Security Evaluation and Enhancement of Bistable Ring PUFs

Xiaolin Xu[1], Ulrich Rührmair[2]
Daniel E. Holcomb[1], and Wayne Burleson[1]

[1] ECE Department, University of Massachusetts Amherst
{xiaolinx,dholcomb,burleson}@umass.edu
[2] Horst Görtz Institute for IT-Security, Ruhr Universität Bochum
ruehrmair@ilo.de

Abstract. The Bistable Ring (BR) Physical Unclonable Function (PUF) is a newly proposed hardware security primitive in the PUF family. In this work, we comprehensively evaluate its resilience against Machine Learning (ML) modeling attacks. Based on the success of ML attacks, we propose XOR strategies to enhance the security of BR PUFs. Our results show that the XOR BR PUF with more than four parallel BR PUFs is able to resist the ML modeling methods in this work. We also evaluate the other PUF metrics of reliability, uniqueness and uniformity, and find that the XOR function is also effective in improving the uniformity of BR PUFs.

1 Introduction

In the last ten years, physical unclonable functions (PUFs) have established themselves as an alternative to conventional security approaches [4][11]. In a nutshell, a PUF is a disordered, at least partly randomly structured physical system. Due to its random structure that is caused by uncontrollable, small-scale manufacturing variations, it is physically unclonable, i.e., no two specimens can be produced that are physically exactly identical. This limitation applies to both the original manufacturer and to other, potentially adversarial, parties. All PUFs have one basic functionality in common, namely some *challenge-response* mechanism: They can be triggered or excited by signals that are commonly denoted as *challenges* C_i. Upon excitation by a challenge C_i, they react by producing a response R_{C_i} that depends on their internal disorder and usually also on the challenge itself. The tuples $(C_i, R(C_i))$ are called the *challenge-response pairs (CRPs)* of the PUF.

Over the years, different variants or types of PUFs have emerged (see [12] for an overview). They all share the above features of being a disordered structure, possessing physical unclonability, and exhibiting some form of challenge-response mechanism. However, their other security features, together with their intended applications and associated attack scenarios, notably differ. This makes it useful in scientific works to explicitly distinguish these types from each other [12].

The two main PUF-types are often denoted as weak and strong PUFs. Weak PUFs possess essentially a single, fixed challenge C, as for example in the case of SRAM PUFs. They are mainly used for internal key derivation in security hardware. The underlying security assumption is that attackers must not be able to access the internal response of the PUF, for example by reading out the power-up state of the SRAM PUF [5][6]. In opposition to that, strong PUFs are PUFs that have a very large number of possible challenges, too many to read out all corresponding CRPs in feasible time. Their challenge-response mechanism should be complex in the sense that it is hard to derive unknown CRPs from a set of known CRPs. In particular, strong PUFs should not allow *"modeling attacks"*, in which an adversary collects a subset of CRPs, uses them to train a machine learning (ML) algorithm, and later employs the model produced by the ML algorithm to predict unknown CRPs.

Strong PUFs are usually employed with a publicly accessible CRP interface, i.e., anyone holding the PUF or the PUF embedding hardware can apply challenges and read out responses. The lack of access restriction mechanisms on strong PUFs is therefore a key difference from weak PUFs. In recent years, strong PUFs have turned out to be a very versatile cryptographic and security primitive: First of all, by using a fixed set of challenges, they can be employed for internal key derivation, just like weak PUFs. But they can do more: They can also implement a host of advanced cryptographic protocols, ranging from identification [11][9] to key exchange [20][1] to oblivious transfer [1].

Their many possible applications make the secure construction of secure strong PUFs a worthwhile and rewarding research target. Unfortunately, it is a non-trivial one, too: A large number of powerful attacks on some first-generation electrical strong PUFs have been published recently. They include the above mentioned modeling attacks [13][14]; side channel attacks [15]; and also optical characterization techniques [19]. Most of these attacks target the first electrical strong PUF, the so-called Arbiter PUF [4][11] and variants thereof, for example XOR Arbiter PUFs and Feed-Forward Arbiter PUFs. For this reason, alternative silicon architectures have been proposed in recent years. One such alternative is the *"Bistable Ring PUF" (BR PUF)* [2][3], which was designed to have a more complex internal response-generating mechanism in hopes of making ML attacks harder.

At TRUST 2014, Hesselbarth and Schuster [16] succeeded in revealing some basic vulnerabilities of the BR PUF against ML techniques. They proved that BR PUFs can be attacked by a single layer artificial neural network (ANN) with prediction errors between close to 0% and 20%, varying from hardware instance to instance. Among the 20 FPGA instances examined, 14 could be predicted with errors less than 10%. This puts close limits on the security usability of the BR PUF on FPGAs. Schuster and Hesselbarth subsequently proposed a small design improvement, so-called twisted BR PUFs (TBR PUFs), which they conjectured to possess better security. Using their own ANN algorithm, they were also able to attack TBR PUFs again. However, the TBR PUF shows average higher prediction errors with respect to ANNs, indicating that TNR PUFs has some improvements

over plain BR PUFs. It remained open in the work of Schuster and Hesselbarth whether said improvement is sufficient for secure practical usage of the TBR PUF.

Our Contributions In this paper, we re-examine the security of the BR PUF and TBR PUF closely, again using FPGA implementations. We thereby make the following new contributions:

- We implement 8 instances of the BR PUF and the TBR PUF on FPGA. To achieve a more comprehensive ML analysis, we implement bitlengths other and larger than 64, namely also 32, 128 and 256. These bitlengths had never before been implemented in silicon and studied in the literature.
- We develop the first analytical models for the BR PUF and the TBR PUF.
- We use these new models in order to apply, for the first time, support vector machines (SVMs) to the BR PUF and the TBR PUF. This more powerful ML-tool drastically improves the ML predication rates relative to previous work. None of our 8 instances has a prediction error exceeding 5%. This result answers the open question of Hesselbarth and Schuster whether certain individual and particularly hard instances of the BR PUF or TBR PUF could be used securely in practice: In our findings, this was not the case.
- We then propose a new, efficient strategy for the secure practical use of the BR PUF: namely the employment of l instances in parallel, whose l outputs are XORed at the end in order to produce one single-bit PUF-response. We call the resulting structure XOR BR PUF. We show that even for small values of l such as 4, this structure cannot be machine learned by our current techniques, while it is still sufficiently stable in practice. This work is the first study of XOR BR PUFs in the literature.

Organization of This Paper This paper is organized as follows. Section 2 discusses our attacks on the BR PUF, while Section 3 details our attacks on the TBR PUF. Section 4 suggests the use of XOR BR PUFs and evaluates their performance improvement. Section 5 concludes the paper.

2 SVM Attack on BR PUFs

2.1 Mechanism of BR PUF

A ring oscillator (RO) is a device composed of an odd number of logically-inverting delay elements. Since the output of the last element is always the logical "NOT" of the first input, an RO will continually oscillate. Derived from the non-settling structure of RO, BR PUF is a ring comprising an even number of inverting cells. Such a design behaves like a memory cell and will fall into one of two possible stable states: either "101010..." or "010101...".

As depicted in Fig. 1, a 64-bit BR PUF is composed of 64 stages, where each stage has two inverting delay elements (NOR gates as an example). A challenge vector $C = \{c_1, c_2, \ldots, c_n\}$ selects the NOR gates used in each bistable

ring configuration by providing values to the MUX and DEMUX gates of the stages. Since each NOR gate has unique process variation, each different challenge vector creates a unique bistable ring configuration, and in total 2^{64} different configurations can be created. A common "RESET" signal is added to each stage to establish a known "all-0" state before letting the ring stabilize to produce its response. Evaluation of the response begins when "RESET" is released and the ring starts to oscillate through the selected NOR gates. Once a stable state is reached, the outputs of any two adjacent stages will be logical compliments of each other, either "10" or "01". The choice among the two possible stable states of the ring depends on noise and the process variation of the NOR gates used in the ring configuration. Any interconnection node between two stages can be used as an output port, and in this work we use the half bit-length port to read out the response (Fig. 1).

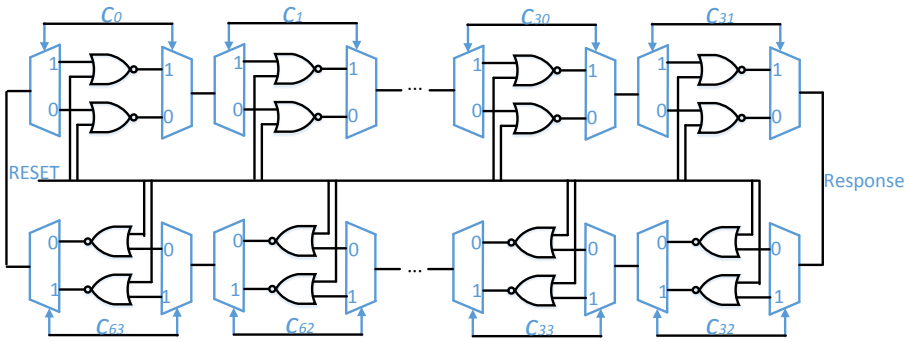


Fig. 1: Schematic of a single BR-PUF with 64 stages.

2.2 Intuition for Modeling BR PUF

The intuition for our modeling attack is that the response can predicted based on a summation of weights. Such an additive model is commonly used for predicting the responses of Arbiter PUFs, where the weights represent stage delays [8]. An additive model has also been used for predicting the resolution of metastability [7], with weights representing the strength with which different cells pull toward a particular outcome. We similarly use an additive model in this work; the weight we associate with each gate represents the difference between its pull-up strength and pull-down strength. The weights are summed across all gates used by a challenge to find the overall favored response for that challenge; a positive sum indicates a preference for the positive response. Note that the summation of weights requires negative and positive polarities because the positive overall response is favored by the pull-up strength of even stages and the pull-down strength of odd stages.

2.3 Model

Let the difference between the pull-up and pull-down strength of the top NOR gate in the i^{th} stage be represented by t_i, and in the bottom NOR gate in the i^{th} stage be represented by b_i. The even stages will contribute toward the positive response with strength t_i (or b_i if the challenge bit selects the bottom NOR gate of the stage), and the odd stages will contribute toward the positive response with strength $-t_i$ (or $-b_i$). To account more generally for even-ness or odd-ness, the strength of the i^{th} stage toward the positive response can be written as $-1^i t_i$ if the challenge bit is 0, and $-1^i b_i$ if the challenge bit is 1. For a given 64-bit challenge, the total strength pulling toward the positive response is the summation of 64 t_i and b_i weights.

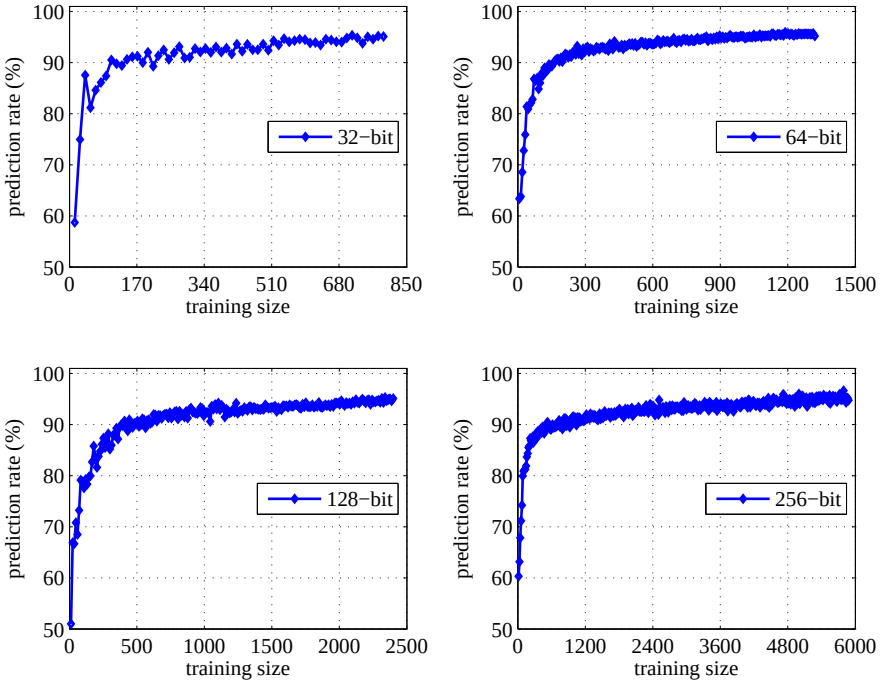


Fig. 2: Prediction rate of SVM modeling attacks on BR PUFs. When the length of the BR PUF increases, more CRPs are required to train the model to achieve 95% prediction. Note that the scale of the x-axes are not consistent across the subfigures.

For convenience we define α_i and β_i (Eq. 1) such that $\alpha_i + \beta_i = -1^i t_i$ and $\alpha_i - \beta_i = -1^i b_i$. This notation allows the pull of the i^{th} stage toward the positive response to be written generally as $\alpha_i + c_i \beta_i$ with challenge bit $c_i \in \{-1, 1\}$. The summed strengths toward the positive response for any challenge vector C is

$r(C)$ as shown in Eq. 2. According to our formulation, if the t_i and b_i weights were known explicitly, then the response could be predicted by the sign of $R(C)$ (Eq. 2).

$$\alpha_i = -1^i \left(\frac{t_i - b_i}{2} \right) \qquad \beta_i = -1^i \left(\frac{t_i + b_i}{2} \right) \tag{1}$$

$$R(C) = sgn(\sum_{i=0..n-1} \alpha_i + c_i \beta_i) \tag{2}$$

Given that weights are not known, since there are only two possible responses of BR PUFs, based on the model above, we can convert the response prediction of BR PUFs into a classification problem. Support Vector Machines (SVM) are powerful learning tools that can perform binary classification of data, the classification is realized with building a hyperplane separating surface. While digesting the known input and output data sets, the hyperplane separating surface will be curved to minimize the error of predicted values.

Known CRPs are used to train the classifier to predict responses from challenges. In the SVM formulation, first note that the α_i terms in Eq. 2 can be discarded because they are constant for a given PUF instance across all challenges. Only $c_i \beta_i$ terms remain, and from these terms the response must be predicted. Given a set of challenges and associated responses, the training examples therefore have as their feature vector an applied challenge $C_j \in \{-1, 1\}^n$ and as their labels the observed response $R(C_j) \in \{-1, 1\}$ to challenge C_j. Note that β_i terms do not appear explicitly in the SVM formulation as the classifier simply works to find the maximum margin hyperplane to separate the challenges into two classes according to their responses.

2.4 SVM Attacks on BR PUFs

To explore the effectiveness of SVM attacks, we implemented on a Xilinx Spartan-VI FPGA board, 8 BR PUFs with lengths of 32-, 64-, 128- and 256 bits, and collected 1,000,000 CRPs from each of them (to decrease the impact of measurement noise, all of the final CRPs are formulated by majority voting from 11 repeated measurements). SVM attacks are implemented with a linear kernel to mimic the operation of single BR PUFs (note that to attack XOR BR PUFs, SVM model with a polynomial kernel is utilized, where the poly-order of the model is set as the XORing complexity of BR PUFs). The results of SVM attacks are shown as Fig. 2. To demonstrate the relationship between prediction rate and CRPs used for different PUF lengths, we utilize 95% as a threshold prediction rate. It is clear that while the size of BR PUF is increasing, the demand for CRPs is also increasing to build its ML model. However, for any tested size of BR PUF, the SVM modeling attack is successful in predicting responses. This means a single BR PUF is not secure, even if it has a large number of stages.

3 Twisted BR PUFs Attack

3.1 Model of TBR PUFs

Uniformity, or fractional Hamming weight, is an important feature of PUFs. A good PUF that produces an equal number of 0 and 1 responses will have a uniformity of around 0.50. However, the uniformity of CRPs of BR PUF implementations has been found to be biased in previous work [16] (see also Sec. 4.3 in this work). To compensate for this drawback, TBR-PUF was proposed in [16]. Compared to the BR PUF, the TBR-PUF has a more compact design; when applying a challenge vector to the TBR PUF, all of its $2n$ inverting elements are used in the ring. By contrast, in the standard BR PUF, half of the NOR gates in the circuit are unused for any given challenge. Taking the TBR PUF in Fig. 3 as an example, using challenge bit $c_0 = 1$ or $c_0 = 0$ will change the location of D_0^0 and D_1^0 in the ring, but in either case D_0^0 and D_1^0 will both contribute in some polarity to the response.

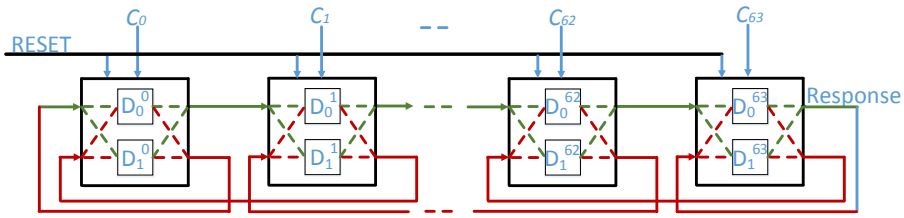


Fig. 3: Schematic of a single TBR-PUF with 64 stages.

From Sec. 2, we know that a ring composed of an even number of inverting elements will stabilize according to the summed strength of the pull-up and pull-down strengths of each gate. The TBR PUF uses pull-up and pull-down strengths of all inverting components in the circuit, but only the polarity (i.e. even-ness or odd-ness) of each element toward the overall ring response changes with the challenge vector. According to the interconnections of the 64-bit TBR PUF, the two NOR gates in the i^{th} stage are the i^{th} and $127 - i^{th}$ element in the overall ring. Because one element is odd in the overall ring, and one is even, the pull-up strength of the top and bottom gates in each stage are working against each other. Therefore, the overall contribution toward the positive response is β_i (Eq. 3), or $-\beta_i$ if the i^{th} challenge bit is negative. The overall sum of weights pulling toward the positive response for challenge C is therefore $R(C)$ (Eq. 4). Eq. 2 and Eq. 4 differ only in in the physical meaning of β_i, and in Eq. 2 having an additional offset term of $\sum_i \alpha_i$, but in terms of ML modeling they are actually the same identical model. Therefore, the complexity of ML attacks on the TBR PUF is the same as the complexity of attacking the BR PUF.

$$\beta_i = -1^i(t_i - b_i) \tag{3}$$

$$R(C) = sgn(\sum_{i=0...n-1} c_i \beta_i) \tag{4}$$

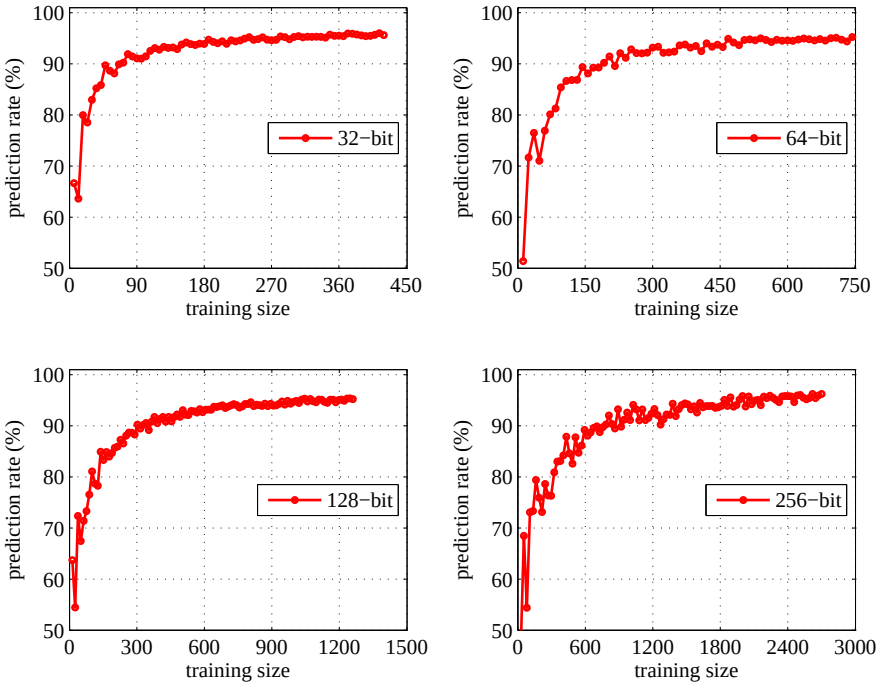


Fig. 4: Prediction rate of SVM modeling attacks on TBR PUFs of different bit lengths. As in Fig. 2, to achieve same prediction rate, a larger PUF requires more CRPs.

3.2 SVM Attacks on TBR PUFs

Given that we have shown the model of a TBR PUF to be the same as that of a BR PUF, we can again train an SVM classifier to predict its responses to each challenge. Eight TBR PUFs are implemented with Spartan-VI FPGA boards, and 1,000,000 CRPs are collected from each of them. For each CRP, majority voting over 11 repeated measurements of the response to a challenge are performed in order to reduce the impact of noise.

Following the experiment in Sec. 2.4, SVM attacks with polynomial kernel are applied on TBR PUFs of 32-, 64-, 128- and 256 bit-length (the poly-order of the model is set as the XORing complexity). The results in Fig. 4 show that the modeling attacks succeed in modeling all different sizes of the TBR PUF, with prediction rate no lower than 95%.

4 XORing BR PUFs to Enhance the Security

It is possible using ML to model the behavior of a single strong PUF like the Arbiter PUF [8]. To thwart modeling attacks, an XOR function was proposed as a way to enhance security of Arbiter PUFs [17] and lightweight PUFs [10]. In an XOR PUF, the same challenge vector is applied to l single PUFs in parallel, and their outputs are XORed together to form a one-bit response. XORing is an efficient method to enhance the security of strong PUFs, because the XOR function obfuscates the CRPs of the individual PUFs [17]. Inspired by this idea, we propose to use XOR strategies on BR PUFs to improve their resistance to modeling attacks.

4.1 Review of Existing Attacks on XOR PUFs

The addition of XOR functions increases the resistance of strong PUF against modeling attacks. Both the training time and number of CRPs required to train a model increase exponentially with the number of XORed PUFs [13]. Attacking XOR-based Arbiter PUFs with more than five parallel Arbiter PUFs was stated as difficult based on pure ML modeling [14]. Later works devised a more powerful class of hybrid attacks that combine side channels with ML [15, 21]. Power and timing side-channels allow information about the sub-responses (i.e. the responses of single PUFs before the final XOR) of XORed PUFs to be extracted and used to improve the prediction rate of ML models. In light of these hybrid attacks, if the side-channel information of BR PUFs can also be measured, then the use of XOR will not be an effective way to enhance the security.

4.2 SVM Modeling Attacks on XORed BR PUF

Adopting the model of single BR PUF in Sec. 2, for an XOR BR PUF employing l BR PUFs, the XORed response to a challenge C can be described by Eq. 5. Note the similarity between this formula and the formula of the single BR PUF (Eq. 2). The only modification is that now each stage has l different α and β terms, one for each of the PUFs. The overall response is based on how many of the individual PUFs have a positive response.

$$R(C) = sgn\Big(\prod_{j=0}^{l-1}\Big(\sum_{i=0}^{n-1}\alpha_{i,j} + c_i\beta_{i,j}\Big)\Big) \tag{5}$$

In applying SVM to XOR BR PUF, it is found that we can only break the complexity up to 2 XOR for 128-bit length and 3 XOR for 64-bit length. The number of CRPs and runtime[1] for SVM modeling attacks against XOR BR PUFs are listed in Tab. 1. We can surmise that XOR BR PUFs with 4 or more XORed outputs are beyond the reach of current SVM modeling attacks.

[1] The computer used has a common Intel 3630QM quadcore processor.

No. of XORs	Bit Length	CRPs ($\times 10^3$)	Predict. Rate	Training* Time
2	32	0.8	95%	3 sec
	64	4	95%	10 sec
	128	18	95%	6 mins
	256	—	50.8%	—
3	32	1.2	95%	5 sec
	64	7.2	95%	24 sec
	128	—	50.1%	—
	256	—	50.1%	—
4	32	—	50.1%	—
	64	—	50.3%	—
	128	—	49.8%	—
	256	—	50.1%	—

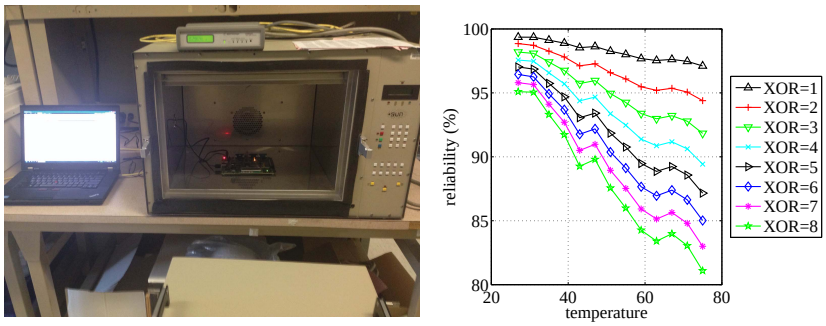
Table 1: The run times and number of CRPs that are required for SVM attacks on the XOR BR PUFs of different sizes. Prediction rates around 50% imply that the SVM model can not break XOR BR PUFs of these complexity. *Note that the training time is greatly determined by the computational systems.

4.3 Performance Evaluation of XORed BR PUF

While the basic motivation of XORing BR PUF is to resist modeling attacks, the impact of the XOR on other key metrics must also be considered. In this section, we evaluate the impact of the XOR function on reliability, uniqueness, and uniformity.

Reliability Reliability is the ratio of consistent CRPs when a PUF is operating in different environment conditions such as temperature. To evaluate the reliability of XOR BR PUFs, 8 BR PUFs are measured across different temperatures between 27°C and 75°C, with a 4°C step, using a Sun Electronics EC12 Environmental Chamber [18] to control the temperature (Fig. 5a). Reliability is evaluated by comparing CRPs collected at 27°C to CRPs collected at other temperatures. For a XOR PUF, any unstable sub-response can cause the XORed response to be unreliable. Therefore, the reliability at any temperature will decrease with the number of PUFs that are XORed together (Fig. 5b). According to the first BR PUF paper [3], an effective solution to solve this problem is employing CRPs that settle down quickly, since those CRPs are less sensitive to noise.

Uniqueness Uniqueness is the capability of a PUF to distinguish itself from other instances. Uniqueness is quantified as the fraction of responses that are different across instances when the same challenges are applied. Thus for m PUF instances, a total of $\frac{m*(m-1)}{2}$ uniqueness values are obtained. To better explore the uniqueness of XOR BR PUF, we compute its within-class (response



(a) experimental platform

(b) reliability across different temperatures

Fig. 5: Evaluating reliability across different temperatures. Because the reliability of each single BR PUF decreases with temperature, the reliability of the XOR BR PUF results degrade significantly.

flipping by noise, temperature noise here) and between-class uniqueness (response difference between instances), these results are depicted in Fig. 6.

Uniformity Uniformity denotes the average response of a PUF, the ideal value of which is 0.5, meaning equal amount of "1" and "0" responses. Uniformity that is far away from 0.5 will have less response entropy and be easier to attack with modeling [22]. In our experiment, the uniformity of a single BR PUF is found to be highly biased, and this phenomenon was also reported in [16] [22]. The XOR function helps to remove this bias. To validate the uniformity improvement from the XOR function, we collected the CRPs from eight 64-bit BR PUFs from FPGA (without CRP majority voting). It is found that some PUF instances show extreme bias, but XORing more single BR PUFs together decreases response bias (Fig. 7).

5 Conclusion and Future Works

In this work, we studied two relatively new PUF variants: BR PUF and its derived architecture TBR PUF. Their resilience against ML modeling attacks is explored and it is shown that their response can be predicted with success rate exceeding 95% using reasonable runtime and less than 10k CRPs in all cases. Our work confirms that neither a single BR, nor TBR, PUF is secure. To strengthen the BR PUF against modeling attacks, we proposed and evaluated an XOR BR PUF variant. It is found that XORing 4 or more BR PUFs together produces a behavior that is beyond the modeling capability of current SVM ML attacks, and also improves other key PUF metrics like uniformity. Future work will explore the effectiveness of other modeling attacks, like Evolutionary Strategy and Logistic Regression methods.

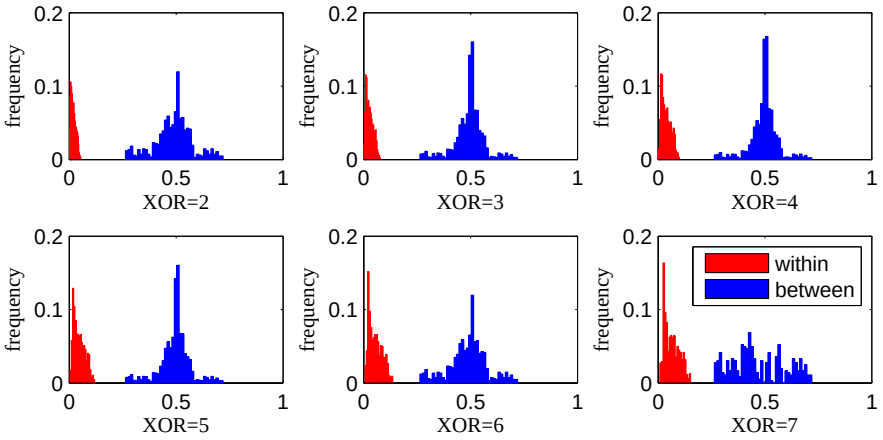


Fig. 6: The between-class and within-class Hamming distance of XOR PUFs. Even when XORing together more BR PUFs, the within-class and between-class Hamming distances can still be differentiated. The results are based on 8 BR PUFs, thus there is only one 8 XOR BR PUF and no uniqueness is formulated for it.

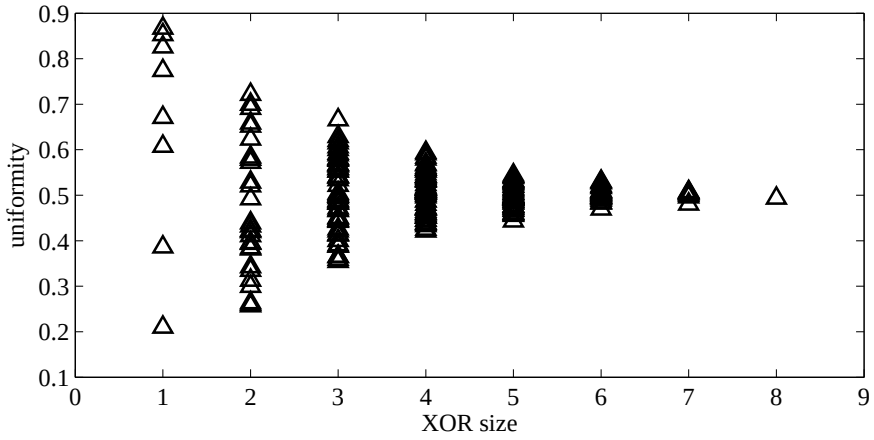


Fig. 7: The response uniformity of a single BR PUF (represented by "XOR=1" in plot) is highly biased. When more BR PUFs are XORed together, the uniformity is closer to 0.5.

References

1. Brzuska, C., Fischlin, M., Schröder, H., and Katzenbeisser, S. Physically uncloneable functions in the universal composition framework. In *Advances in Cryptology–CRYPTO 2011*. Springer, 2011, pp. 51–70.
2. Chen, Q., Csaba, G., Lugli, P., Schlichtmann, U., and Ruhrmair, U. The bistable ring puf: A new architecture for strong physical unclonable functions. In *Hardware-Oriented Security and Trust (HOST), 2011 IEEE International Symposium on* (2011), IEEE, pp. 134–141.
3. Chen, Q., Csaba, G., Lugli, P., Schlichtmann, U., and Ruhrmair, U. Characterization of the bistable ring puf. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2012* (2012), IEEE, pp. 1459–1462.
4. Gassend, B., Clarke, D., Van Dijk, M., and Devadas, S. Silicon physical random functions. In *Proceedings of the 9th ACM Conference on Computer and Communications Security* (2002), ACM, pp. 148–160.
5. Guajardo, J., Kumar, S., Schrijen, G., and Tuyls, P. FPGA intrinsic PUFs and their use for IP protection. *Cryptographic Hardware and Embedded Systems* (2007).
6. Holcomb, D. E., Burleson, W. P., and Fu, K. Power-up SRAM State as an Identifying Fingerprint and Source of True Random Numbers. *IEEE Transactions on Computers* (2009).
7. Holcomb, D. E., and Fu, K. Bitline PUF: Building Native Challenge-Response PUF Capability into Any SRAM. In *Cryptographic Hardware and Embedded Systems (CHES 2014)* (Sept. 2014), L. Batina and M. Robshaw, Eds., vol. 8731 of *Lecture Notes in Computer Science*, pp. 510–526.
8. Lim, D. Extracting secret keys from integrated circuits, MSc Thesis, 2004.
9. Lofstrom, K., Daasch, W. R., and Taylor, D. Ic identification circuit using device mismatch. In *Solid-State Circuits Conference, 2000. Digest of Technical Papers. ISSCC. 2000 IEEE International* (2000), IEEE, pp. 372–373.
10. Majzoobi, M., Koushanfar, F., and Potkonjak, M. Lightweight secure PUFs. In *Proceedings of the 2008 IEEE/ACM International Conference on Computer-Aided Design* (2008), IEEE Press, pp. 670–673.
11. Pappu, R., Recht, B., Taylor, J., and Gershenfeld, N. Physical one-way functions. *Science 297*, 5589 (2002), 2026–2030.
12. Rührmair, U., and Holcomb, D. E. PUFs at a glance. In *Proceedings of the conference on Design, Automation & Test in Europe* (2014), European Design and Automation Association, p. 347.
13. Rührmair, U., Sehnke, F., Sölter, J., Dror, G., Devadas, S., and Schmidhuber, J. Modeling attacks on physical unclonable functions. In *Proceedings of the 17th ACM conference on Computer and communications security* (2010), ACM, pp. 237–249.
14. Rührmair, U., Sölter, J., Sehnke, F., Xu, X., Mahmoud, A., Stoyanova, V., Dror, G., Schmidhuber, J., Burleson, W., and Devadas, S. PUF modeling attacks on simulated and silicon data. *Information Forensics and Security, IEEE Transactions on* (2013).
15. Rührmair, U., Xu, X., Sölter, J., Mahmoud, A., Majzoobi, M., Koushanfar, F., and Burleson, W. Efficient power and timing side channels for physical unclonable functions. In *Cryptographic Hardware and Embedded Systems–CHES 2014*. Springer, 2014, pp. 476–492.

16. Schuster, D., and Hesselbarth, R. Evaluation of bistable ring PUFs using single layer neural networks. In *Trust and Trustworthy Computing*. Springer, 2014, pp. 101–109.
17. Suh, G. E., and Devadas, S. Physical unclonable functions for device authentication and secret key generation. In *Proceedings of the 44th annual Design Automation Conference* (2007), DAC '07, ACM, pp. 9–14.
18. Sun Electronic Systems, I. *Model EC1X Environmental Chamber User and Repair Manual*, 2011.
19. Tajik, S., Dietz, E., Frohmann, S., Seifert, J.-P., Nedospasov, D., Helfmeier, C., Boit, C., and Dittrich, H. Physical characterization of arbiter PUFs. In *Cryptographic Hardware and Embedded Systems–CHES 2014*. Springer, 2014, pp. 493–509.
20. Van Dijk, M. E. System and method of reliable forward secret key sharing with physical random functions, Jan. 26 2010. US Patent 7,653,197.
21. Xu, X., and Burleson, W. Hybrid side-channel/machine-learning attacks on PUFs: a new threat? In *Proceedings of the conference on Design, Automation & Test in Europe* (2014), European Design and Automation Association, p. 349.
22. Yamamoto, D., Takenaka, M., Sakiyama, K., and Torii, N. Security evaluation of bistable ring PUFs on FPGAs using differential and linear analysis. In *Computer Science and Information Systems (FedCSIS), 2014 Federated Conference on* (2014), IEEE, pp. 911–918.