Machine Learning Attack on Arbiter PUF

Albert DiCroce, David Langus Rodríguez, Dean Ward
Department of Electrical and Computer Engineering
University of Florida
Gainesville, FL, USA

***Abstract*—This paper will discuss the use of machine learning algorithms to attack physical unclonable functions (PUFs). Several machine learning algorithms are discussed and analyzed as PUF models. The more accurate an ML based PUF model, the better the algorithm can predict the challenge-response pairs of a particular PUF.**

This paper Logisitic Regression and Support Vector Machine algorithms were used. Each algorithm is discussed in its efficiency in modeling a 64-bit Arbiter PUF response and analysis results compared.

Keywords—Arbiter PUF, Machine Learning Algorithm, Logistic Regression, Support Vector Machine,

I. INTRODUCTION

A. *Motivation and Background*

Hardware security is becoming an increasing problem with the rise of more and more electronic devices in everyday life. Integrated Circuits (ICs) are used for a variety of applications, from consumer electronics to high security military applications. In each application, there is the risk of security and privacy attacks to try and obtain secure information or stimulate malicious activity.

Many methods to increase security rely on some sort of cryptographical method. Most cryptography methods require a key generation in order to verify the recipient of secure data. The need for this key generation introduced the Physical Unclonable Function (PUF). A PUF utilizes the physical characteristics of the IC fabrication process to induce a response that is unique to each particular circuit. PUFS are designed using digital gates that generate a unique response that is hard to predict and unique. A PUF is characterized by its challenge-response pair (CRP). For a given PUF, a challenge input bit sequence is fed into the system and the response bit(s) can be observed.

There are several different types of PUF architectures, including ring oscillator PUFs and arbiter PUFS. An arbiter PUF consists of *N*-stages of digital logic MUX circuits. The final MUX stage feeds into a flip-flop which then generates the response. A high-level schematic of an arbiter PUF can be found in *Fig 1*.

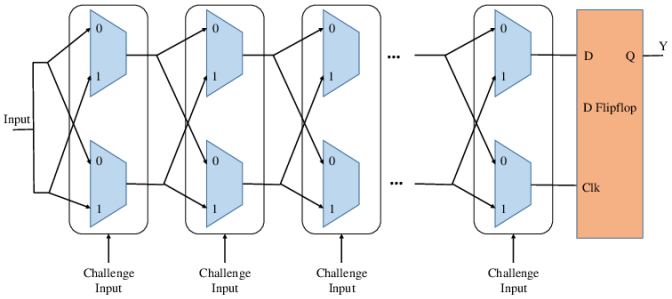

Fig. 1. Arbiter PUF Circuit [1]

B. *Problem*

Ideally, a PUF response should be difficult to model. Past research indicates that certain machine learning techniques can accurately predict and model PUF CRPs. This is a threat to applications that use PUFs for hardware security, such as key generation. An attacker can collect CRPs for a particular PUF design and derive a model.

C. *Previous Work*

This paper uses the research done by Rührmair et al. [2] by modeling different machine learning algorithms against different PUF architectures. Lim et al. [3] also provides a design for an arbiter based PUF. It describes the potential weakness of an arbiter based PUF with certain machine learning algorithms.

II. METHODS

Two machine learning algorithms were used to model an arbiter PUF. Logistic regression and support vector machines.

A. *Logistic Regression*

The logistic regression (LR) algorithm is a model that is primarily used for predictive analysis of data belonging to distinct classes. Logistic regression is known as a probabilistic discriminative classifier where the algorithm will find the probability of input features that distinguish between classifications. This is done mathematically by calculating the posterior probability $p(x|C_i)$ to map input x to the class label C. For logistic regression, a typical two class problem is modeled linearly as the log likelihood ratio of each posterior probability, shown below.

$$\log\left(\frac{P(x|C_1)}{P(x|C_2)}\right) = w^T x + b$$

This statement is only true if the classes are Gaussian-distributed. Using Baye's rule, this can be further simplified to the following, which.

$$\log\left(\frac{P(x|C_1)}{P(x|C_2)}\right) + \log\left(\frac{P(C_1)}{P(C_2)}\right) = \text{logit}\, P(C_1|x)$$

$$w^T + b + \frac{P(C_1)}{P(C_2)} = \text{logit}\, P(C_1|x)$$

This can be rearranged into the get the sigmoid function:

$$y = P(C_1|x) = \frac{1}{1 + \exp\left(-(w^T + w_0)\right)}$$

$$\Phi(z) = \frac{1}{1 + \exp(-z)}$$

The plot for the sigmoid function is shown in *Fig 2.*

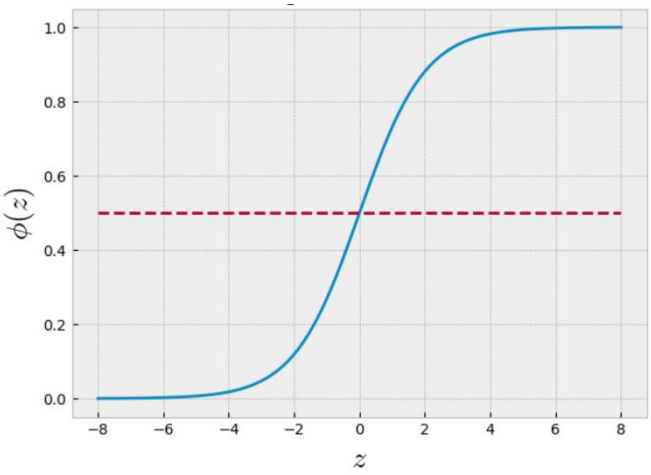

Fig. 2. Logistic Regression Sigmoid

The sigmoid function has several characteristics. The sigmoid function approaches 1 as z approaches infinity. On the other hand, as the sigmoid function approaches 0, z approaches negative infinity. This allows for a class model to be transformed into the range of [0,1] with an intercept of 0.5 to divide between classes. This probability can then be further simplified into a binary outcome with a unit step function, shown below.

$$\hat{t} = y = \begin{cases} 1, & z \geq 0 \\ 0, & z < 0 \end{cases}$$

$$= \begin{cases} 1, & \mathbf{w}^T x + w_0 \geq 0 \\ 0, & \mathbf{w}^T x + w_0 < 0 \end{cases}$$

For logistic regression, the mean squared error is used as its cost function. This can be represented as the following formula.

$$J(\mathbf{w}, w_0) = \sum_{i=1}^{N} -t_i \log \phi(z_i) - (1 - t_i) \log(1 - \phi(z_i))$$

This cost function is used to maximize the probability by minimizing the cost function. A decrease in the cost function will inherently increase the likelihood a particular sample is contained in one class. The cost function can be graphed and visually represented in *Fig 3.*

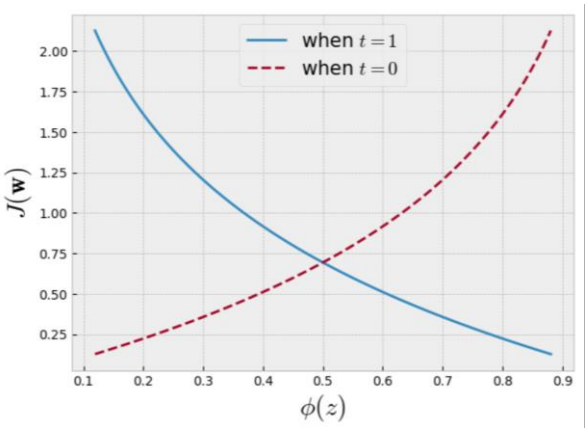

Fig. 3. Logistic Regression Sigmoid

As the cost function approaches 0, it is probable that a sample is in class 1. On the other hand, as the *y* axis approaches 0. It is probable that a sample is in class 2. There is a risk that the prediction can be wrong, and the cost function will approach infinity. In order to penalize wrong predictions, gradient decent is used as a search method for local optima. The gradient decent needs an initial value and a learning rate, or step size. The gradient decent function can be shown with the following derived functions.

$$\mathbf{w}^{(t+1)} \longleftarrow \mathbf{w}^{(t)} + \eta \sum_{i=1}^{N} (t_i - y_i) x_i$$

$$\mathbf{w_0}^{(t+1)} \longleftarrow \mathbf{w_0}^{(t)} + \eta \sum_{i=1}^{N} (t_i - y_i)$$

The logistic regression model used for the PUF CRP had the following parameters TBD.

B. Support Vector Machines (SVM)

The other machine learning method used in this PUF analysis was Support Vector Machines (SVM). The SVM algorithm is used in both regression and classification models. The main objective of the SVM algorithm is to find a hyperplane in an N-dimensional space where N represents the feature size. The hyperplane chosen to separate each class of data should have the maximum margin between data samples of each class. This maximization is key so that future data samples can be classified with increased confidence. An example of margins can be shown in *Fig 4.*

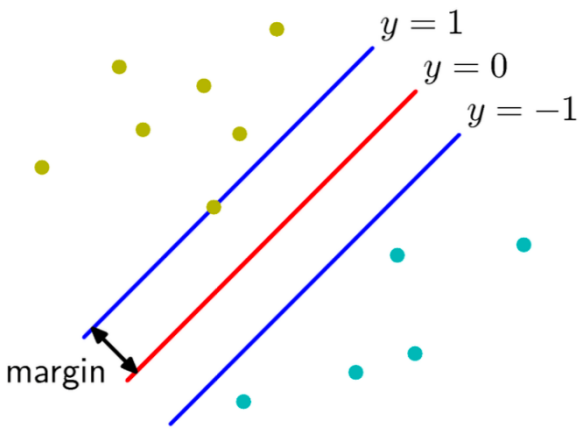

Fig. 4. Ex

These decision boundaries are what separate classes of data points and make decision for what class a data sample belongs to. Mathematically, the maximization function looks like the following formula.

$$\arg_{w,b} \max \left\{ \frac{1}{\|w\|} \min_n \left[t_n (w^T \phi(x_n) + b) \right] \right\}$$

The support vectors are made up of the data samples that are closest to the decision boundaries. The support vectors are crucial for maximizing the margin, and their direction and magnitude directly relate to the margin of the SVM classification model. The support vectors can be modeled as the following constraints.

$$t_n (w^T \phi(x_n) + b) \geq 1, n = 1, 2, \ldots, N$$

This is known as the canonical representation of the decision hyperplane. For any given datapoints where the statement is true, the constraints are said to be active while all others are inactive.

C. Implementation

The arbiter PUF attacked, was a 64-bit PUF. The dataset consisted of 12,000 challenge response pairs. Due to the nature of PUFs and their resulting CRPs, a classification model is the best machine learning model to use for attacking a PUF. Knowing the architecture of the PUF and the results of each individual stage, an algorithm model can map each stage result to the probability of a PUF response belonging to a certain response class, 0 or 1. Logistic regression is the best regression model for

classification. The logistic regression model used for this analysis used a grid search cross validation to tune the hyperparameters. Thet hyperparameters for the grid search are shown in *Table 1*.

Table 1
Hyperparameters passed for cross validation.

Hyperparameter	*Tuned Parameter*
Penalty	L2 penalty term
Stopping tolerance	10e-6 to 10e-1
Inverse Regularization Strength	0.01, 0.1, 1, 10

The overall flowchart diagram for the logistic regression model can be found in *Fig 5*.

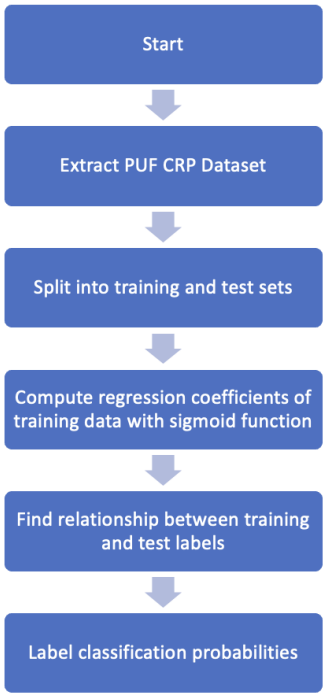

Fig. 5. Logistic Regression Flowchart

The SVM model was chosen as the other algorithm to model a PUF attack. SVM is used for both regression and classification tasks, but it is better suited for classification. SVM is better suited for when logistic regression performs poorly, or the dataset is not linearly separable. If the dataset is not linearly separable, a kernel function can be used to assist with classification. A non-linear SVM model was used for this dataset, since the CRP data was not linearly separable. The kernel used was a radial based function kernel. The SVM algorithm was pipelined with a standard scaler function to standardize the data features. The flowchart diagram for the SVM model used in the PUF attack can be found in *Fig 6.*

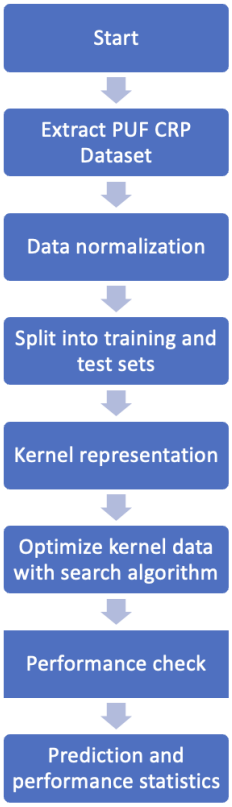

Fig. 6. SVM Flowchart

III. Results

The PUF CRP dataset was split using several ratios and compared to see how each model performed with different training and test datasets.

A. *Logistic Regression*

The logistic regression model overall performed poorly. *Tables 2 and 3* show the results of logistic regression with 10,000 samples for training out of 12,000 samples.

Table 2
Logistic regression training results with 16% split

Classifier	Results		
	Precision	Recall	F1-Score
0	57%	79%	60%
1	55%	30%	39%

Table 3

Logistic regression test results with 16% split

Classifier	Results		
	Precision	Recall	F1-Score
0	56%	79%	65%
1	55%	29%	38%

The overall accuracy against the training data was 57% and 56% against the test data. *Tables 4 and 5* show the results of logistic regression with 6,000 samples for training out of 12,000 samples.

Table 4

Logistic regression training results with 50% split

Classifier	Results		
	Precision	Recall	F1-Score
0	58%	80%	67%
1	56%	30%	39%

Table 5

Logistic regression test results with 50% split

Classifier	Results		
	Precision	Recall	F1-Score
0	56%	79%	65%
1	52%	26%	35%

The overall accuracy against the training data was 57% and 55% against the test data. *Tables 6 and 7* show the results of logistic regression with 2,000 samples for training out of 12,000 samples.

Table 6

Logistic regression training results with 80% split

Classifier	Results		
	Precision	Recall	F1-Score
0	58%	91%	71%
1	62%	20%	30%

Table 7

Logistic regression test results with 80% split

Classifier	Results		
	Precision	Recall	F1-Score
0	55%	88%	67%
1	51%	15%	23%

The overall accuracy against the training data was 59% and 54% against the test data. The overall accuracy was always around 60% for all training datasets and a little lower for the test data. Changing the training/test split didn't impact accuracy, since the logistic model struggled with the non-linearly separable data.

B. SVM

The SVM model overall performed better than the logistic regression. *Tables 8 and 9* show the results of logistic regression with 10,000 samples for training out of 12,000 samples.

Table 8

SVM training results with 16% split

Classifier	Results		
	Precision	Recall	F1-Score
0	79%	93%	85%
1	89%	70%	78%

Table 9

SVM test results with 16% split

Classifier	Results		
	Precision	Recall	F1-Score
0	60%	77%	67%
1	61%	41%	49%

The overall accuracy against the training data was 82% and 60% against the test data. *Tables 10 and 11* show the results of SVM with 6,000 samples for training out of 12,000 samples.

Table 10

SVM training results with 50% split

Classifier	Results		
	Precision	Recall	F1-Score
0	81%	95%	87%
1	92%	73%	81%

Table 11

SVM test results with 50% split

Classifier	Results		
	Precision	Recall	F1-Score
0	59%	75%	66%
1	57%	38%	45%

The overall accuracy against the training data was 85% and 58% against the test data. *Tables 12 and 13* show the results of SVM with 2,000 samples for training out of 12,000 samples.

Table 12
SVM training results with 80% split

Classifier	Results		
	Precision	Recall	F1-Score
0	84%	99%	91%
1	98%	77%	86%

Table 13
SVM test results with 80% split

Classifier	Results		
	Precision	Recall	F1-Score
0	55%	81%	66%
1	51%	24%	33%

The overall accuracy against the training data was 89% and 54% against the test data. The training datasets always performed better compared to the test dataset. This indicates that the training overfits the model and didn't perform well during the testing phase.

IV. Conclusion

In conclusion, the PUF attack was best modeled by the SVM algorithm. The logistic regression model struggled with the given dataset since it was not linearly separable. The SVM model performed better but tended to overfit the dataset during training. Compared to Rhrmair, the methods performed there included a larger dataset and the use of RProp gradient descent for tuning hyperparameters for logistic regression.

References

[1] Yanambaka, Venkata Prasanth & Mohanty, Saraju & Kougianos, Elias. (2018). Making Use of Manufacturing Process Variations: A Dopingless Transistor Based-PUF for Hardware-Assisted Security. IEEE Transactions on Semiconductor Manufacturing. PP. 1-1. 10.1109/TSM.2018.2818180.

[2] Rhrmair, Ulrich, et al. "PUF modeling attacks on simulated and silicon data." IEEE Transactions on Information Forensics and Security 8.11 (2013): 1876-1891

[3] Lim, Daihyun. Extracting secret keys from integrated circuits. Diss. Massachusetts Institute of Technology, 2004.