The Shift PUF: Technique for Squaring the Machine Learning Complexity of Arbiter-based PUFs: Work-in-Progress

Yi Tang
New York University, NY
yt1433@nyu.edu

Donghang Wu
Peking University, China
sdlwwdh@pku.edu.cn

Yongzhi Cao
Peking University, China
caoyz@pku.edu.cn

Marian Margraf
Freie Universität Berlin, Germany
Marian.Margraf@fu-berlin.de

The *physically unclonable function* (PUF) is a hardware cryptographic primitive that provides identifications based on inevitable manufacturing variations, thus, as the name states, being physically unclonable. The *arbiter PUF* (APUF, [5]) is a well-studied PUF design based on variational signal delays in silicon/electronic components. Using APUFs as building blocks, *XOR APUF* ([12]), *lightweight secure PUF* ([9]), *feed forward APUF* ([6], [7]), etc. are proposed and expected to be more secure PUF designs. However, it is discovered that all of these canonical arbiter-based PUF designs suffer from machine learning modeling attacks ([1], [3], [11]). Recently, the *interpose PUF* (iPUF, [10]) is proposed as a new arbiter-based PUF design that is resilient to state-of-the-art machine learning attacks. In this paper we propose a new PUF design called *shift PUF* that directly enhances APUF (which, to remark, is the building block of all arbiter-based PUF designs) by, as a conjecture, squaring its machine learning complexity, and consequently brings the same squaring benefit to all arbiter-based PUFs as well. To emphasize, the shift PUF itself is not a secure PUF design, and the technique of substituting APUFs with shift PUFs also not necessarily turns insecure PUF designs into secure PUF designs (the notion of security immediately follows in the next paragraph); nevertheless the technique greatly benefits already secure arbiter-based PUF designs with squared machine learning complexities.

We start with reviewing some of the most desirable properties of an ideally secure PUF design. As an identification primitive, a PUF instance is identified by its behavior as a (potentially probabilistic) mapping from inputs (often referred to as *challenges*) to outputs (as *responses*). Judging from practical considerations, a PUF design is desired to have the following properties:

- Being *lightweight*, which means the time and the circuit complexities of the hardware is efficient (often, linear) with regard to the size of challenges;
- Being *strong*, which means the challenge space is huge (often, exponential) compared to the size of challenges, and thus the PUF instance space is enormously huge (double exponential);
- *Reliability*, which means the same challenge results in the same response with high probability;
- Most importantly and demandingly, *security*, which means it is hard to clone, i.e., to simulate with high accuracy the challenge-response behavior of a PUF instance given any reasonable partial information of its challenge-response behavior.

Due to the versatility of identification primitive, an ideal PUF design leads to various meaningful applications ([5]–[8], [12]), especially in cloud computing and IoT settings where the lightweight-ness and the security are both of great importance.

In this paper we focus on arbiter-based PUF designs. The APUF acts as a central building block in this category. The challenge-response behavior $r(\mathbf{c})$ of an m-bit APUF can be expressed by a linear classification model

$$\Delta(\mathbf{c}) = \mathbf{w}^\top \mathbf{p} \,, \quad r(\mathbf{c}) = [\Delta(\mathbf{c}) \geq 0] \,, \tag{1}$$

where $\mathbf{w}$ is an $(m+1)$-dimensional vector that is function of the signal delays in the APUF instance, and $\mathbf{p}$ is the corresponding *parity* vector of the challenge vector $\mathbf{c}$; formally,

$$p_i = \prod_{j=i}^{m} (-1)^{c_j}, 1 \leq i \leq m \,, \quad p_{m+1} = 1 \,. \tag{2}$$

Since APUFs are captured by this linear classification model, they can be easily modeled and simulated by machine learning attacks (using e.g. logistic regression or support vector machine) given some reasonable number of *challenge-response pairs* (CRPs).

Using APUFs as building blocks, an XOR APUF parallelizes multiple APUF instances, passes the same challenge to all of them, and XOR-sums up their responses as the overall response. It is straightforward that the model of an m-bit k-XOR APUF is

$$r(\mathbf{c}) = \bigoplus_{j=1}^{k} r_j(\mathbf{c}) \,, \tag{3}$$

where r_j is the model of the j-th m-bit APUF instance. The complexity of modeling XOR APUFs is believed to be exponential in k if given merely CRPs ([11], [13]). Nevertheless, [3] proposes a novel reliability-based machine learning approach that leverages the *reliability* information, i.e., the probability of getting the same response under a challenge. It is empirically verified ([1], [3]) that the complexity of modeling

XOR APUFs is no longer exponential in k following this reliability-based approach.

The iPUF uses two XOR APUF instances as building blocks; an (x, y)-iPUF works as described by the model

$$r(\mathbf{c}) = r_2(\mathbf{c}_1, r_1(\mathbf{c}), \mathbf{c}_2) \,, \tag{4}$$

where r_1 and r_2 are the models of respectively an x-XOR APUF and a y-XOR APUF, and $(\mathbf{c}_1, \mathbf{c}_2)$ is a (fixed) split of $\mathbf{c}$. (x, y)-iPUF is believed to enjoy the same machine learning complexity as $(x/2+y)$-XOR APUF while moreover resilient to reliability-based attacks. Despite the fact that [2], [14] find novel attack strategies against iPUFs, [14] acknowledges that certain iPUF variants with more complicated structures are still considered secure.

We now present the shift PUF, an enhancement to the APUF. The shift PUF composes the APUF with a (without loss of generality, left) *circular shift* operation, as illustrated by Figure 1; formally, an m-bit shift PUF contains an underlying m-bit APUF, and its challenge-response behavior is modeled by

$$\Delta_{\text{Shift}}(\mathbf{c}) = \Delta(\mathbf{c}^{(\ell)}) \,, \quad r(\mathbf{c}) = [\Delta_{\text{Shift}}(\mathbf{c}) \geq 0] \,, \tag{5}$$

where $\mathbf{c}^{(\ell)}$ is the (left) circular shift of $\mathbf{c}$, with ℓ being the displacement of the shift. Obviously, if the displacement ℓ is known by the attacker, then the enhancement brought by the shift PUF completely degenerates as the attacker could easily preprocess out the effect of the circular shift. Recall that securely generating some secret ℓ is exactly among the applications of PUFs, so we propose to use PUF-based key generation (see e.g. [8, Chapter 6], based on fuzzy extractors ([4])) on the underlying APUF[1] to determine ℓ; also note that ℓ is only $\log m$ bits long, and thus this approach will not harm the efficiency badly.

We then elaborate some evidences supporting the conjecture that the shift PUF enhances the APUF by squaring m, i.e., an m-bit shift PUF is no less secure than a $\Theta(m^2)$-bit APUF. As a remark, the enhancement of $\Theta(m^2)$ is the best one could expect for the shift PUF, as the attacker can always enumerate all m possibilities of the displacement ℓ (and assume any machine learning attack requires $\Omega(m)$ efforts). We are going to show that the attacker cannot shortcut this enumeration in certain sense. By definition,

$$\Delta(\mathbf{c}^{(\ell)}) = \mathbf{w}^\top \mathbf{p}^{(\ell)} \,, \tag{6}$$

(note that $\mathbf{p}^{(\ell)}$ is the corresponding parity vector of $\mathbf{c}^{(\ell)}$ instead of the circular shift of $\mathbf{p}$) and by calculation,

$$\mathbf{p}^{(\ell)} = \begin{bmatrix} p_{\ell+1}(p_1 p_{\ell+1}) \cdots p_m(p_1 p_{\ell+1}) & p_1 p_{\ell+1} \cdots p_\ell p_{\ell+1} & 1 \end{bmatrix}^\top . \tag{7}$$

The challenge-response behavior has the form of a linear classification model in terms of $\mathbf{p}^{(\ell)}$, but however ℓ is unknown to the attacker and therefore[2] linear methods are not applicable. Since it does not matter to add unrelated parameters in linear classification, the attacker could form a linear classification models using all $\mathbf{p}^{(\ell)}$ for all ℓ to eliminate the unknown ℓ (which is essentially enumerating all possibilities of ℓ). By listing all $\mathbf{p}^{(\ell)}$ for all $\ell = 0, 1, \ldots, m-1$ together as follows,

$$\begin{bmatrix} \mathbf{p}^{(0)} & \mathbf{p}^{(1)} & \cdots & \mathbf{p}^{(m-1)} \end{bmatrix} = \begin{bmatrix} p_1 & p_1 & p_1 & \cdots & p_1 & p_1 \\ p_2 & p_1p_2p_3 & p_1p_3p_4 & \cdots & p_1p_{m-1}p_m & p_mp_1 \\ p_3 & p_1p_2p_4 & p_1p_3p_5 & \iddots & p_{m-1}p_1 & p_mp_2 \\ \vdots & \vdots & \iddots & \iddots & \vdots & \vdots \\ p_{m-1} & p_1p_2p_m & p_3p_1 & \cdots & p_{m-1}p_{m-3} & p_mp_{m-2} \\ p_m & p_2p_1 & p_3p_2 & \cdots & p_{m-1}p_{m-2} & p_mp_{m-1} \\ 1 & 1 & 1 & \cdots & 1 & 1 \end{bmatrix} , \tag{8}$$

it can be observed that all terms except the first and last lines are distinct and there are $m(m-1)+2 = \Theta(m^2)$ different terms. Hence the attacker cannot circumvent the $\Theta(m^2)$ enhancement.

By substituting APUFs with shift PUFs, all arbiter-based PUF designs might benefit from this squaring enhancement. We remark that the enhancement is not necessarily turning insecure design into secure design (for polynomial squared is still polynomial), but is targeting at substantially adding difficulties to the attacks, or reducing the time and/or the circuit complexities of the hardware while preserving the machine learning complexity. Taking the XOR APUF as an example, in response-based attacks, its machine learning complexity is believed to be $m^{\Theta(k)}$. Then squaring m is equivalently doubling k, which might help alleviate the manufacturing issue that k-XOR APUF for large k has dramatically worsening reliability due to the interference among the internal APUF instances ([13]). More interestingly, for (empirically) secure PUF designs such as iPUF, the squaring enhancement could further consolidate the intractability of attacks on them, further protecting them from smarter brute-force attacks, or alternatively, as previously mentioned, reduce the hardware costs while maintaining a matching machine learning complexity.

ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China (No. 61772035).

REFERENCES

[1] Georg T. Becker. The gap between promise and reality: On the insecurity of XOR arbiter pufs. In *CHES*, volume 9293 of *Lecture Notes in Computer Science*, pages 535–555. Springer, 2015.

[2] Durba Chatterjee, Debdeep Mukhopadhyay, and Aritra Hazra. Interpose PUF can be PAC learned. *IACR Cryptol. ePrint Arch.*, 2020:471, 2020.

[3] Jeroen Delvaux and Ingrid Verbauwhede. Side channel modeling attacks on 65nm arbiter pufs exploiting CMOS device noise. In *HOST*, pages 137–142. IEEE Computer Society, 2013.

[1] It is also valid to use a separate APUF instance or some other kind of PUF to securely generate ℓ. This approach might (slightly) improve the time performance due to the parallelism, but is less efficient in terms of circuit complexity and potentially opens extra side channels to the attackers.

[2] Say if ℓ is generated using PUF-based key generation with the underlying APUF, then the attacker needs to model the underlying APUF accurately to compute ℓ, while the attacker also needs to know ℓ first in order to model the APUF using linear methods.

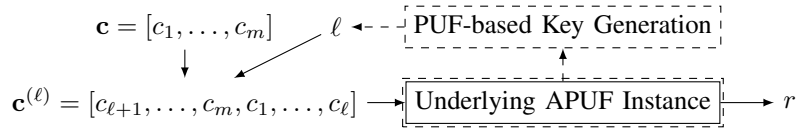

Fig. 1. Structure of the shift PUF: the challenge $\mathbf{c}$ is (left-)shifted by ℓ bits before being sent into the underlying APUF instance, whose response r is the overall response of the shift PUF; ℓ might be determined using PUF-based key generation on the underlying APUF instance.

[4] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam D. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.*, 38(1):97–139, 2008. Preliminary version in EUROCRYPT 2004.

[5] Blaise Gassend, Dwaine E. Clarke, Marten van Dijk, and Srinivas Devadas. Silicon physical random functions. In *ACM Conference on Computer and Communications Security*, pages 148–160. ACM, 2002.

[6] Blaise Gassend, Daihyun Lim, Dwaine E. Clarke, Marten van Dijk, and Srinivas Devadas. Identification and authentication of integrated circuits. *Concurr. Pract. Exp.*, 16(11):1077–1098, 2004.

[7] Daihyun Lim, Jae W. Lee, Blaise Gassend, G. Edward Suh, Marten van Dijk, and Srinivas Devadas. Extracting secret keys from integrated circuits. *IEEE Trans. Very Large Scale Integr. Syst.*, 13(10):1200–1205, 2005.

[8] Roel Maes. *Physically Unclonable Functions - Constructions, Properties and Applications*. Springer, 2013.

[9] Mehrdad Majzoobi, Farinaz Koushanfar, and Miodrag Potkonjak. Lightweight secure pufs. In *ICCAD*, pages 670–673. IEEE Computer Society, 2008.

[10] Phuong Ha Nguyen, Durga Prasad Sahoo, Chenglu Jin, Kaleel Mahmood, Ulrich Rührmair, and Marten van Dijk. The interpose PUF: secure PUF design against state-of-the-art machine learning attacks. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2019(4):243–290, 2019.

[11] Ulrich Rührmair, Frank Sehnke, Jan Sölter, Gideon Dror, Srinivas Devadas, and Jürgen Schmidhuber. Modeling attacks on physical unclonable functions. In *ACM Conference on Computer and Communications Security*, pages 237–249. ACM, 2010.

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

[13] Nils Wisiol and Marian Margraf. Why attackers lose: design and security analysis of arbitrarily large XOR arbiter pufs. *J. Cryptographic Engineering*, 9(3):221–230, 2019. Preliminary version in CHES 2017.

[14] Nils Wisiol, Christopher Mühl, Niklas Pirnay, Phuong Ha Nguyen, Marian Margraf, Jean-Pierre Seifert, Marten van Dijk, and Ulrich Rührmair. Splitting the interpose PUF: A novel modeling attack strategy. *IACR Cryptol. ePrint Arch.*, 2019:1473, 2019.