Estimating Delay Differences of Arbiter PUFs Using Silicon Data

S. V. Sandeep Avvaru, Chen Zhou, Saroj Satapathy, Yingjie Lao, Chris H. Kim, Keshab K. Parhi
Department of Electrical and Computer Engineering
University of Minnesota
Minneapolis, MN 55455 USA

***Abstract*—This paper presents a novel approach to estimate delay differences of each stage in a standard MUX-based physical unclonable function (PUF). Test data collected from PUFs fabricated using 32 nm process are used to train a linear model. The delay differences of the stages directly correspond to the model parameters. These parameters are trained by using a *least mean square* (LMS) adaptive algorithm. The accuracy of the response using the proposed model is around 97.5% and 99.5% for two different PUFs. Second, the PUF is also modeled by a perceptron. The perceptron has almost 100% classification accuracy. A comparison shows that the perceptron model parameters are *scaled* versions of the model derived by the LMS algorithm. Thus, the delay differences can be estimated from the perceptron model where the scaling factor is computed by comparing the models of the LMS algorithm and the perceptron. Because the delay differences are challenge independent, these parameters can be stored on the server. This will enable the server to issue random challenges whose responses need not be stored. An analysis of the proposed model confirms that the delay differences of all stages of the PUFs on the same chip belong to the same Gaussian probability density function.**

I. Introduction

Physical unclonable functions (PUFs) consist of small circuits that can be used for hardware authentication and identification of individual integrated circuit (IC) chips. Several types of PUFs exist; some examples include multiplexer PUFs, ring oscillator PUFs, and SRAM based PUFs. The process variations lead to unique characteristics of these circuits and these unique features are extracted and used as signature of the chip. In a multiplexer (MUX) PUF, the delay difference between the two possible paths of a MUX stage is different for each stage in a chip. These variations are primarily due to layout mismatch and process variations.

This paper, for the first time, presents a novel approach to estimate the delay difference of each MUX stage in a PUF using measured data from PUF chips fabricated in 32 nm process. A total of six PUF chips were fabricated and 96 MUX PUF circuits were implemented on each chip. The layout was identical for each PUF circuit. The challenge response test data are used to model the delay differences of each MUX stage using a simple *least mean square (LMS)* adaptive filtering algorithm. It is shown that the delay differences indeed form a Gaussian distribution. It is also shown that the delay differences of the various stages of the PUF circuits in each chip and among all chips are unique. These delay differences have not been estimated in any prior analysis of MUX PUFs. The fact that the delay differences belong to the same Gaussian PDF has also not been confirmed from test data before. An advantage of the proposed modeling approach is its simplicity with respect to computation time.

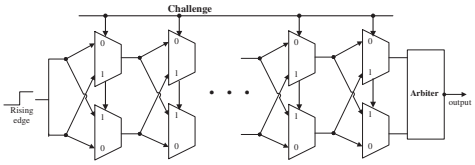


Fig. 1: An arbiter based PUF structure

The rest of the paper is organized as follows. In Section II, we present a review of arbiter based PUFs, attacks on the PUFs and a brief introduction to the LMS algorithm. Section III details the steps involved in modeling an arbiter PUF based on the LMS algorithm. In Section IV, we describe the experimental setup and present the results. In Section V, we analyze the results and discuss their implications. Conclusions are drawn in Section VI.

II. Background

A. MUX Based Arbiter PUF

Silicon PUFs were first introduced in [1], and are based on the idea of exploiting statistical variations inherent in manufacturing processes. An arbiter PUF is a *delay based PUF* and an example of a Silicon PUF. The input to an arbiter PUF is called a *challenge*, which is a binary vector whose length is typically same as the number of stages in the PUF. The output is called a *response*, which depends on the path delays of the circuit. As discussed in [2], there are sufficient manufacturing process variations across different implementations in order to uniquely identify each chip. A randomly chosen set of challenges and their corresponding responses are called *challenge-response pairs (CRPs)*. An example MUX PUF is illustrated in Fig. 1. In an N-stage PUF, a rising edge applied to the PUF races through a possible 2^N paths that depend on the N-bit challenge and reaches the arbiter. At each stage, there is a multiplexer which acts as a switch controlled by the corresponding challenge bit, thereby modifying the path. The arbiter block at the end determines the response based on which rising edge arrives first. Accordingly, the response is either a '0' or a '1'.

978-3-9815370-6-2/DATE16/

B. Attacks on Arbiter PUFs

As discussed in [3], possible attacking strategies on PUFs can be classified into 3 kinds: *Prediction attacks*, *Reverse engineering attacks* and *Collision attacks*. A Reverse engineering attack attempts to learn the behavior of a PUF by studying the input-output relation between several challenge response pairs. The knowledge of the PUF architecture and the amount of CRPs available usually have a significant effect on the feasibility of an attack. Several reverse engineering attack strategies have been proposed in the past. Lee *et al* [2] have demonstrated the vulnerability of arbiter PUFs to modeling attacks and executed an attack strategy using a machine learning technique. Several other modeling attacks have been proposed subsequently, but most of them utilize CRPs generated by simulated arbiter PUFs instead of silicon data. In [4] and [5], responses from physically realized integrated circuit chips have been used to model attacks using machine learning algorithms. The results in [5] were based on Logistic Regression. In [4], the authors investigate artificial neural network (ANN) and support vector machine (SVM) classifiers.

C. Least Mean Square (LMS) Algorithm

Gradient descent or the *method of steepest descent* [6] is an adaptive optimization algorithm used to minimize (or maximize) a given function $\mathbf{J}(n)$, referred as the *cost function*. Starting from an arbitrary *tap-weight vector* ($\mathbf{w}$), the solution improves iteratively. In each iteration, the weight vector is adjusted in the direction of the steepest descent or the direction opposite to the gradient of the cost function. Ultimately, under appropriate conditions, the solution converges to the *Wiener solution*.

D. Artificial Neural Networks

Machine learning techniques are computer algorithms that are used to compute a model by *learning* complex input-output relations from available information. These models are then used to predict outputs for other inputs. We need to provide sufficiently large amount of data to a machine learning algorithm, in order to learn an input-output mapping. The data in our case are the available CRPs. The machine learning technique we implement in this paper is *Artificial Neural Networks*. The simplest version of an ANN is called a *single layer perceptron*. A perceptron, the basic processing element in any ANN, defines a hyperplane which can be used to divide the input space into two parts [7]. For instance, this can be applied as such to train a binary classifier. In this work, a perceptron structure with a hard-limit transfer function was implemented.

III. LMS ALGORITHM BASED MODELING

As our aim is to predict the response bit of an arbiter PUF from a challenge vector, inputs to our models are the challenge vectors and outputs are the corresponding responses. In the proposed model, each challenge or response bit that is 1 or 0 is mapped to -1 or 1, respectively. As described in the previous section, the LMS algorithm is essentially a tool to estimate the parameters (called weights) which optimally express a given input-output relation by minimizing a so called cost function. In other words, our objective is to minimize the mean squared error between the predicted response and the desired response.

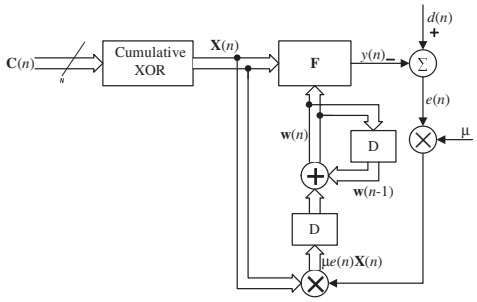


Fig. 2: Block diagram of the LMS algorithm

A. Additive linear delay model

It has been established that an arbiter based PUF circuit can be expressed as a linear model [8] [9]. According to additive linear delay model, the delay difference at the last stage of the PUF can be modeled as the sum of individual delay differences at each stage. It is important to note that the sign (or direction) of the delays at each stage is modified by the corresponding challenge bits. If Δ_i denotes the delay difference at the *i*-th stage and C_i denotes the challenge bit at the *i*-th stage of an arbiter PUF with N stages, the overall delay difference Δ can be computed as

$$\Delta = \sum_{i=1}^{N} \Delta_i X_i \tag{1}$$

where each X_i is computed as a cumulative XOR of the successive challenge bits, i.e., $X_i = C_{i+1} \oplus C_{i+2} ... \oplus C_N$ and $X_N = C_N$, and Δ_i represents the delay difference in stage-i.

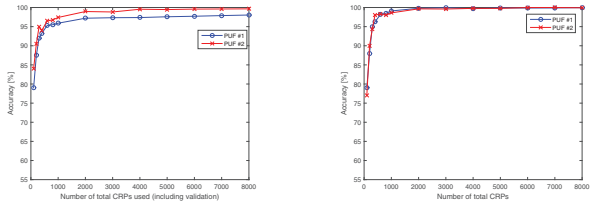
B. Algorithm

An arbiter PUF is a delay based model and can be parameterized by additive linear delay model. These parameters, i.e., the delay differences at each stage or the values of Δ_i's, are considered as the weights of an adaptive filter. Fig. 2 shows the block diagram which illustrates the sequence of steps involved in estimating these weights based on the LMS algorithm from a given set of CRPs. The input vector is concatenated with a '1' to account for the bias. So, the tap-weight vector is of size $N+1$ if the MUX has N stages. The filter output $y(n)$ is then computed and compared to the desired response bit $d(n)$. The difference $e(n)$ is then used to update the weights.

IV. EXPERIMENTAL SETUP AND RESULTS

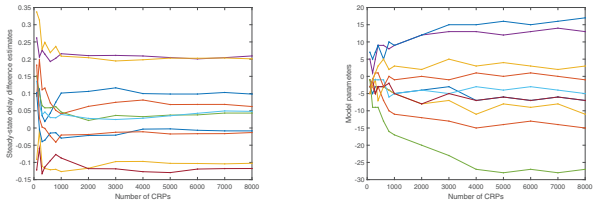
A. PUF Specifications and Data

A total of six chips with 96 MUX PUF circuits with 32 stages per chip were fabricated on IBM 32 nm HKMG process. The MUX PUFs were divided into two types: linear MUX



(a) LMS (b) Perceptron

Fig. 3: Prediction accuracy of a model based on LMS and perceptron plotted against total number of CRPs.



(a) LMS (b) Perceptron

Fig. 4: Estimated model parameters plotted against number of CRPs used for modeling PUF 1 from chip 1

PUFs and feed-forward MUX PUFs. This paper presents the results of modeling and analysis of the linear MUX PUF based on data measured from two chips. 10000 challenges were randomly generated and the circuits were tested multiple times with each challenge. The source voltage was 0.9 volts. All the measurements collected for this experiment were recorded at an ambient temperature of 25 degrees Celsius.

B. Modeling results

As we have a 32 stage PUF, each challenge is a 32-bit vector. So, 33 tap-weights are defined, including a weight for the bias term, and are initialized to random values between 0 and 1. The number of CRPs used for training the model is varied in order to observe its effect on model accuracy and model parameters. In order to obtain reliable estimates, a five-fold cross validation scheme is used to validate the trained models. This means that 80% of the available data is used to train the model and the remaining 20% is used to validate it, and this is repeated for five times, until all the data are tested. The response of the model is decided to be either '0' or '1' based on whether the output is positive or negative. These model responses are compared with the true responses to compute *classification accuracies*. The average accuracy across the five folds is considered as accuracy of the model.

This paper considers modeling of the MUX PUF using ANN and the LMS algorithm. The results of the two modeling approaches are compared. The LMS algorithm applies the gradient descent with adaptive learning rate to train the network. Mean square error is used as the cost function. The data are divided into 5 folds out of which, 3 folds are used for training while one fold is used for validation and the remaining fold is used to test the accuracy of the model. The validation fold is required in order to test the convergence of the network and decide when to stop training. These are implemented using MATLAB neural network toolbox.

Fig. 3(a) shows how the accuracy of the models based on the LMS varies with the number of CRPs available. Fig. 3(b) shows the variation of testing accuracy of models trained using single layer perceptron, as a function of number of CRPs. Each figure has two plots corresponding to the data from PUFs on two different chips.

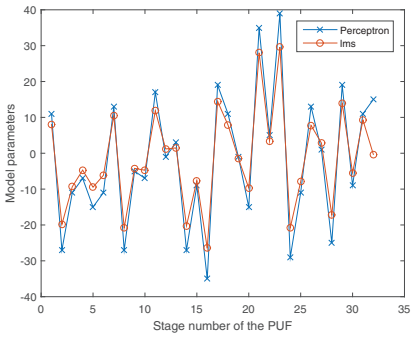


Fig. 5: Comparison of parameters estimated using LMS and perceptron based modeling of PUF 1 from chip 1

C. Convergence of Δ values

The Δ values cannot be considered as estimates of parameters of a PUF if they are not reliable and consistent. In order to validate the reliability of these estimates, the PUFs have been modeled multiple times and convergence of the estimates has been verified. Initially they are assigned a random value between 0 and 1. As the training progresses, they converge to their true values. Fig. 4(a) shows how the estimated delay differences change as more challenges are used for training. The values oscillate initially and gradually attain a stable value.

Fig. 4(b) shows the model parameters estimated using the perceptron algorithm as a function of number of CRPs used. Initially, the model parameters *vary* considerably with change in challenges but they gradually stabilize as the CRPs increase in number.

Moreover, the delay differences estimated using LMS algorithm have been observed to be scaled versions of the model parameters of perceptron. Fig. 5 presents a comparison of estimated delay differences using both techniques. Each value plotted corresponds to delay difference at a given stage (represented on the X-axis). A scale factor of -100 has been multiplied to the LMS estimates. Fig. 5 clearly illustrates a one-to-one correspondence between the models. Thus, delay differences can be estimated from either the LMS model or the perceptron model.

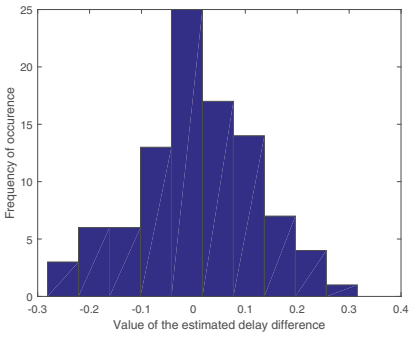


Fig. 6: Distribution of estimated delay differences at the 17th stage

V. DISCUSSION

In this section we discuss about the implications of estimating the delay differences of an arbiter PUF from a security as well as an authentication point of view.

A. Implications on predictability

Even though predicting responses of an arbiter PUF has been studied previously, the LMS algorithm based approach establishes that sophisticated machine learning techniques are not required for creating a model for a MUX PUF. As we can observe from Fig. 3(b), by exploiting a slightly more complex structure of neural networks, we are able to predict the response almost with certainty. Though the LMS approach is not 100% accurate, it provides a significant accuracy of around 97.5% for one PUF and around 99.5% for the other. A notable advantage of this approach, as compared to machine learning methods, is its simplicity. It has also been observed that LMS method requires considerably less training time per iteration as compared to the above mentioned ANNs.

B. Implications on authentication

Besides the susceptibility of arbiter PUFs to modeling attacks, the ability to parameterize the PUFs and estimate their individual delay differences at each stage is very significant. 96 PUFs were fabricated on a chip and 1000 CRPs were recorded for each of these PUFs. Based on these CRPs, PUF parameters (Δ values) were estimated and recorded. Fig. 6 shows the distribution of the estimated delay differences for 96 PUFs from one chip at the 17th stage. It has been observed that at every stage, the delay difference values follow a similar Gaussian distribution. In a prior statistical analysis of MUX based PUFs [10], the delays were modeled as an independently identically distributed (i.i.d.) random variable. The histogram in Fig. 6 validates those assumptions.

Usually, a large set of CRPs corresponding to each chip is stored in a server for the purpose of authenticating the ICs. But storing these Δ values instead of CRPs provides certain benefits. First, the storage memory requirement is considerably reduced. Moreover, it is impractical to store all the possible CRPs since the number increases exponentially as the size (number of stages) of the PUF increases. Storing the Δ values enables the server to verify the responses of a randomly chosen subset of challenges. The model parameters for the perceptron can be stored for better accuracy as these parameters are challenge independent. The advantage of the proposed model is that the server can issue random challenges without requiring challenge-response pairs to be stored.

VI. CONCLUSION

We have proposed an LMS based algorithm to estimate the delay differences, referred to as Δ values, at each stage of an arbiter PUF. Arbiter based PUFs can be modeled as additive delay models that are parameterized by these individual delay differences. It is shown that the delay estimates can be obtained by scaling the model parameters of the perceptron model. These Δ values were used to compute approximate software clones for two 32 nm arbiter PUFs with 32 stages. In both cases, the proposed model achieved significant accuracy. One conclusion of these experiments is that only the model parameters, as opposed to challenge-response pairs, can be stored on the server. This reduces memory size at the expense of time to compute the response. Future work will be directed towards modeling the feed-forward MUX PUFs based on test data from the fabricated chips.

ACKNOWLEDGMENT

This research has been supported by the National Science Foundation under grant number CNS-1441639 and the semiconductor research corporation under contract number 2014-TS-2560.

REFERENCES

[1] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, "Silicon physical random functions," in *Proceedings of the 9th ACM Conference on Computer and Communications Security*, 2002, pp. 148–160.

[2] J. Lee, D. Lim, B. Gassend, G. Suh, M. van Dijk, and S. Devadas, "A technique to build a secret key in integrated circuits for identification and authentication applications," in *Symposium on VLSI Circuits Digest of Technical Papers*, June 2004, pp. 176–179.

[3] M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Testing techniques for hardware security," in *Proceedings of IEEE International Test Conference (ITC)*, Oct 2008, pp. 1–10.

[4] G. Hospodar, R. Maes, and I. Verbauwhede, "Machine learning attacks on 65nm arbiter PUFs: Accurate modeling poses strict bounds on usability," in *IEEE International Workshop on Information Forensics and Security (WIFS)*, Dec 2012, pp. 37–42.

[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 and Security*, vol. 8, no. 11, pp. 1876–1891, Nov 2013.

[6] S. Haykin, *Adaptive Filter Theory (3rd Ed.)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1996.

[7] E. Alpaydin, *Introduction to Machine Learning*, 2nd ed. The MIT Press, 2010.

[8] D. Lim, J. Lee, B. Gassend, G. Suh, M. van Dijk, and S. Devadas, "Extracting secret keys from integrated circuits," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 13, no. 10, pp. 1200–1205, Oct 2005.

[9] M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Lightweight secure PUFs," in *IEEE/ACM International Conference on Computer-Aided Design*, Nov 2008, pp. 670–673.

[10] Y. Lao and K. K. Parhi, "Statistical analysis of MUX-based physical unclonable functions," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, no. 5, pp. 649–662, May 2014.