Proceedings of the 2014 Federated Conference on Computer Science and Information Systems pp. 911–918
DOI: 10.15439/2014F122
ACSIS, Vol. 2

Security Evaluation of Bistable Ring PUFs on FPGAs using Differential and Linear Analysis

Dai Yamamoto*†, Masahiko Takenaka
* Fujitsu Laboratories Ltd.
Kanagawa, Japan
{yamamoto.dai, ma}@jp.fujitsu.com

Kazuo Sakiyama
† The University of Electro-Communications
Tokyo, Japan
sakiyama@uec.ac.jp

Naoya Torii
Fujitsu Laboratories Ltd.
Kanagawa, Japan
torii.naoya@jp.fujitsu.com

***Abstract*—Physically Unclonable Function (PUF) is expected to be an innovation for anti-counterfeiting devices for secure ID generation, authentication, etc. In this paper, we propose novel methods of evaluating the difficulty of predicting PUF responses (i.e. PUF outputs), inspired by well-known differential and linear cryptanalysis. According to the proposed methods, we perform a first third-party evaluation for Bistable Ring PUF (BR-PUF), proposed in 2011. The BR-PUFs have been claimed that they have a resistance against the response predictions. Through our experiments using FPGAs, we demonstrate, however, that BR-PUFs have two types of correlations between challenges and responses, which may cause the easy prediction of PUF responses. First, the same responses are frequently generated for two challenges (i.e. PUF inputs) with small Hamming distance. A number of randomly-generated challenges and their variants with Hamming distance of one generate the same responses with the probability of 0.88, much larger than 0.5 in ideal PUFs. Second, particular bits of challenges in BR-PUFs have a great impact on the responses. The value of responses becomes '1' with the high probability of 0.71 ($>$ 0.5) when just particular 5 bits of 64-bit random challenges are forced to be zero or one. In conclusion, the proposed evaluation methods reveal that BR-PUFs on FPGAs have some correlations of challenge-response pairs, which helps an attacker to predict the responses.**

I. Introduction

RECENTLY, the concept of Internet of things (IoT) has been widely spread. Various things such as vehicles, home appliances, medical devices and sensing devices are connected to the Internet. It is expected that this provides us a lot of new services and products in the field of industry, education, healthcare, agriculture, etc. First of all, the secure IoT requires us to realize an authentication system for things. This is because counterfeiting the things causes serious security problems for the services and products based on the IoT concept. Generally, hardware-based approaches are often used for the authentication of things. For example, cryptographic hardware using integrated circuits (ICs) stores a secret key in its internal memory. A secure cryptographic protocol using the cryptographic hardware enables us to authenticate the things, making the secret key invisible from the outside. This approach prevents a leakage of the key outside, and makes it impossible to counterfeit things. However, recent research has found that the secret key could be revealed by de-packaging the IC and analyzing the IC mask design [2]. Therefore, further techniques are necessary to protect the cryptographic hardware storing the secret keys.

Recently, Physically Unclonable Functions (PUFs) have been focused as a solution to the secure authentication for things [3]. PUFs are realized in individual IC chips, and have a completely identical circuit structure. In spite of the identical circuit, PUFs generate the unique output values (responses) to the same input value (challenge) for each individual IC. This uniqueness is provided by process variations in memory characteristics or wire/gate delay occurring in the manufacturing process of each IC chip [4] [5]. Even if an attacker de-packages and analyzes ICs of PUFs, she cannot analyze the process variations due to the identical PUF structure. As a result, she cannot reveal the challenge-response pairs of PUFs. Therefore, PUFs can be utilized for a secure authentication system for things.

There are two categories in the PUFs on ICs: *memory-based* PUFs utilizing the memory characteristics and *delay-based* PUFs utilizing wire/gate delay variations [6]. One of the most feasible and secure memory-based PUFs is latch-based PUFs (LPUFs) [7] [8]. The LPUF generates an N-bit response based on N outputs from N RS latches. The RS latch is composed of cross-coupled logic gates, and is similar to a memory cell. Each bit of the response is generated from each latch output in a stable state after a metastable state. The metastable state is affected by the memory characteristics, thus the latch outputs (i.e. response bits) are also unique for each individual IC. One of the most famous delay-based PUFs is Ring Oscillator PUFs (RO-PUFs) [9]. The RO-PUF has M number of ROs, one of which is composed of odd number of cascaded inverters as a ring. The RO-PUF derives 1-bit responses from the difference of oscillator frequencies between two arbitrary ROs. Consequently, 1-bit response becomes zero or one, depending on which RO has a larger frequency. The number of responses is ${}_M\mathrm{C}_2$, which corresponds to the number of combinations of M ROs taken 2 at a time. The oscillator frequencies are affected by the wire/gate delays, which makes the responses unique for each individual IC.

Bistable Ring PUFs (BR-PUFs), having both properties of memory-based and delay-based PUFs, were proposed and self-evaluated by Chen et al. (hereinafter called "developers") [10] [11]. There are two major differences between BR-PUFs and

The preliminary version of this paper was presented in a Japanese domestic symposium without peer review [1].

978-83-60810-58-3/$25.00

RO-PUFs: (1) the structure of a ring, (2) response generation. (1) A BR-PUF is composed of cascaded inverters as a ring (hereinafter called "primitive BR-PUF"). A primitive BR-PUF is similar to an RO-PUF in terms of the ring of cascaded inverters. The difference is that the number of the inverters is not odd but even. Hence the primitive BR-PUF does not keep oscillation, but make the transition from metastable to stable state like memory-based PUFs. The primitive BR-PUF derives a 1-bit response from which stable state the ring is, e.g. '10101010' or '01010101' in a ring of 8 inverters. (2) A primitive BR-PUF generates just one 1-bit response because it consists of one ring, while an RO-PUF includes multiple rings. To generate multiple 1-bit responses, the BR-PUF has a *basic component* instead of the inverters in the ring. The basic component consists of two logic gates, either of which is selected by 1 bit of challenge [10]. The BR-PUF with 64 basic components, for example, has 64-bit challenge to select the logic gates. The BR-PUF is organized by 2^{64} different types of rings, the logic gates of which are differently selected depending on the values of challenges. Therefore, the BR-PUF can generate multiple challenge-response pairs without having multiple rings like the RO-PUF.

In this paper, we evaluate security aspects of BR-PUFs implemented on FPGAs. The reason why we focus on this PUF is that BR-PUFs have the following advantages with both memory-based and delay-based PUFs, as claimed by the developers:

- BR-PUFs are similar to delay-based PUFs in that the number of challenge-response pairs is exponential to the bit length of challenges, which makes difficult the predictions of responses.
- BR-PUFs also have the resistance against a machine learning attack and a modeling attack like memory-based PUFs.

BR-PUFs are evaluated by developers themselves. These self-evaluation results are very useful for users to understand the effectiveness of the proposed PUFs. However, the evaluation results may be different depending on PUF implementations since PUFs are based on physical characteristics in ICs. Hence it is quite important to evaluate and analyze newly proposed PUFs by third-party researchers as attackers. These BR-PUFs with excellent characteristics have not been evaluated by other researchers yet.

A. Our Contributions

In order to evaluate the security of PUFs, we focus on the difficulty of predicting responses. This difficulty is one of requirements for PUFs. Consequently, responses for unknown challenges should be unpredictable and non-biased even when some challenge-response pairs are known. In this paper, we propose novel two methods of evaluating this difficulty:

Our Evaluation Method (i):
> What is the probability that PUFs generate the same responses for two challenges with small Hamming distance?

Our Evaluation Method (ii):
> What is the probability that PUFs generate the same responses for multiple challenges whose particular bits are forced to be zero or one?

These methods are inspired by well-known cryptanalysis methods: differential cryptanalysis [12] and linear cryptanalysis [13]. The Evaluation Method (i) focuses on how differences in the challenge (plaintext) lead to differences in the response (ciphertext). The Evaluation Method (ii) is based on the idea that the response (ciphertext) is linearly approximated by particular bits of the challenge (plaintext). If the probabilities in these methods are close to 0.5, the evaluated PUFs are highly secure because they can generate non-biased responses independently of the values of challenges.

In this paper, we evaluate the security of BR-PUFs on Xilinx FPGAs (Spartan-6) according to our two evaluation methods. We analyze a number of challenge-response pairs obtained from BR-PUFs consisting of 64 inverters implemented on FPGAs. As a result, our case study supports that BR-PUFs on FPGAs have undesirable performance in the differential and linear evaluations; the probability is far from ideal 0.5. The differential evalaution shows that two types of rings for the challenges with small Hamming distance have many common logic gates (i.e. similar circuit characteristics), which are likely to generate the same response. The linear evaluation implies that BR-PUFs have some special inverter gates which have a great impact on the responses.

This paper is the first time that BR-PUFs on FPGAs have some security issues of response predictions. More importantly, our two evaluation methods can be used as universal methods for evaluating the security of other types of PUFs.

B. Organization of the Paper

The rest of the paper is organized as follows. Section II gives an outline of the BR-PUF. Section III proposes two evaluation methods of the difficulty of predicting responses. Section IV evaluates the BR-PUFs implemented on an FPGA platform according to the evaluation methods. Finally, in Section V we summarize our work and comment on future directions.

II. Bistable Ring PUF

The BR-PUF was proposed by Chen et.al. in 2011 [10]. Figure 1 shows the basic mechanism of the BR-PUF. The BR-PUF consists of the even (e.g. eight in Fig. 1) number of inverters (INVs), which are connected as a ring. After voltage is supplied, the ring has two possible stable states, '10101010' ('A'-state) or '01010101' ('5'-state), enumerating inverter's outputs beginning from INV_1. The ring generates 1-bit response according to which state the ring falls into. The BR-PUF is similar to the delay-based RO-PUF in terms of having inverter rings. It also has the same characteristic with the memory-based Latch-PUF, having two possible states.

Figure 2 shows the circuit structure of the BR-PUF, presented in [10]. The inverter in Fig. 1 is implemented by a *BR-S*, which is a basic component of the BR-PUF. The l-th BR-S, i.e. BR-S_l ($1 \leq l \leq 64$), is composed of two NOR gates,

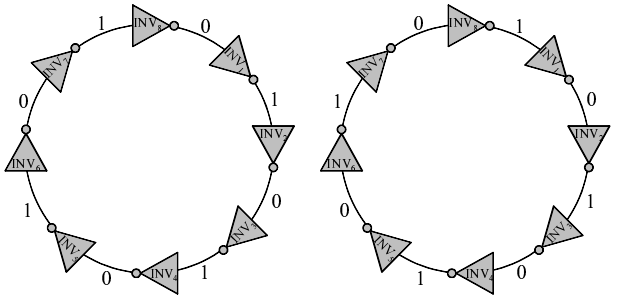


Fig. 1. Two possible stable states on a primitive BR-PUF with 8 inverters.

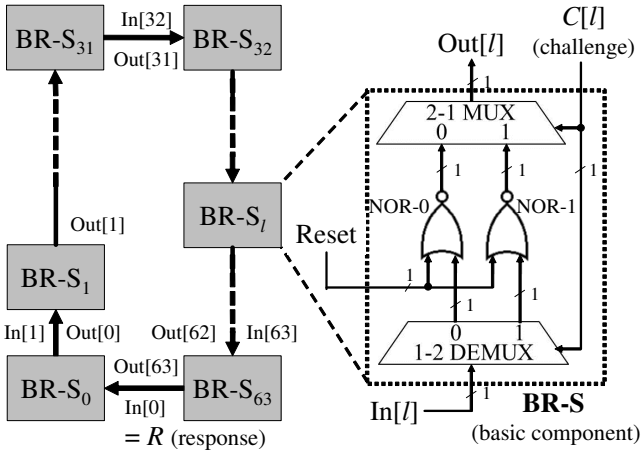


Fig. 2. Circuit structure of Bistable Ring PUF.

a 2-to-1 MUX and a 1-to-2 DEMUX. A 1-bit challenge $C[l]$ is input to the BR-S$_l$ to select either of the NOR gates. The BR-PUF with 64 BR-Ss has 64-bit challenges, which means 2^{64} different types of rings can be organized. Each NOR gate has different characteristics, i.e. drive capability or gate/wire delay. Hence the value of challenges has a great impact on the decision of stable states, either A-state or 5-state, as claimed by the developers. A 1-bit response is extracted from an arbitrary signal between two BR-Ss, e.g. the output from BR-S$_{63}$, i.e. Out[63] (=In[0]) in Fig. 2. The BR-S$_l$ works as an inverter when reset signal equals to 0. In contrast, the input and output of BR-S$_l$, In[l] and Out[l], can be forced to zero when the reset signal is 1 (i.e. neither A-state nor 5-state). This enables us to generate responses at any time after power up. In conclusion, BR-PUF has multi-bit challenges and generates a number of challenge-response pairs at any time.

III. Proposed Evaluation Methods

It is well known that responses of some delay-based PUFs are predictable through a machine learning attack [14]. The developers of BR-PUFs claim that BR-PUFs have a resistance against such an attack [10]. This resistance is based on the complex and non-linear behavior of BR-PUFs, different from other delay-based PUFs. Hence they claim that an attacker cannot predict responses of BR-PUFs. For the verification of this resistance, we consider that the correlation among challenge-response pairs should be evaluated experimentally. This evaluation, unfortunately, has not been performed by other researchers yet.

In this paper, we propose novel two methods of evaluating PUFs in terms of the resistance against response predictions. In the proposed method (i), we evaluate whether or not challenges with small Hamming distance result in highly correlated responses. In the proposed method (ii), we evaluate whether or not we obtain the same responses with high probability if certain bits of challenges are forced to zero or one. In the following, we explain these methods, assuming the case of evaluating BR-PUFs.

A. Proposed Method (i): Differential PUF Analysis

A group of challenges with small Hamming distance may cause the problem that most of NOR gates are selected commonly, so the characteristics impacting on the responses are also similar one another. In detail, let R_j's be the responses obtained from 64-bit challenges C_j's. Here, let $\tilde{R}_j^{(1,i)}$ be the response obtained from $\tilde{C}_j^{(1,i)} (1 \leq i \leq 64)$ whose Hamming distance is one from C_j (i.e. the only i-th bit from least significant bit (LSB) is different). In ideal PUFs, $\tilde{R}_j^{(1,i)}$ and R_j have little correlation. If a correlation exists, $\tilde{R}_j^{(1,i)}$ has a possibility to be easily predicted by an attacker when challenge-response pairs (C_j, R_j) are known. This means that the implemented BR-PUFs have a serious security issue.

The proposed method (i) is inspired by the well-known cryptanalysis method: differential cryptanalysis [12]. The differential cryptanalysis evaluates the avalanche effect: the effects of the changes of plaintext bits on ciphertext bits. In the proposed method (i), we evaluate how differences in the challenge (plaintext) lead to differences in the response.

B. Proposed Method (ii): Linear PUF Analysis

We consider that some logic gates and wires may be quite different from many other ones. This is because of the process variations in the circuit characteristics such as drive capability or gate/wire delay. If such gates and wires exist in a ring of the BR-PUF, the stable state falls into either state with high probability. As a result, the number of independent challenge-response pairs is very small, which is a security problem for PUFs.

This method is inspired by linear cryptanalysis [13]. In this cryptanalysis, an attacker tries to find linear equations with plaintext bits and ciphertext bits which have a high bias. This provides us with the inspiration of the general method of evaluating PUFs in terms of response predictions.

From the view point of designers of PUFs, they must design a secure PUF whose responses cannot be predicted by an attacker. Of course, machine learning attacks can evaluate a tolerance against the response predictions. However, a concrete method of designing such a secure PUF has not been established yet. Therefore, in this paper, the proposed

evaluation methods provide fundamental principles to design secure PUFs, in terms of correlations between challenges and responses. Next section experimentally evaluates the BR-PUFs implemented on FPGAs according to our proposed methods.

IV. Experimental Evaluation

A. Experimental Setup

Figure 3 shows our experimental system, which consists of two boards: a custom-made board with a Xilinx Spartan-6 FPGA (XC6SLX16-2CSG324C) and a commercially-available Spartan-3E starter kit board with a Xilinx Spartan-3E FPGA (XC3S500E-4FG320C). We implemented the BR-PUF circuit with 64 BR-Ss on the Spartan-6 FPGA, and the peripheral circuits such as the block RAM and RS232C module on the Spartan-3E FPGA. An Spartan-6 FPGA chip was put on a socket of the custom-made board, being therefore easily replaceable by another chip. We evaluated 4 BR-PUFs implemented on 4 Spartan-6 FPGA chips: $\mathrm{FPGA}_x (1 \leq x \leq 4)$.

Our response acquisition process was as follows. When the RS232C module in the Spartan-3 FPGA received a start command from a user PC, the module sent a start signal to a CTRL module. The CTRL module got a 64-bit linear feedback shift register (LFSR) to generate 2,048 random challenges $C_j (1 \leq j \leq 2,048)$. According to [15], the tap sequence of the LFSR was set to [64, 63, 61, 60], and the initial value was set to '0x123456789ABCDEF0'. The 64-bit challenge was divided into four 16-bit values, which were sent and stored to the flip-flops (FFs) on Spartan-6 FPGA. The reset signal to the BR-PUF was changed from 1 to 0, then the response acquisition was started. Not only 1-bit output but also all of 64-bit output from BR-Ss was stored into the 64-bit flip-flop. This enables us to confirm whether or not the response is stable; if the 64-bit value has at least two consecutive 1's/0's, the response is regarded as unstable state, vice versa. In our experiment, the 64-bit value was stored after sufficient time (i.e. approximately 6 ms) from the reset signal changing to 0 in order to make the response as stable as possible. The 64-bit value was sent to a block RAM on the Spartan-3E bit-sequentially, and was transmitted to the user PC through an RS232C port.

Both design and implementation of the BR-PUF are very important because they have a large impact on the eventual response behavior of the PUF itself. Hence we take great care of the symmetric layout of the BR-PUF as follows. Figure 4 shows our custom layout of a BR-PUF with 64 BR-Ss on a Spartan-6 FPGA. The 64 BR-Ss were implemented on the ring-shaped neighboring CLBs (configurable logic blocks), expecting that the wire lengths between all BR-Ss are identical. This symmetric layout is expected to make a uniform ring and a bias of responses as small as possible.

Before we perform an experimental evaluation, we verify the implemented BR-PUFs according to the responses R_j's for the 2,048 random challenges C_j's. Average Hamming distance between two arbitrary 64-bit challenges among the 2,048 challenges is 32.00. This is extremely close to theoretical value $(= 64/2)$, so our using challenges are enough random. By

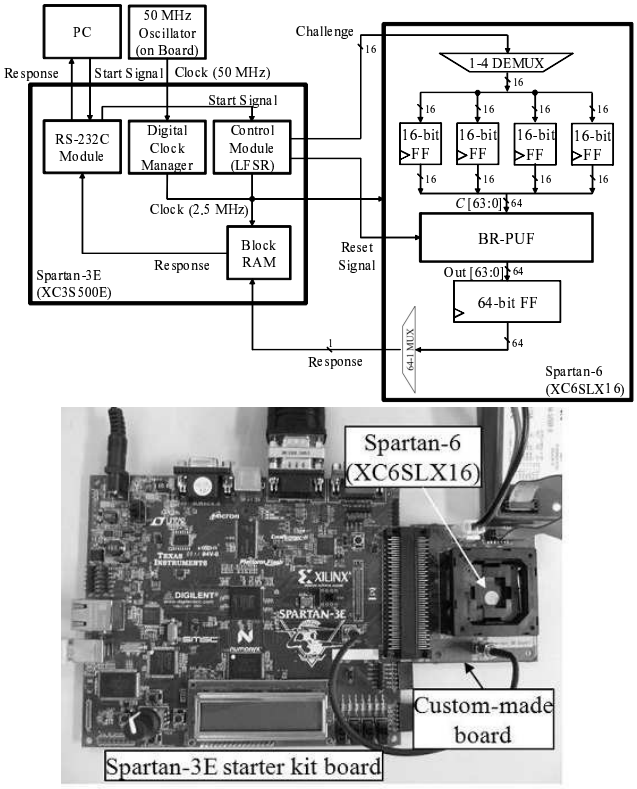


Fig. 3. Experimental system.

using these challenges, we evaluate average hamming distance between two arbitrary responses among the 2,048 responses (i.e. ${}_{2048}\mathrm{C}_2$ combinations). The results are 0.50, 0.49, 0.49 and 0.46 in four BR-PUFs, respectively. These are very close to the ideal value (= 0.5), so our implemented BR-PUFs are verified to generate almost non-biased responses for random challenges.

B. Experimental Results - using Proposed Method (i)

This section evaluates the correlation among the responses obtained from challenges with small Hamming distance. We generate a certain number of challenges $\tilde{C}_j^{(k,i)}$ satisfying the following condition: the Hamming distance between C_j and $\tilde{C}_j^{(k,i)}$ being equal to k, i.e. HD(C_j, $\tilde{C}_j^{(k,i)}$) $= k$. For example, in the case for $k = 1$, we generate 64 challenges $\tilde{C}_j^{(1,i)} (1 \leq i \leq 64)$, where i-th LSB is just different. In our experiment, we evaluate the responses in $k = 1, 2, 4, 8$ and 16. In the case where $k > 1$, however, the number of challenges $\tilde{C}_j^{(k,i)}$ is ${}_{64}\mathrm{C}_k$, which becomes quite large for the value of large k. Due to time constraints, we generate the following two types of challenges $\tilde{C}_j^{(k,i)}$:

Type A

Neighboring k bits are different between C_j and $\tilde{C}_j^{(k,i)}$ as shown in Fig. 5(I).

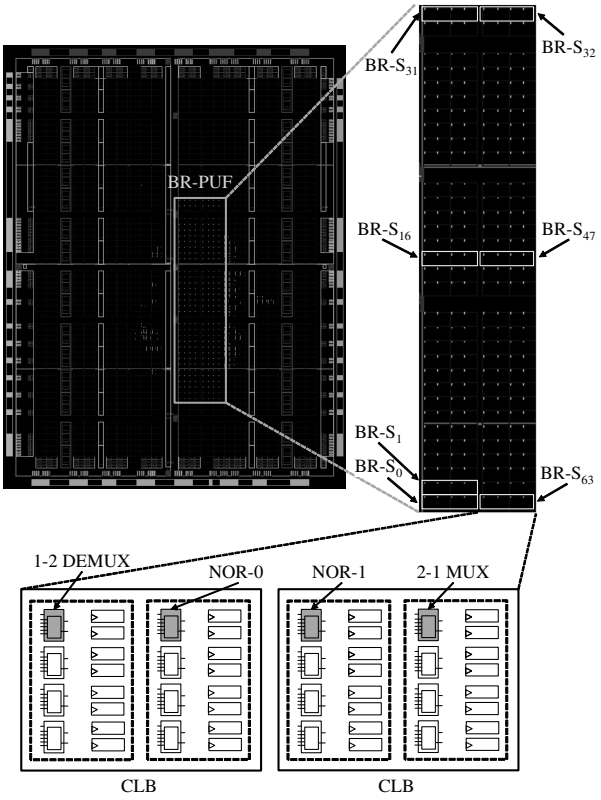
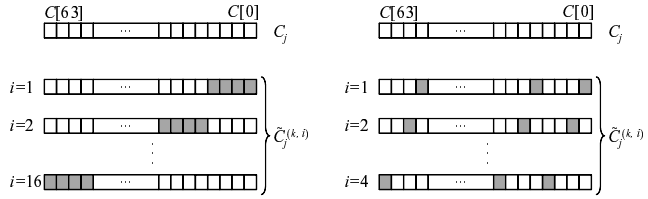


Fig. 4. Implementation of our BR-PUF with 64 BR-Ss on a Spartan-6 FPGA.

Type B
: Intervals of $64/k$ bits are different as shown in Fig. 5(Ⅱ).

Table I shows the number of $\tilde{C}_j^{(k,i)}$ in each k according to the aforementioned types. We generate 184 ($= 124 + 60$) challenges $\tilde{C}_j^{(k,i)}$ for each of 2,048 C_j's. Hence we obtain the total of 378,880 ($= 2,048 \times 185$) challenge-response pairs from each BR-PUF.

Figure 6 shows the ratios of the challenges $\tilde{C}_j^{(k,i)}$ which generate the same responses as each C_j. These results are the means of 4 implemented BR-PUFs. From the result of Type A in $k = 1$, 88.0% of challenges $\tilde{C}_j^{(1,i)}$ lead to the same responses as C_j. This ratio should be around 50% in secure PUFs. The larger the value of k is, the lower the ratios of such challenges are. However, even in the Type A of $\tilde{C}_j^{(16,i)}$ where HD(C_j, $\tilde{C}_j^{(16,i)}$)=16, the probability is approximately 0.665, which is larger than ideal 0.5. Additionally, there is almost no difference between both types in Fig. 6. This indicates that the similarity of responses depends not on the locations of the different bits, but just on the Hamming distance of the challenges. Consequently, if a challenge-response pair is known to an attacker, she has a high possibility to predict the responses for challenges with small Hamming distances by using the known challenge.



(I) Type A ($k = 4$) (Ⅱ) Type B ($k = 16$)

Fig. 5. Two types of challenges $\tilde{C}_j^{(k,i)}$ (Colored bits are different between C_j and $\tilde{C}_j^{(k,i)}$).

TABLE I
NUMBER OF CHALLENGES FOR k IN BOTH TYPES.

k	Type A	Type B
1	64	N/A
2	32	32
4	16	16
8	8	8
16	4	4
Sum	124	60

Different from other PUFs, the generation time of responses, i.e. the duration period for stable states, is quite different depending on values of challenges in BR-PUFs [10]. The generation time has a strong impact on the *reliability* and *uniqueness* of the responses, systematically defined as PUF performance metrics in [16]. Especially, the responses obtained in a short transient time have little uniqueness[1]: a small difference among BR-PUFs because circuit layout influences the responses strongly. Hence we should select and use the only responses with long transient time, as presented in [10]. In the above-mentioned evaluation we focus on all of challenge-response pairs without consideration of the transient time. We anticipate that highly-unique responses with the long transient time have a lower similarity, even if the challenges have a small Hamming distance. To confirm this we obtain the 64-bit outputs of BR-Ss, i.e. responses for 2,048 C_j's, in a short time of approximately 70μs after the reset signal to the BR-PUF is zero. 1,658 (approximately 80.96%) out of 2,048 C_j's lead to stable responses with alternate bits. Here, we focus only on the remaining of 390 C_j's and perform the same evaluation as above mentioned, whose results are shown in Fig. 7. The correlation between the value of responses and the Hamming distance of challenges becomes small, as we expected. However, the correlation still exists: 68.1% of challenges $\tilde{C}_j^{(1,i)}$ lead to the same responses as C_j's. This indicates that the responses of BR-PUFs are predictable even if we use the selection of challenge-response pairs, presented by developers. In conclusion, this dependency of the responses on the Hamming distance of challenges might facilitate an attacker to succeed in her modeling attack, and predict most of unknown responses.

[1]According to the BR-PUFs on ASICs self-evaluated by the developers through SPICE simulations in [11], the PUF performances such as reliability and uniqueness are not affected by the generation time of responses.

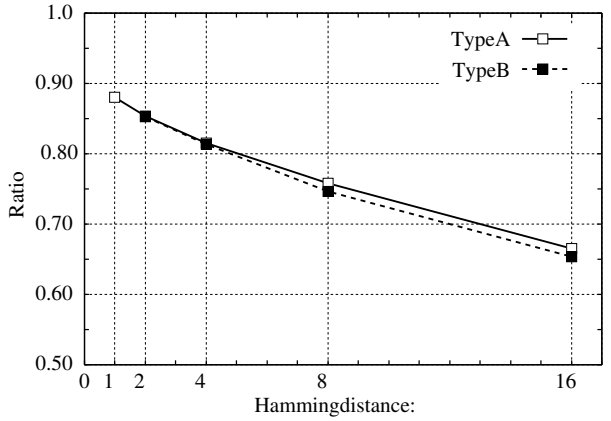


Fig. 6. Ratios of challenges $\tilde{C}_j^{(k,i)}$ generating the same responses as C_j for k in both types.

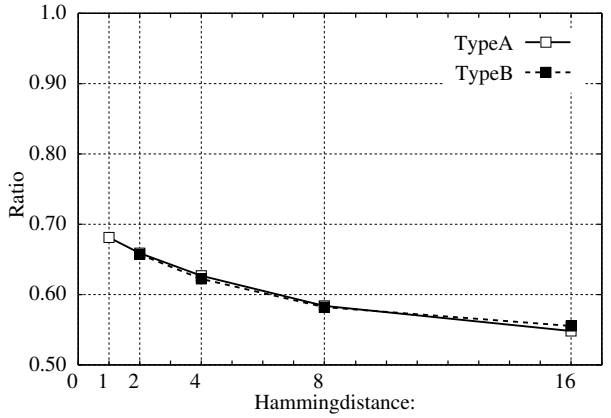


Fig. 7. Ratios of challenges $\tilde{C}_j^{(k,i)}$ generating the same responses as C_j whose transient time is longer than 70 μs.

C. Experimental Results - using Proposed Method (ii)

This section evaluates whether or not BR-PUFs have a BR-S with an *influential* NOR gate, which has a decisive impact on the value of responses. We anticipate that the influential NOR gate has quite different circuit characteristics from other NOR gates. The location of the influential NOR gate is defined by the following two parameters: *enforced bit* ($\leq$ 64) and *enforced value* (0/1). The enforced bit means the location of the BR-S including the influential NOR gate. The enforced value represents either NOR gate in the BR-S. For example, if the enforced bit is 33 and the enforced value is 1, the influential NOR gate is the NOR-1 gate in BR-S_{33}.

As a preliminary experiment to confirm the existence of influential NOR gates, we analyze the 2,048 challenge-response pairs (C_j, R_j) same as Section IV-B. 64-bit challenges of BR-PUFs correspond to the way of selecting NOR gates in BR-Ss. We extract part of C_j's from 2,048 ones whose certain m ($1 \leq m \leq 5$) bits are the same one another, i.e. common NOR gates are selected. Our software program searches all patterns of selecting m NOR gates (${}_{64}\mathrm{C}_m \cdot 2^m$ combinations). Due to time constraints, we set m to less than 6. Table II shows the number of responses (=‘1’s) for the part of C_j's. We explain how to read the table with the specific example of $m = 3$, as follows. Out of 2,048 there are 236 C_j's whose 58th, 13rd and 6th LSBs are 1, 0 and 1, respectively. The number of responses whose values are ‘1’s is 205, which is 86.9% of 236 R_j's. Hence these three NOR gates are predicted to be influential NOR gates, i.e. (enforced bit, enforced value)= $(58, 1), (13, 0), (6, 1)$. Table II also shows the 6 patterns of influential NOR gates for each m. From Table II, we see that more than 65% of responses become 1 in the BR-PUF with just one influential NOR gate (i.e. $m = 1$). The number of the influential NOR gates is around 10 in the 64 BR-Ss. The larger the number of influential NOR gates ($= m$) is, the larger the percentage of responses (=‘1’s) is, i.e. the larger impact on the responses. Especially, all responses become 1 when $m = 5$. In conclusion, according to the analysis of 2,048 C_j's, we demonstrate that our BR-PUF on FPGA has influential NOR gates with a decisive impact on the values of responses.

Above-mentioned results are obtained from a BR-PUF on FPGA_1. We also confirm that the other three BR-PUFs on FPGA_2, FPGA_3 and FPGA_4 have influential NOR gates. BR-PUFs on FPGA_1, FPGA_2 and FPGA_3 generate responses biased to one, while the BR-PUF on FPGA_4 outputs responses biased to zero. The locations of influential NOR gates are different from each FPGA. These are caused by the characteristics of BR-Ss.

As a further experiment, we evaluate the responses for much larger number of challenges than 2,048. First, additional 2^{15} C_j's ($1 \leq j \leq 2^{15}$) are obtained by using the LFSR on the Spartan-3E FPGA. Next, we generate $\hat{C}_j$'s whose enforced bits are changed to the enforced values according to Table II. This means that influential NOR gates are definitely included in the rings of our BR-PUF, and the other NOR gates are selected randomly. Figure 8 shows the ratio of responses equal to 1 for $\hat{C}_j$'s. The line graph represents the average result of six patterns of influential NOR gates as shown in Table II. The upper and lower bounds for error-bars mean the maximum and minimum results of the six patterns, respectively. From Fig. 8, we see that the responses are biased to one when our BR-PUF includes influential NOR gates. The probability of responses being one is 71.4% and 54.5% when the number of influential NOR gates is set to 5 and 1, respectively. The reason why the degree of the bias is smaller than in Table II is more likely that responses are affected by other influential NOR gates not shown in Table II. In conclusion, an attacker who knows some challenge-response pairs could reveal the properties (i.e. influential NOR gates) of her target BR-PUF like Table II. After that, she has a high possibility to predict unknown challenge-response pairs. To minimize the impact of the influential NOR gates, special layout and implementation custom-designed for each BR-PUF are required, however, increase the manufacturing costs dramatically.

TABLE II
INFLUENTIAL NOR GATES AND THEIR IMPACT ON A BIAS OF RESPONSES.

m	**Influential NOR gate(s)** Enforced bit (i-th LSB) : Enforced value (0/1)	# of responses (= 1) / # of responses for challenges with left-column's NORs
1	53:0	701 / 1046 (67.0%)
	25:0	716 / 1044 (68.6%)
	19:0	700 / 1041 (67.2%)
	18:1	678 / 1008 (67.3%)
	06:1	682 / 1011 (67.5%)
	01:0	709 / 1037 (68.4%)
2	53:0, 25:0	411 / 539 (76.3%)
	52:1, 01:0	384 / 505 (76.0%)
	37:0, 06:1	384 / 502 (76.5%)
	25:0, 18:1	400 / 514 (77.8%)
	15:0, 09:0	402 / 528 (76.1%)
	09:0, 06:1	384 / 504 (76.2%)
3	58:1, 13:0, 06:1	205 / 236 (86.9%)
	54:1, 25:0, 18:1	204 / 239 (85.4%)
	53:0, 17:0, 11:0	215 / 252 (85.3%)
	43:0, 37:0, 06:1	219 / 257 (85.2%)
	25:0, 20:1, 19:0	234 / 275 (85.1%)
	25:0, 18:1, 01:0	231 / 271 (85.2%)
4	63:0, 59:0, 37:0, 06:1	124 / 132 (93.9%)
	58:1, 52:1, 13:0, 06:1	112 / 120 (93.3%)
	54:1, 25:0, 18:1, 01:0	114 / 121 (94.2%)
	53:0, 28:1, 11:0, 00:1	122 / 131 (93.1%)
	43:0, 40:1, 32:1, 01:0	123 / 132 (93.2%)
	27:0, 25:0, 18:1, 06:1	123 / 132 (93.2%)
5	53:0, 51:0, 45:0, 18:1, 07:0	79 / 79 (100%)
	58:1, 41:0, 32:1, 19:0, 06:1	76 / 76 (100%)
	59:0, 43:0, 32:1, 13:0, 01:0	61 / 61 (100%)
	63:0, 59:0, 45:0, 18:1, 17:0	61 / 61 (100%)
	52:1, 51:0, 35:1, 20:1, 01:0	45 / 45 (100%)
	48:1, 32:1, 26:1, 10:1, 02:1	41 / 41 (100%)

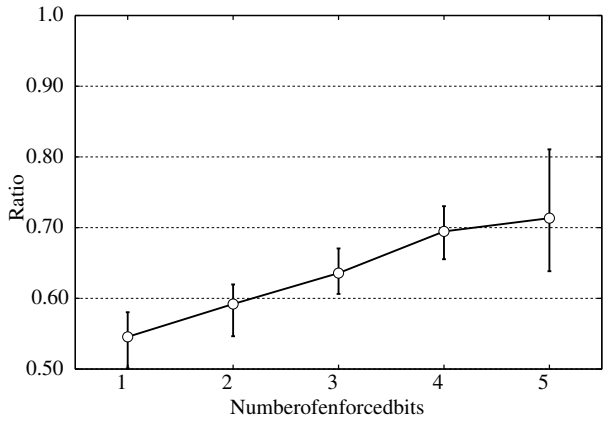


Fig. 8. Ratio of responses (= 1) for $\hat{C}_j$'s whose m-bit enforced bits are changed to enforced values according to Table II.

D. Discussion

1) **Are the wire lengths between all BR-Ss identical?***:* There is no evidence that all of the wire lengths are completely identical. The reason is that it is difficult to exactly control the wire length because logic gates on FPGAs are fixed on grid-pattern layouts. As shown in Fig. 4, we implement BR-PUFs on FPGAs as carefully as possible. In spite of the careful implementations, our BR-PUFs have security issues: correlations of challenge-response pairs. This indicates that it is difficult for many designers to implement BR-PUFs on FPGAs securely. Therefore, this paper gives useful information for designers of BR-PUFs.

2) **Is the implementation of BR-PUFs in this paper the same as that in original paper?***:* We derive 64-bit outputs from all of the 64 BR-Ss, instead of just one in original. We consider that the original implementation is not the best option. This is because deriving only one output may lead to unbalance of capacitive loads on the output of each BR-S, which causes influential gates. We derive outputs from all of BR-Ss in order to prevent this unbalance.

3) **Is it appropriate to derive general results from a single implementation on a specific FPGA family (Xilinx Spartan 6)?***:* Xilinx Spartan 6 FPGAs are relatively newly-released, and have almost the same structure as other types of FPGAs such as Xilinx Virtex 6. We expect, therefore, that similar results are confirmed on the Xilinx FPGA family. In contrast, we need further evaluations on other FPGA families (e.g. developed by Altera) or ASICs because their structures are completely different from that of the Xilinx FPGA family.

4) **Why BR-PUFs are evaluated?***:* It is ture that BR-PUFs are not one of the most famous PUFs. However, we consider BR-PUFs to be excellent and promising PUFs bacause BR-PUFs have advantages of both memory-based and delay-based PUFs. That is why we focus on BR-PUFs in this paper. Our main contribution is to propose the evaluation methods for PUFs: differential and linear PUF analyses. The experimental evaluation of BR-PUFs is a case study of PUF evaluations based on the proposed methods. The proposed methods can be used to evaluate not only BR-PUFs but also other types of PUFs, e.g. arbiter-based PUFs [17].

V. CONCLUSION

In this paper, we proposed the evaluation methods for PUFs: differential and linear PUF analyses. Based on these methods, we experimentally analyzed responses obtained from BR-PUFs using 64 BR-Ss, composed of two NOR gates, implemented on Xilinx Spartan-6 FPGAs. We evaluated the probability of a prediction of the responses R_j for challenge C_j ($1 \leq j \leq 2,048$). According to differential analysis for BR-PUFs, we demonstrated that approximately 88.0% and 66.5% of responses become 1 for challenges with Hamming distance of 1 and 16, respectively. These results are much larger than about 50% in ideal BR-PUFs. Hence an attacker has a high possibility to predict the responses for challenges with small Hamming distances from her known challenge-response pairs. According to linear PUF analysis, we demonstrated that BR-PUFs have some *influential NOR gates*, which cause a strong bias of responses. The probability of responses being one is 71.4% and 54.5% when the number of influential NOR gates is 5 and 1, respectively. An attacker has a high possibility to predict unknown challenge-response pairs by specifying the location of influential NOR gates. Our experimental results are the first time that BR-PUFs present undesirable PUF behavior due to the response prediction, and

compromise the whole security of a system based on BR-PUFs. More importantly, our two evaluation methods can be used as universal framework for evaluating the security of other PUFs (e.g. Arbiter PUFs).

Other implementations of BR-PUFs would probably not behave likewise Spartan-6 FPGAs. For example, the bias of responses has a possibility to improve if BR-PUFs are implemented on other types of FPGAs or ASICs. Future work should include a discussion of security evaluation of BR-PUFs on various platforms.

REFERENCES

[1] D. Yamamoto, M. Takenaka, and N. Torii, "Performance and Security Evaluation of BR-PUF on FPGAs (in Japanese)," in *The 30th Symposium on Cryptography and Information Security (SCIS 2013)*, 2013.

[2] R. Torrance and D. James, "The State-of-the-Art in IC Reverse Engineering," in *Cryptographic Hardware and Embedded Systems (CHES 2009)*, 2009. doi: 10.1007/978-3-642-04138-9_26 pp. 363–381. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-04138-9_26

[3] R. S. Pappu, "Physical One-Way Functions," Ph.D. dissertation, Massachusetts Institute of Technology, 2001.

[4] B. Gassend, D. Clarke, D. Lim, M. van Dijk, and S. Devadas, "Identification and Authentication of Integrated Circuits," *Concurrency - Practice and Experience*, vol. 16, no. 11, pp. 1077–1098, 2004. doi: 10.1002/cpe.805. [Online]. Available: http://dx.doi.org/10.1002/cpe.805

[5] B. Gassend, D. E. Clarke, M. van Dijk, and S. Devadas, "Silicon physical random functions," in *ACM Conference on Computer and Communications Security*, 2002. doi: 10.1145/586110.586132 pp. 148–160. [Online]. Available: http://doi.acm.org/10.1145/586110.586132

[6] R. Maes and I. Verbauwhede, "Physically Unclonable Functions: A Study on the State of the Art and Future Research Directions," in *Towards Hardware Intrinsic Security: Foundation and Practice*. Springer, 2010, pp. 3–37. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-14452-3_1

[7] Y. Su, J. Holleman, and B. Otis, "A 1.6pJ/bit 96% Stable Chip-ID Generating Circuit using Process Variations," in *IEEE International Solid-State Circuits Conference (ISSCC 2007)*, 2007. doi: 10.1109/ISSCC.2007.373466 pp. 406–611. [Online]. Available: http://dx.doi.org/10.1109/ISSCC.2007.373466

[8] Y. Su, J. Holleman, and B. P. Otis, "A Digital 1.6pJ/bit Chip Identification Circuit Using Process Variations," *Solid-State Circuits, IEEE Journal of*, vol. 43, no. 1, pp. 69–77, 2008. doi: 10.1109/JSSC.2007.910961. [Online]. Available: http://dx.doi.org/10.1109/JSSC.2007.910961

[9] G. E. Suh and S. Devadas, "Physical Unclonable Functions for Device Authentication and Secret Key Generation," in *Design Automation Conference (DAC 2007)*, 2007. doi: 10.1109/DAC.2007.375043 pp. 9–14. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/DAC.2007.375043

[10] Q. Chen, G. Csaba, P. Lugli, U. Schlichtmann, and U. Rührmair, "The Bistable Ring PUF: A New Architecture for Strong Physical Unclonable Functions," in *Hardware-Oriented Security and Trust (HOST 2011)*, 2011. doi: 10.1109/HST.2011.5955011. [Online]. Available: http://dx.doi.org/10.1109/HST.2011.5955011

[11] ——, "Characterization of the Bistable Ring PUF," in *Design, Automation & Test in Europe Conference & Exhibition (DATE 2012)*, 2012. doi: 10.1109/DATE.2012.6176596. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/DATE.2012.6176596

[12] E. Biham and A. Shamir, "Differential Cryptanalysis of DES-like Cryptosystems," *J. Cryptology*, vol. 4, no. 1, pp. 3–72, 1991. doi: 10.1007/BF00630563. [Online]. Available: http://dx.doi.org/10.1007/BF00630563

[13] M. Matsui, "Linear Cryptoanalysis Method for DES Cipher," in *EUROCRYPT*, 1993. doi: 10.1007/3-540-48285-7_33 pp. 386–397. [Online]. Available: http://dx.doi.org/10.1007/3-540-48285-7_33

[14] U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, and J. Schmidhuber, "Modeling Attacks on Physical Unclonable Functions," in *ACM Conference on Computer and Communications Security*, 2010. doi: 10.1145/1866307.1866335 pp. 237–249. [Online]. Available: http://dx.doi.org/10.1145/1866307.1866335

[15] R. Ward and T. Molteno, "Table of Linear Feedback Shift Registers," University of Otago, New Zealand, Tech. Rep., 2007. [Online]. Available: http://www.eej.ulst.ac.uk/~ian/modules/EEE515/files/old_files/lfsr/lfsr_table.pdf

[16] A. Maiti, V. Gunreddy, and P. Schaumont, "A Systematic Method to Evaluate and Compare the Performance of Physical Unclonable Functions," in *Embedded Systems Design with FPGAs*. Springer New York, 2013, pp. 245–267. [Online]. Available: http://dx.doi.org/10.1007/978-1-4614-1362-2_11

[17] J. W. Lee, D. Lim, 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 *IEEE VLSI Circuits Symposium 2004*, 2004. doi: 10.1109/VLSIC.2004.1346548 pp. 176–179. [Online]. Available: http://dx.doi.org/10.1109/VLSIC.2004.1346548