (DAC)*. 490–493.

KOUSHANFAR, F., QU, G., AND POTKONJAK, M. 2001. Intellectual property metering. In *Proceedings of the Information Hiding Workshop*. 81–95.

KUMAR, S., GUAJARDO, J., MAES, R., SCHRIJEN, G.-J., AND TUYLS, P. 2008. The butterfly PUF protecting IP on every FPGA. In *Proceedings of the International Workshop on Hardware-Oriented Security and Trust (HOST)*. 67–70.

KUON, I., TESSIER, R., AND ROSE, J. 2008. *FPGA Architecture*. Now Publishers.

LEE, J., DAIHYUN, L., GASSEND, B., SUH, G., VAN DIJK, M., AND DEVADAS, S. 2004. A technique to build a secret key in integrated circuits for identification and authentication applications. In *Proceedings of the Symposium of VLSI Circuits*. 176–179.

LOFSTROM, K., DAASCH, W., AND TAYLOR, D. 2000. IC identification circuits using device mismatch. In *Proceedings of the International Solid-State Circuits Conference (ISSCC)*. 372–373.

MAEDA, S., KURIYAMA, H., IPPOSHI, T., MAEGAWA, S., INOUE, Y., INUISHI, M., KOTANI, N., AND NISHIMURA, T. 2003. An artificial fingerprint device (AFD): a study of identification number applications utilizing characteristics variation of polycrystalline silicon TFTs. *IEEE Trans. Electron. Dev. 50*, 6, 1451–1458.

MAJZOOBI, M., KOUSHANFAR, F., AND POTKONJAK, M. 2008a. Secure lightweight PUFs. In *Proceedings of the IEEE International Conference on Computer-Aided Design (ICCAD)*. 670–673.

MAJZOOBI, M., KOUSHANFAR, F., AND POTKONJAK, M. 2008b. Testing techniques for hardware security. In *Proceedings of the International Test Conference (ITC)*. 1–10.

MEAD, C. A. 1994. Scaling of MOS technology to submicrometer feature sizes. *Analog Integr. Circ. Sig. Process. 6*, 1, 9–25.

OZTURK, E., HAMMOURI, G., AND SUNAR, B. 2008. Physical unclonable function with tristate buffers. In *Proceedings of the IEEE International Symposium on Circuits and Systems*. 3194–3197.

OZTURK, E., HAMMOURI, G., AND SUNAR, B. 2008. Towards robust low cost authentication for pervasive devices. In *Proceedings of the International Conference on Pervasive Computing and Communications*. 170–178.

PAPPU, R., RECHT, B., TAYLOR, J., AND GERSHENFELD, N. 2002. Physical one-way functions. *Science 297*, 2026–2030.

PFLEEGER, C. P. AND PFLEEGER, S. L. 1997. *Security in Computing*. Prentice Hall.

SEDCOLE, P. AND CHEUNG, P. Y. K. 2006. Within-die delay variability in 90nm FPGAs and beyond. In *Proceedings of the International Conference on Field-Programmable Technology (FPT)*. 97–104.

SHAMSI, D., BOUFOUNOS, P., AND KOUSHANFAR, F. 2008. Noninvasive leakage power tomography of integrated by compressive sensing. In *Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED)*. 341–346.

SU, Y., HOLLEMAN, J., AND OTIS., B. 2007. A 1.6J/bit stable chip ID generating circuit using process variations. In *Proceedings of the International Solid State Circuits Conference (ISSCC)*. 606–611.

SUH, G. AND DEVADAS, S. 2007. Physical unclonable functions for device authentication and secret key generation. In *Proceedings of the Design Automation Conference (DAC)*. 9–14.

SUNAR, B., MARTIN, W. J., AND STINSON, D. R. 2007. A provably secure true random number generator with built-in tolerance to active attacks. *IEEE Trans. Comput. 58*, 1, 109–119.

TRIMBERGER, S. 2007. Trusted design in FPGAs. In *Proceedings of the Design Automation Conference (DAC)*. 5–8.

TUYLS, P., SCHRIJEN, G.-J., SKONC, B., VAN GELOVEN, J., VERHAEGH, N., AND WOTTERS, R. 2006. Read-proof hardware from protective coatings. In *Proceedings of the Cryptographic Hardware and Embedded Systems Workshop*. 369–383.

WOLLINGER, T., GUAJARDO, J., AND PAAR, C. 2004. Security on FPGAs: State-of-the-art implementations and attacks. *ACM Trans. Embed. Comput. Syst. 3*, 3.

WONG, J. S. J., SEDCOLE, P., AND CHEUNG, P. Y. K. 2007. Self-characterization of combinatorial circuit delays in FPGAs. In *Proceedings of the International Conference on Field-Programmable Technology*. 17–23.