Testing Techniques for Hardware Security

Mehrdad Majzoobi[†] and Farinaz Koushanfar[†‡] Miodrag Potkonjak[∘]
[†]Electrical and Computer Engineering Dept., and
[‡]Computer Science Dept. Rice University, Houston, TX 77005
[∘]Computer Science Dept. University of California Los Angeles, Los Angeles, CA 90095

***Abstract*—System security has emerged as a premier design requirement. While there has been an enormous body of impressive work on testing integrated circuits (ICs) desiderata such as manufacturing correctness, delay, and power, there is no reported effort to systematically test IC security in hardware. Our goal is to provide an impetus for this line of research and development by introducing techniques and methodology for rigorous testing of physically unclonable functions (PUFs). Recently, PUFs received a great deal of attention as security mechanisms due to their flexibility to form numerous security protocols and intrinsic resiliency against physical and side channels attacks.**

We study three classes of PUFs properties to design pertinent test methods: (i) predictability, (ii) sensitivity to component accuracy, and (iii) susceptibility to reverse engineering.As our case studies, we analyze two popular PUF structures, linear and feed-forward, and show that their security is not adequate from several points of view. The technical highlights of the paper are the first non-destructive technique for PUF reverse engineering and a new PUF structure that is capable of passing our security tests.

I. Introduction

Security has recently emerged as a hardware design objective of paramount importance [21]. Hardware needs to be protected since its insecurities can facilitate attacks on the programs and contents running on it. Furthermore, manufacturing of integrated circuits (ICs) by potentially untrusted foundries using possibly untrusted CAD tools or insecure reusable components, and the need to protect the digital rights have raised the alarm for protecting hardware. Security's importance is on par with or even more crucial than testing prominence: while non-functional or incorrectly manufactured designs will not operate or will produce incorrect results, an IC with compromised security may also reveal sensitive information that can be exploited by the adversary.

Most present hardware security techniques use conventional cryptographic protocols to provide security. In traditional cryptographic protocols, secrecy is provided by trapdoor mathematical functions and digital keys that make the protocols resilient to algorithmic attacks. However, digital hardware security keys can be attacked in a number of ways including side-channel analysis, electromigration, imaging and fault injection [7]. A big source of vulnerability is that storage of digital data (including the cryptographic keys) is inherently unsafe.

A few years ago, a group of researchers at MIT proposed physically unclonable functions (PUFs) that can be used in a number of security tasks such as authentication [20], [10]. In PUFs, the underlying identification is not in digital format; instead, the identifiers (IDs) are implicitly stored in the analog variations of a physical attribute. This results in systems that are stable and impossible to replicate due to their complex physical structure. For example, inherent manufacturing variability in modern and pending silicon technologies inevitably creates PUFs on each IC. In more practical implementations, PUFs are a separate small circuitry with unique timing, power, or other characteristics. It has been demonstrated that PUFs can be used for many digital rights management and cryptographical tasks [10], [13], [9], [5].

Surprisingly, the security of the proposed and manufactured PUFs has not been fully evaluated. The important observation is that the PUF's security can be addressed both at (i) the design (functional) level, where the goal is to create a structure that for a given manufacturing variability model produces a large percentage of PUFs that pass security tests, and (ii) at the individual IC level, where we check for the security of a particular PUF structure realization.

We have three specific strategic technical goals: (i) Introduction of principals that do not just create a methodology for testing security of PUFs at the functional and the IC level, but also define specific techniques and tools that can be extended to other security hardware. We focus on four broad classes of attacks: predictability, collision, fault-injection, and reverse engineering attacks.(ii) Case study of the test method on popular and widely advocated PUF structures. We analyze two delay-based structures: linear (parallel) and feed-forward. Surprisingly, we show that both structures can be reverse engineered and emulated using the introduced tests. (iii) Creation of the first set of safeguarding techniques to build secure PUFs. The goal is to provide intuition along with a specific PUF structure that is capable of passing the security tests. The quantitative tests indicate drastically higher levels of achieved security with respect to conventional linear and feed-forward PUFs.

To be resilient against the introduced attacks, we propose four tests: (i) predictability, (ii) collision, (iii) sensitivity, and (iv) reverse engineering.Predictability test identifies the difficulty of correctly calculating or predicting the PUF output for a given input. Collision test studies how often two different PUFs produce same outputs to the same given inputs. Sensitivity test ensures that the amount of manufacturing variability is large enough such that a PUF operates in a stable way when the components are imperfect. Reverse engineering test determined the hardness of characterizing the PUF circuit components.

$1-4244-4203-0/08/\$20.00$©2008 IEEE

The remainder of the paper is organized as follows. Section II presents a survey of related literature. The preliminaries are described in Section III. We present our testing methodology in Section IV. The details of application of the methodology to the delay-based PUFs and the results of our security studies are shown in Section V. In Section VI we propose a number of safeguards that alleviate the insecurity of the studied PUFs. We conclude in Section VIII.

II. Related Work

The concept of using the complex structure of physical phenomena as the enabling building block for security protocols was first proposed by Pappu et al. [20]. The key competitive advantages of the concept are that the complex physical phenomena is unique and practically impossible to clone, while it is inexpensive, admitting no compact mathematical representation, and intrinsically resilient to attacks that aim to extract the digital key, such as, differential side-channel power/delay analysis, and fault injection [7]. The concept was demonstrated by mesoscopic physics of coherent transport through a disordered 3D medium. A set of challenge/response pairs were used to perform authentication tasks.

Gassend et al. [10] introduced delay-based silicon PUFs that leverage intrinsic manufacturing variability of deep submicron technologies. Note that the use of silicon manufacturing variability for security was introduced earlier [18] and was exploited for other applications [15], [1], [5], [4], [22]. Recently, gate-level characterization of integrated circuits has received a great deal of research interest [24], [14] because of its importance in IC systems and security [4], [3], [2], and post-silicon optimization [6]. The MIT researchers introduced several different PUF structures, and realized the PUFs on both FPGA and ASICs [16], [13]. They also introduced the notion of a controlled PUF, where the challenges are intertwined by the logic surrounding the PUF using a hash function [10]. However, a hash function is not a suitable option for many light weight secure applications for at least two reasons. First, the key-based hash functions on small devices can be attacked by side channel techniques. Second, not only the hash functions have large area and power overheads, but also most standard hash functions (e.g., MD5, SHA-1) are implemented using sequential circuits. Therefore, a delay overhead in the order of multiple clock cycles is inevitable. One of the key properties of the delay-based PUF is that the responses are computed within a single clock cycle, enabling a faster authentication and real-time security checks.

A number of novel applications for PUFs were introduced by the original team and many others, including authenticated identification in smart cards, RFID authentication, proof of execution and code running on a specific processor, remote IC enabling and disabling, FPGA security, and securing processors [10], [11], [13], [9], [5].

So far, the analysis of PUF's behavior has been very limited. A number of preliminary analysis of model building of linear PUFs were reported [10]. Also, the variations of the implemented circuits under varying voltage and temperature conditions were studied [16]. Information-theoretic analysis of the Pappu's PUFs was performed [27]. To the best of our knowledge this is the first study that conducts statistical analysis of different PUFs structures, and successful reverse engineering of several PUF classes.

III. Preliminaries

A. Delay-based PUFs

Silicon PUFs utilize the timing variations of CMOS logic components caused by the manufacturing variability to take input (challenge) vectors and then produce random output (response) vectors. The manufacturing variability is inherent in current and pending silicon processes [25], [8]. The delay-based silicon PUFs have been first proposed in [10]. There are two main reasons why we consider this class of PUFs. First, practical implementations of this structure have been demonstrated on both ASICs and FPGA [16], [12]. Second, a number of researchers employed this structure for real-life applications, e.g., smart cards, RFID security and privacy, and remote IC activation [11], [9], [26], [1], [5], [22], [2]. Third, the concept can be implemented in pure digital structures and thus, it is easily realizable on FPGAs. This is important because the delay is analog in nature.

The delay-based circuits compare the analog timing differences between a logic path and the clock, or between two logic paths. Implementation of the delay-based structures needs a digital component that translates the analog timing difference to a digital value. *Arbiters* are used for this translation. An arbiter is a digital components that takes two inputs: the arbiter output is one if its first input arrives earlier than the second, i.e., the path ending at the first arbiter input has a shorter delay. The arbiter output is zero otherwise.

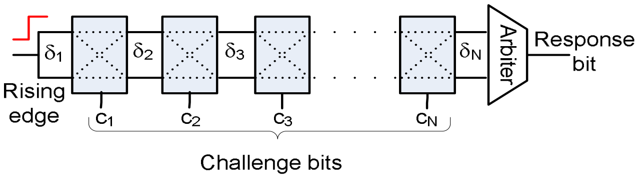

Fig. 1. A parallel PUF block with N challenges and one response bit.

The basic block of a simple parallel delay-based PUF proposed by Lee et al. [16] is shown in Figure 1. Generating one-bit of output requires two parallel paths with multiple segments that terminate at an arbiter. There is a 2-input/2-output switch on each path segment. The path segments are equalized to have the same delay, but their delays differ on actual ICs because of the manufacturing variability. The difference between the top and low path delays on the segment n is represented by δ_n on the figure. To ensure variations, extra delay elements can be inserted on the paths. The challenges are the selector bits of the switches that can have two values. If the challenge bit selector of a switch is zero, the first output path segment is the continuation of the first input path; otherwise, the corresponding two path segments will be switched. The output bit of the arbiter depends on the challenge inputs to the selectors and will be fixed for one IC. The output (response) bit would be different for the same challenge on various ICs. To have multiple response bits, several parallel paths are used.

B. Attack models

In this section, we define and discuss the possible attacks on PUFs. We also identify the desired properties of PUFs that serve as safeguards against attacks. A physically unclonable function can be considered as a transformation that maps a set of challenges to a set of responses. The challenge/response pair is denoted by CRP. The considered attacks are as follows:

- *Prediction attack.* An adversary who has access to the PUF or to a partial database of CRPs may attempt to predict the response to a new challenge by studying the probability, conditional probability, or Hamming distances among the observed CRPs.
- *Reverse engineering attack.* The attacker may attempt to model the input/output behavior of the system by studying a set of CRPs. The reverse engineering attack is most effective when the attacker knows the PUF architecture. To ensure resiliency against this attack, customized tests must be performed on each architecture, since the mathematical model for each PUF architecture is unique.
- *Collision attack.* Collision occurs if two different challenge sets c_1 and c_2 produce the same response on the PUF. Also, two PUFs may collide on their complete or partial CRP space. For a single PUF, the collision attack is relevant when the PUF is used as a hash function. Since the response (hash value) is used as the fixed length signature of the challenges (plain-text), it is desirable to have a unique signature for each plain-text with a very high probability.

IV. Testing Methodology

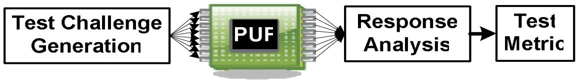

Fig. 2. The flow of each test.

Our testing methodology is dictated by the attack models described in the previous section. More precisely, our methodology covers four tests: *A. predictability test; A. Collision test; C. sensitivity test;* and *D. reverse engineering test*. The diagram in Figure 2 shows the flow of each test. After generating the proper challenge (input) vectors, we study the response (output) of the PUF by performing analysis and developing metrics that define the quality of each test. Note that while some of the tests (e.g., predictability) only analyze the input/output relationship without considering the inside structure of the PUF, a few tests exploit PUF's internal structure.

A. Predictability test

Our first set of tests target predictability. The goal of prediction is to find the CRP relationships such that the response to a new challenge can be guessed with higher than random probability.

1) Single bit probability: The single bit tests target each individual response bit. Considering randomness point of view, for each single output bit, the probability of being 0 (or 1) must be 0.5. Otherwise, the attacker may have a chance at predicting the results. Thus, the evaluation metric for this test is deviation from the random case.

2) Conditional probabilities: These sets of tests study the conditional probability of the response bits with respect to the other response bits and with respect to the input. Since there are many output bits and it is infeasible to study the joint probability of all the outputs and inputs, the test is typically restricted to fewer bits. The range of many possible tests in this category includes the probability of each response bit given the occurrence of each challenge bit, the probability of each response bit conditioned on other response bit(s), and the probability of each response bit conditioned on a subset of challenge or response bits. The internal structure of the PUF may be utilized for determining the independent bits. Again, the evaluation metric is defined as the deviation from the case where bits are completely random.

A class of conditional probabilities that is relevant to PUF is the *conditional transition probability*. The transitional probability targets the probability of a response bit transition, given that a set of challenge or response bits have transitioned from 0 to 1, and vice-versa.

3) Hamming distances: A key predictability factor comes from studying pairs of challenge vectors. For a PUF to be secure, the responses to two challenge vectors that may only differ in one (or a few) bit(s) must be as different as the responses of two completely random challenges. Otherwise, the response to a new challenge vector can be predicted with a higher than random probability. For each PUF under test, we study the relationship between CRPs' Hamming distances (HDs). The density of the response pair's HD vs. the HD of challenges will be analyzed. The deviation of this density compared to the ideal case will be utilized as a metric of predictability.

B. Collision test

This test aims to determine how often a pair of PUFs generate same responses to given challenges. Ideally, if the PUF responses come from uniform distributions, the probability of collision will be only a function of the number of response bits. But in reality the PUF structure can distort the uniformity of responses and introduce a bias.

C. Sensitivity test

This test targets the sensitivity of PUF responses to component imperfections, defects, variations of operational conditions, and ambient noise. For the sensitivity test to work, there is a need to adopt a learn-and-test method. Benchmark studies learn the sensitivity of the PUF to the underlying structure and environmental conditions. While testing, the sensitivity is also tested and compared with the learned benchmark to help diagnose the sensitive parts of the structure.

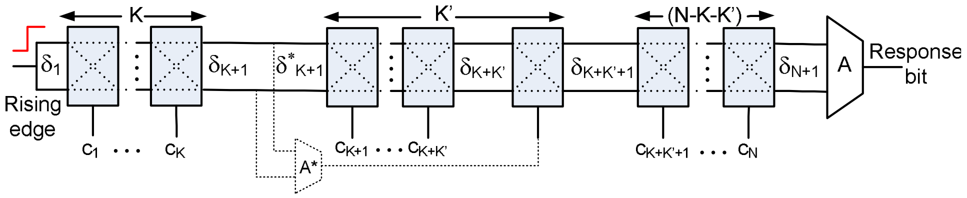

Fig. 3. The feed-forward PUF architecture. The arbiter (A*) introduces nonlinearity.

D. Reverse engineering test

Most physical systems are composed of a finite number of components. The complex interactions among the finite number of components is what defies the reverse engineering attack. We show that a linear system is breakable by using linear optimization (Section V). Addition of nonlinearities is used as a countermeasure. However, maintaining a partial set of inputs constant could fix the nonlinear parts of the structure and thereby, the other parts may be attacked.

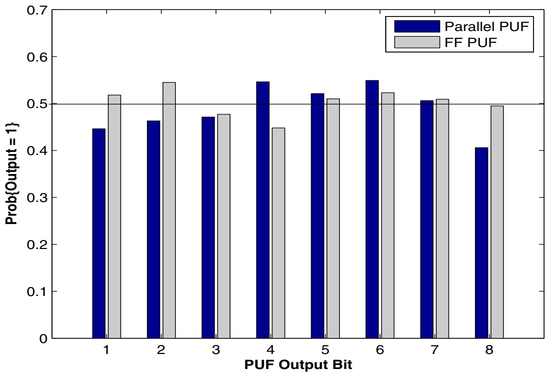

Fig. 4. Response bit probability for one realization of parallel and FF PUFs. Each bar on the x-axis corresponds to a PUF response bit.

V. Case Study:Testing Delay-Based PUFs

We apply the testing methodology introduced in Section IV on two delay-based PUF architectures: (i) the structure shown in Figure 1 to which we refer to as parallel PUF, and (ii) a nonlinear variant of the parallel PUF suggested by Lee et al [16]. The inserted nonlinearity is to prevent reverse engineering attack that can model the linear parallel PUF structure in Figure 1 by a polynomial number (with respect to the PUF size) of measurements. The nonlinearity is introduced by inserting feed-forward arbiter(s) (FFA) such that the output of the inserted arbiter is used as a challenge to a forward selector. An example of adding such a feed-forward arbiter is presented in Figure 3. We call this a feed-forward (FF) PUF structure. The tests reveal a number of security vulnerabilities in the parallel and FF PUFs. In the next section, we introduce a method that can safeguard the PUFs against the diagnosed security holes.

The parallel PUF and FF structures used in our case study consist of 64 and 65 switches respectively. Both structures have 64 input challenges. The FF arbiter input is connected to the switch 20 output and its output goes to switch 40 (i.e., K = 20 and K' = 40 in Figure 3). Each switch is represented by four delays (two delays in the straight connection and two delays in the cross connection states).

A. Predictability test

In predictability studies, we assume that the delay components come from independent identical Gaussian distributions with $\mu = 0.5ns$ and $\sigma = 4ps$. The mean and variance are for the 65nm technology [23]. Studies on correlated delay components are presented in Section V-C.

1) Single bit probability: Figure 4 shows the probability of the output bit being equal to 1 for one parallel and one FF 8-bit PUF circuits. Each bar on the x-axis corresponds to a response bit of the PUF. The probability of each output bit being equal to 1 is presented on the barplot, where each bar is computed by simulating 1,000,000 challenges. The sample PUFs show small variations around the random case (probability=0.5).

To find the density of variations over multiple circuit instances, we simulate the probability of a single response, by applying 1000 challenge sets to 50 different circuit realizations. Since the response from each row is independent of others and the same assumptions are valid for each row, the responses have identical independent distributions. Figure 5 presents the resulting probability density of the response bit. The (μ, σ) of the density is (0.5,0.034) for both parallel and FF architectures. The bell-shaped curve is slightly skewed for the FF architecture.

In all of our tests we adopt a learn-and-test methodology where circuits from a sample population are used to find the distribution of the variations. The rest of the circuits are evaluated against the formed density. Outlying bit probabilities are thus an indication of insensitive arbiters or faults.

2) Conditional probabilities: In this study, we first introduce a transformation that ensures each of the challenge bits directly control the position of the connected path component on the top or bottom path. The *t*op path refers to the route that includes the last upper segment, while the *b*ottom path is the one ending at the last bottom segment right before the arbiter (see Figure 1). For example, if all the challenge bits are zero the top path consists solely of upper segments. The XOR logic block in Figure 6 maps the user's inputs to the PUF challenge bits. Lets define a transformation function as follows:

$$\rho_i^j(c_1, ..., c_N) = \rho_i^j(\underline{c}) = c_i \oplus c_{i+1} \oplus ... \oplus c_j, \qquad (1)$$

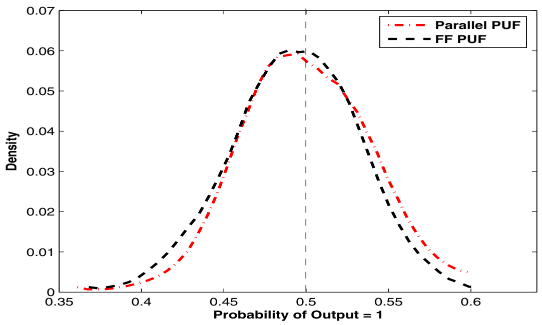

Fig. 5. The probability density of an output bit over multiple PUF circuits.

where $i < j$. By controlling the PUF by ρ's instead of c's (Figure 1), one can directly decide if the segment m would belong to the top path or the bottom path. The XOR logic shown in Figure 6 performs the following transformation: $d_i = c_i \oplus c_{i+1}$ if $i < N$, and $d_N = c_N$. It can be easily verified that $\rho_i(d_1, ..., d_N) = c_i$. Thus, the XOR logic block realizes the inverse transformation of $\rho_i(.)$, denoted by $\rho_i^{-1}(.)$ and the position of each segment is individually controlled.

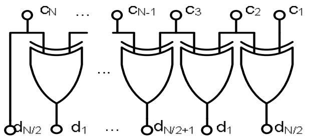

Fig. 6. The logic block mapping the user's inputs to the challenges.

For a PUF circuit with a single response bit, we simulated the probability of output transition conditioned on the challenge bit transitions. In this experiment, we apply 100 challenge pairs to the PUF that differ in the i-th bit, where i = 1,...,64, and record the number of times that the output transitions. Transitions can happen both from 1 to 0 and from 0 to 1. The result is shown in Figure 7, where the x-axis presents the flipping challenge bit number, and the y-axis shows the probability of output transition. The challenge bits are in order, from left to right selector on the PUF. The figure shows that the average conditional probability increases to 0.1 in 0.9 as we move from the leftmost challenge bit to the rightmost one. This can intuitively explained as cumulation of switch delays on the way to the arbiter. Interestingly, the added XOR block flattens the slope for both FF and parallel PUF.

The results for 50 PUF realizations are shown in Figure 8, for parallel PUFs with and without the XOR transformation. Because of space limitation, we do not show the boxplot for the FF PUF but emphasize that the results are similar. Each box in the boxplot represents five numbers that summarize the density: the smallest observation (lower fence), lower quartile (lower box edge), median (middle bar in the box), upper quartile (upper box edge), and largest observation (upper fence). The outliers are indicated by "+" and "." signs below and above the fences.

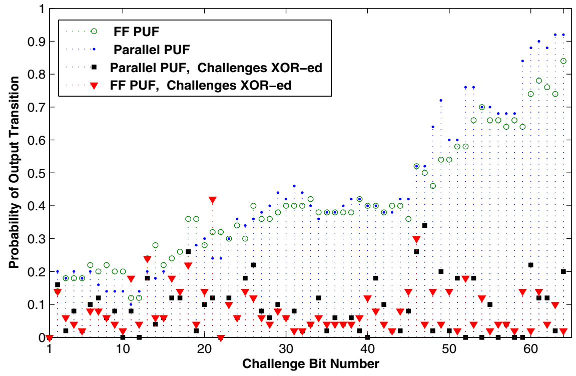

Fig. 7. Conditional probability of output transition for a single chip. The probability is shown for parallel, FF, with and without XOR transformation.

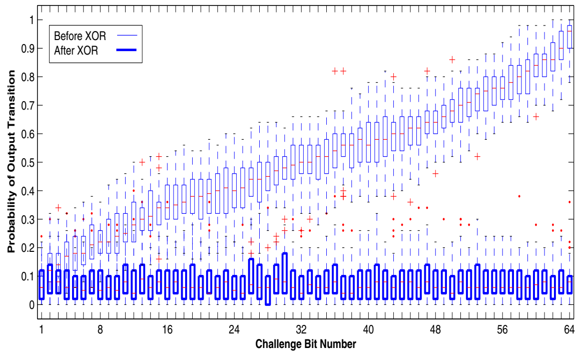

Fig. 8. Conditional probability of output transition on parallel PUF before and after challenge transformation.

The test results for the conditional probability of the response bit value (being equal to 0 or 1) conditioned on the challenge bit value (being equal to 0 or 1) show a flat characteristic for both parallel and FF PUFs (Figure 9). The presented results show prediction vulnerability, as the probability of the response bit transition conditioned on a single challenge bit transition is nonuniform and dependent on the place of the challenge. Adding the transformation circuit makes the conditional probability of transition uniform.

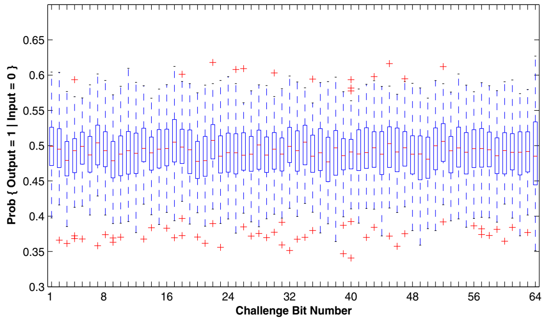

Fig. 9. Probability of the response bit=1, conditioned on the i-th challenge=0.

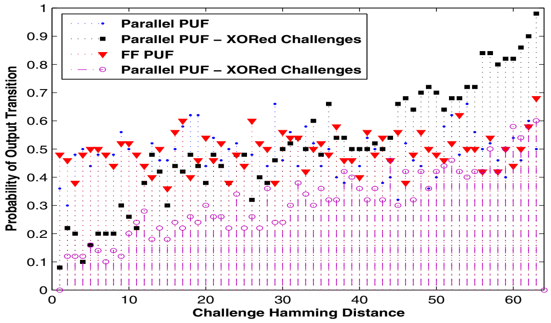

Fig. 10. Probability of output transition conditioned on challenge HDs for a parallel and a FF PUF circuit, before and after the XOR transform.

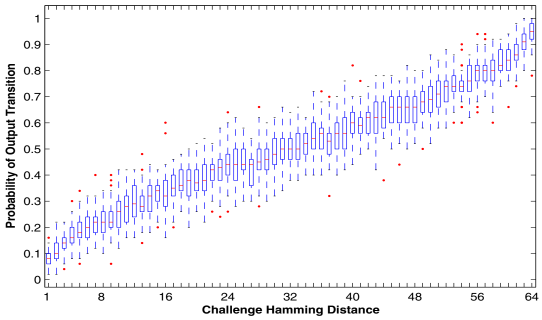

Fig. 12. The probability of output transition conditioned on challenge pair HD after using the XOR logic block for a population of 50 parallel PUF.

3) Hamming distances: We test the delay-based PUF to obtain the probability of output transition conditioned on the Hamming distance (HD) of the challenge pairs. The learn-and-test method is utilized to study the density of the output transition for a sample population of 50 PUF circuits. In this experiment, we apply 100 challenge pairs with HD of i to the PUF, where i= 1,...,64, and record the percentage of times when the output transitions. We also test the PUF with the XOR block attached to its input. The results for a single PUF is shown in Figure 10. The boxplots in Figures 11 and 12 show the distribution of output transition probabilities obtained for 50 PUFs, before and after the XOR transformation. We see that the probability of output transition increases for larger Hamming distances after the XOR transformation. Thus, even though the transformation flattens the probability of outp transition conditioned on a single challenge bit, it introduc vulnerabilities for the cases conditioned on challenge pa HDs.

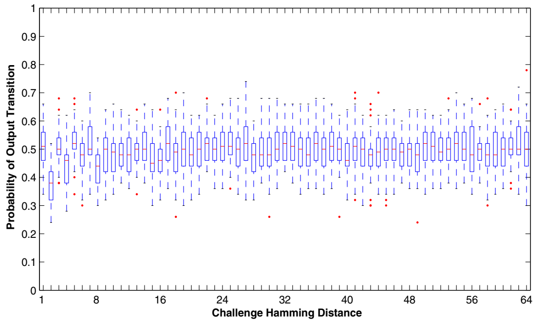

Fig. 11. The density of output transition given challenge pairs' HDs.

B. Collision Test

We test the PUF to obtain the collision probability for different PUF structures. For each given challenge, the PUF responses on various chips must form a uniform distribution to yield the minimum collision probability. The nonlinearity introduced by the FF arbiter distorts the uniformity of output responses and causes higher collision probability, even in presence of completely independent delays and perfect arbiters. Depending on the PUF circuit structure and the location of nonlinearity, there is a lower bound on collision probability.

For a parallel PUF that consists of M response bits (M rows), the minimum collision probability is $\frac{1}{2^M}$. For example, if the PUF has 8 output bits, then the collision probability is $\frac{1}{256}$. Figure 13 shows the collision probability for the FF PUF normalized to that of the parallel PUF (z-axis) vs. feed-forward arbiter input/output locations. The collision probability will be at least 65 times greater than the parallel PUF when the FF arbiter is placed at ($input = 59$, $output = 64$) location (i.e. close to the final arbiter). One way to compensate for this increase in collision probability is to increase the number of responses (PUF rows) to lower the overall collision probability.

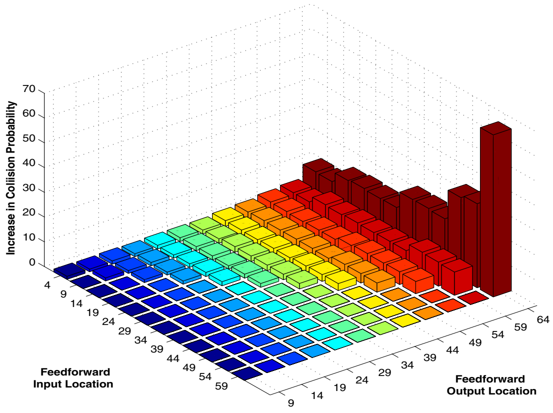

Fig. 13. The increase in the collision probability compared to the parallel PUF vs. feed-forward arbiter's input (x-axis) and output (y-axis) locations.

C. Sensitivity test

We now investigate the effect of spatial correlation of delays and arbiter non-ideality. Arbiters may be non-ideal for two reasons: (i) insensitivity to capture small delay differences, characterized by setup/hold times, and (ii) asymmetry in setup/hold times. The former leads to arbiters producing non-deterministic meta-stable responses if the delay differences of the signals at the arbiter input are smaller than the arbiter setup/hold times. Meta-stable responses vary with environmental fluctuations. Asymmetry in the setup/hold times causes either the 0 or 1 response to be more likely, biasing the PUF output. Figure 14 shows the output bit probability distribution for the parallel and FF PUFs in presence of asymmetry in arbiter setup and hold times. A setup time (ST) three times longer than the hold time (HT) and vise versa are assumed, i.e., (ST,HT) = (30,10) ps and (ST,HT) = (10,30) ps. The arbiters are assumed to produce equally likely 0 and 1 responses if the ST/HT is violated.

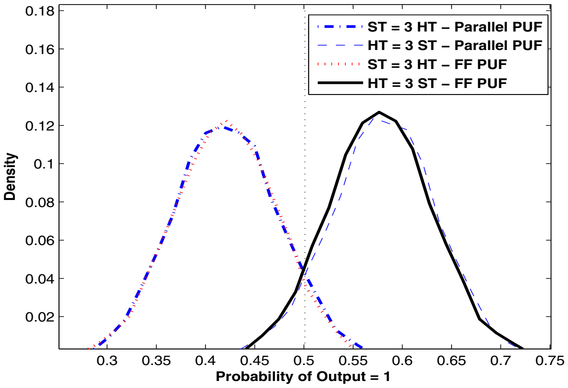

Fig. 14. Output bit probability distribution for unequal arbiter ST and HT's.

Next, the effect of delays' spatial correction is studied. We employ the multivariate Gaussian distribution introduced by Liu [17], who uses the exponential correlogram function ($e^{-\alpha}$) to model spatial correlations, where α is a correlogram function parameter. Note that for an ideal arbiter, the tests in Sections V-A and V-B yield the same results in presence of different degrees of spatial correction. However it can be seen in Figure 15, as the degree of correlation among the element delays increases the delay variance at the arbiter input becomes smaller. Thus, for a fixed (non-ideal) arbiter sensitivity, more responses will be meta-stable. Figure 15 shows the delay variance normalized to the independent case versus the degree of spatial correlation. Larger $e^{-\alpha}$ value indicates a stronger level of spatial correlation. The FF and parallel PUFs have the same level of sensitivity to spatial correlations.

In addition to non-ideal arbiters and delay correlations, PUF switches may also suffer from faults and large delays. Figure 16 shows the conditional probability of output transits for the case where a delay outlier is present in switch 20. The case where no outliers are present is also shown for comparison purposes. It can be seen that the probability values show a sharp transition at switch 20.

D. Reverse engineering test

1) Parallel PUF: The parallel PUF can be easily reverse engineered using a linear number of CRPs and forming a system of linear inequalities. The system can be solved by linear programming in order to find the (differential) path segment delays (δ's). For each challenge input vector $(c_1[l], \ldots, c_N[l])$ used in l-th measurement and the corresponding response bit $r[l]$, one can form an inequality:

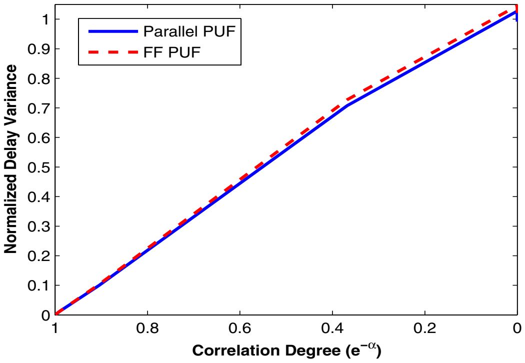

Fig. 15. Delay variance at the input of the arbiter normalized to the delay variance of independent case, versus the degree of correlation.

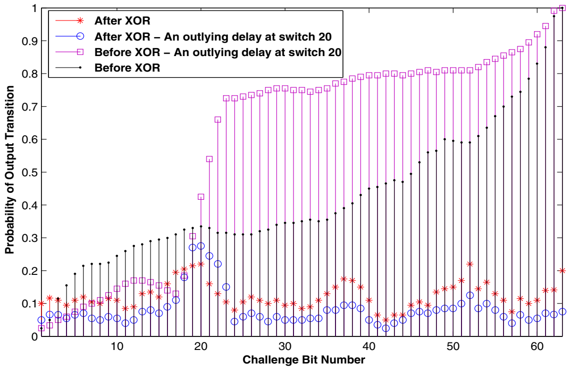

Fig. 16. The probability of output bit transition conditioned on i-th input transition in the presence of a delay outlier at the switch 20; before and after the XOR transformation (Parallel PUF).

$$\forall l, \qquad \sum_{j=1}^{N}(-1)^{\rho_j(.)[l]}\delta_j + \delta_{N+1} \underset{r[l]=1}{\overset{r[l]=0}{\lessgtr}} 0, \tag{2}$$

where $\rho_j(.)$ is defined in Equation 1. The direction of inequality is determined by the PUF response to the l-th challenge vector. In presence of measurement errors, an error term $\epsilon[l]$ is added to the left side of each term in Equation 2. We formulate a linear program (LP) where the set of inequalities in Equation 2 are the constraints and the objective function is to minimize a norm of error over L measurements, e.g., $\min \sum_{l=1}^{L} |\epsilon[l]|$.

We evaluated our reverse engineering approach using a set of CRPs. The box plot in Figure 17 shows the modeling accuracy in percentage versus the number of measurements (CRPs) for a population of 50 chips. We see that by using only 3000 CRPs, the adversary can model the PUF with 99% accuracy. The test set in our experiment contains 10000 CRPs.

We also test the susceptibility to reverse engineering in presence of measurement or arbiter errors. In the experiment we use 3000 CRPs, and randomly inject errors in the response measurements. Figure 18 shows the model accuracy versus

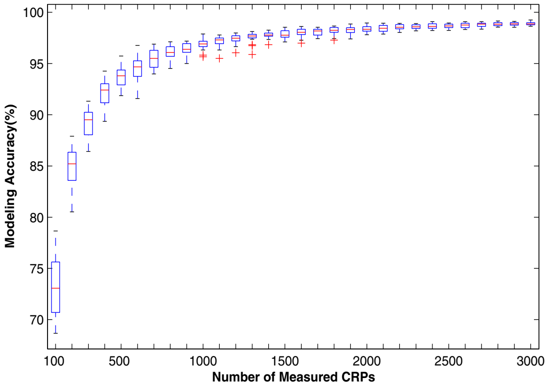

Fig. 17. Evaluation of accuracy distributions versus the number of measurements obtained for a collection of 50 chips.

different degrees of measurement error for a population of 50 PUFs. To improve resiliency against measurement errors, we used the maximum likelihood approach. For example, for a Gaussian delay distribution, the likelihood function would have a quadratic form which changes the LP to a convex programming problem. The improved results are also shown in Figure 18.

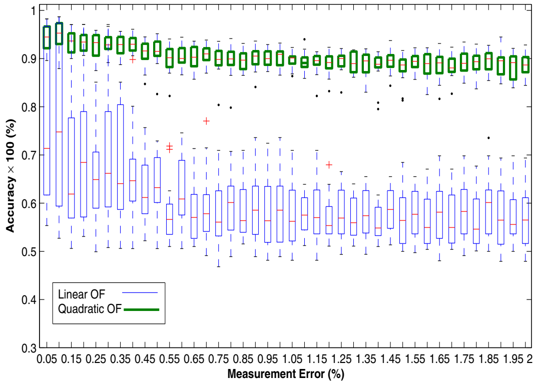

Fig. 18. Modeling accuracy distributions versus the measurement errors. Distributions are obtained for a collection of 50 chips.

2) Feed-forward PUF: To prevent reverse engineering, Lee et al. [16] suggest adding nonlinearities to the circuit. They insert feed-forward arbiters (FFA) in the path such that the added arbiters provide the challenge to a selector as shown in an example in Figure 3. The A^* arbiter's inputs are coming from stage K. The output of A^* is fed-forward to the $K+K'$-th selector challenge bit.

We reverse engineer FFAs in the following way. If we denote the total path delay difference incurred by the signal till the $K+K'$ switch with Δ, then

$$\begin{aligned}\Delta = \quad & \sum_{i=1}^{K}(-1)^{\rho_i^{K+K'}}\delta_i + (-1)^{\rho_{K+1}^{K+K'}}(y_{K+1} + \delta^*_{K+1}) \\ + \quad & \sum_{i=K+2}^{K+K'}(-1)^{\rho_i^{K+K'}}\delta_i + \delta_{K+K'+1}. \qquad (3)\end{aligned}$$

The delay in the segment between the switch K and switch $K+1$ is broken down into two parts, δ_{M+1} and δ^*_{M+1} and therefore the PUF has one more parameter than linear PUF. For the sake of simplicity, the measurement index l (previously defined for Equation 2) is removed. The feed-forward arbiter's result, $c_{K+K'}$, provides another inequality

$$\sum_{i=1}^{K}(-1)^{\rho_i^K}\delta_i + \delta_{K+1} \underset{c_{K+K'}=1}{\overset{c_{K+K'}=0}{\lesseqgtr}} 0. \qquad (4)$$

We also use the following identity that can be directly derived from the definition of ρ_i^j

$$\begin{aligned}\rho_i^{K+K'} &= \rho_i^{K+K'-1} \oplus c_{K+K'} = \rho_i^K \oplus \rho_{K+1}^{K+K'} \\ &= \rho_i^K \oplus \rho_{K+1}^{K+K'-1} \oplus c_{K+K'}. \qquad (5)\end{aligned}$$

Observing that $(-1)^{a\oplus b} = (-1)^a(-1)^b$, Equation 3 is further simplified to

$$\begin{aligned}\Delta = \quad & (-1)^{\rho_{K+1}^{K+K'-1}}|\Delta_{first}| \qquad (6) \\ + \quad & (-1)^{c_{K+K'}}((-1)^{\rho_{K+1}^{K+K'-1}}\delta^*_{K+1} + \Delta_{middle}) + y_{K+K'+1}.\end{aligned}$$

Where $\rho_{K+1}^{K+K'-1}$ is the parity (XOR results) of the challenges to middle stage. Δ_{first}, Δ_{middle} , and Δ_{last} are the first, middle, and last stage differential delays computed respectively as,

$$\Delta_{first} = \sum_{i=1}^{K}(-1)^{\rho_i^K}\delta_i + \delta_{K+1}, \qquad (7)$$

$$\Delta_{middle} = \sum_{i=K+2}^{K+K'-1}(-1)^{\rho_i^{K+K'-1}}\delta_i + \delta_{K+K'}, \qquad (8)$$

$$\Delta_{last} = \sum_{i=K+K'+1}^{N}(-1)^{\rho_i^N}y_i + y_{N+1}. \qquad (9)$$

The total delay can now be expressed as

$$\Delta_{total} = \Delta \times (-1)^{\rho_{K+K'+1}^N} + \delta_{last}. \qquad (10)$$

We complete reverse engineering of FF PUF by using the following observations.
(i) By fixing the selector bits of the switches in first stage (K first switches), we estimate the delays of switch elements in the middle and last stage by solving an LP problem similar to the one in Section V-D1. However, we need to make two assumptions on the FF arbiter output and the LP would have two solutions. The solutions obtained by using these two

assumptions only differ in sign which can be easily resolved later.

(ii) Knowing the delays of switches in the middle and last stages (with a sign ambiguity for the delays of the middle stage) and considering the PUF formulation (Equation 10), we set the challenges to the middle and last segments in a way so that any transition of the final arbiter is closely linked to the transitions of the FF arbiter output. This can be realized by choosing a challenge configuration that yield a large delay difference for the middle stage ($\Delta_{middle} >> 0$), while causing a negligible delay difference at the last stage ($\Delta_{last} \approx 0$).

(iii) While the challenge bits to the middle and last stages are fixed to the appropriate configuration found in (ii), complementary challenges are applied to the first stage switches and transitions of PUF responses (final arbiter response transitions) are recorded. Any time the final arbiter response flips, we obtain a constraint for the LP. Since we are concerned with transitions rather than absolute output values, we need to address two LP problems by trying two different bit assignments. However, the delay values obtained from the incorrect solutions can be easily rejected by cross-validating the results on a few new CRPs.

(iv) Using the estimated delays of the first stage, we can eliminate the ambiguity in the sign of the middle-stage delay difference. Therefore the delays of all switches can be estimated successfully.

In our experiment, the PUF has the structure presented in Figure 3, where $K = 24$, $K' = 20$ and $N = 64$. After reverse engineering the PUF, we validate our model using 10,000 CRPs and measure the accuracy of the modeled PUF. Figure 19 shows the model accuracy versus different number of measurements for 20 PUFs. The first 15,000 CRPs (measurements) are used to estimate the middle and last stage switch delays and the rest are used to estimate the first stage switch delays. Also note that for step (ii), finding the challenge configuration to the middle and last stages that yields the largest and smallest possible delay differences is, in general, an NP-complete problem. But we do not need an exact solution and a rough approximation that gives a very small (large) delay is sufficient. For example, we can try 1000 challenge bit combinations and find the one that gives the minimum delay difference.

VI. Safeguarding The Delay-based PUFs

In order to overcome security limitations of parallel and FF PUFs, we designed a new, interleaved, PUF structure. It consists of R parallel rows of PUF structures, where R is an even integer. The last challenge bit of the PUFs located in even numbered rows are connected to the first challenge bit of the PUFs in the odd rows. In general, the i-th challenge bit of the PUFs in even rows are connected to the $(N - i)$-th challenge bit of the PUFs in odd rows where N is the total number of challenges in one row. The outputs are combined using $R-input$ XOR gates and leave-one-out logic so that a structure consisting of $R+1$ rows, R '$R-input$' XORs generate R outputs. Figure 20 shows an example of the proposed PUF structure with 5 rows.

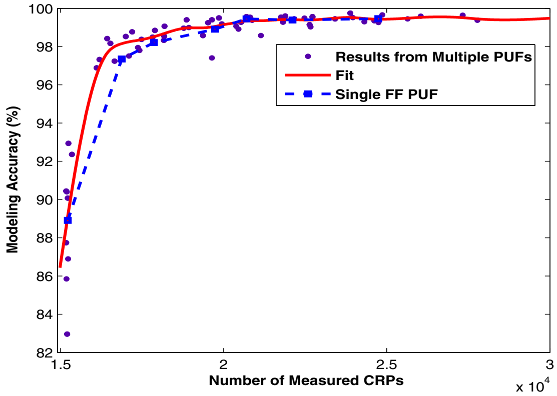

Fig. 19. Modeling accuracy versus the number of measurements obtained for a collection of 20 FF PUFs.

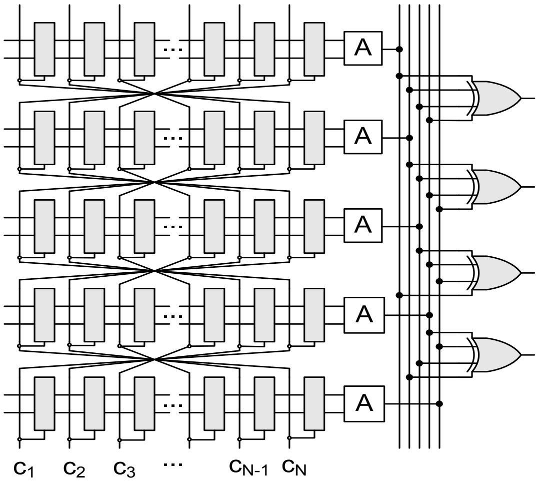

Fig. 20. The proposed interleaved PUF structure. 5 row are shown. R+1 rows are required to produce R responses.

The interleaved PUF mixes the individual Hamming distances and conditional transitional probabilities using the XOR logic. Figure 21 shows the probability of output conditional transition as a function of the i-th input bit transition for structures mixing outputs of 2, 4, and 8 rows. Also the probability of output transition given the input HD is shown in Figure 22. The results in these figures are averaged over 50 PUF realizations. The results suggest that as more number of rows are combined, a flatter characteristic will be achieved. The interleaved PUF has another advantage: it exponentially impedes reverse engineering by creating difficulty in discovering the arbiter outputs. For each challenge, there are always two possible arbiter responses which can produce the same output. So, the adversary must resolve the ambiguity in deducing the true PUF arbiter output for a given challenge. A more detailed and thorough analysis of the proposed PUF which is in fact an instance of a broader class of PUFs is given in [19].

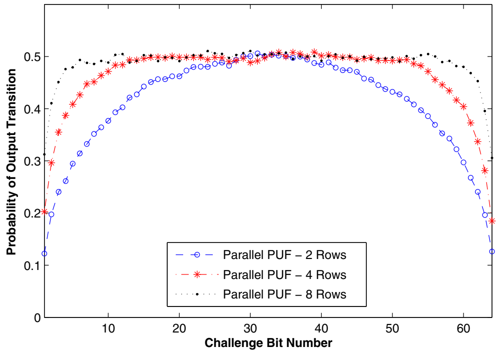

Fig. 21. The conditional probability of output transition after a transition at the i-th input bit, for interleaved PUF mixing 2, 4, and 8 rows.

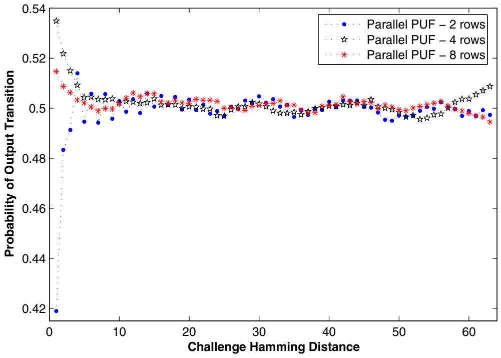

Fig. 22. The probability of output transition given an input HD, for structures mixing 2, 4, and 8 rows.

VII. Acknowledgement

This work is in part supported by the DARPA/MTO Young Faculty Award W911NF-07-1-0198, NSF CT-0716674, NSF CAREER-0644289, and Texas Instruments Leadership University Award.

VIII. Conclusion

We introduce the first system of testing approaches for evaluating security of PUFs by classifying the attacks and by developing techniques for testing several PUFs security properties, including predictability, collision, sensitivity, and susceptibility to reverse engineering. Using our methodology, we demonstrate that popular parallel and feed-forward delay-based PUFs are often easy to predict, reverse engineer, and emulate. The technical highlights of the paper are the first non-destructive technique for reverse engineering of PUFs and a new PUF structure that passes the proposed security tests.

References

[1] Y. Alkabani and F. Koushanfar. Active hardware metering for intellectual property protection and security. In *USENIX Security*, pages 291–306, 2007.

[2] Y. Alkabani and F. Koushanfar. Active control and digital rights management of integrated circuit IP cores. In *Compilers, Architectures, and Synthesis for Embedded Systems (CASES)*, 2008.

[3] Y. Alkabani and F. Koushanfar. N-variant IC design: Methodology and applications. In *Design Automation Conference (DAC)*, 2008.

[4] Y. Alkabani, F. Koushanfar, N. Kiyavash, and M. Potkonjak. Trusted integrated circuits: A nondestructive hidden characteristics extraction approach. In *Information Hiding (IH)*, 2008.

[5] Y. Alkabani, F. Koushanfar, and M. Potkonjak. Remote activation of ICs for piracy prevention and digital right management. In *International conference on computer-aided design (ICCAD)*, pages 674–677, 2007.

[6] Y. Alkabani, T. Massey, F. Koushanfar, and M. Potkonjak. Input vector control for postsilicon leakage current minimization in the presence of manufacturing variability. In *Design Automation Conference (DAC)*, 2008.

[7] R.J. Anderson. *Security Engineering: A guide to building dependable distributed systems*. John Wiley and Sons, 2001.

[8] K. Bernstein, D.J. Frank, A.E. Gattiker, W. Haensch, B.L. Ji, S.R. Nassif, E.J. Nowak, D.J. Pearson, and N.J. Rohrer. High-performance CMOS variability in the 65-nm regime and beyond. *IBM Journal of Research and Development*, 50(4/5):433–450, 2006.

[9] L. Bolotnyy and G. Robins. Physically unclonable function-based security and privacy in rfid systems. In *IEEE International Conference on Pervasive Computing and Communications*, pages 211–220, 2007.

[10] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas. Silicon physical random functions. In *ACM conference on Computer and communications security (CCS)*, pages 148–160, 2002.

[11] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas. Delay-based circuit authentication and applications. In *ACM symposium on Applied computing*, pages 294–301, 2003.

[12] B. Gassend, D. Lim, D. Clarke, M. van Dijk, and S. Devadas. Identification and authentication of integrated circuits. *Concurrency and Computation: Practice and Experience. John Wiley & Sons*, 16(11):1077–1098, 2004.

[13] J. Guajardo, S. Kumar, G. Schrijen, and P. Tuyls. FPGA intrinsic PUFs and their use for IP protection. In *Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, 2007.

[14] F. Koushanfar, P. Boufounos, and D. Shamsi. Post-silicon timing characterization by compressed sensing. In *International Conference on Computer Aided Design (ICCAD)*, 2008.

[15] F. Koushanfar and M. Potkonjak. Cad-based security, cryptography, and digital rights management. In *Design Automation Conference (DAC)*, 2007.

[16] J.W. Lee, L. Daihyun, B. Gassend, G.E. Suh, M. van Dijk, and S. Devadas. A technique to build a secret key in integrated circuits for identification and authentication applications. In *Symposium of VLSI Circuits*, pages 176–179, 2004.

[17] F. Liu. A general framework for spatial correlation modeling in VLSI design. In *Design Automation Conference (DAC)*, pages 817–822, 2007.

[18] K. Lofstrom, W.R. Daasch, and D. Taylor. IC identification circuits using device mismatch. In *International Solid State Circuits Conference (ISSCC)*, pages 372–373, 2000.

[19] M. Majzoobi, F. Koushanfar, and M. Potkonjak. Lightweight secure PUF. In *International conference on computer-aided design (ICCAD)*, 2008.

[20] R. Pappu, B. Recht, J. Taylor, and N. Gershenfeld. Physical one-way functions. *Science*, 297:2026–2030, 2002.

[21] S. Ravi, A. Raghunathan, P. Kocher, and S. Hattangady. Security in embedded systems: Design challenges. *ACM Trans. on Embedded Computing Systems (TECS)*, 3(3):461–491, 2004.

[22] J. Roy, F. Koushanfar, and I. Markov. EPIC: Ending piracy of integrated circuits. In *Design Automation and Test in Europe (DATE)*, 2008.

[23] P. Sedcole and P. Y. K. Cheung. Within-die delay variability in 90nm fpgas and beyond. In *IEEE International Conference on Field Programmable Technology*, 2006.

[24] D. Shamsi, P. Boufounos, and F. Koushanfar. Noninvasive leakage power tomography of integrated circuits. In *International Symposium on Low Power Electronics and Design (ISLPED)*, 2008.

[25] A. Srivastava, D. Sylvester, and D. Blaauw. *Statistical Analysis and Optimization for* VLSI*: Timing and Power*. Series on Integrated Circuits and Systems. Springer, 2005.

[26] G. Suh and S. Devadas. Physical unclonable functions for device authentication and secret key generation. In *DAC*, pages 9–14, 2007.

[27] P. Tuyls, B. Skoric, S. Stallinga, T. Akkermans, and W. Ophey. An information theoretic model for physical uncloneable functions. In *International Symposium on Information Theory (ISIT)*, page 141, 2004.