PUF Modeling Attacks: An Introduction and Overview

Ulrich Rührmair, Jan Sölter

***Abstract*—Machine learning (ML) based modeling attacks are the currently most relevant and effective attack form for so-called Strong Physical Unclonable Functions (Strong PUFs). We provide an overview of this method in this paper: We discuss the basic conditions under which it is applicable; the ML algorithms that have been used in this context; the latest and most advanced results on simulated and silicon data; the right interpretation of existing results; and possible future research directions. The paper corresponds to one of the talks of the hot topic session "How Secure are PUFs Really?" at DATE 2014.**

***Index Terms*—Physical Unclonable Functions, Machine Learning, Modeling Attacks, Cryptanalysis**

I. Introduction

Electronic devices are now pervasive in our everyday life. This makes them an accessible target for adversaries, leading to a host of security and privacy issues. Classical cryptography offers several measures against these problems, but they all rest on the concept of a secret binary key: It is assumed that the devices can contain a piece of information that is, and remains, unknown to the adversary. Unfortunately, it can be difficult to uphold this requirement in practice. Physical attacks such as invasive, semi-invasive, or side-channel attacks, as well as software attacks like API-attacks and viruses, can lead to key exposure and full security breaks. The fact that the devices should be inexpensive, mobile, and cross-linked aggravates the problem.

The described situation was one motivation that led to the development of *Physical Unclonable Functions (PUFs)*. A PUF is a (partly) disordered physical system P that can be challenged with so-called external stimuli or challenges C_i, upon which it reacts with corresponding responses termed R_{C_i}. Contrary to standard digital systems, a PUF's responses shall depend on the nanoscale structural disorder present in the PUF. This disorder cannot be cloned or reproduced exactly, not even by its original manufacturer, and is unique to each PUF. As PUF responses can be noisy, suitable error correction techniques like fuzzy extractors [12] may be applied in practice to obtain stable outputs R'_{C_i}. Assuming successful error compensation, any PUF P can be regarded as an individual function F_P that maps challenges C_i to (stable) responses R'_{C_i} (compare [40]).

Due to its complex and disordered structure, a PUF can avoid some of the shortcomings associated with digital keys. For example, it is usually harder to read out, predict, or derive its responses than to obtain the values of digital keys stored in non-volatile memory. This fact has been exploited for various PUF-based security protocols. Prominent examples include schemes for identification and authentication [32], [14], key exchange or digital rights management purposes [15].

For a more detailed introduction and overview of PUFs, we refer the reader to a recent paper by Rührmair, Devadas and Koushanfar [37], or to the summary paper of the first talk in this DATE 2014 session [35].

II. PUF Modeling Attacks and Their Applicability

What are PUF modeling attacks? Under which conditions and to which PUF types are they applicable? In general, modeling attacks on PUFs presume that an adversary Eve has, in one way or the other, collected a subset of all CRPs of the PUF. She then tries to derive a numerical model from this CRP data, i.e., a computer algorithm which correctly predicts the PUF's responses to arbitrary challenges with high probability. Machine learning (ML) techniques are a natural and powerful tool for this task, but also other methods like linear programming or algebraic techniques have been applied in the past [13], [25], [30], [27], [44], [40], [42], [8]. If and how the required CRPs can be collected, and how relevant modeling attacks are in practice, very strongly depends on the considered type of PUF. It again makes sense to distinguish between Weak PUFs and Strong PUFs here (see [37], [36], [40], [41], [35]).

1) Strong PUFs: Strong are the PUF class for which modeling attacks have been designed originally, and to which they are best applicable. The reason is that they usually have no protection mechanisms that restrict Eve in freely applying challenges and reading out their responses [37], [36], [40], [35]. A Strong PUF's responses are usually not post-processed on chip in a protected environment [32], [45], [28], [16], [24], [26]. Most electrical Strong PUFs furthermore operate at frequencies of a few MHz [24]. Therefore even short physical access periods enable Eve to read-out and collect many CRPs. A yet further CRP source is simple protocol eavesdropping, for example on standard Strong PUF-based identification protocols, where the CRPs are sent in the clear [32]. Both CRP sources are part of the established, general attack model for PUFs.

Once a predictive model for a Strong PUF has been derived, the two main security features of a Strong PUF no longer hold: The PUF is no longer *unpredictable* for parties that are not in physical possession of the PUF; and the *physical unclonability* of the PUF is overcome by the fact that the digital simulation algorithm can be cloned and distributed arbitrarily. Any Strong PUF protocol which is built on these two features is then no longer secure. This includes any standard, widespread Strong

PUF protocols known to the authors. [1]

For example, if Eve can use her intermediate physical access in a PUF-based key exchange protocol [10], [3] to derive a predictive model of the PUF, she can later predict the key that was exchanged between the honest parties. A similar effect occurs in 1-out-of-2 oblivious transfer (OT) protocols [34], [3]: If the OT-receiver can derive a numerical model of the PUF before he physically transfers the PUF to the OT-sender, he can later break the security of the sender, and learn *both* transfered bits b_0 and b_1. Also in the CRP-based, standard identification protocols for Strong PUFs [31], [32], a numerical model can be used to impersonate the original PUF.

Concerning applications where the form factor of the PUF may play a role, such as smartcards, it is important to emphasize that the simple additive simulation models derived in most modeling attacks can be implemented in similar environments as the original PUFs, and with a relatively small number of gates. An active fraudster can come so close to the original form factor in a newly set-up, malicious smartcard hardware that the difference is very difficult to notice in practice.

2) Weak PUFs: Weak PUFs (or POKs) are PUFs with few, fixed challenges, in the extreme case with just one challenge [40], [37]. It is usually assumed that their response(s) remain inside the PUF-carrying hardware, for example for the derivation of a secret key, and are not easily accessible for external parties. Weak PUFs are the PUF class that is the least susceptible to modeling attacks.

They only apply to them under relatively rare and special circumstances: namely if a Strong PUF, embedded in some hardware system and with a not publicly accessible CRP interface, is used to implement the Weak PUF. This method has been suggested in [13], [45]. Thereby only a few (of the very many possible) challenges of the Strong PUF are used for internal key derivation. Our attacks make sense in this context only in the special case that the Strong PUF challenges C_i^* that are used in the key derivation process are not yet fixed in the hardware at the time of fabrication, but are selected later on. For one reason or another, the adversary may learn about these challenges at a point in time that lies after his point of physical access to the PUF. In this case, machine learning and modeling of the Strong PUF can help the adversary to derive the key, even though the points in time where he has access to the PUF and where he learns the challenges C_i^* strictly differ. In order to make our ML methods applicable in this case, one must assume that the adversary was able to collect many CRPs of the Strong PUF, for example by physically probing the internal digital response signals of the Strong PUF to randomly injected challenges, or by malware that abuses internal access to the underlying Strong PUF's interface. We comment that the latter scenarios obviously represent very strong attack models. Under comparable circumstances also many standard Weak PUFs and other secret key based architectures break down.

In any other cases than the above, modeling attacks are not relevant for Weak PUFs. This means that they are not applicable to the majority of current Weak PUF implementations, including the Coating PUF [46], SRAM PUF [17], Butterfly PUF [22], and similar architectures.

We conclude by the remark that this should not lead to the impression that Weak PUFs are necessarily more secure than other PUFs. Other attack strategies can be applied to them, including invasive, side-channel and virus attacks, but they are not the topic of this paper. For example, probing the output of the SRAM cell prior to storing the value in a register can break the security of the cryptographic protocol that uses these outputs as a key. Also physical cloning strategies for certain Weak PUFs have been reported recently [19]. These attacks are discussed in depth in some of the other talks and papers of the session [48], [18].

III. The Process of PUF Modeling and Its Main Challenges

We will now discuss the basic process of machine-learning based modeling and its main challenges in greater detail. The modeling process essentially is a two-step procedure. Its first step consists of setting up an internal, parametric model of the PUF. This requires finding a function $F(\cdot,\cdot)$ which correctly describes the PUF's challenge-response behavior (or input-output behavior). F should take as input (i) a challenge C_i that is applied to the PUF, and (ii) values that describe the internal, unique, fabrication-dependent parameters of the PUF. The latter are usually given by some multidimensional parameter vector $\vec{w}$ with values in the reals or rationals. F then shall output the correct corresponding responses R_{C_i} of the PUF on challenge C_i, i.e., $F(\vec{w}, C_i) = R_{C_i}$.

In a second step, the parametric model F is used together with a suitably chosen ML algorithm for PUF learning. The algorithm takes as input a large set of CRPs of the PUF, the so-called *"training set"*. Its goal is to find a concrete vector $\vec{w}'$ that leads to a good prediction quality of F. This prediction quality is evaluated on a second, independent CRP set, the so-called *"test set"*. [2] One noteworthy aspect is that the vector $\vec{w}'$ derived by the ML algorithm do not need to be equal to the "real" parameter values $\vec{w}$ of the considered PUF instance. Depending on the PUF architecture and the model F, many different vectors can lead to an almost equivalent output behavior.

What are the main challenges in this process? The first issue is finding a suitable model of the considered PUF at all. Given some knowledge about the physical mechanisms underlying the PUF (e.g., about the used integrated circuits or optical systems), however, a parametric model usually can be developed

[1] One sole potential exception are a few recent bit commitment protocols for PUFs that were explicitly designed for the so-called *"bad PUF model"* or the *"malicious PUF model"*. They promise to uphold security even if one or all used PUFs are *not* unpredictable (see partly van Dijk and Rührmair [11] and mainly Damgard and Scafuro [7]). At least some of these protocols are relatively non-standard in a number of aspects, however, such as the assumed input/output lengths of the used PUFs. Asides from these two special protocols, all other practically relevant, widespread Strong PUF schemes straightforwardly break down if the main security feature of the Strong PUF is violated by a modeling attack, namely their unpredictability.

[2] Typically, the test set can have a fixed size of a few thousand CRPs, while the size of the training set required for successful learning usually strongly depends on the PUF and its complexity. It can be linear for certain PUFs (like the Arbiter PUF), and very large, even exponential in some system parameter for others (XOR Arbiter PUF, Lightweight PUF). Overall, the CRP requirements for a sucessful ML attack are mainly determined by the size of the training set.

fairly easily. Finding a model that is computationally efficient can be harder. To that end, the underlying mechanisms in the PUF may need to be simplified or suitably approximated.

A second problem is to identify a suited ML algorithm that performs efficiently, i.e., with a polynomial effort in some system parameter of the PUF (typically in the PUF's challenge bitlength) and in the aspired prediction quality. However, since PUFs cannot be scaled indefinitely due to cost and stability constraints, also mild exponential growth rates may be acceptable in certain situations, as long as they break practically relevant PUF sizes. One well-known example for this effect are the modeling attacks on XOR Arbiter PUFs: The practical stability of these PUFs decreases exponentially in their number of XORs. Current attacks have a complexity that increases exponentially in the same parameter, but still reach practically relevant size and complexity levels (compare [40], [42] and Section IV-C).

The two problems of constructing suitable models and of finding efficient ML algorithms are closely intertwined. First of all, many powerful ML algorithms require certain additional properties of the parametric models in order to be applicable. They may demand, for example, that the model is linearly separable (this allows the application of support vector machines), or that it is differentiable (this enables working with logistic regression, among other techniques). Only the identification of such models makes these powerful ML algorithms applicable. Secondly, the performance of the ML algorithm may depend strongly on the model and its computational efficiency, too. Both the model and the ML algorithm may thus need to be optimized jointly for optimal attacks performance. This often requires some in-depth knowledge about ML.

IV. An Example: Arbiter PUFs and Variants

Let us now illustrate our above discussion by virtue of a concrete example, namely by applying machine learning based modeling to Arbiter PUFs and their variants (i.e., XOR Arbiter PUFs, Lightweight (LW) PUFs, and Feed Forward (FF) Arbiter PUFs). Arbiter PUFs and variants are currently the best investigated electrical Strong PUF type. In their basic form, they were first introduced in [16] [24] [45].

We will discuss the employed machine learning methods in Section IV-A, the attacked PUF types in Section IV-B, and survey the latest results that have been applied in existing publications [40], [42] on simulated and silicon data in Section IV-C. Note in this context that the use of error-free, simulated CRP data obtained by a linear additive delay model is one standard, highly established method of obtaining ML results on Arbiter PUF variants, even though it has one unexpected side effects: It can lead to prediction rates that are higher than the real-world temperature stability of the attacked PUF (compare the discussion in Section II-G of [42]).

A. Employed Machine Learning Methods

Various machine techniques have been applied to PUFs in the literature [13], [25], [30], [27], [44], [40], [42], [8], including Support Vector Machines (SVMs), Logistic Regression (LR), Evolution Strategies (ES), and briefly also Neural Nets and Sequence Learning [40]. The two approaches described in the sequel have been identified as the currently best performing methods.

1) Logistic Regression with Rprop: LR is a well-investigated supervised machine learning framework, which has been described, for example, in [2]. In its application to PUFs with single-bit outputs, each challenge $C = b_1 \cdots b_k$ is assigned a probability $p\left(C, r \mid \vec{w}\right)$ that it generates a output $r \in \{0, 1\}$). The vector $\vec{w}$ encodes the relevant internal parameters, for example the particular runtime delays, of the individual PUF. The probability is given by the logistic sigmoid $\sigma(x) = (1 + e^{-x})^{-1}$ acting on a function $f(\vec{w}, C)$ parametrized by the vector $\vec{w}$ as

$$p\left(C, r \mid \vec{w}\right) = r\sigma(f) + (1 - r)(1 - \sigma(f)). \tag{1}$$

Thereby the decision function f determines through $f = 0$ a decision boundary of equal output probabilities. For a given training set $\mathcal{M}$ of CRPs the boundary is positioned by choosing the parameter vector $\vec{w}$ in such a way that the likelihood of observing this set is maximal, respectively the negative log-likelihood is minimal:

$$\hat{\vec{w}} = \mathsf{argmin}_{\vec{w}} \;\; l(\mathcal{M}, \vec{w}) = \mathsf{argmin}_{\vec{w}} \sum_{(C,\, r) \in\, \mathcal{M}} -ln\, p\left(C, r \mid \vec{w}\right) \tag{2}$$

As there is no analytical solution to determine the optimal parameter vector $\hat{\vec{w}}$, it has to be optimized iteratively, e.g., using the gradient information

$$\nabla l(\mathcal{M}, \vec{w}) = \sum_{(C,\, r) \in\, \mathcal{M}} \left(\sigma\left(f\left(\vec{w}\right)\right) - r\right) \nabla f(\vec{w}) \tag{3}$$

From the different possible optimization methods, RProp [2], [33] has been identified as optimal in earlier ML works on PUFs [40], [42]. RProp makes a very big difference in convergence speed and stability of the LR algorithms (k-XOR Arbiter PUFs for medium or large k were only learnable with RProp).

In general, logistic regression has the asset that the examined problems need not be (approximately) linearly separable in feature space, as is required for successful application of support vector machines, for example, but merely differentiable.

2) Evolution Strategies: Evolution Strategies (ES) [1], [43] belong to an ML subfield known as population-based heuristics. They are inspired by the evolutionary adaptation of a population of individuals to certain environmental conditions. In our case, one individual in the population is given by a concrete instantiation of the runtime delays in a PUF, i.e., by a concrete instantiation of the vector $\vec{w}$ appearing in Eqns. 2 and 3. The environmental fitness of the individual is determined by how well it (re-)produces the correct CRPs of the target PUF on a fixed training set of CRPs. ES runs through several evolutionary cycles or so-called *generations*. With a growing number of generations, the challenge-response behavior of the best individuals in the population better and better approximates the target PUF. ES is a randomized method that neither requires an (approximately) linearly separable problem (like Support Vector Machines), nor a differentiable model

(such as LR with gradient descent); a merely parameterizable model suffices. Since all known electrical PUFs are easily parameterizable, ES is a very well-suited attack method.

B. Parametric Models for Arbiter PUF and Their Variants

It has become standard to describe the functionality of Arbiter PUFs and their variants via an additive linear delay model [25], [40], [42]. The overall delays of the two racing signals are modeled as the sum of the delays in the stages. The final delay difference Δ between the upper and the lower path in an n-bit Arbiter PUF is expressed as

$$\Delta = \vec{w}^T \vec{\Phi}, \tag{4}$$

where $\vec{w}$ and $\vec{\Phi}$ are vectors of dimension $n+1$. The parameter vector $\vec{w}$ encodes the delays for the subcomponents in the Arbiter PUF stages, whereas the feature vector $\vec{\Phi}$ is solely a function of the applied n-bit challenge C [25], [40], [42].

In greater detail, the following holds. We denote by $\delta_i^{0/1}$ the runtime delay in stage i for the crossed (1) respectively uncrossed (0) signal path. Then

$$\vec{w} = (w^1, w^2, \ldots, w^k, w^{n+1})^T, \tag{5}$$

where $w^1 = \frac{\delta_1^0 - \delta_1^1}{2}$, $w^i = \frac{\delta_{i-1}^0 + \delta_{i-1}^1 + \delta_i^0 - \delta_i^1}{2}$ for all $i = 2, \ldots, n$, and $w^{n+1} = \frac{\delta_n^0 + \delta_n^1}{2}$. Furthermore,

$$\vec{\Phi}(\vec{C}) = (\Phi^1(\vec{C}), \ldots, \Phi^k(\vec{C}), 1)^T, \tag{6}$$

where $\Phi^l(\vec{C}) = \prod_{i=l}^{n}(1 - 2b_i)$ for $l = 1, \ldots, n$.

The output r of an Arb-PUF is determined by the sign of the final delay difference Δ:

$$r = \Theta(\Delta) = \Theta(\vec{w}^T \vec{\Phi}). \tag{7}$$

with Θ being the Heaviside step function, i.e., $\Theta(x) = 0$ if $x < 0$ and $\Theta(x) = 1$ if $x \geq 0$. Eqn. 7 shows that the vector $\vec{w}$ via $\vec{w}^T \vec{\Phi} = 0$ determines a separating hyperplane in the space of all feature vectors $\vec{\Phi}$. Any challenges C that have their feature vector located on the one side of that plane give response $r = 0$, those with feature vectors on the other side $r = 1$. Determination of this hyperplane allows prediction of the PUF and can be achieved by setting the decision function f in Eqn. 1 to the linear delay model $f = \vec{w}^T \vec{\Phi}$.

More complex architectures that use k standard Arbiter PUFs in parallel, possibly together with special input or output mappings, can then simply be modelled by using the linear additive delay model for each of the parallel Arbiter PUFs. Overall, this involves k feature vectors $\vec{\Phi}_1, \ldots, \vec{\Phi}_k$ derived from the effective challenges at the individual Arbiter PUFs (given by the input mapping) and k weight vectors $\vec{w}_1, \ldots, \vec{w}_k$:

$$o = \Theta(\prod_{i=1}^{k} \Delta_i) = \Theta(\prod_{i=1}^{k} \vec{w}_i^T \vec{\Phi}_i) \tag{8}$$

Eqn. 8 defines a decision boundary at $\prod_{i=1}^{k} \vec{w}_i^T \vec{\Phi}_i = 0$. Any challenges C that have their feature vector set $\vec{\Phi}_1, \ldots, \vec{\Phi}_k$ located on the one side of the boundary (e.g. $o < 0$ respectively an odd number of individual delays Δ_i smaller than zero) give response $r = 0$, those with feature vectors on the other side $r = 1$. Determination of this boundary allows prediction of the PUF and can be achieved by setting the decision function f in Eqn. 1 to this boundary $f = \prod_{i=1}^{k} \vec{w}_i^T \vec{\Phi}_i$. It implies an optimization along the gradient in Eqn. 3:

$$\nabla f(\vec{w}_j) = \vec{\Phi}_j \prod_{i \neq j} \vec{w}_i^T \vec{\Phi}_i \tag{9}$$

This principle has been applied to XOR Arbiter PUFs and Lightweight PUFs in the literature.

C. Results

The application of the above ML algorithms and models to Arbiter PUFs and their variants (XOR Arbiter PUF, LW PUF, FF Arbiter PUF) has led to the results shown in Table I, which are taken from [42]). The attacks thereby have been carried out both on simulated and silicon CRP data [42]. Only the XOR-based variants (XOR Arbiter PUF, LW PUF) lead to an exponential complexity, while the other types (standard Arbiter PUFs, FF Arbiter PUFs) can be learned successfully with polynomial effort. In greater detail, as analyzed in [40], [42], the complexity of the attacks on XOR Arbiter PUFs and LW PUFs is exponential in the number of XORs (k). It is merely polynomial of degree k in the challenge bitlength, though. At the same time, the stability of these two architectures is exponentially bad in k, too. This makes their ML-resilient implementation a race between machine learners and circuit designers: Can the design be optimized in its noise level, so that it is practically secure against ML existing attacks, while still being stable? Or can the ML algorithms be optimized and run on more powerful hardware, so that they can reach and break all practically stable XOR Arbiter PUF and LW PUF architectures? The authors of [40], [42] estimate that the former is the case, and that XOR Arbiter PUFs and LW PUFs with 8 XORs and bitlength 512 are still stable, but beyond the reach of current ML methods. Future efforts will have to tell whether this estimate was reasonable or perhaps too optimistic.

V. Discussion and Future Work

A. Discussion

Two straightforward, but biased interpretations of the existing modeling attacks would be the following: (i) All Strong PUFs are insecure. (ii) The long-term security of electrical Strong PUFs can be restored trivially, for example by increasing the PUF's size. Both views are simplistic, and the truth is more involved.

Starting with (i), the current attacks are indeed sufficient to break several delay-based PUF implementations. But there are a number of ways how PUF designers might be able fight back in future designs.

For example, increasing the bitlength n in an XOR Arbiter PUF or Lightweight Secure PUF with k XORs increases the effort of the presented attacks methods as a polynomial function of n with exponent k (in approximation for large n and small or medium k). At the same time, it does not worsen the PUF's stability [9]. It has hence been argued in [40], [42] that one might disable attacks through choosing a strongly increased value of n and a value of k that corresponds

PUF-Type	No. of XORs/ FF-Loops	ML Method	Bit Length	CRP Source	CRPs ($\times 10^3$)	Training Time	Prediction Rate
Arbiter PUF	—	LR	128	Simulation	39.2	2.10 sec	99.9%
			64	FPGA	6.5	0.83 sec	99%
			64	ASIC	6.5	0.76 sec	99%
XOR Arbiter PUF	5	LR	128	Simulation	500	16:36 hrs	99%
			64	FPGA	78	39 min	99%
			64	ASIC	78	18:09 min	99%
Lightweight PUF	5	LR	128	Simulation	1000	267 days	99%
FF Arbiter PUF	8	ES	128	Simulation	50	3:15 hrs	99%

TABLE I

SOME MAIN ATTACK RESULTS ON ARBITER PUFS AND VARIANTS THEREOF, TAKEN FROM [42]. BOTH SIMULATED, NOISE-FREE CRPS AND SILICON CRPS FROM FPGAS AND ASICS HAVE BEEN USED IN THE ML EXPERIMENTS. THE PREDICTION RATES AND TRAINING TIMES ARE AVERAGED OVER SEVERAL INSTANCES. ALL PRESENTED TRAINING TIMES ARE CALCULATED AS IF THE ML EXPERIMENT WAS RUN ON ONLY *one single* CORE OF *one single* PROCESSOR. USING k CORES WILL APPROXIMATELY REDUCE THEM BY $1/k$.

to the stability limit of such a construction. For example, an XOR Arbiter PUF with 8 XORs and bitlength of 512 is implementable by standard fabrication processes [9], and currently seems beyond the reach of pure ML-based modeling attacks, so the authors of [40], [42] argue.

Also new design elements that could be added to standard Arbiter PUFs and their variants may raise the attacker's complexity further. One example could be adding nonlinearities, such as AND and OR gates that correspond to MAX and MIN operators [25]. Combinations of Feed-Forward and XOR architectures could be hard to machine learn too, partly because they seem susceptible only to different and mutually-exclusive ML techniques.

Moving away from delay-based PUFs, the exploitation of the dynamic characteristics of current and voltage seems promising, for example in analog circuits [6]. Also special PUFs with a very high information content (so-called SHIC PUFs [38], [39], [21]) could be an option, but only in such applications where their slow read-out speed and their comparatively large area consumption are no too strong drawbacks. Their promise is that they are naturally immune against modeling attacks, since all of their CRPs are information-theoretically independent. Finally, optical Strong PUFs, for example systems based on light scattering and interference phenomena [32], show strong potential in creating high input-output complexity.

Regarding view (ii), PUFs are different from classical cryptoschemes like RSA in the sense that increasing their size often likewise decreases their input-output stability. For example, raising the number of XORs in an XOR Arbiter PUF and Lightweight PUF has an exponentially strong effect *both* on the attacker's complexity *and* on the instability of the PUF. We are yet unable to find parameters that increase the attacker's effort *exponentially* while affecting the PUF's stability merely *polynomially*. Nevertheless, one practically viable possibility is to increase the bitlength of XOR Arbiter PUFs and Lightweight PUFs, as discussed above. Future work will have to show whether the described large polynomial growth of the latter method can persist in the long term, or whether its high degree can be diminished by further analysis.

B. Future Work

The upcoming years will presumably witness some competition between codemakers and codebreakers in the area of Strong PUFs. Similar to the design of classical cryptoprimitives, for example stream ciphers, this process can be expected to converge at some point to solutions that are resilient against the known attacks. Some first attempts into this direction have already been made in [49], [29], [4], [5], [23], but their ML-resilience has not been analyzed thoroughly in the literature yet.

For PUF designers, it may be interesting to investigate some of the concepts that we mentioned above. One major goal will be the development of practical (=stable, area efficient, low cost) Strong PUFs that still possess high modeling resilience. Note in this context that any non-linearity in the PUF design increases its ML-resilience, but usually also worsens its stability, unless special countermeasures are taken.

For PUF breakers, a worthwhile starting point is to improve the attacks presented in this paper through optimized implementations and new ML methods. A performance comparison between our results and earlier approaches that used SVMs and comparable techniques [25], [30] confirms the strong effect of the choice of the right ML-algorithm. Another, qualitatively new path is to combine modeling attacks with extra information obtained from direct physical PUF measurements or from side channels. For example, applying the same challenge multiple times gives an indication of the noise level of a response bit. It enables conclusions about the absolute value of the final runtime difference in the PUF. Such side channel information can conceivably improve the success and convergence rates of ML methods. Some first steps towards these end have been made just recently in a number of works.

REFERENCES

[1] T. Bäck. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford University Press, USA, 1996.

[2] C.M. Bishop et al. *Pattern recognition and machine learning*. Springer New York:, 2006.

[3] C. Brzuska, M. Fischlin, H. Schröder, S. Katzenbeisser. Physical Unclonable Functions in the Universal Composition Framework. *CRYPTO 2011*.

[4] Q. Chen, G. Csaba, P. Lugli, U. Schlichtmann, U. Rührmair. The Bistable Ring PUF: A New Architecture for Strong Physical Unclonable Functions. *HOST 2011*.

[5] Q. Chen, G. Csaba, P. Lugli, U. Schlichtmann, U. Rührmair. Characterization of the Bistable Ring PUF. *DATE 2012*.

[6] G. Csaba, X. Ju, Z. Ma, Q. Chen, W. Porod, J. Schmidhuber, U. Schlichtmann, P. Lugli, and U. Rührmair. Application of mismatched cellular nonlinear networks for physical cryptography. *IEEE CNNA*, 2010.

[7] I. Damgard, A. Scafuro: Unconditionally Secure and Universally Composable Commitments from Physical Assumptions. *Cryptology ePrint Archive, 2013:108*, 2013.

[8] Jeroen Delvaux, Ingrid Verbauwhede: *Side channel modeling attacks on 65nm arbiter PUFs exploiting CMOS device noise*. HOST 2013.

[9] S. Devadas. Physical unclonable functions and secure processors. *Invited Talk, CHES 2009*.

[10] M. van Dijk: *System and method of reliable forward secret key sharing with physical random functions.* US Patent No. 7,653,197, October 2004.

[11] M. van Dijk, U. Rührmair: Physical Unclonable Functions in Cryptographic Protocols: Security Proofs and Impossibility Results. *Cryptology ePrint Archive, 2012:228*, 2012.

[12] Y. Dodis, R. Ostrovsky, L. Reyzin, L., A. Smith: *Fuzzy extractors: How to generate strong keys from biometrics and other noisy data*. SIAM Journal on Computing, 38(1), 97-139, 2008.

[13] B. L. P. Gassend. *Physical random functions*. MSc thesis, MIT, 2003.

[14] B. Gassend, D. Clarke, M. Van Dijk, and S. Devadas. Silicon physical random functions. *ACM CCS 2002*.

[15] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas. Controlled physical random functions. *ACSAC 2002*.

[16] B. Gassend, D. Lim, D. Clarke, M. Van Dijk, and S. Devadas. Identification and authentication of integrated circuits. *Concurrency and Computation: Practice & Experience*, 16(11):1077–1098, 2004.

[17] J. Guajardo, S. Kumar, G.J. Schrijen, and P. Tuyls. FPGA intrinsic PUFs and their use for IP protection. *CHES 2007*.

[18] C. Helfmeier, J.P. Seifert: *XXX*. Design, Automation and Test in Europe (DATE'14), 2014.

[19] C. Helfmeier, C. Boit, D. Nedospasov, J.P. Seifert: Cloning Physically Unclonable Functions. *HOST 2013*.

[20] D.E. Holcomb, W.P. Burleson, and K. Fu. Initial sram state as a fingerprint and source of true random numbers for RFID tags. *Conference on RFID Security*, 2007.

[21] C. Jaeger, M. Algasinger, U. Rührmair, G. Csaba, and M. Stutzmann. Random p-n-junctions for physical cryptography. *Applied Physics Letters*, 96(172103), 2010.

[22] S.S. Kumar, J. Guajardo, R. Maes, G.J. Schrijen, and P. Tuyls. Extended abstract: The butterfly PUF protecting IP on every FPGA. *HOST 2008*.

[23] Raghavan Kumar, Wayne Burleson: *Litho-aware and low power design of a secure current-based physically unclonable function.* IEEE International Symposium on Low Power Electronics and Design (ISLPED), 2013.

[24] J.W. Lee, D. Lim, B. Gassend, G.E. Suh, M. Van Dijk, and S. Devadas. A technique to build a secret key in integrated circuits for identification and authentication applications. *IEEE VLSI Circuits Symposium*, 2004.

[25] D. Lim. *Extracting Secret Keys from Integrated Circuits*. Msc thesis, MIT, 2004.

[26] D. Lim, J.W. Lee, B. Gassend, G.E. Suh, M. Van Dijk, and S. Devadas. Extracting secret keys from integrated circuits. *IEEE Transactions on Very Large Scale Integration Systems*, 13(10):1200, 2005.

[27] M. Majzoobi, F. Koushanfar, and M. Potkonjak. Testing techniques for hardware security. In *International Test Conference (ITC)*, 2008.

[28] M. Majzoobi, F. Koushanfar, and M. Potkonjak. Lightweight secure pufs. *IEEE/ACM Int. Conf. on Computer-Aided Design*, 2008.

[29] M. Majzoobi, M. Rostami, F. Koushanfar, D.S. Wallach, and S. Devadas: Slender PUF Protocol: A Lightweight, Robust, and Secure Authentication by Substring Matching. *IEEE S&P Workshops*, 2012.

[30] Erdinç Öztürk, Ghaith Hammouri, and Berk Sunar. Towards robust low cost authentication for pervasive devices. *IEEE PerCom*, 2008.

[31] R. Pappu. *Physical One-Way Functions*. Phd thesis, MIT, 2001.

[32] R. Pappu, B. Recht, J. Taylor, and N. Gershenfeld. Physical one-way functions. *Science*, 297(5589):2026, 2002.

[33] M. Riedmiller and H. Braun. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. *IEEE international conference on neural networks*, 1993.

[34] U. Rührmair. Oblivious transfer based on physical unclonable functions (extended abstract). *TRUST 2010.* LNCS Vol. 6101, Springer, 2010.

[35] U. Rührmair: *PUFs at a Glance*. Design, Automation and Test in Europe (DATE'14), 2014.

[36] U. Rührmair, H. Busch, S. Katzenbeisser: Strong PUFs: Models, Constructions and Security Proofs. In A.-R. Sadeghi, P. Tuyls (Editors): *Towards Hardware Intrinsic Security: Foundation and Practice.* Springer, 2010.

[37] U. Rührmair, S. Devadas, F. Koushanfar: Security based on Physical Unclonability and Disorder. In M. Tehranipoor and C. Wang (Editors): Introduction to Hardware Security and Trust. Springer, 2011.

[38] U. Rührmair, C. Jaeger, M. Bator, M. Stutzmann, P. Lugli, G. Csaba. Applications of high-capacity crossbar memories in cryptography. *IEEE Transactions on Nanotechnology*, 2011.

[39] U. Rührmair, C. Jaeger, C. Hilgers, M. Algasinger, G. Csaba, M. Stutzmann. Security applications of diodes with unique current-voltage characteristics. Financial Cryptography and Data Security (FC), 2010.

[40] U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, J. Schmidhuber. Modeling Attacks on Physical Unclonable Functions. *ACM CCS 2010*.

[41] U. Rührmair, J. Sölter, F. Sehnke. On the Foundations of Physical Unclonable Functions. Cryptology ePrint Archive, 2009:277, 2009.

[42] U. Rührmair, J. Sölter, F. Sehnke, X. Xu, A. Mahmoud, V. Stoyanova, G. Dror, J. Schmidhuber, W. Burleson, S. Devadas: *PUF Modeling Attacks on Simulated and Silicon Data*. IEEE Transactions on Information Forensics and Security (IEEE T-IFS), 2013.

[43] H.P.P. Schwefel. *Evolution and Optimum Seeking: The Sixth Generation*. John Wiley & Sons, Inc. New York, NY, USA, 1993.

[44] J. Sölter. *Cryptanalysis of Electrical PUFs via Machine Learning Algorithms*. MSc thesis, Technische Universität München, 2009.

[45] G.E. Suh, S. Devadas. Physical unclonable functions for device authentication and secret key generation. *DAC 2007*.

[46] P. Tuyls, G. J. Schrijen, B. Škorić, J. van Geloven, N. Verhaegh, R. Wolters. Read-proof hardware from protective coatings. *CHES 2006*.

[47] P. Tuyls, B. Skoric. Strong Authentication with PUFs. *In: Security, Privacy and Trust in Modern Data Management, M. Petkovic, W. Jonker (Eds.)*, Springer, 2007.

[48] X. Xu, W. Burleson: *XXX*. Design, Automation and Test in Europe (DATE'14), 2014.

[49] M.-D. Yu, D. M'Raïhi, R. Sowell, S. Devadas: Lightweight and Secure PUF Key Storage Using Limits of Machine Learning. *CHES 2011*.