2018 IEEE Asia Pacific Conference on Circuits and Systems

# Attacking Arbiter PUFs Using Various Modeling Attack Algorithms: A Comparative Study

Yue Fang*, Chenghua Wang*, Qingqing Ma*, Chongyan Gu†, Maire O'Neill†, Weiqiang Liu*
*CEIE, Nanjing University of Aeronautics and Astronautics, Nanjing, China, 211106
†CSIT, ECIT, Queen's University Belfast, Belfast, UK, BT3 9DT
Email: {yuef, chwang, martin, liuweiqiang}@nuaa.edu.cn, cgu01@qub.ac.uk, m.oneill@ecit.qub.ac.uk

***Abstract*—In this paper, we investigate the effectiveness of four different modeling attack algorithms, including Logistic Regression (LR), Naïve Bayes, AdaBoost and Covariance Matrix Adaptation Evolutionary Strategies (CMA-ES), on attacking arbiter physical unclonable functions (APUFs). A comparison of experimental results using theses algorithms is presented. The results show that the performance of the algorithms is related to the number of training data, the noise level involved in the APUF design and the number of stages in the generation of each bit response. It is found that the mainstream LR and CMA-ES are worse for a small number of data compared with Naïve Bayes and AdaBoost.**

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

## I. Introduction

Physical unclonable function (PUF) is a promising lightweight security primitive for applications of the internet of things (IoT), which extracts random differences in integrated circuits (ICs) and produces a unique response. To a certain extent, PUF combines the features of biometric-based identity authentication and hardware-based identity authentication. As a new security hardware primitive, PUF is characterized by unpredictability, low cost, and unclonable capability. Currently, PUF has developed several architectures. Depending on the number of challenge response pairs (CRPs), PUFs can be divided into strong PUFs and weak PUFs, which can be applied to low-cost authentication [1] and security key generation [2], respectively. The security of PUF has been one of the main focuses of PUF research. The larger the number of CRPs, the easier the attacker to break a strong PUF. Additionally, most strong PUFs are based on a linear function architecture, which means that it is possible to be attacked with a large number of CRPs. APUF [3] is one of the most widely studied PUF designs. APUF can be successfully attacked by several machine learning algorithms, such as Logistic Regression (LR) and Covariance Matrix Adaptation Evolutionary Strategies (CMA-ES) [4]. With the development of machine learning based techniques, it is interesting to investigate the vulnerability of the PUF designs using advanced machine learning techniques. In this paper, a variety of classical modeling attack algorithms including LR, Naïve Bayes, AdaBoost and CMA-ES is used. Naïve Bayes is a simpler algorithm which is less sensitive to missing data. AdaBoost can combine weak classifiers into strong classifiers ,which has higher prediction rates than single weak classifiers. We perform attacks on APUF with the four algorithms in a variety of cases and found that although the overall performance of CMA-ES and LR are generally excellent, for the case of a small data set, the prediction rate of LR as well as CMA-ES is not as good as Naïve Bayes and AdaBoost. The experiments show that Naïve Bayes and AdaBoost are more effective for small data sets.

The rest of the paper is organized as follows. Section II introduces the classical algorithms in this work. In Section III, the model of a 1-bit APUF design and the result of average prediction rate for APUF using several algorithms under different environment are presented. Finally, a conclusion is draw in Section IV.

## II. Classical Modeling Attack Algorithms

### A. Logistic Regression (LR)

LR is a common machine learning method for PUF attacks [5]. It is a linear classification model based on the maximum likelihood. For a traditional APUF with n stages, challenge $C = c_1c_2\ldots c_n$ corresponds to response $R \in \{0,1\}$. The final decision boundary is decided by the sigmoid function as follows:

$$h_\theta(x) = \sigma(x) = (1+e^{-x})^{-1} \tag{1}$$

where $\theta$ indicates weight of the sample.

For a given training set T of an APUF, one of the samples can be represented as $(x_i, y_i)$. The probability of each sample $(x_i, y_i)$ is represented as follows:

$$P(y_i|x_i) = \prod(P(y_i=1|x_i)^{y_i^{(i)}}(1-P(y_i=1|x_i)^{1-y_i^{(i)}}) \tag{2}$$

When the tag value is "1", the expression represents the probability that $P(y=1,x_i)$; on the other hand, the formula expresses the probability of $P(y=0,x_i)$ when the tag value is "0".

The logarithmic likelihood function can be expressed as

$$l(\theta) = \sum_{i=1}^{m}\sum_{i=1}^{m} y^i log h_\theta(x^{(i)}) + (1-y^{(i)})log(1-h_\theta(x^{(i)})) \tag{3}$$

As the equation is difficult to solve directly, it is usually solved by iterative gradient descent method

$$\theta(x) := \theta - \alpha\nabla_\theta(\theta) \tag{4}$$

978-1-5386-8240-1/18/$31.00 ©2018 IEEE

where $\alpha$ is called learning rate (step size), which determines the rate of gradient descent.

In general, LR is a probabilistic linear regression model. The dependent variable can be two-class or multi-classified.

### B. Naïve Bayes

Naïve Bayes method is a classification method based on Bayesian theorem and feature condition independent hypothesis. The Naïve Bayes Classifier (NBC) originates from classical mathematical theory, which has a stable classification efficiency [7].

The relationship between prior probability and posterior probability can be expressed as:

$$P(y|x_1,\ldots,x_n) = \frac{P(y)P(x_1,\ldots,x_n|y)}{P(x_1,\ldots,x_n)} \tag{5}$$

where $x$ indicates the feature while $y$ indicates the label. $P(y)$ represents the prior probability that can be obtained from the frequency of labels in the training set. The probability can be obtained according to the frequency of the response in the PUF training set.

Conditional independence hypothesis means independence between every pair of features.

$$P(y|x_1,\ldots,x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i|y)}{P(x_1,\ldots,x_n)} \tag{6}$$

Naïve Bayes is one of the classic machine learning algorithms based on probability theory.

### C. AdaBoost

AdaBoost [8] is an iterative algorithm and its core idea is to train different classifiers (weak classifiers) for the same training set and then group these weak classifiers to form a stronger final classifier (strong classifier). The block diagram of AdaBoost is shown in Fig. 1.

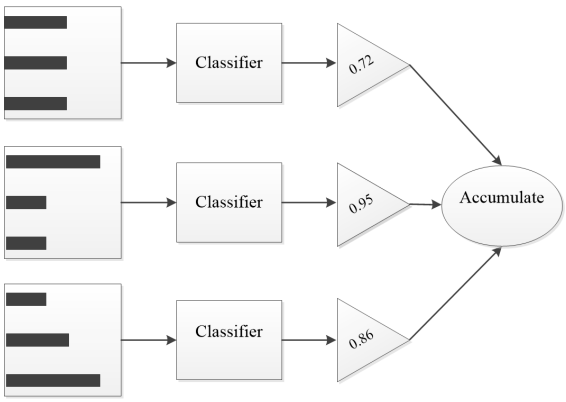


Fig. 1. AdaBoost algorithm structure.

Initially, the weight distribution of the training data is initialized according to number N, and each training sample is initially given the same weight. In this way, the initial weight distribution is as follows

$$D_1(i) = (\omega_1, \omega_2, \ldots, \omega_N) = (\frac{1}{N}, \ldots, \frac{1}{N}) \tag{7}$$

Select a weak classifier $h$ with the lowest error rate as the number $t$ basic classifier, and calculate the error of the weak classifier on the distribution. The error rate is as follows:

$$e_t = P(H_t(x_i) \neq y_i) = \sum_{i=1}^{n} \omega_{ti} I(H_t)(x_i) \neq y_i) \tag{8}$$

Calculate the weight of the classifier in the final classifier (weak classifier weight is denoted by $\alpha$)

$$\alpha_t = \frac{1}{2} ln(\frac{1-e_t}{e_t}) \tag{9}$$

Finally, combine the weak classifiers by weak classifier weights. Through the role of the sign function, the strong classifier can be expressed as follows

$$H = sign(\sum_{y=1}^{T} \alpha_t H_t(x)) \tag{10}$$

The AdaBoost algorithm is a modified boosting algorithm, which can adaptively adjust the errors of weak classifiers.

### D. CMA-ES

CMS-ES is one of the most famous ESs, with good results on medium-sized complex optimization problems [9]. CMA-ES does not use gradient information and performs well on complex optimization problems, while gradientless algorithms in local search are usually slower than gradient algorithms. The core idea of CMA-ES is to handle the dependencies between variables by adjusting the covariance matrix in the normal distribution.

## III. Moldeling Attacks on APUF

### A. Model of a 1-bit APUF Design

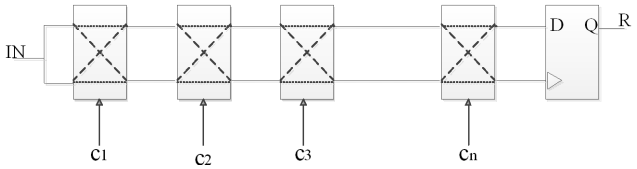


Fig. 2. The structure of APUF [2].

The architecture of a 1-bit APUF is shown in Fig. 2. For an n-bit APUF, an additive linear delay model can be described as [6]. The 1-bit response R, is decided by the final delay difference between the two delay paths, which can be expressed as

$$\Delta = \vec{\omega}^T \vec{\varphi} \tag{11}$$

where the dimension of $\vec{\omega}$ and $\vec{\varphi}$ is $n+1$. The parameter $\vec{\omega}$ represents the delay for the subcomponents in the APUF stages as shown in (12), while the feature vectors $\vec{\varphi}$ as shown in (13) shows the multiply results related to challenge $C$.

$\delta_i^{0/1}$represents the delay in the stage $i$ which includes a crossed path or an uncrossed path.

$$\vec{\omega} = (\omega^1, \omega^2, \ldots, \omega^k, \ldots, \omega^{n+1})^T \tag{12}$$

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

$$\vec{\varphi}(\vec{C}) = (\vec{\varphi}^1(\vec{C}), \ldots, \vec{\varphi}^k(\vec{C}), 1)^T \tag{13}$$

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

According to the difference $\Delta$, we can express output $r$ of the A PUF by the sign function as:

$$r = \theta(\Delta) = \theta(\vec{\omega}^T \vec{\varphi}) \tag{14}$$

where $\theta(x)$ is called Heaviside step function and decide the final output, i.e. $\theta(x) = 0 \quad if \quad x < 0 \quad and \quad \theta(x) = 1 \quad if \quad x \geq 1$.

### *B. Moldeling Attacks on APUF*

The LR and CMA-ES attack results on APUF are shown in [4]. In addition, Naïve Bayes and AdaBoost are presented in this paper. In this experiment, we use Python (version 3.6) simulation to implement APUF sample generation and various machine learning methods. In order to obtain accurate results, the experimental results in this work take the average of 100 repeated samples.

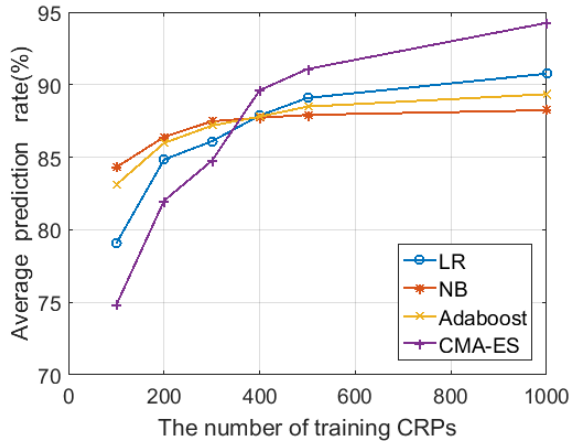


Fig. 3. The average prediction rates for 64-stage APUF.

The results of the above modeling attack algorithms for APUF are shown in Fig. 3. To predict the APUF design using different modeling attack algorithms, a group of tests on different numbers of training samples is performed. To examine the performance of the four algorithms when the number of CRPs is small, the prediction rates is token for 64-stage APUF with the numbers of training sample sets of 100, 200, 300 ,400, 500 and 1000, respectively. The number of test samples is set as the same as the training samples.

For the case of a 64-stage APUF, a small number of data ( $CRPs < 400$ ) is tested, and the average prediction rate of Naïve Bayes is 84.30% and AdaBoost is 83.10%, which are higher compared with LR (79.05%) and CMA-ES (74.8%).

Comparing theses different methods, it can be seen that LR and CMA-ES are suitable for large data sets, especaially CMA-ES has the best performance under large data sets. However, Naïve Bayes and AdaBoost can be applied to smaller data sets. And Naïve Bayes has the best performance for small number of data. Suppose the attacker conducts a model attack on the PUF with limited number of CRPs. For example, only CPRs with less than 300 can be obtained. In this case, using Naïve Bayes is a better choise compared with the other algorithms.

At the same time, we compare the training time of different modeling attack algorithms, as shown in Table 1. Note that Naive Bayes, Adaboost and LR are run on PyCharm(Version 3.6) using Python. While, CMA-ES is run on Matlab (Version 2016a). The comparison can only provide a preliminary insight on the speed performance of the four algorithms. From Table 1, we can see that the training time of Naïve Bayes is the shortest, especially when CRPs=100, the training time is only 0.0007s. Meanwhile, the training time of LR is also very short, about several times that of Naive Bayes. AdaBoost has a longer training time, but can still be completed in 1s when CRPs=10000. For CMA-ES, the training time at CRPs=100 exceeds the training time of other methods at CRPs=10000, reaching 16.36s. It is slower when CRPs=10000, and the training time is 1272s. Therefore, Naïve Bayes has a much faster speed than CMA-ES. In the case of limited time, Naïve Bayes can be very efficient.

TABLE I
TRAINING TIME OF MODELING ATTACKS

| CRPs | Training Time(s) | | | |
|---|---|---|---|---|
| | ***Naïve Bayes*** | ***Adaboost*** | ***LR*** | ***CMA-ES*** |
| 100 | 0.0007 | 0.0753 | 0.0017 | 16.36 |
| 500 | 0.0011 | 0.1167 | 0.0024 | 76.19 |
| 2000 | 0.0025 | 0.1942 | 0.0082 | 256.15 |
| 10000 | 0.0163 | 0.6913 | 0.0698 | 1272 |

In the Fig. 4, various noises are added to the original data in order to simulate the practical APUF under different noise conditions. Gaussian noises with variances of $\sigma = 0$, $\sigma = 0.25$, $\sigma = 0.5$ and $\sigma = 1$ are utilized. The experimental results show that the average prediction rate of the four attack methods has decreased compared with the noise-free case. As the AdaBoost algorithm is more sensitive to the noise, the prediction rate under the impact of noise is studied in detail. The prediction rate when $\sigma = 1$ and CRPs=100 is reduced by nearly 20% compared with that when $\sigma = 0$. When the CRPs is greater than 10,000, the prediction rate tends to be stable. The prediction rate when $\sigma = 0.25$ and $\sigma = 0.5$ is reduced by 2% and 4%, respectively, compared with the noise-free case. It can be seen that the larger the noise, the lower the average prediction rate.

An investigation of the effect of different stages of the APUF using Naïve Bayes and Adaboost is described in Figs. 5-6. The results of the prediction rates useing Naïve Bayes and AdaBoost, are shown with different numbers of CRPs. As the number of stages increases, the prediction rates of the three methods reduce. The prediction rate of APUF with the number of 256 stages is 2% lower than that of 64 stages when

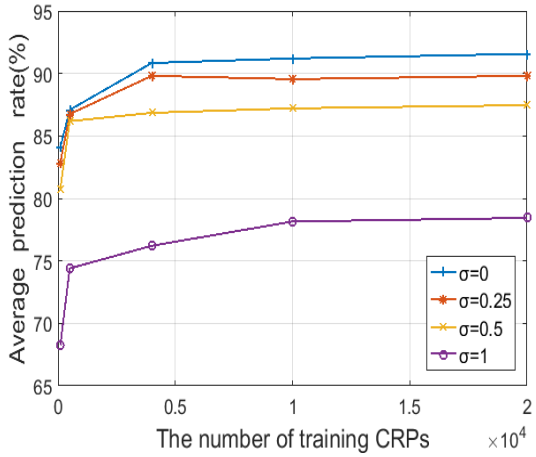


Fig. 4. The average prediction rates for AdaBoost under different noise.

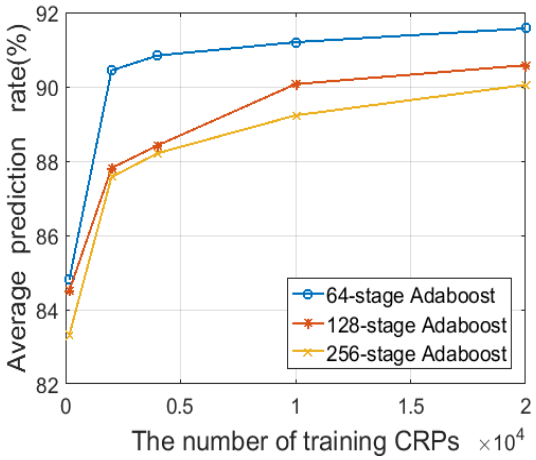


Fig. 6. The average prediction rates for AdaBoost with different stages.

the number of training data is 20,000. The increment of the number of stages has made the attacks more difficult, and the prediction rate has relatively declined. It can be seen that when increasing the number of stages of APUF design, the security of the APUF can be improved.

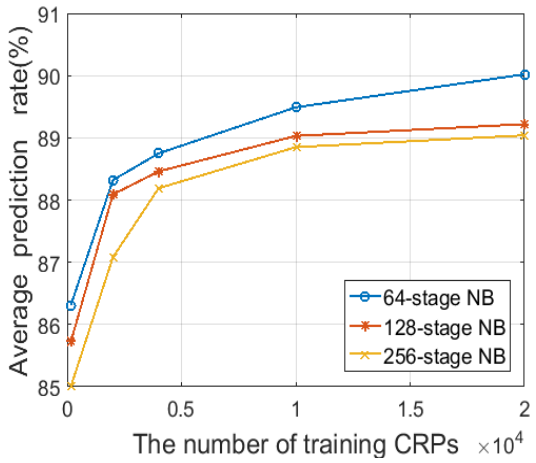


Fig. 5. The average prediction rates for NB with different stages.

## IV. Conclusion

LR and CMA-ES are methods known to be effective in estimating APUF in modeling attacking algorithms. In this paper, we use the machine learning algorithms, including Naïve Bayes and AdaBoost to attack APUF and compare the results with LR and CMA-ES. The average prediction rates of various algorithms under different numbers of CRPs are compared. Moreover, the average prediction rate of the AdaBoost algorithm under different noise conditions is presented. The higher the noise level, the more difficult the APUF to be attacked. The effects of the number of stages on the prediction rates of three algorithms are also demonstrated. The CMA-ES outperforms other methods in general and next is LR. However, when the number of CRPs is small, it is not as good as other methods. In addition, for the 64-stage APUF, the average prediction rate of Naïve Bayes and AdaBoost reached 84.3% and 83.1%, respectively, while the prediction rate of LR and CMA-ES were 79.05% and 74.8%. When the number of training data becomes larger, the prediction rate of CMA-ES is higher than Naïve Bayes and AdaBoost. For the AdaBoost algorithm, it performs well on data sets of various sizes. Naïve Bayes and AdaBoost achieves higher prediction rates than LR and CMA-ES for a small number of training data.

## Acknowledgment

This work has been supported by grants from National Natural Science Foundation of China (61771239), Natural Science Foundatoin of Jiangsu Province (BK20151477) and Six Talent Peaks Project of Jiangsu Province (XYDXX-009).

## References

[1] P. S. Ravikanth, "Physical one-way functions," Science, vol. 297, no. 5589, pp. 2026-2030, 2002.
[2] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, "Silicon physical random functions," in Proc. 9th ACM Conference on Computer and Communications Security (CCS), pp. 148–160, 2002.
[3] W. Liu, L. Zhang, Z. Zhang, C. Gu, C. Wang, M. O'Neill, and F. Lombardi, "XOR-based low-cost Reconfigurable PUFs for IoT Security," ACM Transactions on Embedded Computing Systems. DOI: 10.1145/3274666.
[4] Q. Ma, C. Gu, C. Wang, W. Liu and M. O'Neill, "A machine learning attack resistant multi-PUF design on FPGA," Proc. 23rd Asia and South Pacific Design Automation Conference (ASP-DAC), pp. 97-104, 2018.
[5] U. Rührmair, J. Sölter, F. Sehnke, X. Xu, A. Mahmoud, V. Stoyanova, G. Dror, J. Schmidhuber, W. Burleson, and S. Devadas, "PUF modeling attacks on simulated and silicon data," IEEE Transactions on Information Forensics & Security, pp. 1876-1891, 2013.
[6] D. Lim. "Extracting Secret Keys from Integrated Circuits," Msc Thesis, MIT, 2004.
[7] L. Jiang, H. Zhang, and Z. Cai, "A novel Bayes model: Hidden naive Bayes," IEEE Transactions on Knowledge and Data Engineering, vol. 21, no. 10, pp. 1361-1371, 2009.
[8] T. Hastie, S. Rosset, J. Zhu, and H. Zou, "Multi-class adaboost," Statistics and its Interface, 2009, pp. 349-360.
[9] N. Hansen, S.D. Ller , P. Koumoutsakos, "Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES)," Evolutionary Computation, vol. 11, no. 1, pp. 1-18, 2014.