Time-bounded Authentication of FPGAs

Mehrdad Majzoobi, *Student Member, IEEE,* Farinaz Koushanfar, *Member, IEEE*

***Abstract*—This paper introduces a novel technique to authenticate and identify field programmable gate arrays (FPGAs). The technique uses the reconfigurability feature of FPGAs to perform self-characterization and extract the unique timing of the FPGA building blocks over the space of possible inputs. The characterization circuit is then exploited for constructing a physically unclonable function (PUF). The PUF can accept different forms of challenges including pulse width, digital binary and placement challenges. The responses from the PUF are only verifiable by entities with access to the unique timing signature. However, the authentic device is the only entity who can respond within a given time constraint. The constraint is set by the gap between the speed of PUF evaluation on authentic hardware and simulation of its behavior. A suite of authentication protocols is introduced based on the time-bounded mechanism. We ensure that the responses are robust to fluctuations in operational conditions such as temperature and voltage variations by employing: (i) a linear calibration mechanism that adjusts the clock frequency by a feedback from on-chip temperature and voltage sensor readings, (ii) a differential PUF structure with real-valued responses that cancels out the common impact of variations on delays. Security against various attacks is discussed and a proof-of-concept implementation of signature extraction and authentication are demonstrated on Xilinx Virtex 5 FPGAs.**

***Index Terms*—field-programmable gate arrays; physically unclonable function; delay characterization; time-bounded authentication;**

I. Introduction

Many security mechanisms are based upon the concept of a *secret*. Classic cryptography protocols contain a secret key for reversing trapdoor functions. While such protocols are often secure against attacks at the algorithmic level, it is well-known that digitally stored secret keys can be attacked and cloned. Furthermore, secret key storage has major limitations when applied to conventional FPGAs since the reconfigurable fabric technology cannot easily integrate non-volatile memory (NVM). Thus, the keys need to be stored on off-chip memory which demands secure channels and additional protocols to communicate the keys to- and from- the off-chip components. On-chip key storage requires the overhead of a constant power source. Such reliance not only incurs additional overhead, but increases the vulnerability to attacks.

Physical unclonable functions (PUFs) are efficient mechanisms for many security applications [1], [2]. PUFs exploit secret information inherently embedded in the unclonable physical variations of the silicon devices to produce secure digital keys. Moreover, a PUF provides a unique chip-dependent mapping from a set of digital inputs (challenges) to digital outputs (responses) based on the unique properties of the underlying physical device. PUFs can be employed to provide security at multiple levels and to address a range of problems from securing processors [3], to software protection [4], IP protection [5], and IC authentication [6]. Even though a number of methods for realizing PUFs on FPGAs have been proposed [5], [7], the scope of existing FPGA PUFs is limited. Either they have a limited number of challenge-response pairs, or they are limited by the routing constraints on FPGAs, or they introduce noise and other vulnerabilities to the system.

In this paper, we introduce and implement a novel mechanism for FPGA PUFs. First, the delay of each configurable block is characterized for different combinations of inputs. A challenge to the PUF queries the delays of a subset of configurable blocks. Reproducing the responses requires the knowledge of the extracted delays as well as the structure and placement of the PUF circuit. We introduce a dynamic authentication protocol, where the placement of the PUF is randomly changed during each round of authentication, forcing the adversary to constantly reverse-engineer the configuration bit stream to discover the PUF structure and placement. The protocol is based on the fact that it is infeasible to generate the challenge-response signatures through simulation in a short time duration compared to evaluation on hardware.

As an example application, the FPGA can be used as a hardware security token, where the owner of the genuine FPGA can use it to authenticate him/herself. Before distribution of the tokens (FPGAs) to end users, the delay characteristics of the FPGA components are extracted and stored in a publicly known database. A PIN or serial number is associated with each FPGA and its corresponding database entry. Next, the token is handed out to the users. At the time of authentication, the user in possession of the genuine FPGA presents him/herself by the associated serial number. The verifier then sends a configuration bitstream along with a set of challenges to the device. The responses are expected within an alloted time frame. Since the verifier knows the configuration of the PUF, he/she can simulate its behavior with the help of public delay characteristics and compare the simulated response to those received from the user to perform authentication.

Our contributions are as follows:

- We introduce a new mechanism for efficient self-extraction of the unclonable and unique analog timing signature of the FPGA logic blocks. The self extraction only requires ordinary desktop computers and commodity logic analyzers.
- We introduce a new PUF structure built upon the self-extracting mechanism. The introduced PUF can accept different inputs (challenge formats) including timing, digital binary, and placement challenges; it can generate both binary and real-valued outputs (responses).
- A new time-bounded authentication is developed which

F. Koushanfar and M. Majzoobi are with the Department of Electrical and Computer Engineering, Rice University, Houston, TX, 77005 USA e-mail: farinaz@rice.edu and mm7@rice.edu.

exploits the delay gap between the speed of evaluation of authentic hardware and simulation of the hardware behavior. A suit of new authentication protocols is built upon the time-bounded property.
- A linear calibration method is presented that adjusts the clocks to compensate for the impact of variations in operational conditions using feedback from on-chip temperature and voltage sensors.
- To cancel out the common impact of variation, a low overhead differential PUF structure is presented.
- Attacks and countermeasures for the new PUF and the time-bounded authentication are discussed.
- Extensive evaluation and proof-of-concept implementation on vertex 5 FPGA demonstrate the applicability, consistency and efficiency of the new methods.

The remainder of the paper is organized as follows. The next section summarizes the related literature. Section III presents the flow of the proposed methodology. In Section IV, we present the method for signature extraction for each possible challenge. Section V describes how the characterization circuit can be challenged as a PUF. Next, different ways to achieve robustness of PUF responses are presented in Section VI. Section VII introduces time-bounded authentication protocol and its variants. Attacks and countermeasures are discussed in Section VIII. Experimental evaluations are demonstrated in Section IX. Finally, we conclude the paper in Section XI.

II. Related work

The idea of using complex unclonable features of a physical system as an underlying security mechanism was initially proposed by Pappu et al. [1]. The concept was demonstrated by studying mesoscopic physics of coherent transport through a disordered medium. Another group of researchers observed that the manufacturing process variability in modern silicon technology can be utilized for building a PUF. They proposed the arbiter-based PUF architecture based on the variations in CMOS logic delays [2]. The arbiter-based PUF implementation on ASICs was demonstrated, and a number of attacks and countermeasures were discussed [2], [8], [9], [6], [10]. In particular, it was observed that the linear arbiter-based PUF is vulnerable to modeling attacks and the use of nonlinear feed-forward arbiters and hashing to proposed to safeguard against this attack [2]. Moreover, error correcting codes were proposed in [11] to alleviate instability of PUF responses.

Further efforts were made to address the PUF vulnerability issues by adding input/output networks, adding nonlinearities to hinder machine learning and enforcing an upper bound on the PUF evaluation time [7], [12], [13]. The work in [7] demonstrated that even though successful ASIC implementation of arbiter PUF was shown, FPGA implementation of this PUF is troubled due to the routing constraints enforced by the regularities of the underlying FPGA fabric. In particular, asymmetries in routing when implementing arbiter-based PUF cause skews and bias in delays, which in turn introduces bias in responses. To eliminate bias in delays, authors in [14] introduced a precision look-up table based programmable delay lines to tune and balance the delays. For implementing PUFs on FPGA, Ring oscillator (RO) PUFs were also proposed in [6]. The major drawback of the RO PUFs is having only a quadratic number of challenges with respect to the number of RO's [12]. Furthermore, the ROs (while in use) consume significant dynamic power due to frequent transitions during oscillations. SRAM PUFs suffer from the same limitation in terms of the number of possible challenge combinations [12].

This paper is a major extension to the work presented in [15]. It presents the first practical method and proof-of-concept FPGA implementation of a PUF with an exponential number of possible challenges of different types including placement challenges. The new proposed PUF uses the unique cell-by-cell characteristics of the FPGA array. In order to provide resilience against variations in environmental conditions, a linear calibration method as well as a differential signature extraction system are presented. The authors in [13] exploited the time-bounded characteristics of generic public key protocol by PUFs but no practical implementation options were discussed.

Besides the ongoing research on PUFs, several other relevant works on delay characterization serve as the enabling thrust for realization of our novel PUF structures. To perform delay characterization, Wong et al. in [16] proposed a built-in self-test mechanism for fast chip level delay characterization. The system utilizes the on-chip PLL and DCM modules for clock generation at discrete frequencies. The delay fingerprint can be used to detect any malicious modification to the original design due to insertion of hardware Trojan horses [17], [18].

In addition, the use of reconfigurability to enhance system security and IP protection has previously been a subject of research. The work in [19] proposes a secure reconfiguration controller (SeReCon) which provides secure runtime management of designs downloaded to the DPR FPGA system and protects the design IP. The work in [20] introduces methods for securing the integrity of FPGA configuration while [21] leverages the capabilities of reconfigurable hardware to provide efficient and flexible architectural support for security standards as well as defenses against hardware attacks. In this paper, we take advantage of the degree of freedom offered by reconfigurable platforms to carry out random placements across different configurations. Instantaneous reverse engineering of the configuration bitstream is a highly complex and challenging task because of the proprietary encoding of configuration bitstream. The paper uses this observation to establish a time-bound on the authentication.

III. flow

The flow of the proposed methodology is presented in Figure 1. First, a one-time process is conducted to extract the delay parameters and characterize the timing behavior of the FPGA components. Next, the extracted chip-dependent unique and physically unclonable characteristics are registered in a publicly accessible database. Each time authentication is needed, the prover in possession of the genuine hardware responds to the challenges posed by the verifier. The verifier is able to predict and simulate the responses to the challenges based on the pre-stored timing characteristics in the database.

Response simulation/emulation by a verifier would be much slower than the prover with genuine hardware. Authentication is performed by comparing the simulated responses to the actual responses within the alloted time frame.

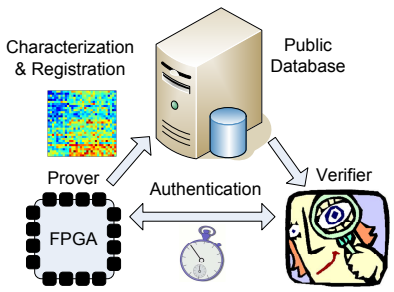


Fig. 1. The flow of the proposed method.

IV. Delay Signature Extraction

To measure the delays of components inside FPGA, we exploit the device reconfigurability to implement a delay signature extraction circuit. A high level view of the delay extraction circuitry is shown in Figure 2. The target circuit/path delay to be extracted is called the *Circuit Under Test (CUT)*. Three flip-flops (FFs) are used in this delay extraction circuit: *launch FF*, *sample FF*, and *Capture FF*. The clock signal is routed to all three FFs as shown on the Figure. Assume for now that the binary challenge input to the CUT is held constant and thus the CUT delay is fixed.

Assuming the FFs in Figure 2 are initialized to zero, a low-to-high signal is sent through the CUT by the launch FF at the rising edge of the clock. The output is sampled T seconds later on falling edge of the clock (T is half the clock period). Notice that the sampling register is clocked at the falling edge of the clock. If the signal arrives at the sample FF before sampling takes place, the correct signal value would be sampled; otherwise, the sampled value would be different and will generate an error. The actual signal value and the sampled value are compared by XOR logic and the result is held for one clock cycle by the capture FF.

A more careful timing analysis of the circuit reveals the relationship between the delay of the CUT (t_{CUT}), the clock pulse width (T), the clock-to-Q delay at the launch FF (t_{clk2Q}), and the clock skew between the launch and sample FFs (t_{skew}). The setup/hold of the sampling register and the setup/hold time of the capture register are denoted by t_{setS}, t_{holdS}, t_{setC}, and t_{holdC} respectively. The propagation delay of the XOR gate is denoted by t_{XOR}. The time it takes for the signal to propagate through CUT and reach the sample

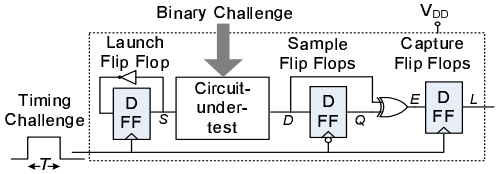


Fig. 2. The timing signature extraction circuit.

flip flop from the moment the launch flip flop is clocked is represented by t_P. Based on the circuit in Figure 2, $t_P = t_{CUT} + t_{clk2Q} - t_{skew}$.

As T approaches t_P, the sample flip flop enters a metastable operation (because of the setup and hold time violations) and its output becomes nondeterministic. The probability that the metastable state resolves to a 0 or 1 is a function of how close T is to t_P. For instance, if T and t_{CUT} are equal, the signal and the clock simultaneously arrive at the sample flip flop and the metastable state resolves to a 1 with a probability of 0.5. If there are no timing errors in the circuit, the following relationships must hold:

$$t_{holdC} < t_P < T - t_{setS} \tag{1}$$

The errors start to appear if t_p enters the following interval:

$$T - t_{setS} < t_P < T + t_{holdS} \tag{2}$$

The rate (probability) of observing timing error increases as t_p gets closer to the upper limit of Inequality 2. If the following condition holds, then timing error happens every clock cycle:

$$T + t_{holdS} < t_P < 2T - (t_{setC} + t_{XOR}) \tag{3}$$

Observability of timing errors follows a periodic behavior. In other words, if t_p goes beyond $2T - (t_{setC} + t_{XOR})$ in Inequality 3, the rate of timing errors begins to decrease again. This time the decrease in the error rate is not due to the proper operation but it is because the timing errors cannot be observed and captured by the capture FF.

Inequality 4 corresponds to the transition from the case where timing error happens every clock cycle (Inequality 3) to the case where no errors can be detected (Inequality 5).

$$2T - (t_{setC} + t_{XOR}) < t_P < 2T + (t_{holdC} - t_{XOR}) \tag{4}$$

$$2T + (t_{holdC} - t_{XOR}) < t_P < 3T - t_{setS} \tag{5}$$

Timing errors no longer stay undetected if t_p is greater than $3T - t_{setS}$. Timing errors begin to appear and can be captured if t_p falls into the following intervals:

$$3T - t_{setupS} < t_p < 3T + t_{holdS} \tag{6}$$

If the following condition holds, then timing error gets detected every clock cycle.

$$3T + t_{holdS} < t_p < 4T - (t_{setC} + t_{XOR}) \tag{7}$$

This periodic behavior continues the same way for integer multiples of T, however it is upper bounded by the maximum clock frequency of the FPGA device. In general, if T is much larger than the XOR and flip flop delays, the intervals can be simplified to $n \times T < t_p < (n+1) \times T$ and timing errors can only be detected for odd values of n.

Notice that in the circuit in Figure 2, high-to-low and low-to-high transitions travel through the CUT every other clock cycle. The propagation delay of these two transitions differ in practice. Suppose that the low-to-high transition propagation delay ($t_p^{l \to h}$) is smaller than the high-to-low transition propagation delay ($t_p^{h \to l}$). Then, for low-to-high transitions, $t_p^{l \to h}$ satisfies Inequalities 1 and for high-to-low transitions, $t_p^{h \to l}$ satisfies Inequality 3. Timing errors in this case happen only

for high-to-low transitions and as a result timing error can only be observed 50% of the times. Thus, the final measurement represents the superposition of both effects.

The top plot in Figure 3 shows the observed/measured probability of timing error as a function of clock pulse width (T). The right most region (R_1) corresponds to the error free region of operation expressed by Inequality 1. Note that the difference between $t_p^{h\to l}$ and $t_p^{l\to h}$ causes the plateau at R_2. The gray regions marked by R_2 and R_4 correspond to the condition expressed by Inequality 2. Region R_5 can be explained by Inequality 3. Metastable regions of R_6 and R_8 relate to inequality 4. Inequality 5 corresponds to the error free region of R_9. Similar to R_3, regions R_7 and R_{11} are due to the difference between high-to-low and low-to-high transition delays. Metastable regions of R_{10} and R_{12} relate to inequality 6 and lastly region R_{13} corresponds Inequality 7.

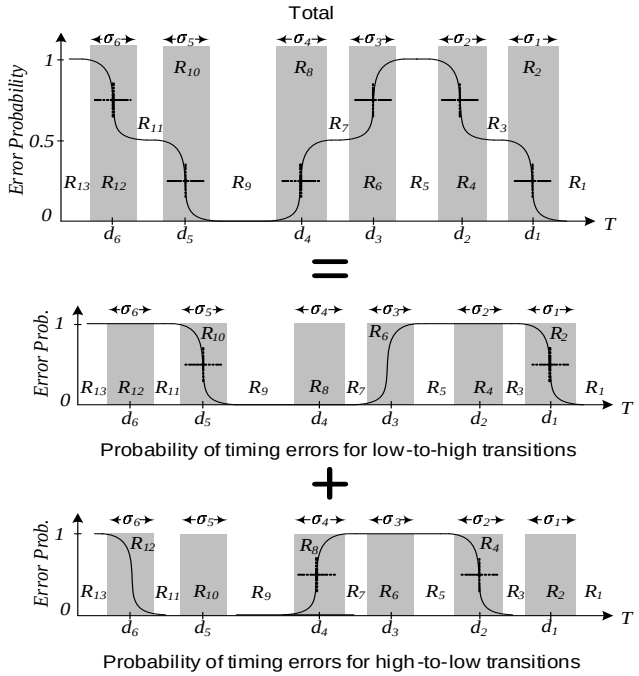


Fig. 3. The probability of observing timing failure as a function of clock pulse width, T.

Notice that similar to t_p, all of the delays defined above for the XOR, flip flops, and clock skew have two distinct values for high-to-low (rising edge) and low-to-high (falling edge) transitions. Nevertheless, all of the inequalities defined in this section hold true for both cases.

We refer to the characterization circuit that includes the CUT as a *characterization cell* or simply a *cell*. Each cell in our implementation is contained in one configurable logic block (CLB). The circuit under test consists of four cascaded look-up tables (LUT) each implementing a variable delay inverter. We explain in Section V how the delay of the inverters can be changed.

A. Signature extraction system

In this subsection, we describe the system that efficiently extracts the probability of observing timing failure as a function of clock pulse width for a group of components on FPGA. The circuit shown in Figure 2 only produces a single bit flag of whether errors happen or not. We require a mechanism to measure the rate or probability at which errors appear at the output of the circuit in Figure 2 to extract the smooth transitions as depicted in Figure 3.

To measure the probability of observing error at a given clock frequency, an error histogram accumulator is implemented by using two counters. The first counter is the error counter whose value increments by unity every time an error takes place. The second counter counts the clock cycles and resets (clears) the error counter every 2^N clock cycles, where N is the size of the binary counters. The value of the error counter is stored in the memory exactly one clock cycle before it is cleared. Now, the stored number of errors normalized to N would yield the error probability value.

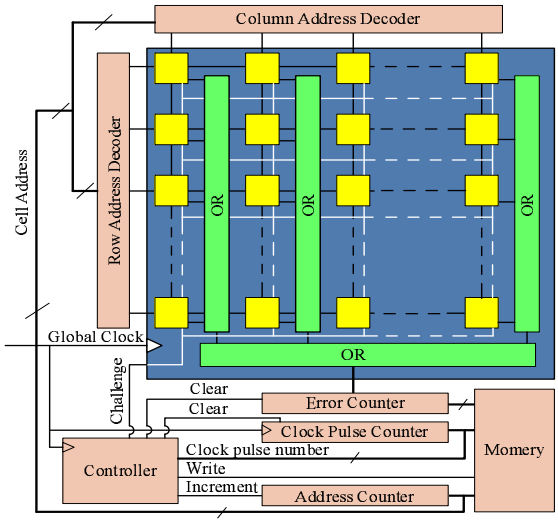


Fig. 4. The architecture for chip level delay extraction of logic components.

The clock frequency to the system is swept linearly and continuously in T_{sweep} seconds from $f_i = \frac{1}{2T_i}$ to $f_t = \frac{1}{2T_t}$, where $T_t < t_p < T_i$. A separate counter counts the number of clock pulses in each frequency sweep. This counter acts as an accurate timer that bookmarks the frequency at which timing errors happen. The value of this counter is retrieved every time the error counter content is written into memory. This action happens every 2^N clock cycles. For further details on clock synthesis see [22].

The system shown in Figure 4 is used for extracting the delays of an array of CUTs on the FPGA. Each square in the array represents the characterization circuit (or *cell*) shown in Figure 2. Any logic configuration can be utilized within the CUT in the characterization circuit. In particular, the logic inside the CUT can be made a function of binary challenges, such that its delay varies by the given inputs. The system in Figure 4 characterizes each cell by sweeping the clock frequency once. Then, it increments the cell address and moves to the next cell. The cells are characterized in serial. The row and column decoders activate the given cell while the rest of the cells are deactivated. Therefore, the output of the deactivated cells remain zero and the output of the OR function solely reflect the timing errors captured in the activated cell.

Each time the data is written to the memory, three values are stored: the cell address, the accumulated error value, and the clock pulse number at which the error has occurred. The clock counter is then for each new sweep. The whole operation iterates over different binary challenges to the cells. Note that the scanning can also be performed in parallel to reduce the characterization time [22].

B. Characterization accuracy

The timing resolution, i.e., the accuracy of the measured delays, is a function of the following factors: (i) the clock jitter and noise, (ii) the number of frequency sample points, and (iii) the number of pulse samples at each frequency. Recall that the output of the characterization circuit is a binary zero/one value. By resending multiple clock pulses of the same width to the circuit and summing up the number of ones at the output, a real-valued output can be obtained. The obtained value represents the rate (or the probability when normalized) at which the timing errors happen for the input clock pulse width. Equivalently, it represents a sample point on the curve shown in Figure 3. The more we repeat the input clock pulse, the higher sample resolution/accuracy can be achieved along Y axis. Now suppose that the clock pulse of width T is sent to the PUF for M times. Due to clock jitter and phase noise, the characterization circuit receives a clock pulse of width $T_{eff} = T + T_j$, where T_j is additive jitter noise. Let us assume T_j is a random variable with zero mean and a symmetric distribution. Since the output probability is a smooth and continuous function of T_{eff}, estimating the probability by averaging will be an asymptotically unbiased estimator as $M \to \infty$. Finally, the minimum measurable delay is a function of the maximum speed at which the FFs can be driven (maximum clock frequency). When performing a linear frequency sweep, a longer sweep increases (ii) and (iii) and thus the accuracy of the characterization. A complete discussion on characterization time and accuracy for this method is presented in [22].

C. Parameter extraction

So far, we have described the system that measures the probability of observing timing errors for different clock pulse widths. The error probability can be represented compactly by a set of few parameters. These parameters are directly related to the circuit component delays and flip flop setup and hold time. It can be shown that the probability of timing error can be expressed as the sum of shifted Gaussian CDFs [7]. The Gaussian nature of the error probabilities can be explained by the central limit theorem. Equation 8 shows the parameterized error probability function.

$$f_{\mathbf{D},\Sigma}(t) = 1 + 0.5 \sum_{i=1}^{|\Sigma|-1} -1^{\lceil i/2 \rceil} \left[Q(\frac{t - d_i}{\sigma_i}) \right] \qquad (8)$$

where $Q(x)=\frac{1}{\sqrt{2\pi}} \int_x^\infty \exp\left(-\frac{u^2}{2}\right)$ and $d_{i+1} > d_i$. To estimate the timing parameters, f is fit to the set of measured data points (t_i,e_i), where e_i is the error value recorded when the pulse width equals t_i.

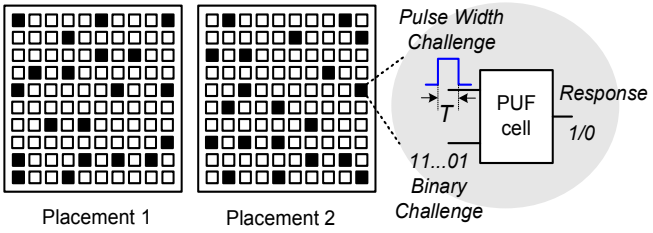


Fig. 5. Two random placement of PUF cells on FPGA.

V. Timing PUF

To enable authentication, a mechanism for applying challenge inputs to the device and observing the evoked responses is required. In this section, we present a PUF circuit based on the delay characterization circuit shown in Figure 2. The response is a function of the clock pulse width T, the delay of circuit under test, t_{CUT}, and flip flop characteristics, σ_i. In the following, we discuss three different ways to challenge the PUF.

A. Pulse challenge

One way to challenge the PUF is to change the clock pulse width. The clock pulse width can be considered as an analog input challenge to the circuit in Figure 2. The response to a given clock pulse of width T is either 0 or 1 with the probability given by Equation 8 or the plot in Figure 3.

However, the use of clock pulse width as challenge has a number of implications. First, the response from the PUF will be predictable if T is either too high and too low compared to the nominal circuit under test delay t_{CUT}. Predictability of responses makes it easy for the attacker to impersonate the PUF without knowledge of the exact value of t_{CUT}. As another example, suppose that the response to multiple clock pulses of the same width, T_1, are equal to '0'; then, the attacker can deduce that T_1 is in either region R_1 or R_9 in Figure 3 with high confidence. If the nominal boundaries of these regions (R_1,...,R_{13}) are known, the attacker can determine which region T_1 belongs by just comparing it to the boundaries $T_{R_i} < T_1 < T_{R_{i+1}}$. Knowing the correct region, it becomes much easier to predict the response to the given pulse width, especially for odd regions $R_1, R_3, ..., R_{13}$.

Within the thirteen regions shown in Figure 3, the six regions that include transitions produce the least predictable responses. Setting the challenge clock pulse width to the statistical median of the center points of transitions in Figure 3 would maximize the entropy of the PUF output responses. In other words, there are only six independent pulse widths that can be used for challenges and the results for other pulse widths are highly predictable. As it can be seen, the space of possible independent challenges for this type of challenge is relatively small.

Another limitation of pulse challenges is that depending on the available clocking resources, generating many clock pulses with specific widths can be costly. Under such limitations, the verifier may prefer to stick to a fixed pulse width. In the next sections, we look into other alternatives to challenge the PUF.

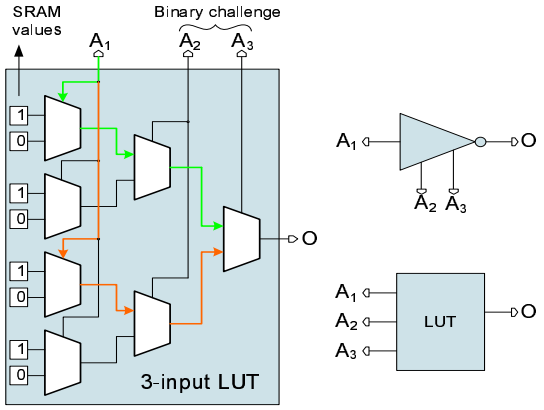


Fig. 6. The internal structure of LUTs. The signal propagation path inside the LUTs change as the inputs change.

B. Binary challenge

An alternative method to challenge the PUF is to change the t_{CUT} while the clock pulse width is fixed. So far, we assumed that the delay of CUT is not changing. To change t_{CUT}, one must devise an input vector to the circuit-under-test that changes its effective input/output delay by altering the signal propagation path inside the CUT. In other words, the binary input challenge vector alters the CUT delay by changing its internal signal propagation path length, hence affecting the response.

In this work, we introduce a low overhead method to alter the CUT delay by tweaking the LUT internal signal proportion path. We implement the CUT by a set of LUTs each implementing an inverter function. Figure 6 shows the internal circuit structure of an example 3-input LUT. In general, a Q-input LUT consists of 2^Q-1 2-input MUXs which allow selection of 2^Q values stored in SRAM cells. The SRAM cell values are configured to implement a pre-specified functionality.

In this example, the SRAM cell values are configured to implement an inverter. The LUT output is only the function of A_1, i.e., $O = f(A_1)$, disregarding values on A_2 and A_3. However, changing the inputs A_2 and A_3 can alter the delay of the inverter due to the modifications in the signal propagation paths inside the LUT. For instance, two internal propagation path for the values of $A_2A_3 = 00$ and $A_2A_3 = 11$ are highlighted in Figure 6. As it can be seen, the path length for the latter case is longer than the former, yielding a larger effective delay. The LUTs in Xilinx Virtex 5 FPGAs consist of 6 inputs. Five inputs of the LUT can be used to control and alter the inverter delay resulting in 2^5 = 32 distinct delays for each LUTs. Finally, note that the delays for each binary input must be measured prior to authentication. The response to the PUF is then predicted by the verifier based on the configured delay and the input clock pulse width.

C. Placement challenge

Another important type of challenge which can be implemented solely on reconfigurable platforms is the placement challenge. This type of challenge is enabled by the degree of freedom in placing the PUF cells on FPGA in each configuration. During characterization, a complete database of all CUT delays across the FPGA is gathered. At the time of authentication, only a subset of these possible locations within the FPGA array are selected to implement and hold the PUF cells. The placement challenge is equivalent to choosing and querying a subset of PUF cells, where the selection input is embedded in the configuration bitstream.

Figure 5 shows two random placements of 20 PUF cells across the FPGA array. Each black square in the figure contains a PUF cell which receives a pulse and binary challenge. The high degree of freedom in placement of PUF cells across the FPGA results in a huge challenge/response space. In our implementation, each PUF cell can be fit into a CLB on FPGA. With *N* CLBs on FPGA, there will be $\binom{N}{k}$ different ways to place *k* PUF cells on FPGA. The smallest Xilinx Virtex 5 FPGA (LX30) has 2400 CLBs which enables $\binom{2400}{512}$ number of possibilities to place 512 PUF cells on the FPGA.

VI. Response Robustness

Although PUF responses are functions of chip-dependent process variations and input challenges, they can also be affected by variations in operational conditions such as temperature and supply voltage. In this section, we discuss two techniques to provide calibration and compensation to make responses resilient against variations in operational conditions.

The first method takes advantage of on-chip sensors to perform linear calibration of the input clock pulse width challenge, while the second method uses a differential structure to cancel out the fluctuations in operational conditions and extract signatures that are less sensitive to variations in operational conditions. We will discuss the advantages and disadvantages of each method. The existing body of research typically addresses this issue mainly through the use of error correction techniques [11] and fuzzy extractors [23]. The error correction techniques used for this purpose rely on a syndrome which is a public piece of information being sent to the PUF system along with the challenge. The response from the PUF and the syndrome are input to the ECC to produce the correct output response. The methods discussed in this section help reduce the amount of errors in responses and they can be used along with many other error correction techniques.

A. Linear Calibration

The extracted delay signatures at characterization phase are subject to changes due to aging of silicon devices, variations in the operating temperature, and supply voltage of the FPGA. Such variations can undermine the reliability of the authentication process. The proposed method performs calibration on clock pulse width according to the current operating conditions. Fortunately, many modern FPGAs are equipped with built-in temperature and core voltage sensors. Before authentication begins, the prover is required to send to the verifier the readings from the temperature and core voltage sensors. The prover, then based on the current operating conditions, adjusts and calibrates the clock frequency. The presented calibration method linearly adjusts the pulse width

using the Equations 9 and 10.

$$T_{calib} = \alpha_{tmp} \times (tmp_{cur} - tmp_{ref}) + T_{ref} \quad (9)$$

$$T_{calib} = \alpha_{vdd} \times (vdd_{cur} - vdd_{ref}) + T_{ref} \quad (10)$$

tmp_{ref} and vdd_{ref} are the reference temperature and FPGA core voltage measured during the characterization phase. tmp_{cur} and vdd_{cur} represent the current operating conditions. The responses from the PUF to the clock pulse width T_{calib} are then treated as if T_{ref} were sent to the PUF at reference operating condition. The calibration coefficients α_{tmp} and α_{vdd} are device specific. These coefficients can be determined by testing and characterizing each single FPGA at different temperatures and supply voltages. For example, if $d_i^{tmp_1}$ and $d_i^{tmp_2}$ are i-th extracted delay parameter under operating temperatures tmp_1 and tmp_2, then

$$\alpha_{tmp,i} = \frac{d_i^{tmp_1} - d_i^{tmp_2}}{tmp_1 - tmp_2}, \alpha_{vdd,i} = \frac{d_i^{vdd_1} - d_i^{vdd_2}}{vdd_1 - vdd_2} \quad (11)$$

Note that for each delay parameter on each chip, two calibration coefficients can be defined (one for temperature and one for voltage supply effect) and the clock pulse width can be calibrated accordingly. Ideally, with the help of a more sophisticated prediction model (potentially a nonlinear model) trained on a larger number of temperature and voltage supply points (instead of two points as in Equation 11), highly accurate calibration can be performed on the clock frequency. In reality, due to limitations on test time and resources, it is impractical to perform such tests for each FPGA device. Instead, calibration coefficients can be derived from a group of sample devices and a universal coefficient can be defined for all devices by averaging the coefficients. In Section IX, we demonstrate reliability of authentication for universal calibration coefficients. Note that in Equations 9 and 10, we assume that only one type of operational condition variation is happening at a time and both temperature and voltage supply do not fluctuate simultaneously. However, if we consider these effects independently, we can superimpose the effects by applying Equation 9 to the output of Equation 10. A more general approach would be to consider a 2D nonlinear transformation given by:

$$T_{calib} = f(vdd_{cur}, tmp_{cur}, T_{ref}) \quad (12)$$

The main disadvantage of calibration methods is the time and effort required to characterize the delay at various operational conditions. Hence, more effort spent on building and training the regression model, the more accurate calibration and a higher robustness in responses can be achieved.

B. Differential Structure

In this section, we introduce a differential PUF structure, that compensates for the common mode variation induced by the impact of fluctuations in operational conditions on the delays. The goal of the method is to extract a signature that is invariant to fluctuations in operational conditions.

The PUF introduced previously receives a clock pulse and a binary challenge to produce a binary response. Here, instead of looking at the output responses from a single PUF cell,

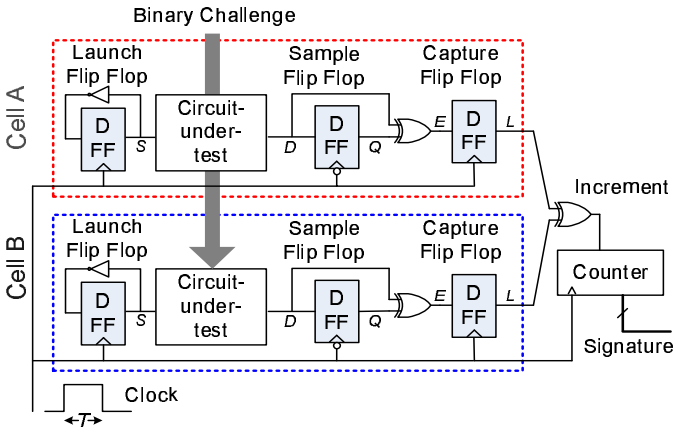


Fig. 7. The differential signature extraction system.

we consider the difference of the responses from two adjacent PUF cells. More specifically, the outputs of the capture flip flops from the two cells drive an XOR logic. Assuming i_1 and i_2 are the inputs and O is the output of the XOR logic, then the probability of output being equal to '1' , ρ_O, as a function of the probability of inputs being equal to '1', ρ_1 and ρ_2, can be written as:

$$\rho_O = \rho_1 + \rho_1 - 2 \times \rho_1 \times \rho_2 \quad (13)$$

ρ_1 and ρ_2 are functions of the clock pulse width (T) and the binary challenge as explained in Section IV. The resulting output probability is shown in Figure 8 (see the red dashed line) for two sample PUF cells under (a) normal operating condition and (b) low operating temperature of -10^oC. As it can be seen, since both PUF cell delay parameters are shifted together under the same operational conditions, the resulting XOR output probability retains the shape, with only a scalar shift along the x axis. To extract robust signatures, one needs to look into shift invariant features that are less sensitive to environmental variables. Features such as the high/low region widths of the resulting XOR probability plot, or the total area under the XOR output probability plot can be used for this purpose. In this work, we use the area under the XOR

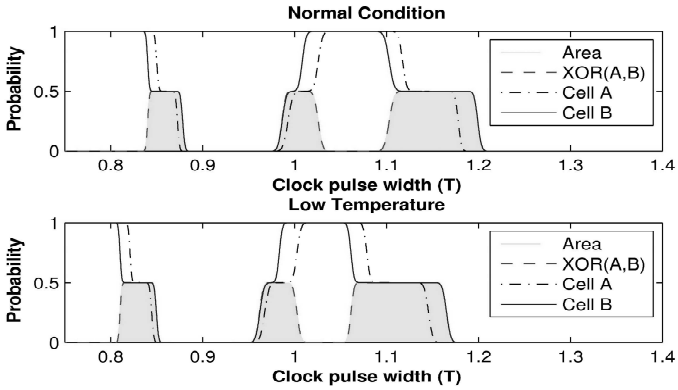


Fig. 8. The timing error probability for two sample PUF cells and the resulting XOR output probability under (a) normal operating condition and (b) low operating temperature of -10^oC.

output probability curve. The area is shaded in Figure 8 for

the two operating conditions. The area under the curve can be calculated by integrating the probability curve from the lowest to highest clock pulse width. We use the Riemann sum method to approximate the total area underneath the XOR probability curve in hardware. The result of the integration is a resilient real valued signature extracted from the PUF cell pairs.

In order to find a quick approximation to this integral in hardware, we sweep the input clock frequency linearly from frequency $f_l = 1/2T_u$ to $f_u = 1/2T_l$ where $T_l \ll D_{min}$, $T_u \gg D_{max}$, D_{min} and D_{max} represent lowest and highest bounds on delay parameters under all operational conditions. In other words, the sweep window must always completely contain all parts of the curve. The output of the XOR is connected to a counter as shown in Figure 7. The aggregate counter value after a complete sweep is a function of the area under the curve. Please note that this value is not exactly equal to the area under the curve and is only proportional to the integral. Also, a longer sweep time results in a larger number of clock pulses and thus more accurate approximation of the signature. This is analogous to using a larger number of narrower subintervals when approximating the area under curve with the Riemann sum to achieve a smaller approximation error.

Although the generated responses are less sensitive to variations in operational conditions, it should be noted that the responses are a function of the difference in the timing characteristics of the two PUF cells. The area under the curve loses a lot of information about the shape of the curve and also some information is lost on each individual probability curve through the difference operation. Therefore, the responses have a lower entropy compared to the linear calibration method. To obtain the same amount of information, more PUF cell pairs must be challenged and scanned. Another limitation of this structure is the length of the input challenge. To estimate the area under the curve with a high accuracy, the whole interval from the lowest to the highest frequency must be swept in fine steps and thus, it would require more clock pulses compared to the other method. Using few clock pulses leads to a larger area estimation error, lower probability of detection, and higher probability of false alarm. Finally, the pairing of the PUF cells introduces another degree of freedom to the system where a set of challenges can specify pairing of the PUF cells.

VII. Authentication Protocol

In this section, we show how the extracted cell characteristics in Section IV can be utilized for FPGA authentication. The following terminology is used in the remainder of the paper. The *verifier* (V) authenticates the *prover* (P) who owns the genuine FPGA device. The verifier authenticates the device by verifying the unique timing properties of the device.

A. Classic Authentication

The registration and authentication processes for the classic authentication case are demonstrated in the diagram in Figure 9(a) and (b) (disregard the darker boxes for now). The minimum required assumptions for this case are: (i) the verifier is not constrained in power, (ii) it is physically impossible to clone the FPGA, and (iii) the characteristics of the FPGA owned by the prover is a secret only known to the prover and verifier.

As shown in Figure 9(a), during the registration phase, the verifier extracts and securely stores the cell delay parameters by performing characterization as explained in Sections IV. By knowing the FPGA-specific features in addition to the structure and placement of the configured PUF, the verifier is able to predict the responses to any challenges to the PUF. After registrations, the FPGA along with the pertinent PUF configuration bitstream is passed to the end-user.

At the authentication, end-user (prover) is queried by the verifier to make sure she is the true owner of the FPGA. Classic authentication is shown in Figure 9(b). To authenticate the ownership, the verifier utilizes a random seed and generates a set of pseudorandom challenge vectors for querying the prover. The prover responds to the challenges she receives from the verifier by applying them to the configured FPGA hardware. The verifier then compares the received responses from the prover with the predicted ones, and authenticates the chip if the responses are similar.

To ensure robustness against errors in measuring the delays and the changes in operational conditions, the registration entity may also compute the error correction information for the responses to the given challenges. To prevent information leakage via the error correction bits, secure sketch techniques can be used. A secure sketch produces public information about its input that does not reveal the input, and still permits exact recovery of the input given another value that is close to it [24].

The device is authenticated if the response after error correction would be mapped to the verifier-computed hash of responses. Otherwise, the authentication will fail. Alternatively, the verifier can allow for some level of errors in the collected responses and remove the error correction and hashing from the protocol. However, accepting some errors in the responses makes the verifier be more susceptible to emulation/impersonating attacks [2], [25].

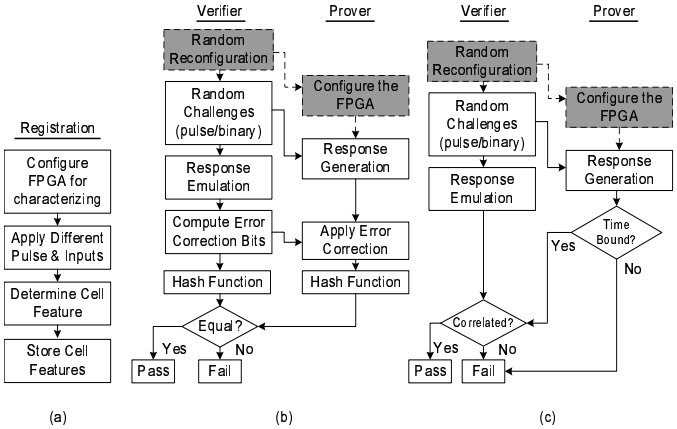


Fig. 9. (a) FPGA registration (b) Classic authentication flow (c) Time-bound authentication flow.

B. Time-bounded Authentication Using Reconfigurability

After the FPGA registration, the verifier is able to compute and predict the responses to any set of challenges by knowing (i) the cell-level features of the pertinent FPGA, (ii) the circuit structure, and (iii) placement of the PUF circuit. The information on the PUF circuit structure and placement is embedded into the configuration bitstream. In the classic authentic method, the bitstream is never changed. A dishonest prover, off-line and given enough time and resources can (i) extract the cell-level delays of the FPGA, and (ii) reverse engineer the bitstream to discover the PUF structure and its placement on the FPGA. During the authentication, the dishonest prover can compute the responses to the given challenges online by simulating the behavior of the PUF on the fly and producing the responses that pass the authentication.

A stronger set of security protocols can be built upon the fact that the prover is the only entity who can compute the correct response to a random challenge within a specific time bound since he has access to the actual hardware. In this protocol, prior to the beginning of the authentication session, the FPGA is blank. The verifier then sends a bitstream to the device in which a random subset of LUTs are configured for authentication. After the device is configured, the verifier starts querying the FPGA with random challenges. The verifier accepts the responses that are returned back only if $\Delta t \leq \Delta t_{\max}$ where Δt is the time elapsed on the prover device to compute the responses after receiving the configuration bitstream, and $\Delta t_{\max}$ is the upper bound delay estimated computation of responses by the authentic FPGA prover device, which is composed of device configuration, response generation, error correction, and hashing time all performed in hardware.

The verifier will authenticate the device only if the time the device takes to generate the response is less than $\Delta t_{\max}$. We denote the minimum emulation time by $t_{\min}^{\text{emu}}$, where $t_{\min}^{\text{emu}} >> \Delta t_{\max}$. Time-bounded authentication protocol can be added to the authentication flow, as demonstrated in Figure 9(c). Compared to the classic authentication flow, a time bound check is added after the hash function. While performing the above authentication, we emphasize on the assumption that the time gap between hardware response generation and simulation (or emulations) of the prover must be larger than the variation in the channel latency. The time-bound assumption would be enough for providing the authentication proof [7], [13], [26].

1) Estimating the time-bound: Now let us look at Δt, the time elapsed on the prover device to compute the responses. Before proceeding, note that the characterization is a one-time offline operation which happens prior to authentication phase and its time complexity does not affect the time-bound discussed here. Δt is the sum of time required to configure the FPGA, T_{conf}, and the time spent on evaluating the PUF, T_{eval}, i.e., $\Delta t = T_{conf} + T_{eval}$. During the PUF evaluation, N_p clock pulses at N_f distinct frequencies are sent to N_{cell} PUF cells in serial with an average pulse width of T_{avg}, therefore the average evaluation time is, $T_{eval} = N_p \times N_{cell} \times N_f \times T_{avg}$. For instance, in our experiments, $N_p = 8$, $N_p = 6$, $N_{cell} = 1024$, and $T_{avg} = 2ns$, yielding $T_{eval} \simeq 98\mu$seconds.

Configuration time varies for different configuration schemes and depends on the configuration file size, configuration port width, and frequency of the driving clock. Configuration time can roughly be estimated by $T_{conf} = \frac{L_b}{f_c \times P_w}$, where L_b is the configuration bitstream length in bits, f_c is the clock frequency in Hz, and P_w is the configuration port width in bits. For example, In our experiment on Xilinx Virtex 5 FPGAs (LX110), L_b=3.5MB, f_c=50MHz and P_w=16bit, the configuration time equals 350 milli-seconds. Faster clocks can expedite the configuration process.

VIII. Attacks and Countermeasures

Perhaps the most dangerous attack to an authentication system is impersonation attack. Impersonation attack aims at deceiving the verifier to get through the authentication by reverse-engineering and simulation of the authentic device behavior, or storing and replaying the communication, or random guessing. Storage and reply attacks are impractical as long as the verifier uses a new random challenge every time. Random guessing and prediction attacks pose a threat if the responses have a low entropy and are predictable. As we mentioned in Section V, by setting the input clock pulse widths to the statistical median of the center of transition regions, the entropy of the responses can be maximized. For a fixed binary challenge, there are not more than six independent input clock pulse widths to be tried. In other words, the responses to other input clock pulse widths would lack sufficient entropy. To obtain more response bits, more binary challenges must be used instead.

Among the aforementioned threats, the reverse engineering and simulation attacks are the most critical attacks to address. The time-bounded protocol discussed in Section VII is constructed based on secrecy in placement of the PUF and the connection of the input challenges to the CUTs. The secret expires within the given time bound. To provide the correct response to a new challenge, the adversary has to reverse engineer the bitstream to decipher the placement and connection of the input challenges to the PUF. Next, he has to simulate (or emulate) the PUF behavior using the public timing characterization. These two steps must be performed within the given time constraint. Even after many years of research in rapid simulation technologies for hardware design and validation, accurate simulation or emulation of a hardware architecture is extremely slow compared to the real device. In addition, even though bitstream reverse-engineering have partially been performed on some FPGAs [27], performing it would require a lot of simulations and pattern matching. Thus, it would take many more cycles than the authentic hardware where the verifying time is dominated by the bitstream configuration time (in the order of 100 mili seconds).

IX. Experimental evaluations

In this section, the implementation details of the signature extraction system are presented. We demonstrate results obtained by measurements performed on Xilinx FPGAs and further use the platform to carry out authentication on the available population of FPGAs. For delay signature extraction,

the system shown in Figure 4 is implemented on Xilinx Virtex 5 FPGAs. The system contains a 32× 32 array of characterization circuits as demonstrated in Figure 2. The CUT inside the characterization circuit consists of 4 inverters each being implemented using one 6-input LUT. The first LUT input (A_1) is used as the input of the inverter and the rest of the LUT inputs (A_2,...,A_6) serve as the binary challenges which alter the effective delay of the inverter. The characterization circuit is pushed into 2 slices (one CLB) on the FPGA. In fact, this is the lower bound on the characterization circuit hardware area. The reason is that the interconnects inside the FPGA force all the flip flops within the same slice to operate either on rising edge or falling edge of the clock. Since the launch and sample flip-flops must operate on different clock edges, they cannot be placed inside the same slice. In total, 8 LUTs and 4 flip flops are used (within two slices) to implement the characterization circuit. The error counter size (N) is set to 8. To save storage space, the accumulated error values are stored only if they are between 7 and 248.

We use an ordinary desktop function generator to sweep the clock frequency from 8MHz to 20MHz and afterwards shift the frequency up 34 times using the PLLs inside the FPGA. The sweeping time is set to 1 milli seconds (due to the limitations of the function generator, a lower sweeping time could not be reached). The measured accumulated error values are stored on an external memory and the data is transferred to a PC for further processing. Notice that the storage operation can easily be performed without the logic analyzer by using any off-chip memory.

The system is implemented on twelve Xilinx Virtex 5 XC5VLX110 chips and the measurements are taken under different input challenges and operating conditions. The characterization system in total uses 2048 slices for the characterization circuit array and 100 slices for the control circuit out of 17,280 slices.

The measured samples for each cell are processed and the twelve parameters as defined in Section IV-C are extracted. Figure 10 shows the measured probability of timing error versus the clock pulse width for a single cell and a fixed challenge. The (red) circles represent original measured sample points and the (green) dots show the reconstructed samples. As explained earlier, to reduce the stored data size, error samples with values of 0 and 1 (after normalization) are not written to the memory and later are reconstructed from the rest of the sample points. The solid line shows the Gaussian fit on the data as expressed in Equation 8.

Parameter extraction procedure is repeated for all cells and challenges. Figure 11 shows the extracted parameters d_1 and σ_1 for all cells on chips #9 and #10 while the binary challenge is fixed. The pixels in the images correspond to the cells within the 32×32 array on FPGA. Some levels of spatial correlation among d_1 parameters can be observed on the FPGA fabric.

The boxplots in Figure 12(a) show the distribution of the delay parameters d_i for i=1,2,...,6 over all 12 chips and 1024 cells and 2 challenges. The central mark on the boxplot denotes the median, the edges of the boxes correspond to the 25th and 75th percentiles, the whiskers extent to the most

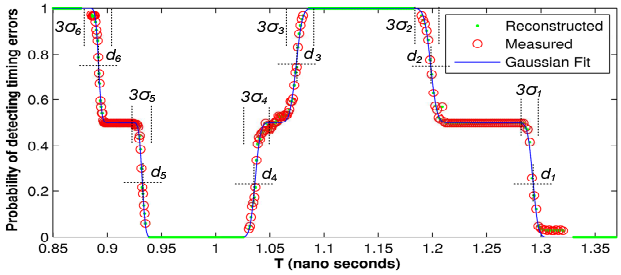


Fig. 10. The probability of detecting timing errors versus the input clock pulse width T. The solid line shows the Gaussian fit to the measurement data.

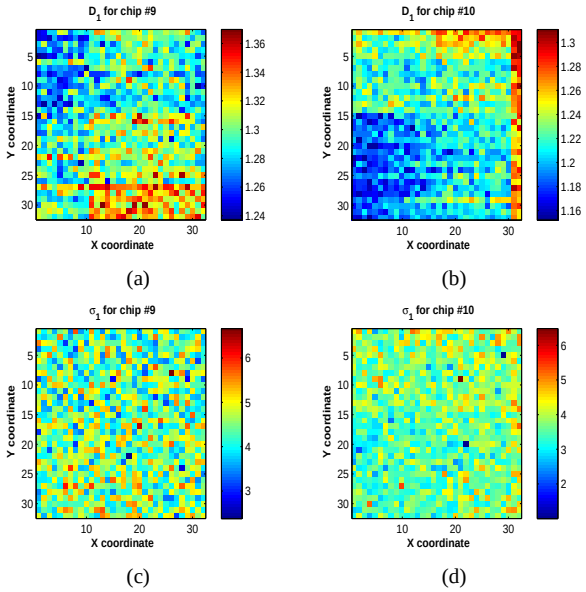


Fig. 11. The extracted parameters d_1 (a,b) and σ_1 (c,d) for chips 9 and 10.

extreme data points and the plus signs show the outlier points.

Using the measured data from the twelve chips, we investigate different authentication scenarios. The authentication parameters substantially increase the degree of freedom in challenging the PUF. These parameters include the number of clock pulses to send to the PUF (N_p), the number of binary challenges to apply to the PUF (N_c), the challenge clock pulse width (T), and the number of PUF cells (N_{cell}) to be queried. In other words, in each round of authentication, N_c challenges are applied to N_{cell} PUF cells on the chip and then N_p pulses of width T are sent to to these PUF cells. The response to each challenge consists of N_p bits. For ease of demonstration, the response can be regarded as the percentage of ones in the N_p response bits, i.e., an integer between 0 and N_p.

To quantify the authentication performance, we study the effect of N_{cell} and T on the probability of detection (p_d) and false alarm (p_f). Detection error occurs in cases where the test and target chips are the same, but due to instability and noise in responses, they fail to be authenticated as the same. On the other hand, false alarm corresponds to the cases where the test and target chips are different, but they are identified as the same chips. During this experiment, the binary challenges to PUF cells are fixed and the number clock pulses is set to $N_p = 8$. The clock width (T) is set to each of the medians

of the values shown in Figure 12 (a). Setting the clock pulse width to the median values results in least predictability of responses. All N_{cell}=1024 PUF cells are queried. The same experiment is repeated for 10 times to obtain 10 response vectors (each vector is $N_p = 8$ bits) for each chip. Therefore, each clock pulse generates 8×1024 bits of responses from every chip. After that, the distance between the responses from the same chips (intra-chip distance) over repeated evaluations is measured using the normalized L_1 distance metric. The distance between responses from different chips (inter-chip distance) is also measured. If the distance between the test chip and the target chip responses is smaller than a pre-specified detection threshold, then the chip is successfully authenticated. In the experiments, the detection threshold is set at 0.15.

Table I shows the probability of detection and false alarm for different clock pulse widths and number of queried PUF cells. To calculate the probabilities, the distance between the response of every distinct pair of FPGAs are calculated. The number of pairs with a response distance of less than 0.15, normalized to the total number of pairs yield the probability of false alarm. To find the probability of detection, the distance between the responses from the same chip acquired at different times are compared to 0.15. The percentage stay within the threshold determine the probability of detection. As it can be observed, the information extracted from even the smallest set of cells is sufficient to reliably authenticate the FPGA chip if the pulse width is correctly set.

(a)

N_{cell}	Challenge Pulse Width					
	1.23	1.15	1.06	1.03	0.9	0.87
64	0.96	0	0	0	0	1.52
128	2.04	0	0	0	0	1.52
256	4.55	0	0	0	0	1.52

(b)

N_{cell}	Challenge Pulse Width					
	1.23	1.15	1.06	1.03	0.9	0.87
64	93.3	96.2	100	100	100	100
128	94.2	98.8	100	100	100	100
256	99.85	100	100	100	100	100

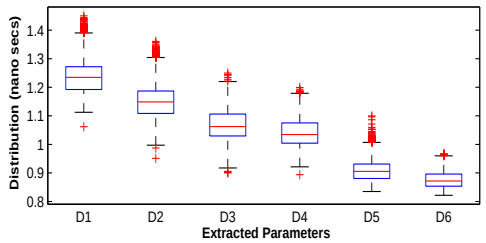
TABLE I
(A) PROBABILITY OF FALSE ALARM (B) PROBABILITY OF DETECTION.

In the next experiment, we study the effect of fluctuations in the operating conditions (temperature and core supply voltage) on the probabilities of detection and false alarm. Moreover, we demonstrate how linear calibration of the challenge clock pulse width can improve the reliability of detection. To calculate the calibration coefficient defined by Equation 11, we repeat the delay extraction process and find the delay parameters for all twelve chips at temperature -10^oC and core voltage 0.9 Volts. The chip operates at the temperature 37^oC and core voltage of 1 volts in the normal (reference) condition. We use the built-in sensors and the Xilinx Chip Scope Pro package to monitor the operating temperature and core voltage. To cool down the FPGAs, liquid compressed air is consistently sprayed over the FPGA surface. Figure 12 (b) depicts the changes in the distribution of the first delay parameter (d_1) at the three different operating conditions.

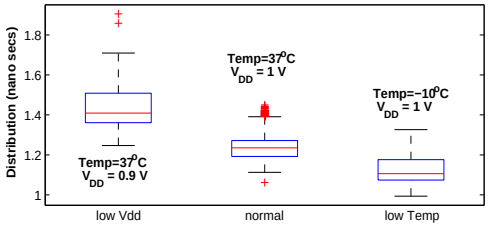
The probabilities of detection and false alarm are derived before and after performing calibration on the challenge pulse width for different clock pulse widths and number of binary challenges to the cells. In this experiment, all 1024 PUF cells on the FPGA are queried for the response. N_p =8 as before. As it can be seen in Table II, the detection probabilities are significantly improved after performing linear calibration based on the coefficients extracted for each chip. The variables v_{low} and t_{low} correspond to -10^oC temperature and 0.9 supply voltages respectively. The reported probabilities of Table II are all in percentage. Also note that for the challenge pulse width of $T = 0.87\ ns$, the probability of detection reaches 100% and probability of alarm falls to zero after calibration. The same holds true for $N_c = 2$ and $T = 0.87, 0.9, 0.95$. Thus, increased level of reliability can be achieved during authentication with proper choice of pulse width and number of challenges.

Figure 13 shows how performing calibration decreases the intra-chip response distances in presence of temperature changes. The histogram corresponds to $T = 0.95ns$ and $N_c = 2$ in Table II before and after calibration.

Next, we examine the differential signature extraction system presented in Section VI-B. To extract the signature, the base frequency is swept from 8 to 20 MHz in a linear fashion in 1 mili second and shifted up 34 times using the FPGA internal PLLs. The sweep is repeated for the 512 pairs of PUF cells producing a real-valued signature vector of size 512. A large number of pulses ($\sim 10^7$) are generated in a complete sweep. The signature as explained in Section VI-B is the accumulation of the timing errors over a complete sweep. To achieve an accurate approximation of the area under the curve, a large number of clock pulses must be tried. This is the main disadvantage of this method compared to the singled ended method. To extract the shift invariant parameters such the



(a)



(b)

Fig. 12. (a) Distribution of delay parameters d_i. (b) The distribution of d_1 for normal, low operating temperature, and low core voltage.

		No Calibration								Calibrated							
		N_C=1				N_C=2				N_C=1				N_C=2			
		v_{low}		t_{low}		v_{low}		t_{low}		v_{low}		t_{low}		v_{low}		t_{low}	
		p_d	p_f	p_d	p_f	p_d	p_f	p_d	p_f	p_d	p_f	p_d	p_f	p_d	p_f	p_d	p_f
T	1.23	18.4	0	33.3	16.7	18.4	0	33.3	22.29	100	0	75	0	100	0	75	0
	1.06	18.4	0	18.4	0	18.4	0	18.4	0	50	0	50	0	57.3	0	50	0
	1.01	18.4	0	16.7	0	18.4	0	16.7	0	66.6	0	75	0	68.2	0	75	0
	0.95	18.4	0	16.7	0	18.4	0	16.7	0	66.7	0	100	0	84.9	0	100	0
	0.9	16.7	0	25	0	16.7	0	25	0	83.3	0	91.7	0	83.4	0	100	0
	0.87	25	0	25	0	25	1.5	25	0	100	0	100	0	100	0	100	0

TABLE II
THE PROBABILITY OF DETECTION AND FALSE ALARM BEFORE AND AFTER PERFORMING CALIBRATION ON THE CHALLENGE PULSE WIDTH IN PRESENCE OF VARIATIONS IN TEMPERATURE AND CORE VOLTAGE.

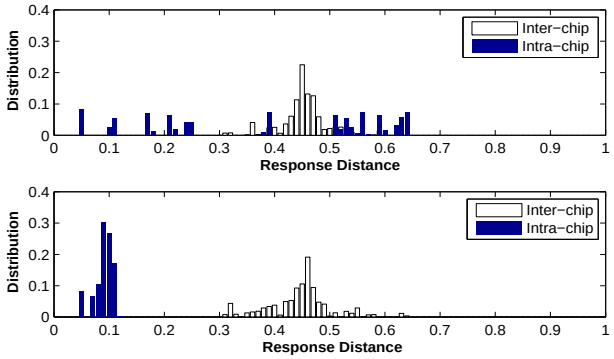


Fig. 13. The inter-chip and intra-chip response distances for $T = 0.95\ ns$ and $N_c = 2$ before (top) and after (bottom) calibration against changes in temperature.

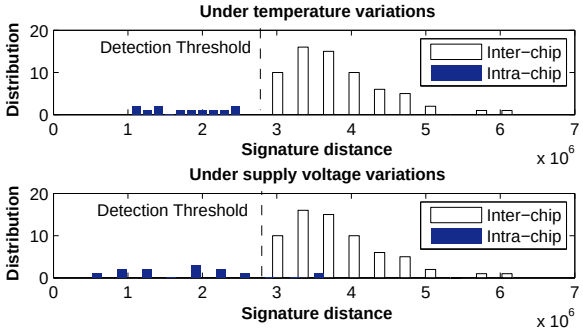


Fig. 14. The distribution of the intra- and inter-chip signature L_1 distances

region width and/or area under the probability curve probing the PUF circuit at single frequency points will not yield sufficient information. Therefore, a complete sweep covering the regions with high information content is needed. The L_1 distance of the signatures from the same chip under different operational conditions (intra-chip distance) and the distance of the signatures from different chips (inter-chip distance) are calculated. Figure 14 shows the distribution of intra and inter-chip distance of signatures under variations in temperature and supply voltage for the twelve Virtex 5 chips. As it is shown in the figure, the distance among signatures obtained at room temperature and -10^oC temperature from the same chip is always smaller than those from different chips, resulting in 100% probability of detection and 0% false alarm probability. However, with 10% variations in voltage supply, the intra- and inter-chip distributions overlap slightly.

X. CONCLUSIONS

We presented a technique for FPGA authentication that takes advantage of the unclonable timing variability present in FPGAs, the reconfigurability feature, and its unprecedented speed. Authentication is composed of two phases; namely registration and authentication. During registration, cell level timing features are extracted and stored in a database. Later at the authentication phase, the verifier generates a random configuration bitstream and sends it to the prover. A unique aspect of the new method is its high degree of freedom in placing the PUF cells and selection of challenges. The protocol relies on the fact that online reverse-engineering of the bitstream is a non-trivial task. A new calibration method for improving robustness to temperature and voltage fluctuations was demonstrated. Evaluations on Xilinx V5 FPGA show the effectiveness and practicality of the new timing signature extraction and authentication method.

XI. ACKNOWLEDGEMENT

This research is in part supported by the Office of Naval Research (ONR) grant under grant No. R16480 and National Scient Foundation Career Grant under grant No. 0939766, and Semiconductor Research Center (SRC) under the grant No. 1836.039. The authors would like to thank Ahmed Elnably for assistance in data collection process.

REFERENCES

[1] R. Pappu, B. Recht, J. Taylor, and N. Gershenfeld, "Physical one-way functions," *Science*, vol. 297, pp. 2026–2030, 2002.
[2] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, "Silicon physical random functions," in *Conference on Computer and Communications Security (CCS)*, 2002, pp. 148–160.
[3] G. Suh, C. O'Donnell, and S. Devadas, "AEGIS: A single-chip secure processor," *IEEE Design & Test of Computers*, vol. 24, no. 6, pp. 570–580, 2007.
[4] M. Atallah, E. Bryant, J. Korb, and J. Rice, "Binding software to specific native hardware in a VM environment: the PUF challenge and opportunity," in *Workshop on Virtual Machine Security*, 2008, pp. 45–48.
[5] 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, pp. 63–80.
[6] G. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *Design Automation Conference (DAC)*, 2007, pp. 9–14.

[7] M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Techniques for design and implementation of secure reconfigurable PUFs," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 2, no. 1, pp. 1–33, 2009.

[8] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, "Controlled physical random functions," in *Annual Computer Security Applications Conference (ACSAC)*, 2002, pp. 149–160.

[9] J. Lee, L. Daihyun, B. Gassend, G. Suh, M. van Dijk, and S. Devadas, "A technique to build a secret key in integrated circuits for identification and authentication applications," in *Symp. of VLSI*, 2004, pp. 176–179.

[10] D. Holcomb, W. Burleson, and K. Fu, "Power-up SRAM state as an identifying fingerprint and source of true random numbers," *IEEE Trans. on Computers*, 2009.

[11] B. Gassend, "Physical random functions," S.M. Thesis, MIT, 2003.

[12] U. Ruhrmair, J. Solter, and F. Sehnke, "On the foundations of physical unclonable functions," *Cryptology ePrint Archive*, 2009.

[13] U. Ruhrmair, "SIMPL system: on a public key variant of physical unclonable function," *Cryptology ePrint Archive*, 2009.

[14] M. Majzoobi, F. Koushanfar, and S. Devadas, "FPGA PUF using programmable delay lines," in *IEEE Workshop on Information Forensics and Security*, 2010, p. in press.

[15] M. Majzoobi, A. Elnably, and F. Koushanfar, "FPGA time-bounded unclonable authentication," in *Information Hiding Conference (IH)*, vol. 6387, 2010, pp. 1–16.

[16] J. S. J. Wong, P. Sedcole, and P. Y. K. Cheung, "Self-measurement of combinatorial circuit delays in FPGAs," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 2, no. 2, pp. 1–22, 2009.

[17] Y. Jin and Y. Makris, "Hardware trojan detection using path delay fingerprint," in *International Symposium on Hardware-Oriented Security and Trust (HOST)*, 2008, pp. 51–57.

[18] D. Rai and J. Lach, "Performance of delay-based trojan detection techniques under parameter variations," in *International Symposium on Hardware-Oriented Security and Trust (HOST)*, 2009, pp. 58–65.

[19] K. Kepa, F. Morgan, K. Kosciuszkiewicz, and T. Surmacz, "Serecon: A secure dynamic partial reconfiguration controller," *IEEE Computer Society Annual Symposium on VLSI*, vol. 0, pp. 292–297, 2008.

[20] J. B. Webb and J. B. Webb, "Methods for securing the integrity of FPGA configurations," Tech. Rep., 2006.

[21] G. Gogniat, T. Wolf, W. Burleson, J.-P. Diguet, L. Bossuet, and R. Vaslin, "Reconfigurable hardware for high-security/ high-performance embedded systems: The safes perspective," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 16, no. 2, pp. 144 –155, 2008.

[22] M. Majzoobi, E. Dyer, A. Elnably, and F. Koushanfar, "Rapid FPGA delay characterization using clock synthesis and sparse sampling," in *International Test Conference (ITC)*, 2010, p. in press.

[23] C. Bösche, J. G. A. Sadeghi, J. Shokrollahi, and P. Tuyls, "Efficient helper data key extractor on FPGAs," in *Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, 2008, pp. 181–197.

[24] Y. Dodis, L. Reyzin, and A. Smith, "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data," in *EUROCRYPT*, 2004, pp. 523–540.

[25] M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Testing techniques for hardware security," in *International Test Conference (ITC)*, 2008, pp. 1–10.

[26] N. Beckmann and M. Potkonjak, "Hardware-based public-key cryptography with public physically unclonable functions," in *Information Hiding Conference (IH)*, 2009, pp. 206–220.

[27] J. Note and E. Rannaud, "From the bitstream to the netlist," in *International Symposium on Field-Programmable Gate Arrays (FPGA)*, 2008, pp. 264–274.

Mehrdad Majzoobi received his B.Sc. and M.Sc. degree in Electrical and Computer Engineering from University of Tehran, Iran in 2006 and Rice University, Texas in 2009 respectively. He is currently a Ph.D. candidate in the Electrical and Computer Engineering Department at Rice University. His research interests include statistical modeling and optimization, low power embedded systems, sensor-based and reconfigurable systems, VLSI testing, hardware security, and hardware/software intellectual property (IP) protection and watermarking.

Farinaz Koushanfar received the Ph.D. degree in electrical engineering and computer science from the University of California at Berkeley in 2005. She also received the M.A. degree in statistics from the University of California, Berkeley, the M.Sc. degree in electrical engineering from the University of California, Los Angeles, and the B.A.Sc. degree in electrical engineering from the Sharif University of Technology, Iran. She is currently an Assistant Professor in the Department of Electrical and Computer Engineering at Rice University where she is also the director of the Texas Instruments (TI) DSP Leadership University program. Prior to joining Rice University in July 2006, she held the Coordinate Science Lab (CSL) fellowship at the University of Illinois, Urbana Champaign. Her research interests include data integrity, statistical modeling and optimization, embedded systems, sensor-based and reconfigurable systems, hardware security, and hardware/software intellectual property (IP) protection. Dr. Koushanfar is a recipient of the Presidential Early Career Awards for Scientists and Engineers (PECASE), the Office of Naval Research (ONR) Young Investigator Program Award (2009), MIT Technology Review's Top 35 Young Innovators Award (2008), the Defense Advanced Research Projects Agency (DARPA) Young Faculty Award (2007), the National Science Foundation (NSF) CAREER Award (2007), an Intel Open Collaborative Research Fellowship, and a Best Paper Award at Mobicom.