Machine learning attacks on 65nm Arbiter PUFs: Accurate modeling poses strict bounds on usability — Source link

Gabriel Hospodar, Roel Maes, Ingrid Verbauwhede

Institutions: Katholieke Universiteit Leuven

Published on: 01 Jan 2012 - International Workshop on Information Forensics and Security

Topics: Physical unclonable function, Arbiter and Hardware security module

Machine Learning Attacks on 65nm Arbiter PUFs: Accurate Modeling poses strict Bounds on Usability

Gabriel Hospodar, Roel Maes, Ingrid Verbauwhede

ESAT/SCD-COSIC and IBBT, KU Leuven, Belgium

{firstname.lastname}@esat.kuleuven.be

Abstract—**Arbiter Physically Unclonable Functions (PUFs) have been proposed as efficient hardware security primitives for generating device-unique authentication responses and cryptographic keys. However, the assumed possibility of modeling their underlying challenge-response behavior causes uncertainty about their actual applicability. In this work, we apply well-known machine learning techniques on challenge-response pairs (CRPs) from 64-stage Arbiter PUFs realized in 65nm CMOS, in order to evaluate the effectiveness of such modeling attacks on a modern silicon implementation. We show that a 90%-accurate model can be built from a training set of merely 500 CRPs, and that 5000 CRPs are sufficient to perfectly model the PUFs. To study the implications of these attacks, there is need for a new methodology to assess the security of PUFs suffering from modeling. We propose such a methodology and apply it to our machine learning results, yielding strict bounds on the usability of Arbiter PUFs. We conclude that plain 64-stage Arbiter PUFs are not secure for challenge-response authentication, and the number of extractable secret key bits is limited to at most 600.**

I. Introduction

Implementations of classical cryptographic primitives rely heavily on the ability to securely store secret information. Regardless of higher level abstractions and protection mechanisms, the lowest representation of a secret is always of a physical nature. In modern digital implementations, this typically comes down to a binary vector stored in a silicon memory, which hence requires physical security measures. This turns out to be a non-trivial requirement since standard cryptographic techniques cannot be used any longer, and security is often reverted back to obscurity, e.g. by hiding secret data in a complex chip layout or beneath dense metal layers to prevent visual scrutiny. Silicon Physically Unclonable Functions [1], or PUFs, have been proposed as a physically more secure alternative to storing secrets in a digital memory. Instead of binary values, they use random nanoscale structures which occur naturally in silicon devices in order to store secrets. This offers a higher level of security against physical attacks, and additionally has the potential to be more efficient since costly physical protection measures can be avoided.

A structure is said to show PUF-behavior if it efficiently generates a response when challenged, with the challenge-response behavior depending in an unpredictable and unique way on the challenge and on the considered physical instantiation. A number of different integrated circuits (ICs) designs exhibiting PUF-behavior have been proposed. We refer to [2] for an extensive overview. A particularly interesting construction is the so-called Arbiter PUF, which is able to generate an exponential number of response bits based on the random outcome of a race condition on a silicon chip. Lee et al. already showed for their original Arbiter PUF implementation in 0.18μm CMOS that it was vulnerable to modeling attacks [3], which severely reduces the number of unpredictable response bits. In later work, more elaborate modeling attacks on Arbiter PUFs, including anti-modeling countermeasures, were proposed [4], [5]. However, these results only use data obtained from mathematical or circuit simulations, not from actual implementations, and the achieved modeling results should be considered in this perspective. Moreover, neither Lee et al. nor later works on Arbiter PUF modeling formally specify what the implication of their modeling attacks is on the actual security of using Arbiter PUFs.

The main goal and contribution of this work is twofold: *i)* The susceptibility to modeling for simple and 2-XOR Arbiter PUF implementations in 65nm CMOS is evaluated by assessing the model building performance of two different machine learning techniques: Artificial Neural Networks and Support Vector Machines. This evaluation shows how the results from Lee et al. scale for an independent implementation in a modern silicon technology, and moreover provides a physical verification for the simulation-based modeling results. *ii)* The usability of a PUF in the presence of modeling attacks is evaluated for challenge-response authentication and for secret key generation. This results in practical security bounds for these applications. The proposed methodology is generic, but the presented bounds are specifically for our modeling results on the Arbiter PUF implementation.

This paper is organized as follows. Section II presents background information on Arbiter PUFs and machine learning. Section III details the PUF implementation and the modeling experiments and results. A discussion on the implications of PUF modeling for security applications is put forward in Section IV. Finally, Section V concludes the work.

II. Background

A. Arbiter PUFs

Arbiter PUFs [3] are a type of silicon PUFs for which the PUF-behavior is caused by the intrinsic manufacturing variability of the production process of ICs. They are constructed as a concatenation of stages, with each stage passing two

WIFS'2012, December, 2-5, 2012, Tenerife, Spain. 978-1-4244-9080-6/10/$26.00 ©2012 IEEE.

inputs to two outputs, either straight or crossed depending on a challenge bit. The propagation of two signals through an ℓ-stage Arbiter PUF is determined by an ℓ-bit challenge vector. By careful design, the nominal delays of both paths are made identical. However, the effective delays of both paths in a particular implementation are never exactly deterministic, but are subject to random delay mismatch caused by the IC manufacturing variability. As a consequence, one of both paths will propagate a signal slightly faster or slower than the other, depending on the considered physical implementation, and depending on the applied challenge vector. An *arbiter* sitting at the end of both paths determines on which of the outputs a rising edge, applied simultaneously to both inputs, arrives first. The arbiter outputs a one-bit response accordingly.

An Arbiter PUF implementation generates 1-bit responses from ℓ-bit challenges, and is hence able to produce up to 2^ℓ different CRPs. Lee et al. [3] immediately realised that these 2^ℓ different response bits of an Arbiter PUF are not independent, but can be modeled by an additive linear delay model with a limited number of unknown parameters. An adversary can attempt to estimate these parameters for a particular Arbiter PUF from a set of q_{train} known CRPs, e.g. using machine learning techniques. Once an accurate model of the PUF is built, the remaining $2^\ell - q_{\text{train}}$ response bits are not random anymore but can be predicted by the adversary. Lee et al. successfully constructed a model of their 0.18μm Arbiter PUF implementation which achieved prediction success rates up to 97%. More efficient and more accurate model building attacks on Arbiter PUFs, based on different modeling techniques, were subsequently proposed in [4], [5]. However, besides the initial work from Lee et al., all later model building attempts use responses generated by *simulated* Arbiter PUFs, instead of measurements from physical implementations. Although these are valuable contributions for introducing new PUF modeling techniques and hypothesizing powerful attacks, their numerical outcomes were (to the best of our knowledge) never verified on a modern physical implementation. One of the main goals of this work is to provide this physical verification.

Parallel to modeling attacks on Arbiter PUFs, countermeasures were introduced aimed at preventing modeling. Their basic idea is to disturb the linearity of the delay model by adding non-linear elements to the response generation, such as feed-forward challenge bits [3] and exclusive-or (XOR) combinations of responses [6], [7]. Nonetheless, it was demonstrated based on simulations of PUFs [5] that advanced machine learning techniques are still able to generate good models after training with a larger number of CRPs. Besides the simple Arbiter PUF as described above, we will consider the k-XOR Arbiter PUF [5]–[7] consisting of k equally challenged simple Arbiter PUFs, with the response bit of the k-XOR Arbiter PUF being the XOR of the separate k arbiter outputs.

B. Machine Learning (ML)

Machine learning (ML) [8] is concerned with computer algorithms that automatically learn a complex behavior from a limited number of observations, by trying to generalize the underlying interactions from these examples. Since the apparently complex challenge-response behavior of a PUF is the result of an underlying physical system with a limited number of unknowns, appropriate ML techniques could be able to learn this behavior from a relatively small training set of q_{train} known CRPs and use it to make accurate predictions of unknown responses. In this work, the ML techniques of Artificial Neural Networks (ANN) and Support Vector Machines (SVM) are tested for their ability to model physically realized Arbiter PUFs and k-XOR Arbiter PUFs. An advantage of ANN and SVM is their flexibility to learn any model, as opposed to techniques assuming a prior model with unknown parameters, which are more restrictive. In this work, the used ML techniques were heuristically tuned to give good results without introducing unnecessary complexity.

1) Artificial Neural Networks (ANN): ANNs are adaptive systems formed by interconnected computing nodes called neurons, which are typically structured in feedforward layers. A strong motivation to use ANN is given by the Universal Approximation Theorem [9], which in short states that a two-layer feedforward ANN containing a finite number of hidden neurons can approximate any function with arbitrary precision. However, the theorem does not hint at how to tune a network to efficiently reach such an approximation. The simplest form of an ANN consists of a single layer of neurons and is called the single layer perceptron (SLP) [10]. In each neuron, all input vector values are weighed, summed, biased and applied to an activation function to generate an output. In SLP training, a neuron's weights and bias are updated according to a linear feedback function of the prediction error on a known training set. The training process stops when the prediction error reaches a predefined value or a predetermined number of iterations is completed. SLPs are only capable of solving linearly separable classification problems. Multilayer ANNs are required for nonlinear problems. In this work, the multilayer ANNs are trained by the resilient backpropagation (RProp) [11] training algorithm because it offers fast convergence. RProp is an improved version of the SLP training algorithm based on the gradient descent method. The important *tuning parameters* that should be set to create accurate ANN models are: the number of layers and neurons in each layer, the activation function of each neuron and the training algorithm.

2) Support Vector Machines (SVM): SVM is a ML technique able to learn a binary classification pattern from a set of training examples. In the learning phase, known training examples are mapped into a higher dimensional space to relax the classification task. The learning algorithm tries to find a good separating hyperplane allowing to linearly solve classification problems that are not linearly separable in the original input space. The separating hyperplane should have the largest possible distance between input vectors belonging to different classes, and the inputs with minimal distance to the separating hyperplane are called support vectors. The separating hyperplane is constructed with the help of two parallel supporting hyperplanes through the corresponding support vectors. The distance between the supporting hyperplanes is called the margin. The basic idea of building a good SVM is to maximize the margin while minimizing the classification error. These conflicting goals are traded-off by a regularization parameter γ. Trained SVMs rely heavily on the self inner product of the mapping function, called kernel, evaluated respectively on the support vectors and on the challenge to be classified. Three commonly used kernels $K(\cdot,\cdot)$ are: i) Linear: $K(w,z) = z^T w$ (no mapping – solves only linearly separable problems); ii) Radial Basis Function (RBF): $K(w,z) = \exp\left(\frac{-\|w-z\|_2^2}{\sigma^2}\right)$; iii) Multilayer Perceptron (MLP): $K(w,z) = \tanh(\kappa_1 z^T w + \kappa_2)$. The important *tuning parameters* for a good SVM classifier are: γ, and σ^2 (RBF) or (κ_1, κ_2) (MLP).

III. Experiments and Results

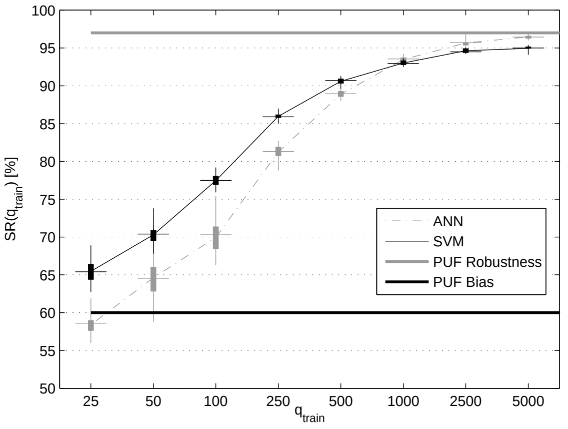
A. PUF Implementation and Experiment Setup

The studied Arbiter PUFs were implemented in silicon using TSMC's 65nm CMOS process. To minimize systematic bias, the layout of the delay line stages and of the arbiter element was fully custom designed. A test board for the ICs provides a convenient digital interface to the PUFs allowing efficient collection of CRPs using a standard PC. A total of 192 ICs were produced, each one implementing 256 simple 64-stage Arbiter PUFs as described in Section II-A. At nominal operating conditions, the measured response bits have a robustness of 97% (ratio of error-free response reconstructions) and, despite the design effort, still exhibit a 60% bias towards zero. The uniqueness, measured as the rate of differing bits generated by the same PUF and challenge but on different ICs, is 48%. The responses of 2-XOR Arbiter PUFs were obtained by XORing the outputs of pairs of simple Arbiter PUFs from the same IC using the same challenges. The resulting 2-XOR Arbiter PUF responses have a robustness around 94%, a bias towards zero around 55% and a uniqueness close to 50%.

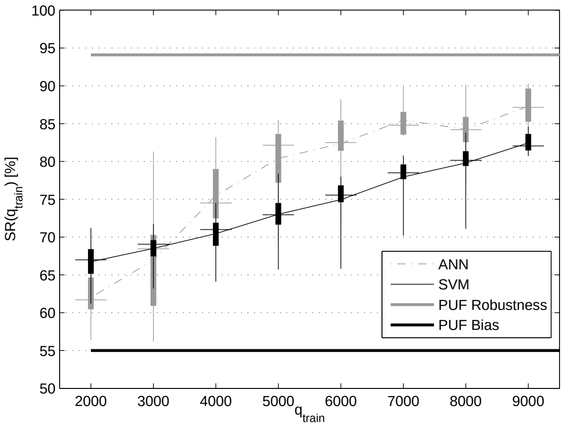
For the machine learning attacks 20 CRP data sets were used, obtained from four different Arbiter PUFs on five distinct ICs, with each data set containing $10{,}000$ randomly selected challenges and 10 independent measurements of every response bit. A model's performance is assessed by its success rate after training with a set of q_{train} CRPs. This success rate $\text{SR}(q_{\text{train}})$ is defined as the ratio of correct response predictions for CRPs from an independent test set. In a realistic attack scenario, q_{train} would be the number of CRPs an adversary needs to obtain, e.g. by eavesdropping on a protocol, in order to build a model of the used PUF. To give an idea of the complexity: all models were trained in less than one minute on a standard machine (dual core @ 3 GHz, 4 GB of RAM).

B. Modeling Attacks and Results

The performance of ML attacks on the simple Arbiter PUF was evaluated for training set sizes ranging from $q_{\text{train}} = 25$ up to $5{,}000$. In order to obtain a meaningful estimate on the results of an attacker's model, for each value of q_{train} ten independent experiments are performed using different randomly selected training sets of size q_{train} from each of the 20 data sets. Motivated by the fact that the adversary has freedom to scrutinize the training set, and as we verified that the modeling performance depends strongly on the used training CRPs, 100 different models were created for each experiment. Each model was respectively built and validated on subsets formed by random splits of a training set containing 70% and 30% of the training CRPs. The model with the best validation results was selected to evaluate the success rate using a test set of $5{,}000$ previously unseen CRPs.



(a) Simple Arbiter PUF.



(b) 2-XOR Arbiter PUF.

Fig. 1. Box plots of obtained $\text{SR}(q_{\text{train}})$ of our ML attacks.

In the Arbiter PUF additive delay model, e.g. as detailed in [5], [7], the response is shown to be linearly dependent on the *cumulative XORs* of the challenge bits, rather than on the challenge bits directly. Performing this nonlinear operation prior to training the ML algorithms substantially improves their performance: the ANN models use fewer neurons and the SVM models count on fewer support vectors. Consequently, as the models get simpler, fewer training CRPs are required. The results for the ANN and SVM modeling attacks on the simple Arbiter PUF are shown in Fig. 1(a). The used ANNs consist of a single neuron SLP using a threshold comparator as the activation function. The SVM models were based on linear kernels with $\gamma = 0.1$. The graph shows the box plots of $\text{SR}(q_{\text{train}})$ for both techniques over all performed experiments on all 20 data sets. Also shown are the Arbiter PUF's robustness and bias which indicate practical lower and upper bounds for the achievable success rates.

On average, SVM yields more accurate Arbiter PUF models than ANN for $q_{\text{train}} \leq 500$, but ANN outperforms SVM for larger training sets. SVM achieves $\text{SR}(50) \approx 70\%$ and for $q_{\text{train}} = 500$, both SVM and ANN are able to predict responses with an accuracy close to 90%. For $q_{\text{train}} \geq 5{,}000$, ANN is able to perfectly model an Arbiter PUF by achieving success rates arbitrarily close to the PUF's robustness. The decreasing height of the box plots indicates that the estimation of $\text{SR}(q_{\text{train}})$ gets more accurate as q_{train} increases.

Similarly, the modeling performance of ANN and SVM on 2-XOR Arbiter PUFs is evaluated. As their behavior is more complex, more training CRPs are required for effective modeling and we use training set sizes ranging from $q_{\text{train}} = 2{,}000$ up to $9{,}000$ CRPs, and $1{,}000$ CRPs for the test set. Ten models are created for each experiment. The used ANNs consist of two layers with respectively four and one neurons, and respectively using hyperbolic tangent and linear activation functions. The SVM models were based on RBF kernels with $(\gamma = 10, \sigma^2 = 3.16)$ for experiments with $q_{\text{train}} \leq 6{,}000$ and on MLP kernels with $(\gamma = 2.7, \kappa_1 = 0.015, \kappa_2 = -1.2)$ for $q_{\text{train}} > 6{,}000$. Figure 1(b) shows the box plots of the obtained $\text{SR}(q_{\text{train}})$ for all the experiments for the ANN and SVM models. This shows that SVM performs better than ANN when $q_{\text{train}} \leq 3{,}000$, but ANN outperforms SVM for $q_{\text{train}} > 3{,}000$. ANN achieves $\text{SR}(9{,}000) \approx 87\%$ and even up to 90% in certain experiments. Although this is still below the upper bound given by the PUF robustness, the steadily rising trend of both graphs suggests that ANN and SVM *possibly* reach near-perfect models respectively for $q_{\text{train}} \approx 12{,}000$ and $q_{\text{train}} \approx 14{,}000$, if a plateau does not happen before these values. The large spread of the observed $\text{SR}(q_{\text{train}})$ values, as shown by the long box plot whiskers, indicates that in certain cases modeling is considerably harder than in the average experiment. This observation suggests that ad hoc adjustments of the ML tuning parameters can significantly improve the results. For k-XOR Arbiter PUFs with $k > 2$, the considered ML techniques perform considerably worse. ANN achieves $\text{SR}(9{,}000) \approx 75\%$ for a 3-XOR Arbiter PUF. We emphasize that the modeling results can be optimized if q_{train} increases and if more specific models are created, e.g. by fine tuning the parameters for each attack or using other ML techniques.

IV. Discussion

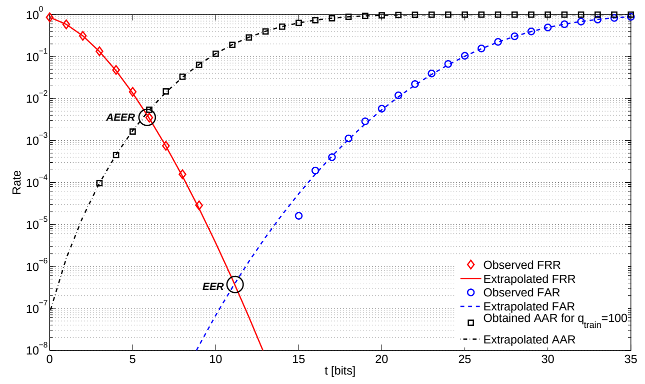
In this section, the implications of model building attacks on the security of PUF-based applications are discussed. The provided results are specifically based on our modeling results of the Arbiter PUFs, but the introduced methodology can be applied to any type of PUF which suffers from model building.

A. *Implications on Challenge-Response Authentication (CRA)*

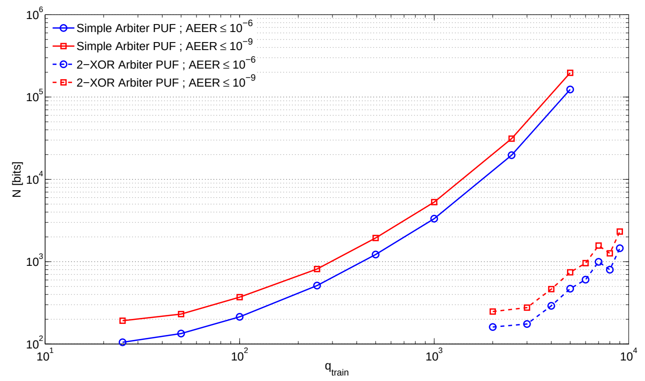
We first consider the implications of modeling attacks on a PUF-based CRA scheme [12]. More efficient and/or practical variants of this scheme have been introduced, but the core idea remains the same: during an enrollment phase, CRPs are collected from every device and stored in a verifier's database; and in the verification phase, a device authenticates itself by proving that it can recreate (almost) the same PUF responses stored by the verifier. The assumed *unclonability* of PUFs ensures that only enrolled devices can be authenticated.

A PUF response is not perfectly reconstructible and a verifier needs to take this into account by allowing a number of errors when matching the regenerated with the stored response bits. This is often done by *forgiving* bit errors, or alternatively by applying some form of error correction on the responses, up to a certain error threshold t. If t is set too low, authentic PUFs that happen to have too many bit errors will be rejected, this is called *false rejection*; while setting t too high will cause non-authentic PUFs to be accepted when their responses happen to be too close to that of an authentic PUF, which is called *false acceptance*. The rates of false rejections (FRR) and false acceptances (FAR) cannot be optimized simultaneously, and setting t is a careful trade-off between security and reliability requirements. In this sense, a good performance indicator of a PUF-based CRA scheme is the point where FAR and FRR are equal and the corresponding failure rate is called the *equal error rate* (EER). In Fig. 2(a), FAR and FRR of 64 response bits obtained from our Arbiter PUF implementations are plotted as a function of t. To determine EER, both plots are extrapolated with a fitted binomial cumulative distribution to find their intersection at $(t_{\text{EER}} = 11, \text{EER} = 5.0 \cdot 10^{-7})$.

In a modeling attack on a PUF-based CRA scheme, an adversary tries to fool the verifier into believing he possesses an enrolled PUF, while in reality he only has a (possibly accurate) model thereof. He might have trained his model using eavesdropped CRPs from previous successful runs of the CRA protocol. We define the *adversary acceptance rate* (AAR) as the probability that an adversary, without direct access to an enrolled PUF, achieves a successful authentication. FAR is a lower bound for AAR, since an adversary can always try to authenticate with an unenrolled PUF. However, in general AAR $>$ FAR, especially if the adversary possesses an accurate model of an enrolled PUF. Figure 2(a) also shows the AAR for a CRA scheme using $N = 64$ simple Arbiter PUF response bits, considering our modeling results as described in Section III-B. For the adversary's PUF model, we considered the ML technique with the best median success rate after being trained with $q_{\text{train}} = 100$ random CRPs. It is clear from this graph that these modeling attacks severely reduce the security of the CRA scheme. If the verifier keeps allowing up to $t_{\text{EER}} = 11$ bit errors then the probability of a successful attack becomes as high as $\text{AAR}(t = 11) = 19.2\%$! Aware of the existence of these attacks, it is wiser to select the point where FRR $=$ AAR as a performance indicator for the CRA scheme. We will call this point the *attack equal error rate* (AEER) and for the considered example it lies at $(t_{\text{AEER}} = 6, \text{AEER} = 5.3 \cdot 10^{-3})$. We note that the actual value of AEER strongly depends on the considered adversary. We have evaluated AEER using the results of our ML attacks as



(a) FRR, FAR and AAR graphs for a simple Arbiter PUF with $N = 64$.



(b) Lower bounds on N to respectively reach AEER $\leq 10^{-6}$ and $\leq 10^{-9}$.

Fig. 2. PUF-based challenge-response authentication (CRA).

reported in Section III-B, but AEER will increase when better modeling attacks are found.

The design parameter of a CRA scheme directly affecting the value of AEER is the number N of used response bits per authentication, with AEER decreasing as N increases. In the discussed example, AEER $= 5.3 \cdot 10^{-3}$ for $N = 64$. However, a reasonable security-reliability trade-off in practice requires that AEER $\leq 10^{-6}$ down to $\leq 10^{-9}$. To obtain these bounds in the considered example, N needs to be increased to 214 or 371 respectively. When q_{train} increases, the adversary's model becomes more accurate and even more response bits are required to obtain practical security levels. Figure 2(b) shows the evolution of the lower bounds on the required number of response bits to achieve AEER $\leq 10^{-6}$ and $\leq 10^{-9}$ respectively for increasing q_{train}. A particularly pessimistic conclusion from this plot is the observation that $N > q_{\text{train}}$ for all considered training set sizes. This implies that a simple Arbiter PUF can be authenticated at most once with a CRA scheme, since an adversary learns more than enough response bits from eavesdropping on one protocol run, to build an accurate model which can impersonate the PUF during subsequent authentications. An *adaptive* adversary, capable of building and evaluating a model during a run of the CRA scheme, might even be able to accurately impersonate an Arbiter PUF during its very first authentication attempt.

Figure 2(b) also shows the same N vs. q_{train} analysis for our ML results on the 2-XOR Arbiter PUF. The conclusion is not as pessimistic as for the simple Arbiter PUF since $N < q_{\text{train}}$ for all training set sizes, though not by a large factor, indicating that the number of possible secure authentications is also strictly limited. Moreover, the plots from Fig. 2(b) are lower bounds on N based on our ML attack results, and any improvement upon our attacks will further increase them.

B. Implications on Secure Key Generation (SKG)

In this second analysis we investigate the effects of modeling on the usability of an Arbiter PUF in a Secure Key Generation (SKG) algorithm. We refer to [13] for an extensive background on how PUF responses can be considered as *fuzzy secrets* from which secure keys can be extracted. In line with [13], the implications of modeling attacks on PUF-based SKG are discussed from an information-theoretical viewpoint.

In the following, we denote a vector of N response bits as a random variable $X^N = (X_1, X_2, \dots, X_N)$ with X_i a single response bit, and a subvector consisting of the first j bits as $X^{(j)} \equiv (X_1, \dots, X_j)$. By p_i we mean the conditional probability of X_i after observing $x^{(i-1)}$: $p_i \equiv \Pr(X_i = 1|x^{(i-1)})$. The operators $H(.)$ and $I(.;.)$ respectively stand for entropy and mutual information, and the binary entropy function is defined as $h(p) \equiv -p \cdot \log_2 p - (1-p) \cdot \log_2(1-p)$.

The premise of considering PUF modeling attacks is the assumption that different response bits generated by the same PUF are not independent. The real conditional probability p_i cannot be learned, but it can be approximated as $\tilde{p}_i \equiv \Pr(X_i = 1|\tilde{x}_i)$, with $\tilde{x}_i$ the response bit predicted by a modeling attack trained on $x^{(i-1)}$. The unknown value of p_i is bounded as $h(p_i) \leq h(\tilde{p}_i)$. Moreover, $\tilde{p}_i$ will be equal to $\text{SR}(i-1)$ or $1 - \text{SR}(i-1)$ depending on $\tilde{x}_i$, and in any case $h(\tilde{p}_i) = h(\text{SR}(i-1))$. In the following, we use as values for $\text{SR}(q)$ the linear interpolation of the median success rates from Section III-B of the best ML technique given q.

In earlier work on SKG, the *secrecy capacity* $S(X)$ of a fuzzy secret X is defined as the theoretical maximum number of secure key bits that can be extracted from X [14], and it is shown that $S(X) = I(X;X')$ with X and X' two noisy realisations of the same fuzzy secret. We calculate this mutual information bound of X^N as $I(X^N;X'^N) = H(X^N) - H(X^N|X'^N)$ and consider both terms separately. We expand $H(X^N)$ as $\sum_{i=1}^{N} H(X_i|X^{(i-1)})$, and $H(X_i|X^{(i-1)}) \equiv \sum_{x^{(i-1)}} \Pr(x^{(i-1)}) \cdot H(X_i|x^{(i-1)}) = \sum_{x^{(i-1)}} \Pr(x^{(i-1)}) \cdot h(p_i) \leq \sum_{x^{(i-1)}} \Pr(x^{(i-1)}) \cdot h(\text{SR}(i-1)) = h(\text{SR}(i-1))$. From which it follows that: $H(X^N) \leq \sum_{i=1}^{N} h(\text{SR}(i-1))$. To evaluate $H(X^N|X'^N)$, we assume a simple but realistic noise model for the PUF response with bit errors occurring i.i.d. over the different response bits with probability $p_e \equiv \Pr(X_i \neq X'_i)$. This is equivalent to a transmission over a binary symmetric channel and in that case $H(X^N|X'^N) = N \cdot h(p_e)$.

Substituting both results in the secrecy capacity bound leads to $S(X^N) \leq \sum_{i=1}^{N} h(\text{SR}(i-1)) - N \cdot h(p_e)$. An upper bound for the secrecy capacity can thus be calculated using the success rates $\text{SR}(i-1)$ of a PUF response model and the bit

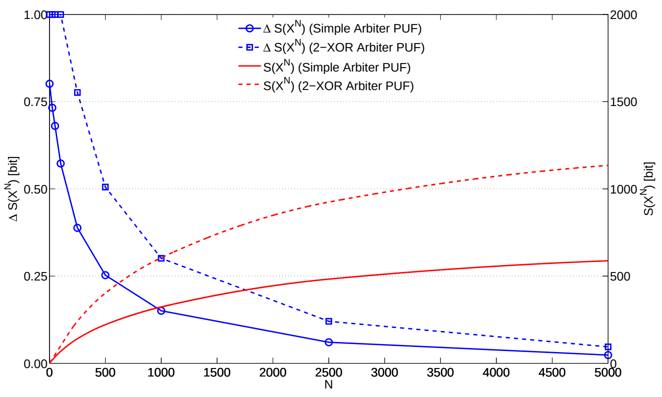


Fig. 3. Upper bounds on the secrecy capacity of Arbiter PUF responses.

error probability p_e estimated from the PUF's statistics. Using our empirical results from Section III, we calculate this bound for increasing N. The result is shown in Fig. 3. Also shown is an upper bound on the *incremental secrecy capacity* $\Delta S(X^N)$, which indicates a bound on how much $S(X^N)$ increases by considering a single additional response bit. It is clear that $\Delta S(X^N)$ decreases steadily as N grows and approaches 0 for $N \geq 5000$. The $S(X^N)$ upper bound of our simple Arbiter PUF implementation reaches 600 bits for $N = 5000$ and will not increase substantially for larger N.

We also performed the $S(X^N)$ vs. N analysis for the 2-XOR Arbiter PUF results, considering the fact that from an information-theoretical viewpoint, $\Delta S(X^N)$ of a 2-XOR Arbiter PUF response bit can never be larger than $2 \times \Delta S(X^N)$ of a simple Arbiter PUF as calculated earlier. Moreover, $\Delta S(X^N)$ of a single PUF response bit can never be larger than 1. The results are also shown in Fig. 3.

Again, the SKG results shown in Fig. 3 express rather loose upper bounds on the number of secure key bits which can be generated in practice. First of all, $S(X^N)$ expresses a theoretical maximum, but no efficient algorithms are known to reach this maximum. Secondly, we did not calculate $S(X^N)$ exactly but only an upper bound thereof. Finally, any improvement upon our ML attacks will further decrease these upper bounds.

V. Conclusion

We have demonstrated the susceptibility of an actual 65nm CMOS Arbiter PUF implementation to modeling attacks based on machine learning. Summarizing, even after training a model with merely a couple of dozen CRPs, it can predict responses from simple Arbiter PUFs with a success rate significantly better than random guessing. After 1,000 training CRPs the prediction accuracy is already $> 90\%$, and after 5,000 training CRPs the prediction is perfect up to the robustness of the PUF. For the 2-XOR Arbiter PUF a prediction accuracy close to 90% is achieved after training with 9,000 CRPs.

Additionally, we have proposed a methodology for assessing the implications of modeling attacks on PUF-based security applications and applied it to our modeling results on Arbiter PUFs. We conclude that simple Arbiter PUFs cannot be securely used for PUF-based challenge-response authentication and the applicability of 2-XOR Arbiter PUFs is also limited. For PUF-based secure key generation, we find that the number of information-theoretically secure key bits which a simple Arbiter PUF can generate is at most about 600 and for 2-XOR Arbiter PUFs at most twice that amount. Moreover, the secure key material contributed by each additional CRP decreases rapidly and approaches zero after about 5,000 CRPs.

We stress that these numerical limitations on the usability of Arbiter PUFs, both for authentication as for key generation, are merely bounds. These bounds will become tighter when better modeling attacks are found, or when the PUF's robustness decreases, e.g. as a consequence of varying temperature and supply voltages which we did not consider here. Future work based on our proposed methodology will reveal tighter bounds on the actual applicability of Arbiter PUFs.

Acknowledgment

This work was supported by the Research Council K.U.Leuven: GOA TENSE (GOA/11/007), by the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy), by the European Commission through the ICT programme under contract ICT-2007-216676 ECRYPT II, by the Flemish Government through FWO G.0550.12N and the Hercules Foundation AKUL/11/19, by the European Commission through the ICT programme under contract FP7-ICT-2011-284833 PUFFIN and HINT, and FP7-ICT-2007-238811 UNIQUE. Roel Maes is funded by IWT-Flanders grant no. 71369.

References

[1] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, "Silicon physical random functions," in *ACM CCS*, 2002, pp. 148–160.
[2] R. Maes and I. Verbauwhede, "Physically Unclonable Functions: a Study on the State of the Art and Future Research Directions," in *Towards Hardware-Intrinsic Security*, D. Naccache and A.-R. Sadeghi, Eds. Springer, 2010, pp. 3–37.
[3] 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 application," in *Symposium on VLSI Circuits*, 2004, pp. 176–159.
[4] E. Ozturk, G. Hammouri, and B. Sunar, "Physical unclonable function with tristate buffers," 2008, pp. 3194–3197.
[5] U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, and J. Schmidhuber, "Modeling attacks on physical unclonable functions," in *ACM CCS*, 2010, pp. 237–249.
[6] G. E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *DAC*, 2007, pp. 9–14.
[7] M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Lightweight secure PUFs," in *ICCAD*, 2008, pp. 670–673.
[8] T. M. Mitchell, *Machine Learning*. USA: McGraw-Hill, 1997.
[9] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Networks*, vol. 4, no. 2, pp. 251–257, 1991.
[10] F. Rosenblatt, *The perceptron – a perceiving and recognizing automaton*, ser. Cornell Aeronautical Laboratory report, 1957.
[11] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: The RProp algorithm," in *IEEE Intl. Conference on Neural Networks*, 1993, pp. 586–591.
[12] R. S. Pappu, B. Recht, J. Taylor, and N. Gershenfeld, "Physical one-way functions," *Science*, vol. 297, pp. 2026–2030, 2002.
[13] T. Kevenaar, P. Tuyls, and B. Skoric, Eds., *Security with noisy data*. Springer, 2007.
[14] U. Maurer, "Secret key agreement by public discussion from common information," *IEEE Transactions on Information Theory*, vol. 39, no. 3, pp. 733–742, 1993.