Attacks on Three PUF-Based Authentication Protocols: PolyPUF, RPUF, and PUF–FSM*

Jeroen Delvaux

imec-COSIC, KU Leuven, Belgium, jdelvaux@esat.kuleuven.be

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 three 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 RPUF protocol of Ye, Hu, and Li, as presented at the IEEE conference AsianHOST 2016, and (3) 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 2017.

Keywords: physically unclonable function · 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 boundaries of machine learning attacks on PUF-based protocols. The previous analyses, however, are not up-to-date with more recent proposals. 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 RPUF protocol of Ye et al. [YHL16], and the PUF–FSM protocol of Gao et al. [GMA+17] are

*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.

presented. All three protocols attempted 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 aforementioned authentication protocols. Section 4 discusses the aftermath from the perspective of a system provider. Section 5 concludes this work.

2 Preliminaries

2.1 Notation

Variables are denoted by a character from the Latin alphabet: a, b, c, etc. Constants are denoted by a character from the Greek alphabet: α, β, γ, etc. Vectors are denoted by a bold-faced, lowercase character, e.g., $\mathbf{x} = (x_1\, x_2)$. All vectors are row vectors. The all-zeros vector is denoted by $\mathbf{0}$. Matrices are denoted by a bold-faced, uppercase character, e.g., $\mathbf{X}$. 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., $\mathbf{X} = \mathrm{diag}(x_1, x_2)$. A random variable is denoted by an uppercase character, e.g., X. A multivariate normal random variable X with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ is denoted by $X \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. The expected value of a random variable X is denoted by $\mathbb{E}_{x \leftarrow X}[X]$. 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 $\{0,1\}^\lambda$. 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 Figure 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.

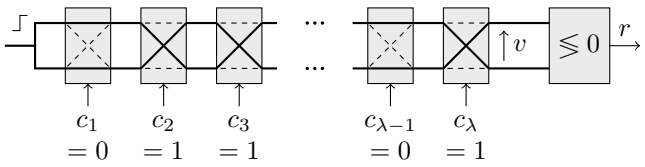


Figure 1: An Arbiter PUF with λ stages [Lim04].

If delay difference $v > 0$, the single-bit response $r = 1$; otherwise, $r = 0$. 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 value of $\mathsf{HD}(\mathbf{r}, \tilde{\mathbf{r}})$, averaged over numerous challenges $\mathbf{c}$, typically lies between 0.05η and 0.15η. 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 Figure 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].

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

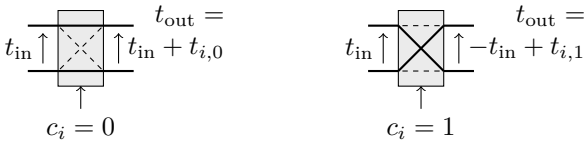


Figure 2: The delay behavior of a single stage of an Arbiter PUF.

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 $M \sim N\big(\mathbf{0}\,\boldsymbol{\Psi}, \sigma_t^2\,\boldsymbol{\Psi}^T\,\mathbf{I}_{2\lambda}\,\boldsymbol{\Psi}\big) \sim N\big(\mathbf{0}, \sigma_t^2\ \mathrm{diag}(1/2, 1, 1, \cdots, 1, 1/2)\big)$. Given that only the sign of v matters in determining the nominal value of 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 either Arbiter PUFs or *random oracles* [BR93] substantially differ in their statistical properties. For the latter population, the probability $\rho_{\text{flip}} = \mathbb{E}_{\mathbf{m} \leftarrow M}[R_1 \oplus R_2]$ equals $1/2$ for any given challenge pair $(\mathbf{c}_1, \mathbf{c}_2)$ where $\mathbf{c}_1 \neq \mathbf{c}_2$. For the former population, 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, Figure 12].

Another manifestation of the correlated structure is that machine learning algorithms training on a relatively small set of CRPs, i.e., $\{(\mathbf{c}_1, \mathbf{r}_1), (\mathbf{c}_2, \mathbf{r}_2), \cdots, (\mathbf{c}_\omega, \mathbf{r}_\omega)\}$ 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 $\mathbf{c}_{\omega+1}$. 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*, and *logistic regression*. 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. 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.

To prevent an attacker from successfully modeling an Arbiter PUF, several authentication protocols either keep the response bits r 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

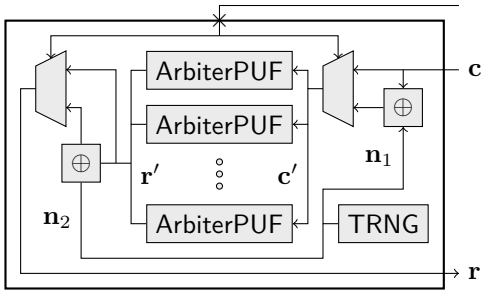
To facilitate the understanding of the analyzed authentication protocols for a visually oriented reader, Figure 3 shows the hardware of a PUF-enabled device. The implementation efficiency is evidently reflected but is of secondary importance in light of the newly revealed security issues. Intermediary 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. Although the protocols are specified and attacked in chronological order, there is no problem in reading Sections 3.1 to 3.3 in a different order.

3.1 PolyPUF

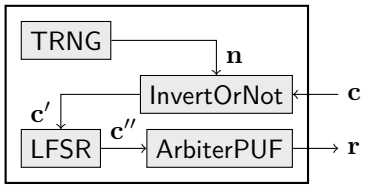
3.1.1 Specification

The so-called PolyPUF protocol of Konigsmark, Chen, and Wong [KCW16], where "Poly" stands for "Polymorphic", is specified in Figure 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 α CRPs $(\mathbf{c}', \mathbf{r}')$ and trains a predictive model $\hat{\mathbf{m}}$ for each Arbiter PUF. A suggested value for α is 5000. After the enrollment, direct access to the CRPs $(\mathbf{c}', \mathbf{r}')$ is irreversibly disabled.

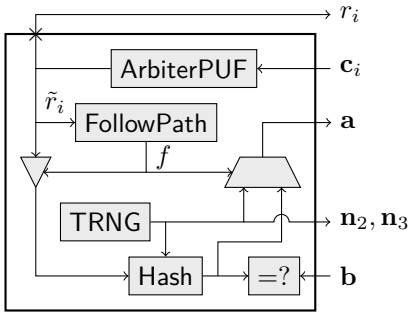
To prevent machine learning by an attacker, a device that is deployed in the field XORs the received challenge $\mathbf{c} \in \{0,1\}^\lambda$ with λ/γ concatenated copies of a nonce $\mathbf{n}_1 \in \{0,1\}^\gamma$ in order to form the PUF input $\mathbf{c}'$. Likewise, the released response $\mathbf{r} \in \{0,1\}^\lambda$ 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



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



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



(c) PUF–FSM of Gao et al. [GMA+17]

Figure 3: The hardware of a PUF-enabled device for the analyzed authentication protocols.

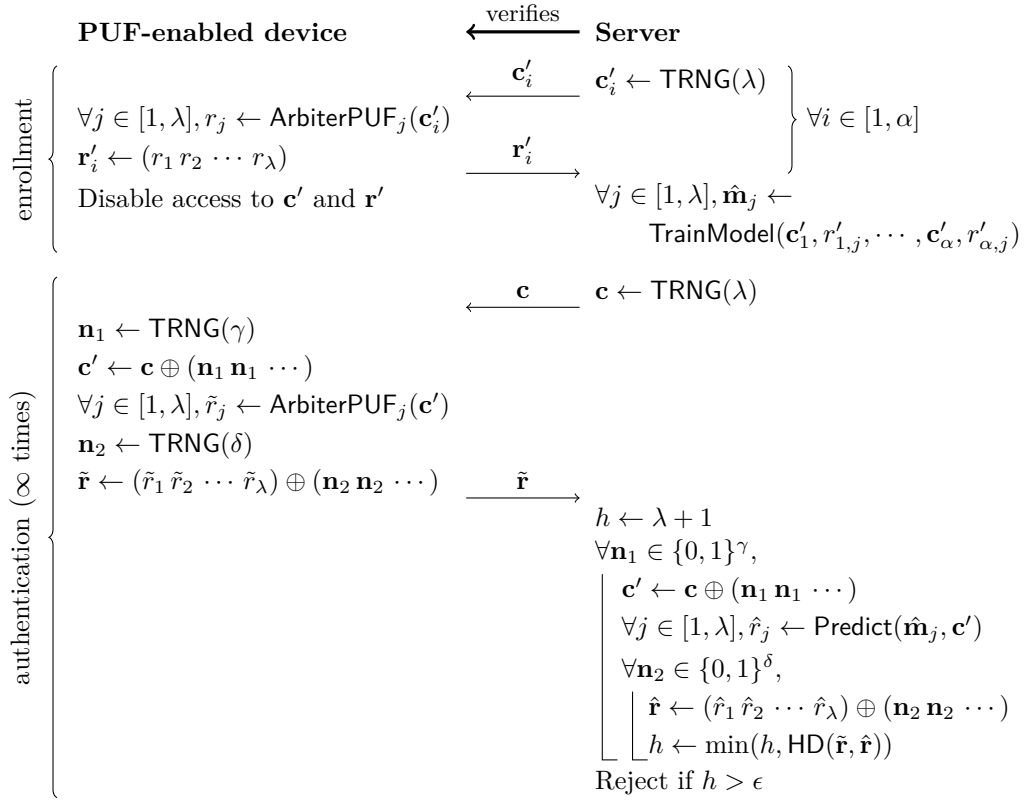


Figure 4: The unilateral authentication protocol of Konigsmark et al. [KCW16].

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 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 random 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 ω training CRPs $(\mathbf{c}_i, \mathbf{r}_i)$ where the nonces $(\mathbf{n}_1, \mathbf{n}_2)$ 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 $i \in [2, \omega]$, are randomly chosen such that $\mathsf{HD}(\mathbf{c}_i, \mathbf{c}_{i-1}) = 1$. 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. It is a triumph that, even with $\beta = 10^8$ device queries and 10–30 neurons in the hidden layer, the obtained modeling accuracies do not significantly exceed the ideal value of 50%.

3.1.2 Attack

Ironically, the authors overlook that their 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 $\mathsf{HD}(\mathbf{r}_i, \mathbf{r}_{i-1}) \leq \epsilon_1$, where ϵ_1 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 Figure 5 demonstrates for $\lambda = 64$, $\gamma = 2$, and $\delta = 3$ 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.

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 ϵ_2. In order to obtain all unique values of R_i, each response $\mathbf{r}$ is XORed with 2^δ possible patterns $(\mathbf{n}_2\, \mathbf{n}_2 \cdots)$. 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 $\beta \gg w$ queries to the PUF-enabled device. 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.

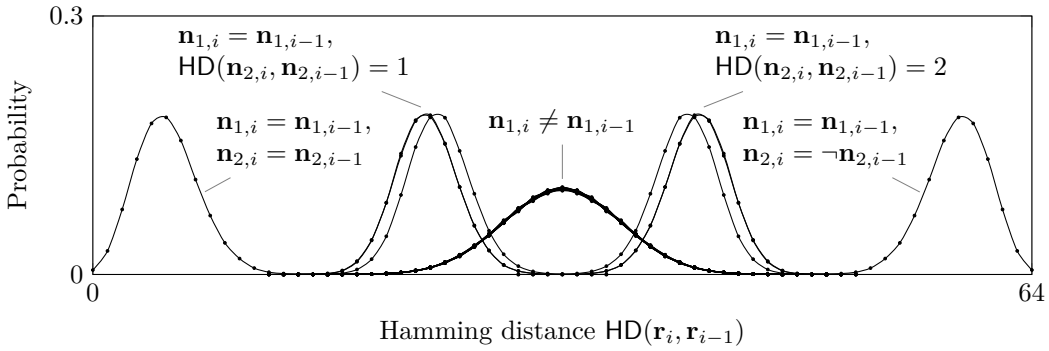


Figure 5: Feasibility study of an attack on the protocol of Konigsmark et al. [KCW16], where $\lambda = 64$, $\gamma = 2$, and $\delta = 3$.

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 $i < \beta$ **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 $h_{(1)} \leq h_{(2)} \leq \cdots \leq h_{(2^\delta)}$
    $f \leftarrow \left(h_{(1)} \leq \epsilon_1\right)$
    $f \leftarrow f \wedge \left(h_{(2)} - h_{(1)} \geq \epsilon_2\right)$
  $i \leftarrow i + j$
  if $f = 1$ **then**
    $w \leftarrow w + 1$
    $\mathbf{c}_w \leftarrow \mathbf{c}$
    $\mathbf{r}_w \leftarrow \mathbf{a}_{(1)}$

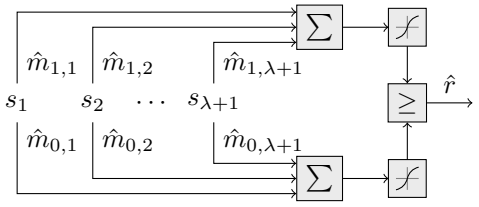


Figure 6: A pair of single-neuron networks.

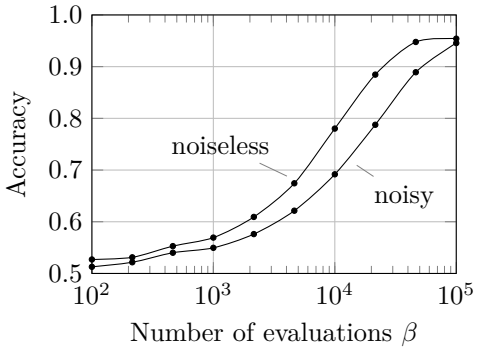


Figure 7: The accuracy of modeling an Arbiter PUF that is used in the protocol of Konigsmark et al. [KCW16], where $\lambda = 64$, $\gamma = 2$, and $\delta = 3$.

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 Figure 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 as a function of the approximate number of device queries β. Fewer than $\beta = 10^5$ queries suffice to exceed accuracies of 90%, whereas Konigsmark et al. [KCW16] were unable to exceed the ideal value of 50% using $\beta = 10^8$ queries. Each dot corresponds to five runs of Algorithm 1 using different devices and hence displays the averaged average of modeling $5\lambda = 320$ Arbiter PUFs; parameters were configured as $\epsilon_1 = \epsilon_2 = 14$. 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 RPUF

3.2.1 Specification

The so-called RPUF protocol of Ye, Hu, and Li [YHL16], where "R" stands for "Randomized", is specified in Figure 8. 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 $\mathbf{r}$.

To enroll a device, the server requests the response $\mathbf{r}$ to each out of α 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 α 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.

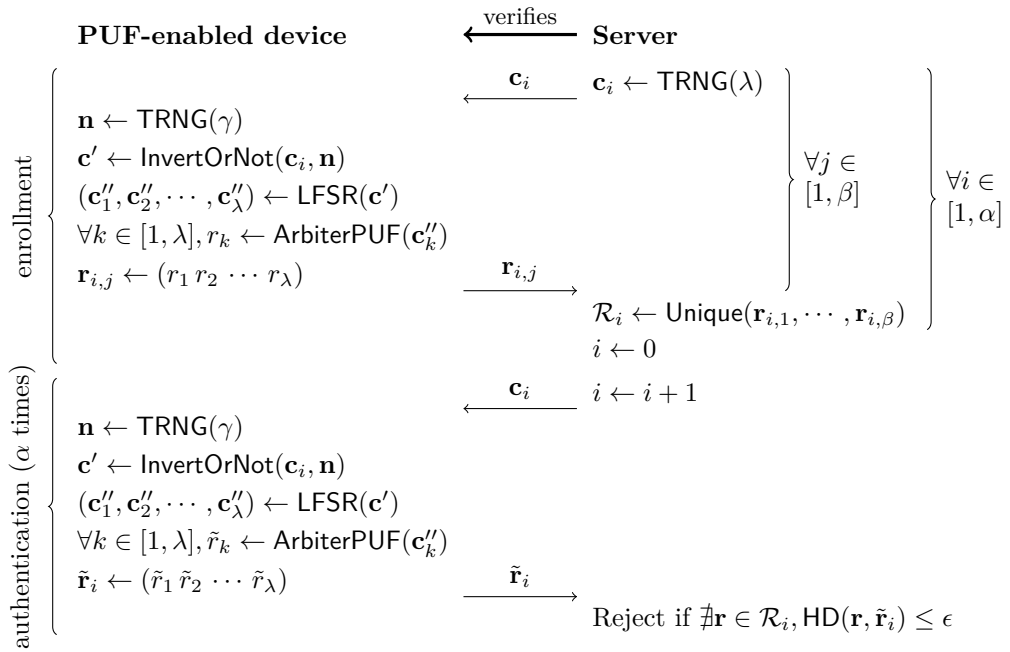


Figure 8: The unilateral authentication protocol of Ye et al. [YHL16].

3.2.2 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 an order 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 q challenges $\mathbf{c} \in \{0,1\}^\lambda$. There are hence $(2^\gamma!)^q$ possibilities for constructing a combined training and testing set of $2^\gamma \cdot q \cdot \lambda$ 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 q genuine protocol runs can iterate over $2^{\gamma \cdot q}$ combined training and testing sets of $q \cdot \lambda$ transformed CRPs $(\mathbf{s}'', r)$ each. Figure 9(a) shows that for either strategy, a relatively limited computational effort corresponds to a relatively large number of CRPs.

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 Figure 9(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 $q = 1$ 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].

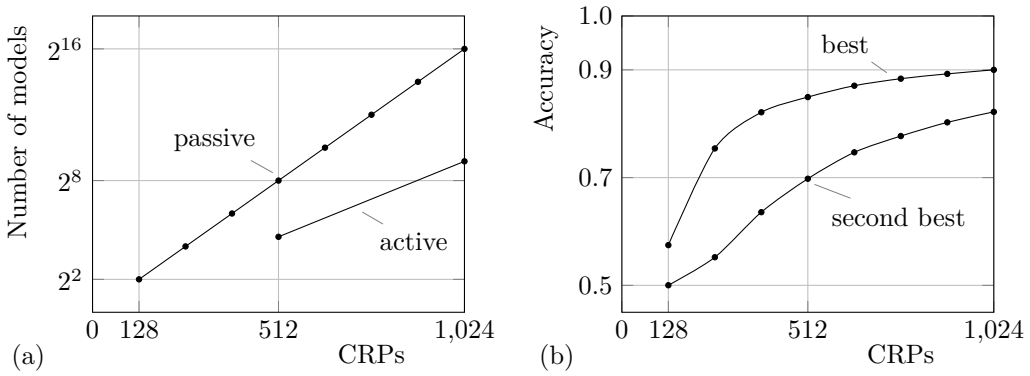


Figure 9: The first phase of an attack on the protocol of Ye et al. [YHL16], where $\lambda = 128$ and $\gamma = 2$. For an either passive or active attacker, subplot (a) shows the number of possible mappings between a given number of transformed challenges $\mathbf{s}''$ and an equal number of response 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 $M \sim N\big(\mathbf{0}, \mathrm{diag}(1/2, 1, 1, \cdots, 1, 1/2)\big)$. 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.

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

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.

3.3 PUF–FSM

3.3.1 Specification

The so-called PUF–FSM protocol of Gao, Ma, Al-Sarawi, Abbott, and Ranasinghe [GMA+17], where "FSM" stands for "finite-state machine", is specified in Figure 10. Each device hosts an Arbiter PUF with λ challenge bits. A suggested value for λ is 64. To enroll a given device, the server collects α CRPs $(\mathbf{c}, r)$ so that an accurate predictive model $\hat{\mathbf{m}}$ can be trained. A suggested value for α is 10^4. Both response bits r, which are the result of a comparison $v \lessgtr 0$, 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.

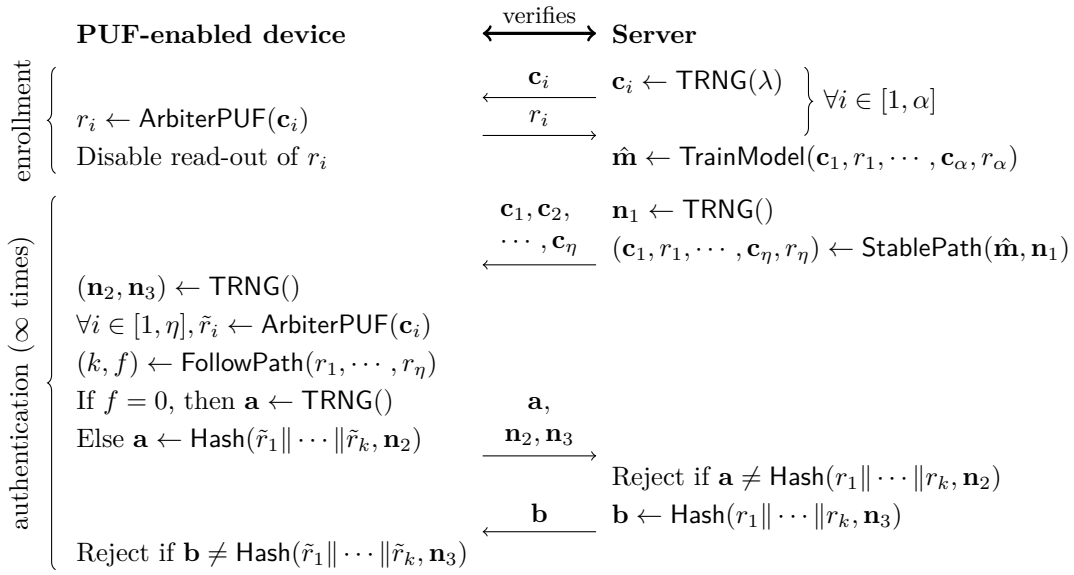


Figure 10: The mutual authentication protocol of Gao et al. [GMA+17].

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}, which is a fairly common technique to obtain a low failure rate [Del17, Chapter 4]. 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 Figure 11, 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 $k \leq \eta$ bits suffices to reach the end state. A value for constant η has not been suggested. The proposed FSM consists of β stages, where constant β is odd. A suggested value for β is 41. Odd- and even-numbered stages, in turn, consist of 1 and $\gamma > 1$ states respectively. A suggested value for γ is 3. Each state transition is defined by a δ-bit substrings of response $\mathbf{x} \in \{0,1\}^\eta$. A total of $k \in [(\beta - 1)\delta, \eta]$ 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 β and stages 1 to $\beta - 1$ 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 k 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 large training set of CRPs and hence model the Arbiter PUF such that only the sign of $\hat{\mathbf{m}}$ remains unknown.

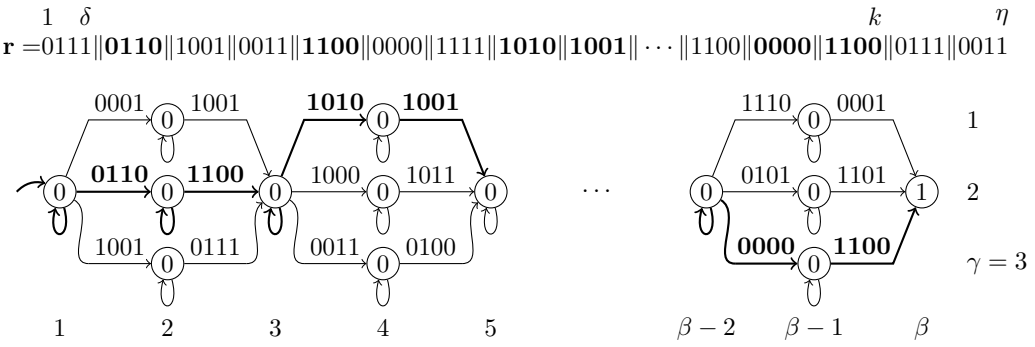


Figure 11: The FSM of Gao et al. [GMA+17].

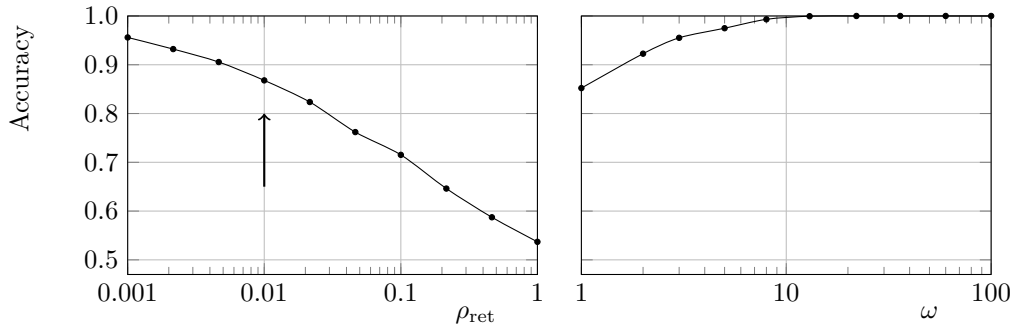
3.3.2 Attack

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 corresponding PUF. This model $\hat{\mathbf{m}}$, in turn, allows the attacker to impersonate either the device or the server a virtually unlimited number of times. Although the authors are aware of Becker's work [Bec15a] and designed their protocol such that not only the response bits r but also their corresponding error rates p_{error} remain internal to the device, 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 $\omega = 1$ challenge $\mathbf{c} \in \{0,1\}^\lambda$, having transformed version $\mathbf{s} \in \{-1,1\}^{\lambda+1}$, the two best guesses for a 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 one bit of entropy and is hence irrelevant from a security perspective. Figure 12(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.

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 $\hat{\mathbf{m}}$. 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. It is crucial to define an appropriate fitness function, i.e., $\mathsf{fitness} : \{0,1\}^{\lambda+1} \to \mathbb{R}$. We instantiate the fitness function as shown in (3).

$$\mathsf{fitness}(\hat{\mathbf{m}}) = \frac{1}{\omega}\sum_{i=1}^{\omega}\left|\mathbf{s}_i\,\hat{\mathbf{m}}^T\right| \Bigg/ \frac{1}{q}\sum_{i=1}^{q}\left|\mathbf{s}_{\text{ref},i}\,\hat{\mathbf{m}}^T\right| . \tag{3}$$

The ω transformed challenges $\mathbf{s}_i$ in its numerator originate from a genuine protocol run and are hence known to have stable response bits r_i. The q 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}. We use the same $q = 1000$ transformed challenges $\mathbf{s}_{\text{ref},i}$ for each evaluation of the fitness function. 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. Default values suffice for all parameters of the adopted CMA-ES algorithm, e.g., the population size is $4 + \lfloor 3\ln(\lambda+1)\rfloor = 16$. For a retention ratio $\rho_{\text{ret}} = 1\%$ and $\omega = 10$ server-defined challenges $\mathbf{c}_i$, Figure 12(b) shows that



(a) $\omega = 1$ (b) $\rho_{\text{ret}} = 0.01$

Figure 12: The accuracy of modeling an Arbiter PUF with $\lambda = 64$ challenge bits that is used in the protocol of Gao et al. [GMA+17]. 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 $\omega \in [1, 100]$ 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. In subplot (a), models $\hat{\mathbf{m}}$ are directly derived from $\omega = 1$ transformed challenge $\mathbf{s}$. In subplot (b), we use CMA-ES. Because its randomized training algorithm does not always converge to an accurate model $\hat{\mathbf{m}}$, we only retain the best out of five trials.

the best out of two 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 long response $\mathbf{r} \in \{0,1\}^\eta$ into a secret key $\mathbf{k} \in \{0,1\}^\kappa$ through a fuzzy extractor [DORS08] and then use a keyed cryptographic algorithm to perform the authentication [Del17, Section 5.2]. Realizations of a fuzzy extractor are usually based on an error-correcting code and requires the storage of public helper data. 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 the protocols of Konigsmark et al. [KCW16], Ye et al. [YHL16], and Gao et al. [GMA+17], taking shortcuts might be fatal for the system security. The irony is that for two out of three protocols, the obtained reductions in hardware footprint are small, if existing at all, and might not even have justified taking the risk.

The protocol of Gao et al. [GMA+17] 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 this approach are known to exist [Bec15b] [Del17, Chapter 5], but several versions still hold up. A final note is that a sizeable helper data string needs to be transferred with every protocol run, or alternatively for a non-reversed fuzzy extractor, stored permanently by a device. The FSM, however, requires the repeated transfer of more than $160 \cdot 64 = 10240$ challenge bits c and is thus more expensive in this regard.

The protocol of Konigsmark et al. [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, Figure 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 more robust array of PUFs.

On the bright side, the protocol of Ye et al. [YHL16] allows for an 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].

5 Conclusion

Through the use of custom-tailored machine learning techniques, we were able to construct an accurate predictive model of the Arbiter PUFs that are used in the protocols of Konigsmark et al. [KCW16], Ye et al. [YHL16], and Gao et al. [GMA+17], and hence enable an impersonation attack.

Acknowledgement

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.

[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.

[GMA+17] 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*, 99(99):5, 2017.

[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.

[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.

[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.

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

[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.