Machine Learning Attacks on PolyPUFs, OB-PUFs, RPUFs, LHS-PUFs, and PUF–FSMs*

Jeroen Delvaux

imec-COSIC, KU Leuven, Belgium,
PACE, Nanyang Technological University, Singapore, jdelvaux@ntu.edu.sg

Abstract. A *physically unclonable function* (PUF) is a circuit of which the input–output behavior is designed to be sensitive to the random variations of its manufacturing process. This building block hence facilitates the authentication of any given device in a population of identically laid-out silicon chips, similar to the biometric authentication of a human. The focus and novelty of this work is the development of efficient impersonation attacks on the following five PUF-based authentication protocols: (1) the so-called PolyPUF protocol of Konigsmark, Chen, and Wong, as published in the IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems in 2016, (2) the so-called OB-PUF protocol of Gao, Li, Ma, Al-Sarawi, Kavehei, Abbott, and Ranasinghe, as presented at the IEEE conference PerCom 2016, (3) the so-called RPUF protocol of Ye, Hu, and Li, as presented at the IEEE conference AsianHOST 2016, (4) the so-called LHS-PUF protocol of Idriss and Bayoumi, as presented at the IEEE conference RFID-TA 2017, and (5) the so-called PUF–FSM protocol of Gao, Ma, Al-Sarawi, Abbott, and Ranasinghe, as published in the IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems in 2018. The common flaw of all five designs is that the use of lightweight obfuscation logic provides insufficient protection against machine learning attacks.

Keywords: physically unclonable functions · entity authentication · machine learning

1 Introduction

Since their advent in the early 2000s [LDT00], *physically unclonable functions* (PUFs) have been used as a building block in numerous authentication protocols. The authentication is either unilateral, i.e., one-way, or mutual, i.e., two-way, and usually takes place between a low-cost, resource-constrained device hosting a PUF and a high-cost, resource-rich server storing a selection of the input–output pairs of this PUF. The selected pairs embody a shared secret between both parties, and a device is hence not required to store a secret key in *non-volatile memory* (NVM). This way, physically invasive attacks that, e.g., optically scan the cell contents of an NVM or microprobe its bus [Sko05], are precluded. The output of a PUF, however, is noisy and hinders the design of a serviceable protocol. Moreover, to avoid the amplification of noise, a PUF is highly constrained in its use of non-linear operations and is therefore prone to machine learning. Stated otherwise, the level of *diffusion* and *confusion* that can be achieved by a PUF is no match for a properly designed cipher.

Delvaux et al. [Del17, Chapter 5] analyzed the security and practicality of 21 PUF-based authentication protocols, thereby revealing numerous problems to the extent that only six candidates survive. In parallel, Becker [Bec15a, Bec15b] and Tobisch [TB15] pushed the

*If you downloaded this file from any source other than https://eprint.iacr.org/, please check the previous link to ensure that your version is the latest one.

boundaries of machine learning attacks on PUF-based protocols. The previous analyses, however, are not up-to-date with proposals beyond the year 2014. In this work, we illustrate that the research field of developing new PUF-based authentication protocols remains a minefield. Efficient attacks on the PolyPUF protocol of Konigsmark et al. [KCW16], the OB-PUF protocol of Gao et al. [GLM+16], the RPUF protocol of Ye et al. [YHL16], the LHS-PUF protocol of Idriss and Bayoumi [IB17], and the PUF–FSM protocol of Gao et al. [GMA+18] are presented. More precisely, our examination reveals that all five designs are unsuccessful attempts to impede machine learning attacks through the use of lightweight obfuscation logic.

The remainder of this paper is organized as follows. Section 2 introduces the notation and provides preliminaries. Section 3 specifies and obliterates the five protocols. Section 4 discusses the aftermath from the perspective of a system provider. Section 5 concludes this work.

2 Preliminaries

2.1 Notation

As exemplified in Table 1, constants are denoted by characters from the Greek alphabet, whereas random variables and their outcomes are denoted by characters from the Latin alphabet. Scalars are denoted by normal lowercase characters. Vectors are denoted by bold-faced, lowercase characters. All vectors are row vectors. The all-zeros vector is denoted by $\mathbf{0}$. Matrices are denoted by bold-faced, uppercase characters. The $\lambda \times \lambda$ identity matrix is denoted by $\mathbf{I}_\lambda$. A diagonal matrix is defined by listing the entries on its main diagonal, e.g., $\mathbf{I}_2 = \mathrm{diag}(1, 1)$.

Table 1: Symbols used to denote constants and variables

	Constant	Outcomes of random variables $A, B, C, \cdots$
Scalar	$\alpha, \beta, \gamma, \cdots$	$a, b, c, \cdots$
Vector	$\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}, \cdots$	$\mathbf{a}, \mathbf{b}, \mathbf{c}, \cdots$
Matrix	$\mathbf{A}, \mathbf{B}, \mathbf{\Gamma}, \cdots$	$\mathbf{A}, \mathbf{B}, \mathbf{C}, \cdots$

A set, often but not necessarily referring to all possible outcomes of a random variable, is denoted by an uppercase, calligraphic character, e.g., $\mathcal{X}$. The set of all λ-bit vectors is denoted by $\{0, 1\}^\lambda$. A multivariate normal random variable X with mean $\boldsymbol{\mu}$ and covariance matrix $\mathbf{\Sigma}$ is denoted by $X \sim N(\boldsymbol{\mu}, \mathbf{\Sigma})$. The expected value of a random variable X is denoted by $\mathbb{E}_{x \leftarrow X}[X]$. For binary vectors, bitwise inversion and bitwise modulo-2 addition are denoted by $\neg\mathbf{x}$ and $\mathbf{x}_1 \oplus \mathbf{x}_2$ respectively. Custom-defined functions are printed in a sans-serif font, e.g., Hamming distance $\mathsf{HD}(\mathbf{x}_1, \mathbf{x}_2)$.

2.2 Arbiter PUF

A PUF maps a binary input, i.e., the so-called challenge $\mathbf{c} \in \{0, 1\}^\lambda$, to a binary, device-specific output, i.e., the so-called response $\mathbf{r} \in \{0, 1\}^\eta$. There is a special interest for PUFs that support a large-sized challenge $\mathbf{c}$, e.g., having $\lambda = 128$, because this facilitates the design of an authentication protocol considerably. Even those who are given unrestricted access to such a PUF can neither gather nor tabulate all of its *challenge–response pairs* (CRPs) within the lifetime of its hosting device. For the well-known Arbiter PUF [Lim04], which quantizes the difference v between the propagation delays of two reconfigurable paths as is shown in Fig. 1, a large λ can be supported. The challenge $\mathbf{c}$ determines for each out of λ switching elements whether path segments are crossed or uncrossed.

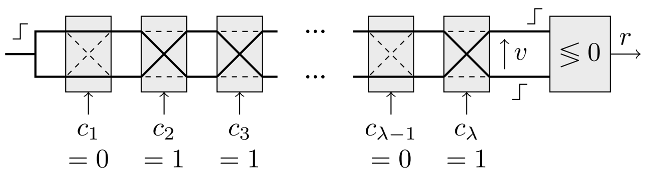


Figure 1: An Arbiter PUF with λ stages [Lim04].

If the delay difference $v > 0$, the single-bit response $r = 1$; otherwise, $r = 0$. To resist brute-force attacks, protocols usually require a long response $\mathbf{r}$, e.g., having $\eta = 128$. This expansion can be achieved either by laying out η Arbiter PUFs in parallel, or by concatenating the response bits r of a single Arbiter PUF that evaluates η challenges $\mathbf{c}$. Unfortunately, noise sources within the device, as well as changes to its external environment, imply that an initially generated response $\mathbf{r}$ slightly differs from its reproduction $\tilde{\mathbf{r}}$. The averaged *bit error rate* $\mathbb{E}_{\mathbf{c}\leftarrow\{0,1\}^\lambda}[\mathsf{HD}(R, \tilde{R})]/\eta$ typically lies between 5% and 15%. A crucial insight is that the reproducibility of the response r to a given challenge $\mathbf{c}$ increases monotonically with the absolute value $|v|$. A continuous spectrum ranging from highly stable to highly noisy response bits hence arises.

2.3 Correlations and Machine Learning

Unfortunately, the 2^λ CRPs $(\mathbf{c}, r)$ of an Arbiter PUF are all determined by the variability of a limited number of circuit elements, and are hence strongly correlated. Numerous authors have experimentally confirmed that the value of v can be accurately described by a dot product: $v = \mathbf{m}\,\mathbf{s}^T$ in (1), where variability model $\mathbf{m} \in \mathbb{R}^{\lambda+1}$ aggregates differences t between the propagation delays of the logic gates that constitute each stage as defined in Fig. 2 and where $\mathbf{s} \in \{-1,1\}^{\lambda+1}$ is the result of an invertible challenge transformation. To incorporate the effect of both internal noise sources and environmental changes, the latter of which are assumed to be centered around a constant nominal value, the quantization can be extended to $(v + n) \lessgtr 0$, where $N \sim N(0, \sigma_n^2)$ with respect to the infinite set of evaluations [Mae13].

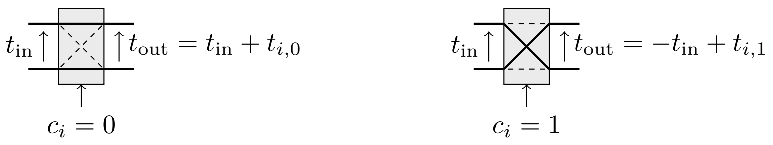

t_{in}
$t_{\text{out}} = t_{\text{in}} + t_{i,0}$
$c_i = 0$
t_{in}
$t_{\text{out}} = -t_{\text{in}} + t_{i,1}$
$c_i = 1$

Figure 2: The delay behavior of a single stage of an Arbiter PUF.

$$v = \mathbf{m}\,\mathbf{s}^T, \quad \text{where } \mathbf{m} = \mathbf{t}\,\mathbf{\Psi},$$
$$\mathbf{t} = \begin{pmatrix} t_{1,0} & t_{1,1} & t_{2,0} & t_{2,1} & \dots & t_{\lambda,0} & t_{\lambda,1} \end{pmatrix}, \mathbf{\Psi} =$$
$$\frac{1}{2}\begin{pmatrix} 1 & -1 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & -1 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & -1 & \dots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots & 1 & 1 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 1 & 1 \end{pmatrix}^T, \tag{1}$$
$$\text{and } \mathbf{s} = \begin{pmatrix} (-1)^{c_1 \oplus c_2 \oplus \cdots \oplus c_\lambda} \\ (-1)^{c_2 \oplus c_3 \oplus \cdots \oplus c_\lambda} \\ \vdots \\ (-1)^{c_\lambda} \\ 1 \end{pmatrix}^T.$$

For a population of ideally manufactured Arbiter PUFs, it holds that $T \sim N(\mathbf{0}, \sigma_t^2\,\mathbf{I}_{2\lambda})$ and hence $M \sim N\big(\mathbf{0}\,\mathbf{\Psi},\, \sigma_t^2\,\mathbf{\Psi}^T\,\mathbf{I}_{2\lambda}\,\mathbf{\Psi}\big) \sim N\big(\mathbf{0}, \sigma_t^2\ \mathrm{diag}(1/2, 1, 1, \cdots, 1, 1/2)\big)$. Given that only the sign of the delay difference v matters in determining the nominal value of its corresponding response r, one may arbitrarily choose $\sigma_t^2 = 1$ as long as σ_n^2 is scaled accordingly. The previous variability model M implies that infinitely large populations of Arbiter PUFs and *random oracles* [BR93] substantially differ in their statistical properties. For the population of random oracles, the probability $\rho_{\text{flip}} = \mathbb{E}_{\mathbf{m} \leftarrow M}[R_1 \oplus R_2] = 1/2$ for any given challenge pair $(\mathbf{c}_1, \mathbf{c}_2)$ where $\mathbf{c}_1 \neq \mathbf{c}_2$. For the population of Arbiter PUFs, however, ρ_{flip} increases roughly proportionally with $\mathsf{HD}(\mathbf{s}_1, \mathbf{s}_2) \in [1, \lambda]$ such that the interval $[0, 1]$ is quasi completely covered [MKP08, Fig. 12].

Another manifestation of the correlated structure is that machine learning algorithms training on a relatively small set of CRPs, i.e., $\{(\mathbf{c}_1, r_1), (\mathbf{c}_2, r_2), \cdots, (\mathbf{c}_\omega, r_\omega)\}$ where $\omega \ll 2^\lambda$, can produce a model $\hat{\mathbf{m}}$ that allows to accurately predict the unseen response $\mathbf{r}_{\omega+1}$ to any given challenge $\mathbf{c}_{\omega+1}$. The probability $p_{\text{acc}} \in [1/2, 1]$ that a prediction for a given Arbiter PUF is correct is referred to as the accuracy, and usually increases monotonically with ω. If pairs $(\mathbf{s}, r)$ instead of pairs $(\mathbf{c}, r)$ are used as training data, the problem of learning $\mathbf{m}$ becomes quasi-linear, i.e., the quantization $v \lessgtr 0$ is the only remaining non-linearity, and hence straightforward to handle for numerous algorithms. This includes the use of *artificial neural networks* (ANNs), *support vector machines* (SVMs), and *logistic regression*. Even for Arbiter PUFs having $\lambda = 128$ stages, a set of $\omega = 10^3$ CRPs $(\mathbf{s}, r)$ suffices to obtain accuracies $\mathbb{E}_{\mathbf{m} \leftarrow M}[P_{\text{acc}}] \geq 90\%$.

Thanks to existing validations with experimental data, it has become a common practice to demonstrate the feasibility of a machine learning attack on randomly generated instances of the mathematical abstraction M. This favors both the reproducibility and the comparability of results, and it also excludes the possibility that a flaw in the circuit or the layout of a given Arbiter PUF implementation facilitates attacks. Noise sources, however, pollute both training and testing data $(\mathbf{s}, r)$, so if omitted from the mathematical abstraction, the reported learning efficiency is usually slightly higher than for experimental data.

In an attempt to resist machine learning attacks, numerous variations of the Arbiter PUF have been proposed. For one version of the so-called γ-XOR PUF [RSS+13], for example, $\gamma > 1$ identically laid-out Arbiter PUFs evaluate a common challenge $\mathbf{c}$ and the eventually released response bit is determined as $r = r_1 \oplus r_2 \oplus \cdots \oplus r_\gamma$. Unfortunately, such variations also increase the bit error rate and the footprint of the PUF to the extent that serviceable, lightweight designs remain learnable. An alternative or complementary line of defense, which is the topic of this paper, is the design of authentication protocols that either keep the response bits r of a PUF internal to its hosting device or obfuscate the link between the public challenges $\mathbf{c}$ and the released response bits r. The latter strategy usually entails the use of a *true random number generator* (TRNG). As demonstrated by Becker [Bec15b] and Tobisch [TB15], however, the release of variables that are correlated to r might still enable a modeling attack. For example, if the protocol leaks the error rate p_{error} of a hidden response bit r, an estimate of the absolute value $|v|$ can still be obtained. Noise sources might hence help rather than hinder an attacker.

2.4 Attacker Model

The analyzed authentication protocols adopt a frequently used attacker model [Del17, Chapter 5]. The enrollment of a PUF-enabled device takes place in a secure environment, and afterwards, an interface for accessing the CRPs might have to be irreversibly disabled. In the field, the protocols should resist both impersonation and denial-of-service attacks. Given that the device comprises a smart card, a *radio-frequency identication* tag, or another mobile entity, it is assumed that an attacker may obtain physical access. The server, however, features both secure computations and secure storage. The communication channel between both parties is assumed to be insecure. This implies that an attacker may not only eavesdrop on a genuine protocol run, but also manipulate, inject, and block messages.

3 Protocols

To facilitate the understanding of the analyzed authentication protocols for a visually oriented reader, Fig. 3 shows the hardware of a PUF-enabled device. The implementation efficiency is evidently reflected but is of secondary importance in light of the newly revealed security issues.

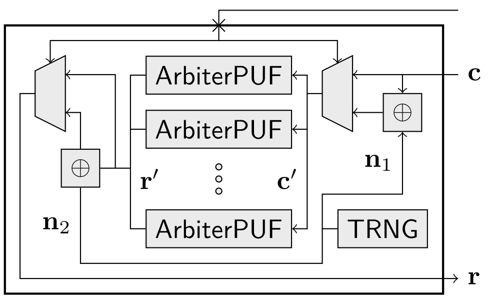
For each protocol, we devise a method for training an accurate predictive model $\hat{\mathbf{m}}$ of the underlying Arbiter PUFs. This model $\hat{\mathbf{m}}$ allows the attacker to successfully impersonate the device. For the LHS-PUF [IB17] and PUF–FSM [GMA+18] protocols, which both aim to provide mutual authentication, the server can be impersonated as well. Although the protocols are specified and attacked in chronological order, there is no problem in reading Sections 3.1 to 3.5 in a different order.

3.1 PolyPUF

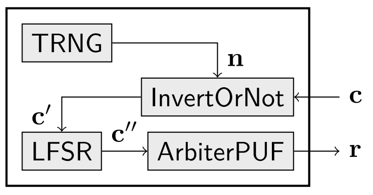
3.1.1 Specification

The so-called PolyPUF protocol of Konigsmark, Chen, and Wong [KCW16], where "Poly" stands for "Polymorphic", is specified in Fig. 4. Each device hosts λ Arbiter PUFs that evaluate a common challenge $\mathbf{c}' \in \{0,1\}^\lambda$. Suggested values for λ are 32 and 64. To enroll a given device, the server collects ω CRPs $(\mathbf{c}', \mathbf{r}')$ and trains a predictive model $\hat{\mathbf{m}}$ for each Arbiter PUF. A suggested value for ω is 5000. After the enrollment, direct access to the CRPs $(\mathbf{c}', \mathbf{r}')$ is irreversibly disabled.

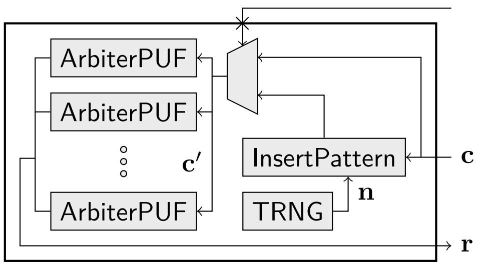
To preclude machine learning attacks, a device that is deployed in the field XORs the received challenge $\mathbf{c} \in \{0,1\}^\lambda$ with λ/γ concatenated copies of a nonce $\mathbf{n}_1 \in \{0,1\}^\gamma$ in order to form the PUF input $\mathbf{c}'$. Likewise, the released response $\mathbf{r} \in \{0,1\}^\lambda$ is the result of XORing the PUF output $\mathbf{r}'$ with λ/δ concatenated copies of a nonce $\mathbf{n}_2 \in \{0,1\}^\delta$. Suggested values for γ and δ are 2 and 3 respectively. The authors do not comment on the fact that $\lambda \in \{32, 64\}$ is not an integer multiple of $\delta = 3$; we therefore assume that one copy of $\mathbf{n}_2$ is truncated to $\mathsf{mod}(\lambda, \delta) \in \{2, 1\}$ bits. To save resources, the $\gamma + \delta$ random bits could be generated by XORing unstable responses bits r rather than through a dedicated



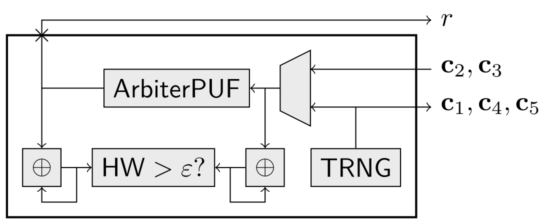
(a) PolyPUF of Konigsmark et al. [KCW16].



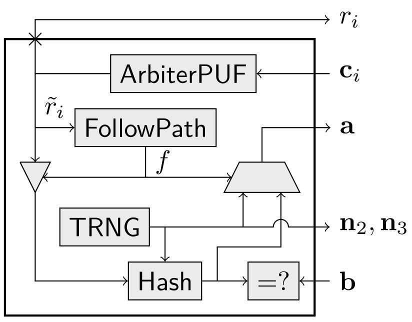
(c) RPUF of Ye et al. [YHL16].



(b) OB-PUF of Gao et al. [GLM+16].


r
ArbiterPUF
$\mathbf{c}_2, \mathbf{c}_3$
$\mathbf{c}_1, \mathbf{c}_4, \mathbf{c}_5$
⊕
HW > ε?
⊕
TRNG

(d) LHS-PUF of Idriss and Bayoumi [IB17].



(e) PUF–FSM of Gao et al. [GMA+18].

Figure 3: The hardware of a PUF-enabled device for the analyzed authentication protocols. Intermediary registers and control logic are not drawn. The symbol $\times$ on the boundary of a device denotes a one-time interface that is irreversibly disabled after the enrollment.

TRNG. This solution, however, requires that a well-chosen challenge $\mathbf{c}'$ is programmed into the device during the enrollment. To authenticate a device, the server checks whether the response $\mathbf{r}$ to a randomly chosen challenge $\mathbf{c}$ matches with at least one out $2^{\gamma+\delta}$ possible responses $\hat{\mathbf{r}}$. To account for the noisiness of the PUFs, only an approximate match is required as reflected by the Hamming distance threshold ε.

The authors experiment with ANNs in order to validate the security of their protocol. Most notably, they attempt to exploit the statistical weaknesses of the underlying Arbiter PUFs in gathering a set of $\omega^\star$ training CRPs $(\mathbf{c}_i, \mathbf{r}_i)$ where the nonces $(\mathbf{n}_1, \mathbf{n}_2)$ are supposed to remain unchanged. For this purpose, challenge $\mathbf{c}_1$ is chosen uniformly at random from $\{0,1\}^\lambda$, and all other challenges $\mathbf{c}_i$, where $i \in [2, \omega^\star]$, are randomly chosen such that

```
PUF-enabled device        <--verifies--        Server

Enrollment
                          <--- $\mathbf{c}'_i$ ---    $\mathbf{c}'_i \leftarrow \mathsf{TRNG}(\lambda)$          } $\forall i \in [1, \omega]$
for $k \leftarrow 1$ to $\lambda$ do                                                                }
    $r'_k \leftarrow \mathsf{ArbiterPUF}_k(\mathbf{c}'_i)$                                                 }
$\mathbf{r}'_i \leftarrow (r'_1\, r'_2\, \cdots\, r'_\lambda)$   --- $\mathbf{r}'_i$ --->                                       }
Disable direct access                          for $k \leftarrow 1$ to $\lambda$ do
  to $\mathbf{c}'$ and $\mathbf{r}'$                                $\hat{\mathbf{m}}_k \leftarrow \mathsf{TrainModel}($
                                                   $\mathbf{c}'_1, r'_{1,k}, \cdots, \mathbf{c}'_\omega, r'_{\omega,k})$

Authentication ($\infty$ times)
                          <--- $\mathbf{c}$ ---     $\mathbf{c} \leftarrow \mathsf{TRNG}(\lambda)$
$\mathbf{n}_1 \leftarrow \mathsf{TRNG}(\gamma)$
$\mathbf{c}' \leftarrow \mathbf{c} \oplus (\mathbf{n}_1\, \mathbf{n}_1\, \cdots)$
for $k \leftarrow 1$ to $\lambda$ do
    $\tilde{r}'_k \leftarrow \mathsf{ArbiterPUF}_k(\mathbf{c}')$
$\tilde{\mathbf{r}}' \leftarrow (\tilde{r}'_1\, \tilde{r}'_2\, \cdots\, \tilde{r}'_\lambda)$
$\mathbf{n}_2 \leftarrow \mathsf{TRNG}(\delta)$
$\tilde{\mathbf{r}} \leftarrow \tilde{\mathbf{r}}' \oplus (\mathbf{n}_2\, \mathbf{n}_2\, \cdots)$   --- $\tilde{\mathbf{r}}$ --->
                                               $h \leftarrow \lambda + 1$
                                               foreach $\mathbf{n}_1 \in \{0,1\}^\gamma$ do
                                                   $\mathbf{c}' \leftarrow \mathbf{c} \oplus (\mathbf{n}_1\, \mathbf{n}_1\, \cdots)$
                                                   for $k \leftarrow 1$ to $\lambda$ do
                                                       $\hat{r}'_k \leftarrow \mathsf{Predict}(\hat{\mathbf{m}}_k, \mathbf{c}')$
                                                   $\hat{\mathbf{r}}' \leftarrow (\hat{r}'_1\, \hat{r}'_2\, \cdots\, \hat{r}'_\lambda)$
                                                   foreach $\mathbf{n}_2 \in \{0,1\}^\delta$ do
                                                       $\hat{\mathbf{r}} \leftarrow \hat{\mathbf{r}}' \oplus (\mathbf{n}_2\, \mathbf{n}_2\, \cdots)$
                                                       $h \leftarrow \min(h, \mathsf{HD}(\tilde{\mathbf{r}}, \hat{\mathbf{r}}))$
                                               if $h > \varepsilon$ then Reject
```

Figure 4: The PolyPUF protocol of Konigsmark et al. [KCW16].

$\mathsf{HD}(\mathbf{c}_i, \mathbf{c}_{i-1}) = 1$. Out of $2^{\gamma+\delta}$ unique responses $\mathbf{r}_i \in \{0,1\}^\lambda$, the one value that minimizes $\mathsf{HD}(\mathbf{r}_i, \mathbf{r}_{i-1})$ is retained. The authors are delighted that, even with $\tau^\star = 10^8$ device queries and 10–30 neurons in the hidden layer, the obtained modeling accuracies p_{acc} do not significantly exceed the ideal value of 50%.

3.1.2 Attack

We point out that the authors' non-functional attack can be functionalized through a minimal modification. Given a proper understanding of the challenge transformation in (1), it is evident that an attacker should choose consecutive challenges $(\mathbf{c}_i, \mathbf{c}_{i-1})$ such that the Hamming distance $\mathsf{HD}(\mathbf{s}_i, \mathbf{s}_{i-1}) = 1$ rather than $\mathsf{HD}(\mathbf{c}_i, \mathbf{c}_{i-1}) = 1$. If nonce $\mathbf{n}_1$ remains unchanged, it holds for the former case that $\mathsf{HD}(\mathbf{s}'_i, \mathbf{s}'_{i-1}) = 1$, and the value of $\mathsf{HD}(\mathbf{r}'_i, \mathbf{r}'_{i-1})$ is hence expected to be small. If nonce $\mathbf{n}_2$ remains unchanged as well, it follows that an equally small Hamming distance $\mathsf{HD}(\mathbf{r}_i, \mathbf{r}_{i-1})$ is output by the device. Thus, an attacker can assume that if $\mathsf{HD}(\mathbf{r}_i,\, \mathbf{r}_{i-1}) \leq \varepsilon_1^\star$, where $\varepsilon_1^\star$ is a well-chosen threshold, that nonces $(\mathbf{n}_1, \mathbf{n}_2)$ remained unaltered.

The main concern, however, is that a single wrongly selected response $\mathbf{r}_i$ could suffice to corrupt the whole training set. The Monte Carlo experiment in Fig. 5 demonstrates that corruptions are not likely to occur. For each out of 10^5 sets of λ randomly generated PUFs $M \sim N\big(\mathbf{0},\, \mathrm{diag}(1/2, 1, 1, \cdots, 1, 1/2)\big)$, a challenge pair $(\mathbf{c}_i, \mathbf{c}_{i-1})$ is randomly chosen such that $\mathsf{HD}(\mathbf{s}_i, \mathbf{s}_{i-1}) = 1$, and nonces $\mathbf{n}_{1,i-1}$ and $\mathbf{n}_{2,i-1}$ are chosen uniformly at random from $\{0,1\}^\gamma$ and $\{0,1\}^\delta$ respectively. For each combination of nonce differences $(\mathbf{n}_{1,i} \oplus \mathbf{n}_{1,i-1}) \in \{0,1\}^\gamma$ and $(\mathbf{n}_{2,i} \oplus \mathbf{n}_{2,i-1}) \in \{0,1\}^\delta$, the estimated *probability mass function* of $\mathsf{HD}(R_i, R_{i-1})$ is shown. It benefits an attacker that the first and the second curves from the left can easily be distinguished. As a side note, the 1-bit offsets among the curves with $\mathsf{HD}(\mathbf{n}_{2,i}, \mathbf{n}_{2,i-1}) \in \{1, 2\}$ exist because λ is not an integer multiple of δ.

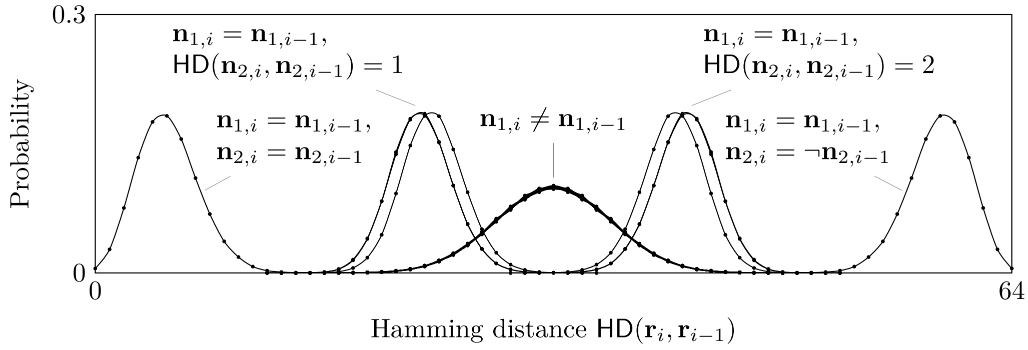


Figure 5: Feasibility study of an attack on the PolyPUF protocol, where $\lambda = 64$, $\gamma = 2$, and $\delta = 3$.

Moreover, an attacker can play safe and only add a new CRP $(\mathbf{c}_i,\, \mathbf{r}_i)$ to the training set if the difference between the smallest and the second smallest value of $\mathsf{HD}(R_i, \mathbf{r}_{i-1})$ is greater than or equal to a well-chosen threshold $\varepsilon_2^\star$. This way, Algorithm 1 is able to produce a training set of w correctly linked CRPs $(\mathbf{c}_i, \mathbf{r}_i)$ from sending approximately $\tau^\star \gg w$ queries to the PUF-enabled device. In order to maximize w for a given τ, each received response $\mathbf{r}$ is XORed with 2^δ possible patterns $(\mathbf{n}_2\, \mathbf{n}_2\, \cdots)$. There are $2^{\gamma+\delta}$ possible pairs of nonces $(\mathbf{n}_1, \mathbf{n}_2)$ that may underlie the w training CRPs $(\mathbf{c}_i, \mathbf{r}_i)$, and the attacker does not know which pair. It can, however, arbitrarily be assumed that $\mathbf{n}_1 = \mathbf{0}$ and $\mathbf{n}_2 = \mathbf{0}$, and the corresponding pairs $(\mathbf{s}_i = \mathbf{s}'_i, \mathbf{r}_i = \mathbf{r}'_i)$ are then used for training λ predictive models $\hat{\mathbf{m}}$, i.e., one for each Arbiter PUF. Given that the server iterates over $2^{\gamma+\delta}$ possible pairs $(\mathbf{n}_1, \mathbf{n}_2)$ to authenticate a device, the previous set of λ models $\hat{\mathbf{m}}$ always suffices for impersonation purposes.

In spite of what Konigsmark et al. [KCW16] suggest, there is no need for the ANN to have 10–30 neurons in the hidden layer. The bare minimum, i.e., a network consisting of a single neuron, suffices to capture the dot product $v = \mathbf{m}\, \mathbf{s}^T$ that underlies an Arbiter PUF. A minor inconvenience is that ANNs inherently serve a regression purpose rather than a classification purpose. To overcome this issue, we use *resilient backpropagation* to independently train two single-neuron networks that approximate response bits r and their inverses $\neg r$ respectively. As shown in Fig. 6, the real-valued outputs of the corresponding *activation functions* are compared to obtain a prediction $\hat{r} \in \{0, 1\}$.

Figure 7 shows the obtained modeling accuracies $\mathbb{E}[P_{\text{acc}}]$ as a function of the approximate number of device queries $\tau^\star$. Fewer than $\tau^\star = 10^5$ queries suffice to obtain accuracies $\mathbb{E}[P_{\text{acc}}] \geq 90\%$, whereas Konigsmark et al. [KCW16] were unable to exceed the ideal value of 50% using $\tau^\star = 10^8$ queries. Each dot corresponds to five runs of Algorithm 1 using

Algorithm 1: PolyPUF training set

```
$i, w \leftarrow 1$
$\mathbf{c}_1 \leftarrow \mathsf{TRNG}(\lambda)$
$\mathbf{r}_1 \leftarrow \mathsf{QueryDevice}(\mathbf{c}_1)$
while $i < \beta^\star$ do
    $j, f \leftarrow 0$
    while $(f = 0) \wedge (j < 2^\gamma)$ do
        $j \leftarrow j + 1$
        $\mathbf{c} \leftarrow \mathsf{TRNG}(\lambda)$ such that
         $\mathsf{HD}(\mathbf{s}, \mathbf{s}_w) = 1$
        $\mathbf{r} \leftarrow \mathsf{QueryDevice}(\mathbf{c})$
        $k \leftarrow 0$
        foreach $\mathbf{n}_2 \in \{0,1\}^\delta$ do
            $k \leftarrow k + 1$
            $\mathbf{a}_k \leftarrow \mathbf{r} \oplus (\mathbf{n}_2\, \mathbf{n}_2\, \cdots)$
            $h_k \leftarrow \mathsf{HD}(\mathbf{r}_w, \mathbf{a}_k)$
        Sort $h_{(1)} \leq h_{(2)} \leq \cdots \leq h_{(2^\delta)}$
        $f \leftarrow \left(h_{(1)} \leq \varepsilon_1^\star\right)$
        $f \leftarrow f \wedge \left(h_{(2)} - h_{(1)} \geq \varepsilon_2^\star\right)$
    $i \leftarrow i + j$
    if $f = 1$ then
        $w \leftarrow w + 1$
        $\mathbf{c}_w \leftarrow \mathbf{c}$
        $\mathbf{r}_w \leftarrow \mathbf{a}_{(1)}$
```

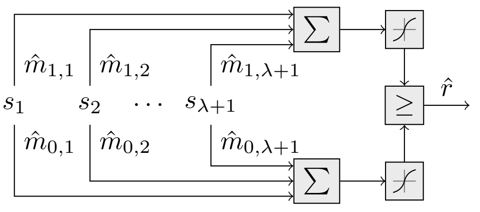


Figure 6: A pair of single-neuron networks.

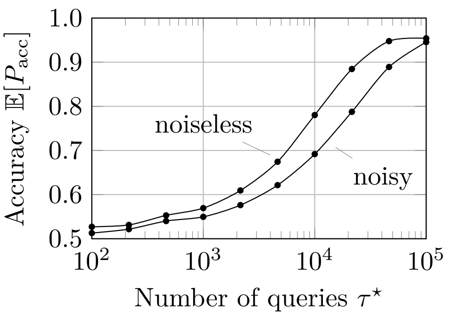


Figure 7: The accuracy of modeling an Arbiter PUF that is used in the PolyPUF protocol, where $\lambda = 64$, $\gamma = 2$, and $\delta = 3$.

different devices and hence displays the averaged accuracy of modeling $5\lambda = 320$ Arbiter PUFs; parameters were configured as $\varepsilon_1^\star = \varepsilon_2^\star = 14$. For the noisy case, the standard deviation $\sigma_n = 0.325\sqrt{\lambda}$ so that the expected error rate between a nominal response r and its reproduction $\tilde{r}$ is approximately 10%. The responses r to 1000 testing challenges $\mathbf{c}$ are all nominal values, which corresponds to the best-case scenario where the server stores infinitely precise predictive models $\hat{\mathbf{m}}$ of the λ Arbiter PUFs that are hosted by a given device.

For the sake of completeness, it is worth mentioning that although Algorithm 1 succeeds as a deobfucation tool, its robustness and its efficiency might still be open for improvement. One idea is to track all $2^\gamma = 4$ values of nonce $\mathbf{n}_1$ instead of a single value only. This implies that, in each algorithm pass, an attacker stores four ordered responses $\mathbf{r}$ to the given challenge $\mathbf{c}$. Ultimately, the four tracks will have to be combined into a single training set of CRPs. There are $(2^\gamma - 1)!\,(2^\delta)^{2^\gamma - 1} = 3072$ non-equivalent combinations of which exactly one results in server-acceptable predictive models $\hat{\mathbf{m}}$. A relatively small-sized exhaustive execution of modeling experiments hence suffices to find the one. A complementary idea is to store real-valued responses $r \in [0, 1]$ that reflect the stability, given that multiple noisy readings for each nonce $\mathbf{n}_1 \in \{0,1\}^\gamma$ might be available anyway. The Hamming distance computation $\mathsf{HD}(\mathbf{r}, \mathbf{a})$ can be generalized to $\sum_{j=1}^{\lambda} |a_j - r_j|$.

3.2 OB-PUF

3.2.1 Specification

The so-called OB-PUF protocol of Gao, Li, Ma, Al-Sarawi, Kavehei, Abbott, and Ranasinghe [GLM+16], where "OB" stands for "Obfuscated", is specified in Fig. 8. Each device hosts η Arbiter PUFs that evaluate a common challenge $\mathbf{c}' \in \{0,1\}^\lambda$. A suggested value for η is 3; a suggested value for λ is 64. To enroll a given device, the server collects ω CRPs $(\mathbf{c}', \mathbf{r})$ and trains a predictive model $\hat{\mathbf{m}}$ for each Arbiter PUF. A value for ω has not been suggested. After the enrollment, direct access to the challenge $\mathbf{c}'$ is irreversibly disabled.

```
PUF-enabled device        $\xleftarrow{\text{verifies}}$        Server

Enrollment
                              $\xleftarrow{\mathbf{c}'_i}$      $\mathbf{c}'_i \leftarrow \mathsf{TRNG}(\lambda)$             ⎫
for $k \leftarrow 1$ to $\eta$ do                                                            ⎬ $\forall i \in [1,\omega]$
    $r_k \leftarrow \mathsf{ArbiterPUF}_k(\mathbf{c}'_i)$                                        ⎪
$\mathbf{r}_i \leftarrow (r_1\, r_2 \cdots r_\eta)$       $\xrightarrow{\mathbf{r}_i}$                                        ⎭
Disable access to $\mathbf{c}'$                        for $k \leftarrow 1$ to $\eta$ do
                                                  $\hat{\mathbf{m}}_k \leftarrow \mathsf{TrainModel}($
                                                      $\mathbf{c}_1, r_{1,k}, \cdots, \mathbf{c}_\omega, r_{\omega,k})$

Authentication ($\infty$ times)
                              $\xleftarrow{\mathbf{c}}$      $\mathbf{c} \leftarrow \mathsf{TRNG}(\lambda - \delta)$
$\mathbf{n} \leftarrow \mathsf{TRNG}(\gamma)$
$\mathbf{c}' \leftarrow \mathsf{InsertPattern}(\mathbf{c}, \mathbf{n})$
for $k \leftarrow 1$ to $\eta$ do
    $\tilde{r}_k \leftarrow \mathsf{ArbiterPUF}_k(\mathbf{c}')$
$\tilde{\mathbf{r}} \leftarrow (\tilde{r}_1\, \tilde{r}_2 \cdots \tilde{r}_\eta)$       $\xrightarrow{\tilde{\mathbf{r}}}$
                                                $h \leftarrow \eta$
                                                foreach $\mathbf{n} \in \{0,1\}^\gamma$ do
                                                    $\mathbf{c}' \leftarrow \mathsf{InsertPattern}(\mathbf{c}, \mathbf{n})$
                                                    for $k \leftarrow 1$ to $\eta$ do
                                                        $\hat{r}_k \leftarrow \mathsf{Predict}(\hat{\mathbf{m}}_k, \mathbf{c}')$
                                                    $\hat{\mathbf{r}} \leftarrow (\hat{r}_1\, \hat{r}_2 \cdots \hat{r}_\eta)$
                                                    $h \leftarrow \min(h, \mathsf{HD}(\tilde{\mathbf{r}}, \hat{\mathbf{r}}))$
                                                if $h > 0$ then Reject
```

Figure 8: The OB-PUF protocol of Gao et al. [GLM+16].

To prevent an attacker from training an accurate predictive model of its PUFs, a device that is deployed in the field extends the received challenge $\mathbf{c} \in \{0,1\}^{\lambda-\delta}$ according to the value of a nonce $\mathbf{n} \in \{0,1\}^\gamma$. For the suggested values $\lambda = 64$, $\gamma = 1$, and $\delta = 5$, each PUF evaluates a challenge $\mathbf{c}' \in \{(0\,1\,0\,1\,0\,c_1\,c_2 \cdots c_{59}), (c_1\,c_2 \cdots c_{59}\,1\,0\,1\,0\,1)\}$. To authenticate a device, the server checks whether the response $\tilde{\mathbf{r}}$ to a randomly chosen challenge $\mathbf{c}$ is equal to at least one out of 2^γ predicted responses $\hat{\mathbf{r}}$. For the suggested response length $\eta = 3$, however, an attacker can easily impersonate a device through random guessing. The authors did not specify a method to scale their protocol to a comfortable security level; we assume that multiple CRPs $(\mathbf{c}, \tilde{\mathbf{r}})$ are collected in order to authenticate a device. Although the authors are aware that the responses of a PUF are noisy, their protocol also lacks a mechanism to deal with noise; we assume that a device is rejected if the number of non-matching CRPs $(\mathbf{c}, \tilde{\mathbf{r}})$ exceeds a certain threshold ε.

The authors experiment with logistic regression in order to validate the security of their protocol. For any given device, they collect the responses $\tilde{\mathbf{r}}$ to $\omega^\star$ randomly chosen challenges $\mathbf{c}$. They consider it a success that even with $\omega^\star = 10^6$ training CRPs $(\mathbf{c}, \tilde{\mathbf{r}})$, the obtained accuracy p_{acc} does not exceed 72%.

3.2.2 Attack

The authors assume that the mediocre accuracy of 72% supports their security claims, but for a conservative cryptologist any value other than 50% is symptomatic of an underlying weakness. Indeed, we now devise a learning strategy that is several orders of magnitude more efficient. Consider an attacker who obtains physical access to a PUF-enabled device and records its response $\tilde{\mathbf{r}}$ to a randomly chosen challenge $\mathbf{c} \in \{0,1\}^{\lambda-\delta}$ not once but $\beta_1^\star \gg 2^\gamma$ times. If a response bit $\tilde{r}_k$, where $k \in [1, \eta]$, remains constant for all $\beta_1^\star$ evaluations, it is likely that the corresponding Arbiter PUF has the same nominal value for the response r to all 2^γ underlying challenges $\mathbf{c}'$, and 2^γ transformed CRPs $(\mathbf{s}', r)$ can hence be appended to a training set for that particular Arbiter PUF. Algorithm 2 applies this mechanism to a list of $\omega_1^\star$ randomly chosen challenges $\mathbf{c}$ and all η Arbiter PUFs of a given device. Constant $\varepsilon_1^\star$, where $\varepsilon_1^\star \ll \beta_1^\star$, represents the maximum number of opposing evaluations such that the nominal value of response r is still deemed constant.

Algorithm 2: OB-PUF training set I

```
$w_1, w_2, \cdots, w_\eta \leftarrow 0$
for $i \leftarrow 1$ to $\omega_1^\star$ do
    $\mathbf{c} \leftarrow \mathsf{TRNG}(\lambda - \delta)$
    $\mathbf{h} \leftarrow \mathbf{0}$
    for $j \leftarrow 1$ to $\beta_1^\star$ do
        $\tilde{\mathbf{r}} \leftarrow \mathsf{QueryDevice}(\mathbf{c})$
        $\mathbf{h} \leftarrow \mathbf{h} + \tilde{\mathbf{r}}$
    for $k \leftarrow 1$ to $\eta$ do
        if $h_k \in [0, \varepsilon_1^\star] \cup [\beta_1^\star - \varepsilon_1^\star, \beta_1^\star]$
         then
            foreach $\mathbf{n} \in \{0,1\}^\gamma$ do
                $w_k \leftarrow w_k + 1$
                $\mathbf{c}'_{w_k} \leftarrow$
                 $\mathsf{InsertPattern}(\mathbf{c}, \mathbf{n})$
                if $h_k \in [0, \varepsilon_1^\star]$ then
                    $r_{w_k} \leftarrow 0$
                else
                    $r_{w_k} \leftarrow 1$
```

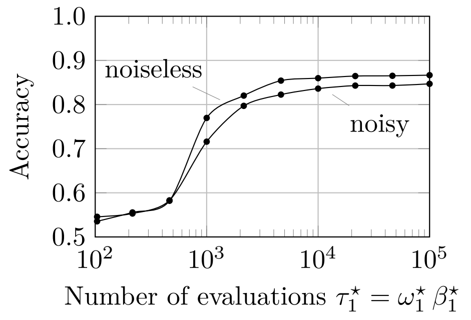


Figure 9: The accuracy of modeling an Arbiter PUF that is used in the OB-PUF protocol, where $\lambda = 64$, $\eta = 3$, and $\gamma = 1$, and $\delta = 5$.

Figure 9 shows the obtained modeling accuracies as a function of the number of device queries $\tau_1^\star = \omega_1^\star \beta_1^\star$. For each dot, we generate 20 PUFs $M \sim N\big(\mathbf{0}, \text{diag}(1/2, 1, 1, \cdots, 1, 1/2)\big)$ and average the obtained accuracies. The machine learning algorithm can be chosen arbitrarily; we opted for linear regression, as specified later-on in (2). For the noiseless case, where $\beta_1^\star = 8$ and $\varepsilon_1^\star = 0$, it can be seen that the authors' highest reported accuracy of 72% can already be exceeded after $\tau_1^\star = 10^3$ queries. From $\tau_1^\star = 10^4$ queries onwards, the accuracy reaches an upper bound of approximately 85%. To incorporate noise, we choose the standard deviation $\sigma_n = 0.325\sqrt{\lambda}$ so that the expected error rate between a nominal response r and its reproduction $\tilde{r}$ is approximately 10%. For $\beta_1^\star = 8$ and $\varepsilon_1^\star = 1$, it can be seen that learning efficiency is only slightly lower than for the noiseless case.

Increasing the value of $\beta_1^\star$ does not help in obtaining accuracies that exceed 85%. Although we confirmed this statement experimentally, a more insightful explanation for the case of a noiseless Arbiter PUF is that the probability that a training CRP $(\mathbf{s}, r)$ is corrupted is $2^{-\beta_1^\star - 1} \approx 0.2\%$ and hence negligible already. Instead, the constraint that the value of a response bit r remains constant for all $\beta_1^\star$ evaluations is presumed to be responsible for the upper bound on the accuracy. Although Algorithm 2 selects challenges $\mathbf{c}$ uniformly at random from $\{0,1\}^{\lambda-\delta}$, the subset of retained challenges $\mathbf{c}$ is not necessarily uniform anymore and might hence not capture the integral behavior of an Arbiter PUF. Optionally, one could use the obtained predictive models $\hat{\mathbf{m}}$ as a deobfuscation tool and gather the responses r to a more uniform set of challenges $\mathbf{c}$. Algorithm 3 selects challenges $\mathbf{c}$ for which all 2^γ predicted responses $\hat{\mathbf{r}}$ are far apart from each other and hence distinguishable.

Algorithm 3: OB-PUF training set II

```
$w \leftarrow 0$
for $i \leftarrow 1$ to $\tau_2^\star$ do
    do
        $\mathbf{c} \leftarrow \mathsf{TRNG}(\lambda - \delta)$
        $n \leftarrow 0$
        foreach $\mathbf{n} \in \{0,1\}^\gamma$ do
            $n \leftarrow n + 1$
            $\mathbf{c}'_n \leftarrow \mathsf{InsertPattern}(\mathbf{c}, \mathbf{n})$
            for $k \leftarrow 1$ to $\eta$ do
                $\hat{r}_k \leftarrow \mathsf{Predict}(\hat{\mathbf{m}}_k)$
            $\hat{\mathbf{r}}_n \leftarrow (\hat{r}_1\, \hat{r}_2 \cdots \hat{r}_\eta)$
        $f \leftarrow 1$
        for $n_1 \leftarrow 1$ to $2^\gamma$ do
            for $n_2 \leftarrow n_1 + 1$ to $2^\gamma$ do
                if $\mathsf{HD}(\hat{\mathbf{r}}_{n_1}, \hat{\mathbf{r}}_{n_2}) < \varepsilon_2^\star$ then
                    $f \leftarrow 0$
    while $f = 0$
    $\tilde{\mathbf{r}} \leftarrow \mathsf{QueryDevice}(\mathbf{c})$
    $f \leftarrow 0$
    for $n \leftarrow 1$ to $2^\gamma$ do
        if $\mathsf{HD}(\hat{\mathbf{r}}_n, \tilde{\mathbf{r}}) < \varepsilon_3^\star$ then
            $f \leftarrow f + 1$
            $\hat{n} \leftarrow n$
    if $f = 1$ then
        $w \leftarrow w + 1$
        $\mathbf{c}'_w \leftarrow \mathbf{c}'_{\hat{n}}$
        $\mathbf{r}_w \leftarrow \tilde{\mathbf{r}}$
```

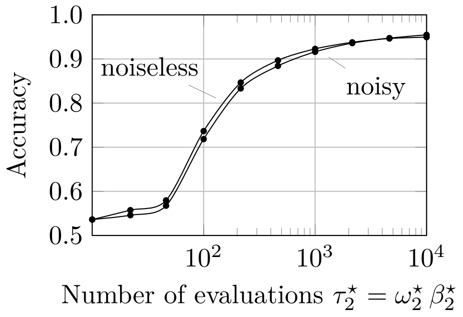


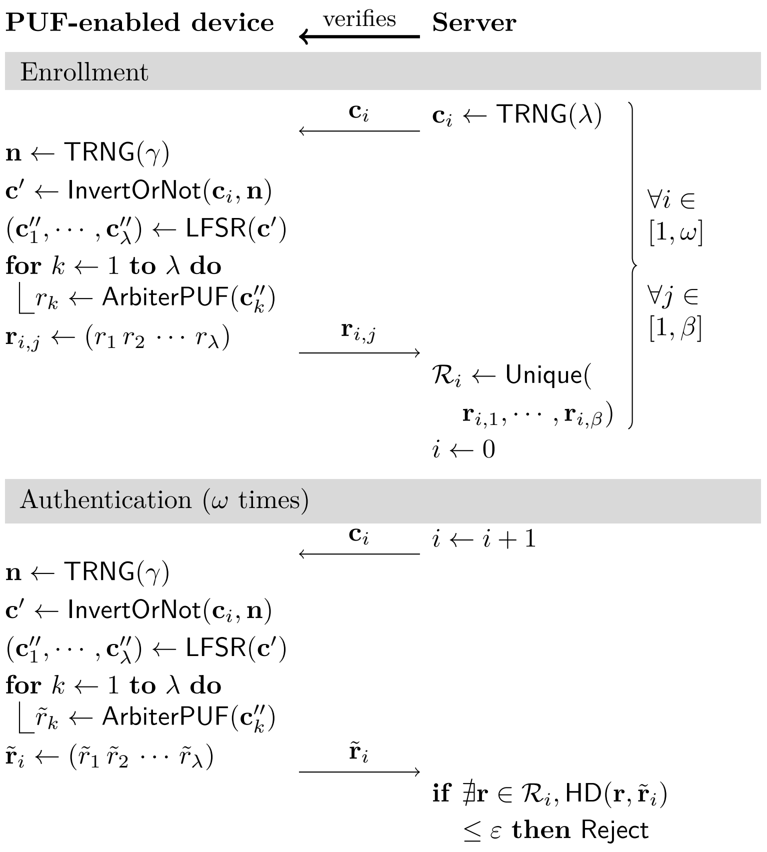
Figure 10: The accuracy of modeling an Arbiter PUF that is used in the OB-PUF protocol, where $\lambda = 64$, $\eta = 3$, $\gamma = 1$, and $\delta = 5$.

Figure 10 shows the obtained modeling accuracies as a function of the number of device queries $\tau_2^\star = \omega_2^\star\,\beta_2^\star$. For each dot, we generate again 20 random PUFs, and average the obtained accuracies. The 85% accurate predictive models for $\tau_1^\star = 10^4$ were used. We used constants $\varepsilon_2^\star = \eta = 3$ and $\varepsilon_3^\star = 2$ It can be seen that accuracies exceeding 90% can now be obtained.

3.3 RPUF

3.3.1 Specification

The so-called RPUF protocol of Ye, Hu, and Li [YHL16], where “R” stands for “Randomized”, is specified in Fig. 11. To prevent the machine learning of its Arbiter PUF, a device either does or does not invert the bits of any received challenge $\mathbf{c} \in \{0,1\}^\lambda$ depending on the value of a nonce $\mathbf{n} \in \{0,1\}^\gamma$. Suggested values for λ are 32, 64, and 128. For $\gamma = 1$, it holds that $\mathbf{c}' \in \{\mathbf{c}, \neg\mathbf{c}\}$. For $\gamma = 2$, it holds that $\mathbf{c}' \in \{\mathbf{c}, (c_1\, c_2\, \cdots\, c_{\lambda/2}\, \neg c_{\lambda/2+1}\, \neg c_{\lambda/2+2}\, \cdots\, \neg c_\lambda), (\neg c_1\, \neg c_2\, \cdots\, \neg c_{\lambda/2}\, c_{\lambda/2+1}\, c_{\lambda/2+2}\, \cdots\, c_\lambda), \neg\mathbf{c}\}$. Larger values of γ are not deemed necessary. The randomized challenge $\mathbf{c}'$ is fed into a *linear-feedback shift register* (LFSR) so that the 1-bit responses r to an expanded list of λ challenges $\mathbf{c}''$ can be concatenated into a λ-bit response $\mathbf{r}$.



```
PUF-enabled device      <--verifies--      Server

Enrollment
                        <--c_i--           c_i ← TRNG(λ)                    ⎫
n ← TRNG(γ)                                                                 ⎪
c' ← InvertOrNot(c_i, n)                                                    ⎪ ∀i ∈ [1, ω]
(c''_1, ⋯, c''_λ) ← LFSR(c')                                                ⎬
for k ← 1 to λ do                                                           ⎪ ∀j ∈ [1, β]
    r_k ← ArbiterPUF(c''_k)                                                 ⎪
r_{i,j} ← (r_1 r_2 ⋯ r_λ)   --r_{i,j}-->                                    ⎪
                                           R_i ← Unique(                    ⎪
                                             r_{i,1}, ⋯, r_{i,β})           ⎭
                                           i ← 0

Authentication (ω times)
                        <--c_i--           i ← i + 1
n ← TRNG(γ)
c' ← InvertOrNot(c_i, n)
(c''_1, ⋯, c''_λ) ← LFSR(c')
for k ← 1 to λ do
    r̃_k ← ArbiterPUF(c''_k)
r̃_i ← (r̃_1 r̃_2 ⋯ r̃_λ)     --r̃_i-->
                                           if ∄r ∈ R_i, HD(r, r̃_i)
                                             ≤ ε then Reject
```

Figure 11: The RPUF protocol of Ye et al. [YHL16].

To enroll a device, the server requests the response $\mathbf{r}$ to each out of ω randomly generated challenges $\mathbf{c}$ not once but $\beta \gg 2^\gamma$ times and collects the 2^γ unique values. A suggested value for β is 100. Evidently, slightly differing responses $\mathbf{r}$ are attributed to the noisiness of the PUF and are not considered unique. To authenticate a device up to ω times, the server checks whether the response $\tilde{\mathbf{r}}$ to a challenge $\mathbf{c}$ is sufficiently close to one out of its 2^γ prerecorded values. The authors emphasize that the nonce N should be uniformly distributed over $\{0,1\}^\gamma$. Otherwise, frequency analysis would allow an attacker to partition the unique responses $\mathbf{r}$ from multiple protocol runs into 2^γ sets that each correspond to a given transformation of the challenge $\mathbf{c}$. The authors collect data from numerous protocol runs and conduct machine learning experiments that do not exceed an accuracy of $\approx 75\%$. They, consequentially, consider their protocol fit for deployment in practical use cases.

3.3.2 Attack

Analogous to the growth of cracks in solid materials, the mediocre accuracy of $\approx 75\%$ should have been a warning of an imminent failure. Indeed, we now devise an alternative learning strategy that is orders of magnitude more efficient, thereby allowing an attacker to impersonate a PUF-enabled device an unlimited number of times. Given physical access to the device, the attacker can obtain the 2^γ unique responses $\mathbf{r} \in \{0,1\}^\lambda$ to each out of α challenges $\mathbf{c} \in \{0,1\}^\lambda$. There are hence $(2^\gamma!)^\alpha$ possibilities for constructing a combined training and testing set of $2^\gamma \cdot \alpha \cdot \lambda$ transformed CRPs $(\mathbf{s}'', r)$ each. When exhaustively applying a machine learning algorithm to each out of these sets, the one and only correct mapping can be observed to result in the highest accuracy. Alternatively, an attacker who eavesdrops on α genuine protocol runs can iterate over $2^{\gamma \cdot \alpha}$ combined training and testing sets of $\alpha \cdot \lambda$ transformed CRPs $(\mathbf{s}'', r)$ each. Figure 12(a) shows that for either strategy, a relatively limited computational effort corresponds to a relatively large number of CRPs.

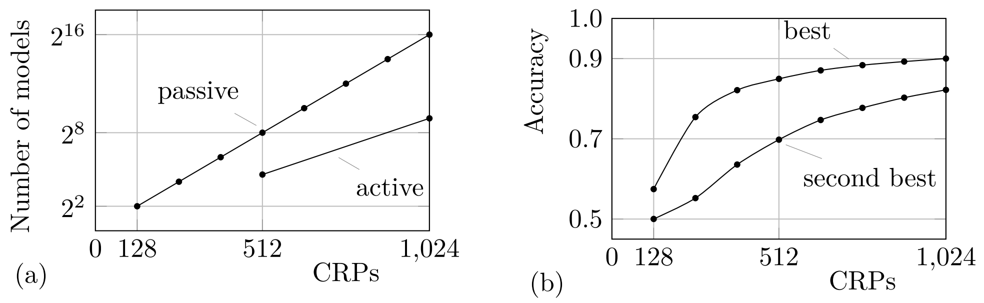


Figure 12: The first phase of an attack on the RPUF protocol, where $\lambda = 128$ and $\gamma = 2$. For an either passive or active attacker, subplot (a) shows the number of possible mappings between a given number of transformed challenges $\mathbf{s}''$ and an equal number of response bits r. For each possible mapping, a model is trained and subsequently tested. Subplot (b) shows the accuracy of the best and second-best models, which are obtained through linear regression according to (2). Both accuracies are averaged over 1000 randomly generated and noiseless PUFs $M \sim N\big(\mathbf{0}, \text{diag}(1/2, 1, 1, \cdots, 1, 1/2)\big)$. For any given challenge $\mathbf{c}$, we use $\mathsf{round}(0.8\lambda) = 102$ and $\mathsf{round}(0.2\lambda) = 26$ transformed CRPs $(\mathbf{s}'', r)$ for training and testing purposes respectively.

We apply *linear regression* [HTF09, 12th printing, Section 4.2] to each set of transformed CRPs $(\mathbf{s}'', r)$. Although the learning capabilities of this deterministic approach are slightly inferior to several randomized training algorithms, its speed is unparalleled and hence favors exhaustive enumeration. As shown in (2), determining the least-squares solution of a system of linear equations is all what is needed. Although Fig. 12(b) demonstrates that a fairly limited brute-force effort already allows for an accuracy of 90%, we suggest adopting a more efficient two-step approach to further improve the accuracy. First, numerous repeated executions of a small-sized exhaustive search, e.g., using $\alpha = 1$ every time, can be used to deobfuscate the mapping between numerous transformed challenges $\mathbf{s}''$ and their corresponding response bits r. Second, a potentially slower training algorithm with superior learning capabilities can be applied to a single large set of deobfuscated pairs $(\mathbf{s}'', r)$. This way, accuracies exceeding 99% can be achieved [RSS+13].

$$\text{Solve } \begin{pmatrix} \mathbf{s}''_1 \\ \mathbf{s}''_2 \\ \vdots \\ \mathbf{s}''_{\omega^\star} \end{pmatrix} (\hat{\mathbf{m}}_1^T \; \hat{\mathbf{m}}_0^T) = \begin{pmatrix} r_1 & \neg r_1 \\ r_2 & \neg r_2 \\ \vdots & \vdots \\ r_{\omega^\star} & \neg r_{\omega^\star} \end{pmatrix}; \text{ predict } \hat{r}_{\omega^\star+1} = \begin{cases} 1, & \text{if } \mathbf{s}''_{\omega^\star+1}\hat{\mathbf{m}}_1^T > \mathbf{s}''_{\omega^\star+1}\hat{\mathbf{m}}_0^T, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

We emphasize that the previously elaborated attack cannot simply be mitigated by increasing the value of γ. To enroll a device, the response $\mathbf{r}$ to every challenge $\mathbf{c}$ needs to be evaluated $\beta \gg 2^\gamma$ times. Therefore, the attacker and the server face a similar workload. A final note is that, depending on the non-specified internals of the LFSR, a more straightforward deobfuscation method might exist. It is intuitive to assume that the LFSR has a λ-bit state that is initialized by the randomized challenge $\mathbf{c}' \in \{0,1\}^\lambda$, and that each out of λ^2 state updates generates a single challenge bit c''. This allows an attacker to choose two challenges $\mathbf{c}$ such that for any given value of nonce $\mathbf{n} \in \{0,1\}^\gamma$, the expanded challenge sequences are $(\mathbf{c}''_1, \mathbf{c}''_2, \cdots, \mathbf{c}''_\lambda)$ and $(\mathbf{c}''_{\lambda/2+1}, \mathbf{c}''_{\lambda/2+2}, \cdots, \mathbf{c}''_{3\lambda/2})$ respectively. The respective responses $\mathbf{r}$ hence have an overlap of $\lambda/2$ bits.

3.4 LHS-PUF

3.4.1 Specification

The so-called LHS-PUF protocol of Idriss and Bayoumi [IB17], where "LHS" stands for "Lightweight Highly Secure", is specified in Fig. 13. The authors do not instantiate their protocol with a specific PUF design, but consistently refer to work on Arbiter PUFs and their variations. Given that no constraints are imposed with respect to the LHS claim, we assume the use of a basic Arbiter PUF. To enroll a given device, the server collects ω CRPs $(\mathbf{c}, r)$ and trains a predictive model $\hat{\mathbf{m}}$ of the Arbiter PUF. After the enrollment, direct access to the response bits r is irreversibly disabled.

To preclude machine learning attacks, response bits r are not directly exposed. Instead, a protocol run releases challenge tuples $(\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3, \mathbf{c}_4, \mathbf{c}_5)$ for which it is known that $r_1 = r_2 \oplus r_3$ and $r_2 = r_4 \oplus r_5$. Long responses $\mathbf{r} \in \{0,1\}^\eta$ are obtained by concatenating the 1-bit responses r to η randomly generated challenges $\mathbf{c} \in \{0,1\}^\lambda$. Suggested values for η are 64 and 128. To preclude trivial impersonation attacks using strongly correlated CRPs, it is imposed for several challenge pairs that $\mathsf{HD}(\mathbf{c}, \mathbf{c}') > \varepsilon_1$.

3.4.2 Attack

A first flaw related to the minimum Hamming distance checks on various challenge pairs $(\mathbf{c}, \mathbf{c}')$ is that these do not detect the use of strongly correlated CRPs in general. For Arbiter PUFs and their variations, two-sided constraints on transformed challenge pairs $(\mathbf{s}, \mathbf{s}')$ would be more appropriate, i.e., $\varepsilon_1 < \mathsf{HD}(\mathbf{s}, \mathbf{s}') < \lambda - \varepsilon_1$. A second flaw is that not sufficient challenge pairs are considered. Due to the dual role of response $\mathbf{r}_2$, an attacker can successfully impersonate a device by transmitting arbitrarily chosen challenges $\mathbf{c}_{1,k}$ and replying $(\mathbf{c}_{4,k}, \mathbf{c}_{5,k}) = (\mathbf{c}_{1,k}, \mathbf{c}_{3,k})$ or $(\mathbf{c}_{4,k}, \mathbf{c}_{5,k}) = (\mathbf{c}_{3,k}, \mathbf{c}_{1,k})$ for all response bit indices $k \in [1, \eta]$. Alternatively, an attacker can transmit challenges $\mathbf{c}_{1,1} = \mathbf{c}_{1,2} = \cdots = \mathbf{c}_{1,\eta}$, which implies $(\hat{\mathbf{r}}_1, \hat{\mathbf{r}}_2, \hat{\mathbf{r}}_3) \in \{(\mathbf{0},\mathbf{0},\mathbf{0}), (\mathbf{0}, \neg\mathbf{0}, \neg\mathbf{0}), (\neg\mathbf{0}, \mathbf{0}, \neg\mathbf{0}), (\neg\mathbf{0}, \neg\mathbf{0}, \mathbf{0})\}$, and thus has a 50/50 chance of successfully impersonating a device by sending replies $\mathbf{c}_{4,1} = \mathbf{c}_{4,2} = \cdots = \mathbf{c}_{4,\eta}$ and $\mathbf{c}_{5,1} = \mathbf{c}_{5,2} = \cdots = \mathbf{c}_{5,\eta}$ later-on. By replaying the messages of the first successful impersonation attempt, the success rate can later be increased to 100%.

The previously described problems are easy-to-fix, but this is not the case for the misassumption that machine learning attacks are precluded by releasing challenges $\mathbf{c}$ only. The modeling resistance is, in fact, equivalent to the serialized version [YHD+16] of a 3-XOR

```
PUF-enabled device            <--verifies-->   Server

Enrollment
                              <-- c_i --       $\mathbf{c}_i \leftarrow \mathsf{TRNG}(\lambda)$   } $\forall i \in [1, \omega]$
$r_i \leftarrow \mathsf{ArbiterPUF}(\mathbf{c}_i)$   -- r_i -->
Disable read-out of $r_i$                      $\hat{\mathbf{m}} \leftarrow \mathsf{TrainModel}(\mathbf{c}_1, r_1, \cdots, \mathbf{c}_\omega, r_\omega)$

Authentication ($\infty$ times)
                              <-- init --
for $k \leftarrow 1$ to $\eta$ do
    $\mathbf{c}_{1,k} \leftarrow \mathsf{TRNG}(\lambda)$
    $\tilde{r}_{1,k} \leftarrow \mathsf{ArbiterPUF}(\mathbf{c}_{1,k})$
$\tilde{\mathbf{r}}_1 \leftarrow (\tilde{r}_{1,1}\, \tilde{r}_{1,2} \cdots \tilde{r}_{1,\eta})$
                              -- $\mathbf{c}_{1,1}, \mathbf{c}_{1,2}, \cdots, \mathbf{c}_{1,\eta}$ -->
                                               for $k \leftarrow 1$ to $\eta$ do
                                                   $\hat{r}_{1,k} \leftarrow \mathsf{Predict}(\hat{\mathbf{m}}, \mathbf{c}_{1,k})$
                                                   do $\mathbf{c}_{2,k} \leftarrow \mathsf{TRNG}(\lambda)$
                                                       $\mathbf{c}_{3,k} \leftarrow \mathsf{TRNG}(\lambda)$
                                                       $\hat{r}_{2,k} \leftarrow \mathsf{Predict}(\hat{\mathbf{m}}, \mathbf{c}_{2,k})$
                                                       $\hat{r}_{3,k} \leftarrow \mathsf{Predict}(\hat{\mathbf{m}}, \mathbf{c}_{3,k})$
                                                   while $(\hat{r}_{2,k} \oplus \hat{r}_{3,k} \neq \hat{r}_{1,k})$
                                                       $\vee(\mathsf{HD}(\mathbf{c}_{2,k}, \mathbf{c}_{3,k}) \leq \varepsilon_1)$
                                               $\hat{\mathbf{r}}_2 \leftarrow (\hat{r}_{2,1}\, \hat{r}_{2,2} \cdots \hat{r}_{2,\eta})$
                              <-- $\mathbf{c}_{2,1}, \mathbf{c}_{3,1}, \mathbf{c}_{2,2}, \mathbf{c}_{3,2}, \cdots, \mathbf{c}_{2,\eta}, \mathbf{c}_{3,\eta}$ --
for $k \leftarrow 1$ to $\eta$ do
    if $(\mathsf{HD}(\mathbf{c}_{1,k}, \mathbf{c}_{2,k}) \leq \varepsilon_1)$
        $\vee(\mathsf{HD}(\mathbf{c}_{1,k}, \mathbf{c}_{3,k}) \leq \varepsilon_1)$
        $\vee(\mathsf{HD}(\mathbf{c}_{2,k}, \mathbf{c}_{3,k}) \leq \varepsilon_1)$
        then Reject
    $\tilde{r}_{2,k} \leftarrow \mathsf{ArbiterPUF}(\mathbf{c}_{2,k})$
    $\tilde{r}_{3,k} \leftarrow \mathsf{ArbiterPUF}(\mathbf{c}_{3,k})$
$\tilde{\mathbf{r}}_2 \leftarrow (\tilde{r}_{2,1}\, \tilde{r}_{2,2} \cdots \tilde{r}_{3,\eta})$
$\tilde{\mathbf{r}}_3 \leftarrow (\tilde{r}_{2,1}\, \tilde{r}_{2,2} \cdots \tilde{r}_{3,\eta})$
if $(\mathsf{HD}(\tilde{\mathbf{r}}_1, \tilde{\mathbf{r}}_2 \oplus \tilde{\mathbf{r}}_3) > \varepsilon_2)$
    then Reject
for $k \leftarrow 1$ to $\eta$ do
    do $\mathbf{c}_{4,k} \leftarrow \mathsf{TRNG}(\lambda)$
        $\mathbf{c}_{5,k} \leftarrow \mathsf{TRNG}(\lambda)$
        $\hat{r}_{4,k} \leftarrow \mathsf{ArbiterPUF}(\mathbf{c}_{2,k})$
        $\hat{r}_{5,k} \leftarrow \mathsf{ArbiterPUF}(\mathbf{c}_{3,k})$
    while $(\hat{r}_{4,k} \oplus \hat{r}_{5,k} \neq \hat{r}_{2,k})$
        $\vee(\mathsf{HD}(\mathbf{c}_{4,k}, \mathbf{c}_{5,k}) \leq \varepsilon_1)$
                              -- $\mathbf{c}_{4,1}, \mathbf{c}_{5,1}, \mathbf{c}_{4,2}, \mathbf{c}_{5,2}, \cdots, \mathbf{c}_{4,\eta}, \mathbf{c}_{5,\eta}$ -->
                                               for $k \leftarrow 1$ to $\eta$ do
                                                   if $(\mathsf{HD}(\mathbf{c}_{2,k}, \mathbf{c}_{4,k}) \leq \varepsilon_1)$
                                                       $\vee(\mathsf{HD}(\mathbf{c}_{2,k}, \mathbf{c}_{5,k}) \leq \varepsilon_1)$
                                                       $\vee(\mathsf{HD}(\mathbf{c}_{4,k}, \mathbf{c}_{5,k}) \leq \varepsilon_1)$
                                                       then Reject
                                                   $\tilde{r}_{4,k} \leftarrow \mathsf{Predict}(\hat{\mathbf{m}}, \mathbf{c}_{4,k})$
                                                   $\tilde{r}_{5,k} \leftarrow \mathsf{Predict}(\hat{\mathbf{m}}, \mathbf{c}_{5,k})$
                                               $\tilde{\mathbf{r}}_4 \leftarrow (\tilde{r}_{2,1}\, \tilde{r}_{2,2} \cdots \tilde{r}_{3,\eta})$
                                               $\tilde{\mathbf{r}}_5 \leftarrow (\tilde{r}_{2,1}\, \tilde{r}_{2,2} \cdots \tilde{r}_{3,\eta})$
                                               if $(\mathsf{HD}(\tilde{\mathbf{r}}_2, \tilde{\mathbf{r}}_4 \oplus \tilde{\mathbf{r}}_5) > \varepsilon_2)$
                                                   then Reject
```

Figure 13: The LHS–PUF protocol of Idriss and Bayoumi [IB17].

PUF, i.e., the attacker is given three challenges for which the response $r = r_1 \oplus r_2 \oplus r_3 = 0$. However, given that training data for $r = 1$ is missing, predictive models might converge to the equivalent of a biased Arbiter PUF that produces 0s exclusively. Therefore, we invert half of the training challenges, i.e., $\mathbf{s}' = (-s_1, -s_2, \ldots, -s_\lambda, 1)$, and flip r accordingly. We adopt a *covariance matrix adaptation* (CMA) variant of an *evolution strategy* (ES) [Han06] and perform minimal changes to its open-source implementation in MATLAB. Similar to Darwin's theory on biological evolution, the fittest candidates in a population of prospective models $\hat{\mathbf{m}}$ recombine and mutate into a new and presumably fitter population. Although default values suffice for all parameters, it is crucial to define an appropriate fitness function, i.e., $\mathsf{fitness} : \{0,1\}^{\lambda+1} \rightarrow \mathbb{R}$. Results for the fitness function in (3) are shown in Fig. 14. Because an accurate model $\hat{\mathbf{m}}$ is not always obtained, we only retain the best out of 10 runs.

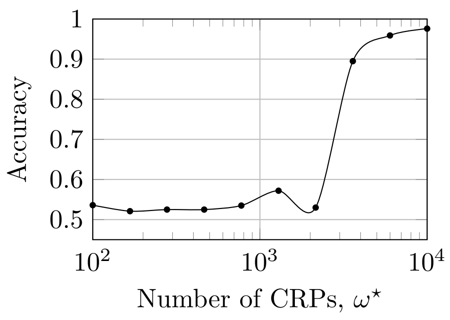


Figure 14: The accuracy of modeling the 3-XOR PUF equivalent of the Arbiter PUF in the LHS-PUF protocol, where $\lambda = 64$.

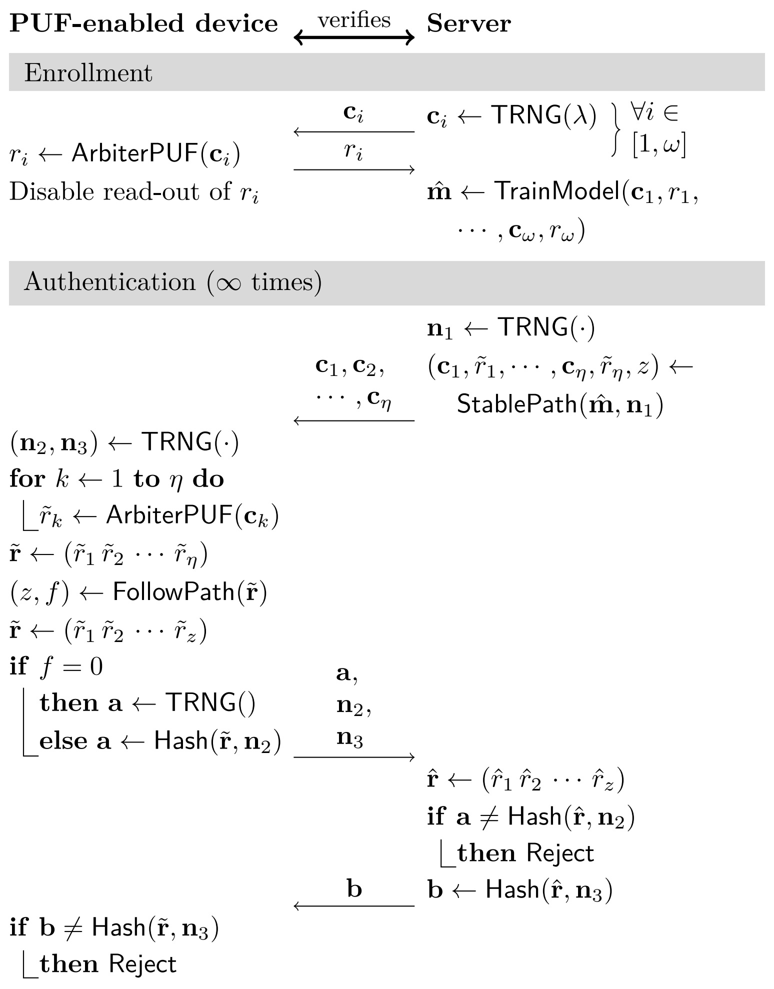
$$\begin{aligned} \mathsf{fitness}(\hat{\mathbf{m}}) = \frac{1}{\omega^\star} \sum_{i=1}^{\omega^\star} & \big(\mathbf{s}'_{1,i}\, \hat{\mathbf{m}}^T > 0\big) \oplus \big(\mathbf{s}'_{2,i}\, \hat{\mathbf{m}}^T > 0\big) \\ & \oplus \big(\mathbf{s}'_{3,i}\, \hat{\mathbf{m}}^T > 0\big) \oplus r'_i \oplus 1. \end{aligned} \tag{3}$$

3.5 PUF–FSM

3.5.1 Specification

The so-called PUF–FSM protocol of Gao, Ma, Al-Sarawi, Abbott, and Ranasinghe [GMA+18], where "FSM" stands for "finite-state machine", is specified in Fig. 15. Each device hosts an Arbiter PUF with λ challenge bits c. A suggested value for λ is 64. To enroll a given device, the server collects ω CRPs $(\mathbf{c}, r)$ so that an accurate predictive model $\hat{\mathbf{m}}$ can be trained. A suggested value for ω is 10^4. Both response bits r, which are the result of a comparison $v \lesseqgtr 0$, and their respective error rates p_{error}, which decrease monotonically with $|v|$, can be predicted. After the enrollment, the interface for reading out response bits r is irreversibly disabled.

During any out of a virtually unlimited number of protocol runs, the server is restricted to using the CRPs $(\mathbf{c}, r)$ that have the lowest error rates p_{error}. Considering the noisiness of their implemented Arbiter PUFs, the authors opt to maintain $1.8 \cdot 10^{17}$ out of 2^{64} CRPs, which corresponds to a retention rate $\rho_{\text{ret}} \approx 1\%$. For a hardwired FSM, having one start state and one end state as shown in Fig. 16, the server randomly selects one out of a large number of paths from start to finish. The corresponding sequence of state transitions defines a sequence of η response bits r, where a variable number of $z \leq \eta$ bits suffices to


```
PUF-enabled device  <--verifies-->  Server

Enrollment
                          <-- c_i --   c_i ← TRNG(λ)  } ∀i ∈ [1, ω]
r_i ← ArbiterPUF(c_i)     -- r_i -->                  }
Disable read-out of r_i                m̂ ← TrainModel(c_1, r_1, ··· , c_ω, r_ω)

Authentication (∞ times)
                                       n_1 ← TRNG(·)
                                       (c_1, r̃_1, ··· , c_η, r̃_η, z) ←
                   <-- c_1, c_2, ··· , c_η --   StablePath(m̂, n_1)
(n_2, n_3) ← TRNG(·)
for k ← 1 to η do
    r̃_k ← ArbiterPUF(c_k)
r̃ ← (r̃_1 r̃_2 ··· r̃_η)
(z, f) ← FollowPath(r̃)
r̃ ← (r̃_1 r̃_2 ··· r̃_z)
if f = 0
    then a ← TRNG()
    else a ← Hash(r̃, n_2)   -- a, n_2, n_3 -->
                                       r̂ ← (r̂_1 r̂_2 ··· r̂_z)
                                       if a ≠ Hash(r̂, n_2)
                                           then Reject
                          <-- b --     b ← Hash(r̂, n_3)
if b ≠ Hash(r̃, n_3)
    then Reject
```

Figure 15: The PUF–FSM protocol of Gao et al. [GMA+18].

reach the end state. A value for constant η has not been suggested. The proposed FSM consists of θ stages, where constant θ is odd. A suggested value for θ is 41. Odd- and even-numbered stages, in turn, consist of 1 and $\phi > 1$ states respectively. A suggested value for ϕ is 3. Each state transition is defined by a δ-bit substrings of response $\mathbf{x} \in \{0,1\}^{\eta}$. A total of $z \in [(\theta - 1)\delta, \eta]$ response bits hence suffices for reach the end state. A suggested value for δ is 4. A flag f indicating whether or not the finish is reached is 1 and 0 for stage θ and stages 1 to $\theta - 1$ respectively.

For a given path-defining response $(r_1\, r_2\, \cdots\, r_\eta)$, the server randomly selects a corresponding sequence of η challenges $\mathbf{c}$ that is subsequently transmitted to the device. The latter party then reconstructs the path from newly generated response bits $\tilde{r}_i$. If the end state is successfully reached, i.e., flag $f = 1$, the first z response bits are used to establish a shared secret with the server. This secret, in addition to nonce $\mathbf{n}_2$ or $\mathbf{n}_3$, is then fed into a cryptographic hash function to perform the authentication. To preserve the secrecy of flag f, an attacker is not allowed to observe whether or not the authentication succeeds. Otherwise, an attacker would be able to replace a server-determined challenge $\mathbf{c}_i$ by an arbitrary challenge $\mathbf{c}_j$, where $\mathbf{c}_j \neq \mathbf{c}_i$, and determine whether or not $r_i = r_j$. Observe that a repeated execution of this swapping mechanism would allow the attacker to gather a

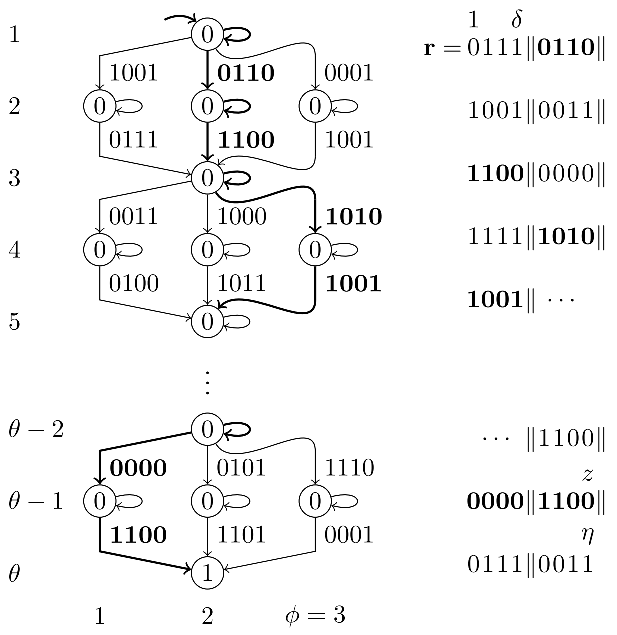


Figure 16: The FSM of Gao et al. [GMA+18].

large training set of CRPs and hence model the Arbiter PUF such that only the sign of $\hat{\mathbf{m}}$ remains unknown.

3.5.2 Attack

It suffices for an attacker to eavesdrop on a single genuine protocol run in order to train an accurate predictive model $\hat{\mathbf{m}}$ of the underlying Arbiter PUF. Although the authors are aware that not only the response bits r but also their corresponding error rates p_{error} should remain internal to the device, given a pre-existing attack by Becker [Bec15a], it is overlooked that other variables that are correlated to the delay difference v are released. Most notably, for each server-determined challenge $\mathbf{c}$, it is known that the absolute value $|v|$ is relatively high. Given $\omega^\star = 1$ challenge $\mathbf{c} \in \{0,1\}^\lambda$, having transformed version $\mathbf{s} \in \{-1,1\}^{\lambda+1}$, the two best guesses for a predictive model are hence $\hat{\mathbf{m}} = (s_1/2 \; s_2 \; s_3 \; \cdots \; s_\lambda \; s_{\lambda+1}/2)$ and $\hat{\mathbf{m}} = -(s_1/2 \; s_2 \; s_3 \; \cdots \; s_\lambda \; s_{\lambda+1}/2)$. The choice between these two models $\hat{\mathbf{m}}$ corresponds to an entropy of one bit, which is negligible in a system-level security analysis.

Figure 17(a) shows that for a retention ratio $\rho_{\text{ret}} = 1\%$, the best out of two models already exceeds an accuracy of 85%, which suffices to consider the protocol broken. For each dot, we generate 100 PUFs $M \sim N\big(\mathbf{0}, \text{diag}(1/2, 1, 1, \cdots, 1, 1/2)\big)$ and average the best accuracies P_{acc} for each out of two reproduced models $\hat{\mathbf{m}}$. Stated otherwise, we show an estimate of $\mathbb{E}[\max(P_{\text{acc}}, 1 - P_{\text{acc}})]$, where P_{acc} is the accuracy for one out of two possible models $\hat{\mathbf{m}}$. For each individual modeling experiment, we select $\omega^\star \in [1, 100]$ training and 1000 test challenges $\mathbf{c}$ uniformly at random from the subset $\mathcal{C}_{stab} \subseteq \mathcal{C}$ that contains the challenges with the most stable responses r, where $|\mathcal{C}_{\text{stab}}|/|\mathcal{C}| = \rho_{\text{ret}}$. We emphasize that for impersonation purposes, an attacker is only required to predict stable response bits r.

During a single protocol run, however, the server releases not one but $\eta \gg 1$ challenges $\mathbf{c}_i$. There is hence plenty of margin to improve the accuracy of model $\hat{\mathbf{m}}$. Becker, Wild, and Güneysu [BWG15] previously addressed a similar learning problem using CMA-ES, but we require a fitness function of which the design is based on different principles, e.g., (4).

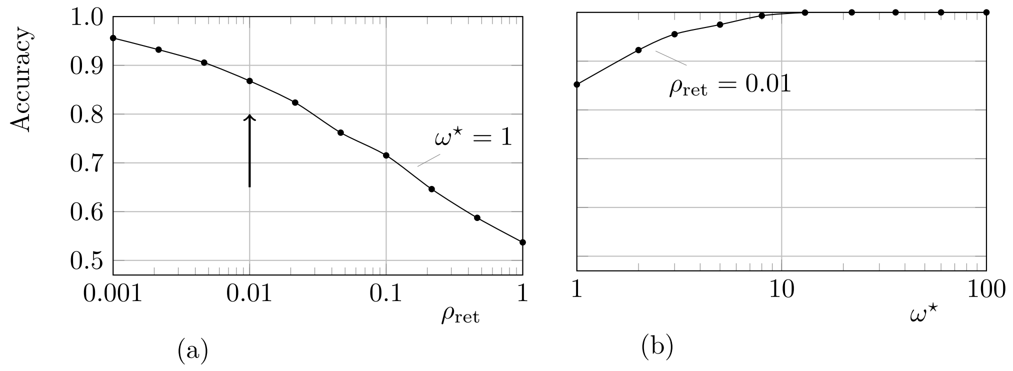


Figure 17: The accuracy of modeling an Arbiter PUF that is used in the PUF–FSM protocol, where $\lambda = 64$.

$$\mathsf{fitness}(\hat{\mathbf{m}}) = \frac{1}{\omega^\star}\sum_{i=1}^{\omega^\star}\left|\mathbf{s}_i\,\hat{\mathbf{m}}^T\right| \Bigg/ \frac{1}{\alpha}\sum_{i=1}^{\alpha}\left|\mathbf{s}_{\text{ref},i}\,\hat{\mathbf{m}}^T\right| . \tag{4}$$

The $\omega^\star$ transformed challenges $\mathbf{s}_i$ in the numerator originate from a genuine protocol run and are hence known to have stable response bits r_i. The α transformed challenges $\mathbf{s}_{\text{ref},i}$ in the denominator are chosen uniformly at random from the set of all 2^λ transformed challenges and hence have response bits r that cover the full spectrum of error rates p_{error}. To ensure that the fitness function allows for a fast evaluation, we use the same α transformed challenges $\mathbf{s}_{\text{ref},i}$ for each evaluation and limit ourselves to $\alpha = 1000$. The scale invariance, i.e., $\forall a \in \mathbb{R}_0, \mathsf{fitness}(a\,\hat{\mathbf{m}}) = \mathsf{fitness}(\hat{\mathbf{m}})$, is desired for positive factors $a \in \mathbb{R}_0^+$, but the inclusion of negative factors $a \in \mathbb{R}_0^-$ once again implies that one bit of entropy always remains present. Because the randomized training algorithm does not always converge to an accurate model $\hat{\mathbf{m}}$, we only retain the best out of five trials. For a retention ratio $\rho_{\text{ret}} = 1\%$ and $\omega^\star = 10$ server-defined challenges $\mathbf{c}_i$, Fig. 17(b) shows that the best out of $2 \cdot 5 = 10$ models approaches the ideal accuracy of 100%.

The previously presented modeling techniques are successful despite disregarding the internal specifics of the FSM. For the sake of completeness, we briefly discuss how this disregarded knowledge could facilitate CMA-ES. For a given model $\hat{\mathbf{m}}$ and a given protocol run, the prospective η-bit response $\mathbf{r}$ could be computed. For this sequence of state transitions, the fitness of the best possible match with an available path can then be computed. Numerous path-matching metrics could be devised but, given that our main objective has already been achieved, we abstain from further exploration.

4 Aftermath

A fairly conservative approach to craft a PUF-based authentication protocol is to convert a noisy response $\mathbf{r} \in \{0,1\}^\eta$ into a stable secret key $\mathbf{k} \in \{0,1\}^\kappa$, where $\eta \gg \kappa$, and then use a keyed cryptographic algorithm to perform the authentication [Del17, Section 5.2]. A *fuzzy extractor* [DORS08] can perform this conversion. Its realizations are usually based on an error-correcting code and require public helper data, which is stored either by the PUF-enabled device or by the server. In the latter case, the helper data is transferred with each protocol run. Under the assumption that response R is uniformly distributed over $\{0,1\}^\eta$ and that the expected bit error rate $\mathbb{E}[P_{\text{error}}] \leq 15\%$, a few thousand response and helper bits usually suffice to derive a uniformly distributed key $\mathbf{k} \in \{0,1\}^{128}$ such that its reconstruction is expected to fail with probability $\mathbb{E}[P_{\text{fail}}] \leq 10^{-6}$ [vdLPvdS12, HYS16].

Designers of PUF-based protocols frequently aim to save resources by avoiding the use of an error-correcting code and/or the cryptographic logic, but as we have demonstrated for five recent proposals, taking shortcuts might be fatal for the system security. The irony is that for three out of five proposals, the obtained reductions in hardware footprint are small, if existing at all, and might not even have justified taking the risk.

The PUF–FSM protocol [GMA+18] requires each PUF-enabled device to implement a cryptographic algorithm, so it suffices to compare the implementation efficiencies of the FSM and an error-correcting code. Although monolithic, large-sized codes require expensive decoders, it is a common practice to construct a large-size code from the repeated execution of one or more small-sized and hence cheaper codes. This refers, for example, to the sliding window of a convolutional code [HYS16] and to the concatenation of a Golay and a repetition code [vdLPvdS12]. Moreover, so-called reverse fuzzy extractors [VHKM+12] only require a PUF-enabled device to implement an encoder, which is considerably cheaper than the corresponding decoder. Protocol-specific and more generic weaknesses for the reversed modus are known to exist [Bec15b] [Del17, Chapter 5], but several versions still hold up. Finally, each run of the PUF–FSM protocol requires the transfer of more than $160 \cdot 64 = 10240$ challenge bits c, which is more expensive then storing or transferring the helper data of a fuzzy extractor.

The PolyPUF protocol [KCW16] requires each device to implement 64 Arbiter PUFs having 64 stages each. Given that the estimated area of a 64-stage Arbiter PUF [Roz16, Fig 7.1] is equivalent to 387 two-input NAND gates, consisting of four transistors each, the whole array consumes 24768 *gate equivalent* (GE). More area-efficient implementations of an Arbiter PUF evidently exist, but the main observation here is that a full-fledged PUF-based key generator easily fits within 5000 GE for the given security level $\kappa \approx 64$ [vdLPvdS12]. When basing all subsequent cryptographic operations on a lightweight cipher such as KATAN64 [CDK09], which adds around 1000 GE to the system, it becomes clear that the conservative authentication approach might be cheaper. For the sole purpose of performing area comparisons, Konigsmark et al. [KCW16] conveniently switch to an alternative protocol version where a single Arbiter PUF generates all 64 response bits. Recall that their machine learning experiments are all conducted on a harder-to-attack array of PUFs.

The LHS-PUF protocol [IB17] might be area-efficient, but even for a modest security level, e.g., $\eta = \lambda = 64$, each protocol run already entails the transmission of 20480 bits. On the bright side, the RPUF protocol [YHL16] allows for an overall efficient implementation. For those who are looking for a similarly sized alternative that remains unbroken to date, we refer to the so-called lockdown protocols of Yu et al. [YHD+16]. We were unable to assess the footprint of the OB-PUF protocol [GLM+16] because its authors did not specify how to expand and deal with the noise of its 3-bit responses $\mathbf{r}$.

Although all five protocols are failed attempts to provide secure, lightweight authentication, the authors still deserve credit for their compliance with Kerckhoffs' principle, i.e., the specification of the obfuscation logic is public. On the contrary, Mispan, Su, Zwolinski, and Halak [MHZ17, MSZH18] proposed three PUF-based authentication protocols of which the specification is private, i.e., they rely on the almost universally reject paradigm of *security through obscurity.* To be clear: obscurity is the only protective measure, which differs from cases where obscurity augments an inherently secure system. If the complete specification of either protocol would be made public, the modeling resistance degrades to a conventional Arbiter PUF. It hence only takes one disgruntled employee of the system provider, or one attacker who recovers the netlist through the side-channel analysis or physical delayering of a device [RR13], to instantly compromise all devices in the field.

5 Conclusion

Through the use of custom-tailored machine learning techniques, we were able to train an accurate predictive model of the Arbiter PUFs that underlie the PolyPUF protocol of Konigsmark et al. [KCW16], the OB-PUF protocol of Gao et al. [GLM+16], the RPUF protocol of Ye et al. [YHL16], the LHS–PUF protocol of Idriss and Bayoumi [IB17], and the PUF–FSM protocol of Gao et al. [GMA+18], and hence enable an impersonation attack.

Acknowledgement

The author is currently with NTU, but most of the work has been performed while affiliated to KU Leuven. This work is partially funded by the Research Council of KU Leuven through C16/15/058 and the European Union's Horizon 2020 research and innovation programme under grant number 644052 (HECTOR) and the ERC Advanced Grant 695305 (CATHEDRAL).

References

[Bec15a] Georg T. Becker. The gap between promise and reality: On the insecurity of XOR arbiter PUFs. In Tim Güneysu and Helena Handschuh, editors, *17th Workshop on Cryptographic Hardware and Embedded Systems (CHES 2015)*, volume 9293 of *Lecture Notes in Computer Science*, pages 535–555. Springer, September 2015.

[Bec15b] Georg T. Becker. On the pitfalls of using arbiter-PUFs as building blocks. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 34(8):1295–1307, August 2015.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *1st Conference on Computer and Communications Security (CCS 1993)*, pages 62–73. ACM, November 1993.

[BWG15] Georg T. Becker, Alexander Wild, and Tim Güneysu. Security analysis of index-based syndrome coding for PUF-based key generation. In *Symposium on Hardware Oriented Security and Trust (HOST 2015)*, pages 20–25. IEEE, May 2015.

[CDK09] Christophe De Cannière, Orr Dunkelman, and Miroslav Knezevic. KATAN and KTANTAN – A family of small and efficient hardware-oriented block ciphers. In Christophe Clavier and Kris Gaj, editors, *11th Workshop on Cryptographic Hardware and Embedded Systems (CHES 2009)*, volume 5747 of *Lecture Notes in Computer Science*, pages 272–288. Springer, September 2009.

[Del17] Jeroen Delvaux. *Security Analysis of PUF-Based Key Generation and Entity Authentication.* PhD thesis, KU Leuven and Shanghai Jiao Tong University, June 2017.

[DORS08] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM Journal on Computing*, 38(1):97–139, March 2008.

[GLM$^+$16] Yansong Gao, Gefei Li, Hua Ma, Said F. Al-Sarawi, Omid Kavehei, Derek Abbott, and Damith C. Ranasinghe. Obfuscated challenge-response: A secure lightweight authentication mechanism for PUF-based pervasive devices. In *14th Conference on Pervasive Computing and Communications (PerCom 2016)*, pages 1–6. IEEE, March 2016.

[GMA$^+$18] Yansong Gao, Hua Ma, Said F. Al-Sarawi, Derek Abbott, and Damith C. Ranasinghe. PUF-FSM: A controlled strong PUF. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(5):1104–1108, May 2018.

[Han06] Nikolaus Hansen. *The CMA Evolution Strategy: A Comparing Review*, volume 192 of *Studies in Fuzziness and Soft Computing*, pages 75–102. Springer, 2006.

[HTF09] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning.* Springer, 2009.

[HYS16] Matthias Hiller, Meng-Day Yu, and Georg Sigl. Cherry-picking reliable PUF bits with differential sequence coding. *IEEE Transactions on Information Forensics and Security (TIFS)*, 11(9):2065–2076, September 2016.

[IB17] Tarek Idriss and Magdy Bayoumi. Lightweight highly secure PUF protocol for mutual authentication and secret message exchange. In *Conference on RFID Technology & Application (RFID-TA 2017)*, pages 1–6. IEEE, September 2017.

[KCW16] Sven Tenzing Choden Konigsmark, Deming Chen, and Martin D. F. Wong. PolyPUF: Physically secure self-divergence. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 35(7):1053–1066, July 2016.

[LDT00] Keith Lofstrom, W. Robert Daasch, and Donald Taylor. IC identification circuit using device mismatch. In *2000 International Solid-State Circuits Conference (ISSCC)*, pages 372–373. IEEE, February 2000.

[Lim04] Daihyun Lim. Extracting secret keys from integrated circuits. Master's thesis, Massachusetts Institute of Technology, May 2004.

[Mae13] Roel Maes. An accurate probabilistic reliability model for silicon PUFs. In Guido Bertoni and Jean-Sébastien Coron, editors, *15th Workshop on Cryptographic Hardware and Embedded Systems (CHES 2013)*, volume 8086 of *Lecture Notes in Computer Science*, pages 73–89. Springer, August 2013.

[MHZ17] Mohd Syafiq Mispan, Basel Halak, and Mark Zwolinski. Lightweight obfuscation techniques for modeling attacks resistant PUFs. In *2nd International Verification and Security Workshop (IVSW 2017)*, pages 19–24. IEEE, July 2017.

[MKP08] Mehrdad Majzoobi, Farinaz Koushanfar, and Miodrag Potkonjak. Testing techniques for hardware security. In *International Test Conference (ITC 2008)*, pages 1–10. IEEE, October 2008.

[MSZH18] Mohd Syafiq Mispan, Haibo Su, Mark Zwolinski, and Basel Halak. Cost-efficient design for modeling attacks resistant PUFs. In *Design, Automation & Test in Europe Conference & Exhibition (DATE 2018)*, pages 467–472. IEEE, March 2018.

[Roz16] Vladimir Rozić. *Circuit-Level Optimizations for Cryptography.* PhD thesis, KU Leuven, September 2016.

[RR13] Matthieu Rivain and Thomas Roche. SCARE of secret ciphers with SPN structures. In Kazue Sako and Palash Sarkar, editors, *Advances in Cryptology – ASIACRYPT 2013*, volume 8269 of *Lecture Notes in Computer Science*, pages 526–544. Springer, December 2013.

[RSS+13] Ulrich Rührmair, Jan Sölter, Frank Sehnke, Xiaolin Xu, Ahmed Mahmoud, Vera Stoyanova, Gideon Dror, Jürgen Schmidhuber, Wayne Burleson, and Srinivas Devadas. PUF modeling attacks on simulated and silicon data. *IEEE Transactions on Information Forensics and Security*, 8(11):1876–1891, November 2013.

[Sko05] Sergei P. Skorobogatov. Semi-invasive attacks – a new approach to hardware security analysis. Technical Report UCAM-CL-TR-630, University of Cambridge, Computer Laboratory, April 2005.

[TB15] Johannes Tobisch and Georg T. Becker. On the scaling of machine learning attacks on PUFs with application to noise bifurcation. In Stefan Mangard and Patrick Schaumont, editors, *RFIDSec 2015: Radio Frequency Identification*, volume 9440 of *Lecture Notes in Computer Science*, pages 17–31. Springer, June 2015.

[vdLPvdS12] Vincent van der Leest, Bart Preneel, and Erik van der Sluis. Soft decision error correction for compact memory-based PUFs using a single enrollment. In Emmanuel Prouff and Patrick Schaumont, editors, *14th Workshop on Cryptographic Hardware and Embedded Systems (CHES 2012)*, volume 7428 of *Lecture Notes in Computer Science*, pages 268–282. Springer, September 2012.

[VHKM+12] Anthony Van Herrewege, Stefan Katzenbeisser, Roel Maes, Roel Peeters, Ahmad-Reza Sadeghi, Ingrid Verbauwhede, and Christian Wachsmann. Reverse fuzzy extractors: Enabling lightweight mutual authentication for PUF-enabled RFIDs. In Angelos D. Keromytis, editor, *16th Conference on Financial Cryptography and Data Security (FC 2012)*, volume 7397 of *Lecture Notes in Computer Science*, pages 374–389. Springer, February 2012.

[YHD+16] Meng-Day Yu, Matthias Hiller, Jeroen Delvaux, Richard Sowell, Srinivas Devadas, and Ingrid Verbauwhede. A lockdown technique to prevent machine learning on PUFs for lightweight authentication. *IEEE Transactions on Multi-Scale Computing Systems (TMSCS)*, 2(3):146–159, July 2016.

[YHL16] Jing Ye, Yu Hu, and Xiaowei Li. RPUF: Physical unclonable function with randomized challenge to resist modeling attack. In *1st Asian Hardware Oriented Security and Trust Symposium (AsianHOST 2016)*, pages 1–6. IEEE, December 2016.