
CS771 : Introduction to Machine Learning

Assignment - 1

Abhishek Sahu

220049

abhisahu22@iitk.ac.in

Naman Sethi

220688

namansethi22@iitk.ac.in

Sneh Sinha

221067

snehsinha22@iitk.ac.in

Mayur Agarwal

220641

mayurag22@iitk.ac.in

Pragati Rathi

220780

pragati22@iitk.ac.in

Zartaf Raza

221237

zartafr22@iitk.ac.in

1

Abstract

This report evaluates the security and predictability of a newly proposed Physical Unclonable Function (PUF) variant known as the Cross-Connection PUF (COCO-PUF). Traditional arbiter PUFs, which are vulnerable to linear modeling attacks, serve as the foundation for the COCO-PUF. Unlike standard arbiter PUFs, the COCO-PUF incorporates two arbiter PUFs and interconnects their output signals, theoretically enhancing complexity and security. This study aims to assess whether this cross-connection design substantially improves COCO-PUF's resistance to linear modeling attacks. We derive mathematical models demonstrating that COCO-PUF responses, similar to those of arbiter PUFs, can be accurately predicted using linear models.

Using a dataset of 40,000 challenge-response pairs (CRPs), we employed linear classifiers, including Logistic Regression and Linear Support Vector Classification (LinearSVC), to predict COCO-PUF responses. Our experimental results show that both models achieve high accuracy, thereby challenging the security claims of COCO-PUF. These findings suggest that despite the added complexity, COCO-PUF remains susceptible to linear modeling attacks. Therefore, similar cryptographic measures to those used for traditional arbiter PUFs should be considered.

1 Derivation for a Simple Arbiter PUF

Let $t_{i,1}^u$ and $t_{i,1}^l$ be the times of upper and lower signals when they leave from i -th MUX. Using the given challenge bit lets say c_i , we can express these times as :

$$t_i^u = (1 - c_i) \cdot (t_{i-1}^u + p_i) + c_i \cdot (t_{i-1}^l + s_i)$$

$$t_i^l = (1 - c_i) \cdot (t_{i-1}^l + q_i) + c_i \cdot (t_{i-1}^u + r_i)$$

To simplify the analysis, we define two new variables, X_i and Y_i , as follows:

$$\begin{aligned} X_i &= t_i^u + t_i^l \\ Y_i &= t_i^u - t_i^l \end{aligned}$$

Using the above, we get:

$$\begin{aligned} t_i^u &= \frac{X_i + Y_i}{2} \\ t_i^l &= \frac{X_i - Y_i}{2} \end{aligned}$$

Simplifying the above expression, we get:

$$X_i = (1 - c_i) \cdot (p_i + q_i) + c_i \cdot (s_i + r_i) + X_{i-1}$$

Similarly:

$$Y_i = (1 - c_i) \cdot (Y_{i-1} + p_i - q_i) + c_i \cdot (-Y_{i-1} + s_i - r_i)$$

To further simplify, we define the following parameters:

$$d_i = (1 - 2 \cdot c_i)$$

$$\alpha_i = \frac{p_i - q_i + r_i - s_i}{2}$$

$$\beta_i = \frac{p_i - q_i - r_i + s_i}{2}$$

$$\gamma_i = \frac{p_i + q_i + r_i - s_i}{2}$$

$$\delta_i = \frac{s_i + r_i - (p_i + q_i)}{2}$$

Rewriting X_i and Y_i by using the above notations as:

Assuming $t_0^u = 0$ and $t_0^l = 0$

$$X_i = (1 - c_i) \cdot (p_i + q_i) + c_i \cdot (s_i + r_i) + X_{i-1} \quad \text{where } X_0 = 0$$

$$Y_i = (1 - 2 \cdot c_i) \cdot (Y_{i-1}) + c_i \cdot (s_i - r_i - (p_i - q_i)) + p_i - q_i$$

Simplifying further, we get:

$$Y_i = d_i \cdot Y_{i-1} + d_i \cdot \alpha_i + \beta_i$$

Thus, the equations for X_i and Y_i can be expressed as:

$$X_i = p_i + q_i + c_i \cdot (-(p_i + q_i) + (s_i + r_i)) + X_{i-1}$$

$$X_i = X_{i-1} - d_i \cdot \delta_i + \gamma_i$$

$$Y_i = d_i \cdot Y_{i-1} + d_i \cdot \alpha_i + \beta_i$$

Solving for $i = 32$ (as the PUF has 32 stages), we obtain:

$$X_{32} = w'_1 \cdot x'_1 + w'_2 \cdot x'_2 + \cdots + w'_{32} \cdot x'_{32} + b'$$

$$Y_{32} = w_1 \cdot x_1 + w_2 \cdot x_2 + \cdots + w_{32} \cdot x_{32} + b$$

Defining the vectors x_i and x'_i as:

$$x_i = d_i d_{i+1} \cdots d_{32}, \quad x'_i = -d_i$$

and the weights w_i and w'_i as:

$$w_i = \alpha_i, \quad w'_i = \delta_i$$

we have:

$$w_i = \alpha_i + \beta_{i-1} \quad (\text{where } i = 1, 2, \dots)$$

Finally, we define the bias terms b and b' as:

$$b = \beta_{32}, \quad b' = \sum_{i=1}^{32} \gamma_i$$

t_{32}^u can be expressed using the derived equation of X and Y :

$$t_{32}^u = \frac{X_{32} + Y_{32}}{2}$$

$$t_{32}^u = \sum_{i=1}^{32} \frac{w_i \cdot x_i + w'_i \cdot x'_i + b + b'}{2}$$

$$t_{32}^u = \frac{w^T \cdot x + b + w'^T \cdot x' + b'}{2}$$

$$t_{32}^u = \frac{(w'^T \cdot x' + b') + (w^T \cdot x + b)}{2}$$

The above equation satisfies the condition for linearity. Thus, we prove that there exists a feature map Φ such that $\Phi : \{0, 1\}^{32} \rightarrow \mathbb{R}^D$.

Let $B = b + b'$, $x = f(c)$ and $x' = g(c)$,

$$\mathbf{W}^T = [w \quad w'^T]$$

$$\phi(\mathbf{c})^T = [x \quad x']$$

Thus, we obtain the formula of : $t^u(c) = W^T \phi(c) + B$

2 Dimensionality for a Simple Arbiter PUF

The equation for upper signal time can be expanded as follows:

$$t_{32}^u = \frac{w_1 \cdot x_1 + w_2 \cdot x_2 + \cdots + w_{32} \cdot x_{32} + b + w'_1 \cdot x'_1 + w'_2 \cdot x'_2 + \cdots + w'_{32} \cdot x'_{32} + b'}{2}$$

Where w and w' are feature coefficients vector, x and x' are feature vectors, and b and b' are bias terms.

Given the property of the feature vectors, we assume x_i can be related to x'_i as follows:

$$x_i = (-1)^i \cdot x'_i \quad \text{so} \quad x_{32} = -x'_{32}$$

Substituting $x_{32} = -x'_{32}$ into the expression for t_{32}^u , we get:

$$t_{32}^u = \frac{w_1 \cdot x_1 + w_2 \cdot x_2 + \cdots + w_{31} \cdot x_{31} + w_{32} \cdot (-x'_{32}) + b + w'_1 \cdot x'_1 + w'_2 \cdot x'_2 + \cdots + w'_{32} \cdot x'_{32} + b'}{2}$$

Simplifying this, we have:

$$t_{32}^u = \frac{w_1 \cdot x_1 + w_2 \cdot x_2 + \cdots + w_{31} \cdot x_{31} + w'_1 \cdot x'_1 + w'_2 \cdot x'_2 + \cdots + (w'_{32} - w_{32}) \cdot x'_{32} + B}{2}$$

From the given expression, we can determine the dimensionality of the model by examining the number of terms in the expression :

$$31 + 31 + 1 = 63$$

Thus, we conclude:

$$\text{Dimensionality} = 63$$

3 Derivation for a COCO-PUF

After analyzing time for signal to reach the finish line, we need to compare the lower and upper signal propagation time for both arbiter 0 and 1. We define the responses r^0 and r^1 as follows:

First, the response r^0 is determined by comparing the time differences of the lower signals between two stages

If $t_{32,0}^l > t_{32,1}^l$, then $r^0 = 1$

else $r^0 = 0$

Therefore:

$$r^0 = \text{bool}(t_{32,0}^l - t_{32,1}^l > 0)$$

Similarly, r^1 is determine by comparing the time differences of the upper signals:

$$r^1 = \text{bool}(t_{32,0}^u - t_{32,1}^u > 0)$$

Next, we express r^0 using the feature coefficient and feature vectors derived from the signal propagation times in the arbiter PUF. Given that t_{32}^l and t_{32}^u are linear functions of the feature vectors and feature coefficient, we have:

$$r^0 = \text{bool}(w_{0,l}^T x'_{0,l} + b'_{0,l} - w_{0,l}^T x_{0,l} + b_{0,l} - w'^T_{1,l} x'_{1,l} + b'_{1,l} + w_{1,l}^T x_{1,l} + b_{1,l})$$

Using the signum function which is defined as $\text{sgn}(x) = \frac{x}{|x|}$. We can represent r^0 as :

$$r^0 = \frac{1 + \text{sgn}(w_{0,l}^T x'_{0,l} + b'_{0,l} - w_{0,l}^T x_{0,l} + b_{0,l} - w'^T_{1,l} x'_{1,l} + b'_{1,l} + w_{1,l}^T x_{1,l} + b_{1,l})}{2}$$

Let $B = b_0 + b'_0 + b_1 + b'_1$, $x_i = f_i(c)$ and $x'_i = g_i(c)$ where $c \in \{0, 1\}^{32}$

$$\begin{aligned}\tilde{\phi}(\mathbf{c})^T &= [x_0 \quad x'_0 \quad x_1 \quad x'_1] \\ \tilde{\mathbf{W}}^T &= [-w_0^T \quad w'^T_0 \quad w_1^T \quad -w'^T_1]\end{aligned}$$

Hence we obtain :

$$r^0(c) = \frac{1 + \text{sgn}(\tilde{\mathbf{W}}_0^T \tilde{\phi}(c) + B_0)}{2}$$

Similarly, for the response r^1 , we have:

$$r^1(c) = \frac{1 + \text{sgn}(\tilde{\mathbf{W}}_1^T \tilde{\phi}(c) + B_1)}{2}$$

where $\tilde{\mathbf{W}} \in \mathbb{R}^{\tilde{D}}$, $\tilde{b} \in \mathbb{R}$ and $\tilde{\phi} : \{0, 1\}^{32} \rightarrow \mathbb{R}^{\tilde{D}}$

4 Dimensionality for a COCO-PUF

Using r^0 and r^1 from part 3 we have :

$$\begin{aligned}r^0 &= \frac{1 + \text{sgn}(w_{0,l}^T x'_{0,l} + b'_{0,l} - w_{0,l}^T x_{0,l} + b_{0,l} - w'^T_{1,l} x'_{1,l} + b'_{1,l} + w_{1,l}^T x_{1,l} + b_{1,l})}{2} \\ r^1 &= \frac{1 + \text{sgn}(w_{0,u}^T x'_{0,u} + b'_{0,u} - w_{0,u}^T x_{0,u} + b_{0,u} - w'^T_{1,u} x'_{1,u} + b'_{1,u} + w_{1,u}^T x_{1,u} + b_{1,u})}{2}\end{aligned}$$

Each weight vector and bias can be considered a separate dimension and for both responses, the term x_{32} is consistent across all four quantities and each feature having a dimension of 32 , resulting in a total dimensionality of $4 \times 31 + 1 = 125$.

Dimensionality = 125

5 Solution for Part 5

Zipped Solution to Assignment 1

6 Performance Analysis for Models

(a) Performance Comparison of LinearSVC with Different Loss Functions:

Loss function	Total Features	Model Train Time (in sec)	Test accuracy 0	Test Accuracy 1
Default	32	10.5585	98.40%	99.43%
Hinge	32	7.9534	98.22%	99.34%
Squared Hinge	32	10.8123	98.59%	99.44%

Analysis:

From the table, we observe the following:

- **Training Time:** The model using the Hinge loss function has the shortest training time of 7.1150 seconds, while the Squared Hinge loss function requires 10.1847 seconds for training. This shows that the Hinge loss function is computationally more efficient.
- **Test Accuracy for Class 0:** The test accuracy for class 0 is slightly lower with the Hinge loss function (98.19%) compared to the Squared Hinge loss function (98.64%). This implies that the Squared Hinge loss function provides a better fit for class 0.
- **Test Accuracy for Class 1:** The test accuracy for class 1 is marginally higher with the Squared Hinge loss function (99.45%) compared to the Hinge loss function (99.29%). This indicates that the Squared Hinge loss function also performs slightly better for class 1.

To sum up, while the Squared Hinge loss function provides a modest improvement in test accuracy for both classes, it also demands more training time. Depending on the application's specific requirements and constraints, one might opt for the Hinge loss function for quicker training or the Squared Hinge loss function for slightly enhanced accuracy.

(b) Performance Comparison based on the value of C

Analysis for LinearSVC with different C values:

Table 1: Performance Comparison of LinearSVC with Different C Values

C Value	Total Features	Model Train Time (in sec)	Test Accuracy 0	Test Accuracy 1
Low	32	12.9694	98.17%	99.29%
Medium	32	10.9830	98.48%	99.47%
High	32	11.2775	98.32%	99.38%

- **Training Time:** As C increases, the training time decreases slightly from 11.4641 seconds for low C to 10.0647 seconds for high C . This suggests that higher regularization (lower C) leads to more computational complexity.
- **Test Accuracy for Class 0:** The test accuracy for class 0 is highest for the medium C value (0.9860), followed by the high C value (0.9841), and lowest for the low C value (0.9817). This indicates that the model's performance improves with increased regularization up to a certain point.
- **Test Accuracy for Class 1:** Similarly, the test accuracy for class 1 is highest for the medium C value (0.9944), followed by the high C value (0.9938), and lowest for the low C value (0.9929). This trend suggests that a medium level of regularization yields the best overall performance.

Analysis for LogisticRegression with different C values:

The evaluated performance of 'LogisticRegression' with varying C values.

Table 2: Performance Comparison of LogisticRegression with Different C Values

C Value	Total Features	Model Train Time (in sec)	Test Accuracy 0	Test Accuracy 1
Low	32	14.2893	97.77%	98.96%
Medium	32	13.8352	98.24%	99.36%
High	32	13.9132	98.24%	99.35%

- **Training Time:** As the value of C increases, the training time slightly decreases, dropping from 13.6300 seconds for a low C to 13.1017 seconds for a high C . This suggests that higher values of C (indicating lower regularization) lead to shorter training times.
- **Test Accuracy for Class 0:** The test accuracy for class 0 improves from 97.77% for a low C to 98.24% for a high C . This indicates that increasing the value of C (reducing regularization) enhances the model's accuracy for class 0.

- **Test Accuracy for Class 1:** Similarly, the test accuracy for class 1 rises from 98.96% for a low C to 99.35% for a high C . This shows that lower regularization improves performance for class 1 as well.

In summary, the performance of both *LinearSVC* and *LogisticRegression* is affected by the regularization parameter C . For *LinearSVC*, a medium C value strikes the best balance between training time and accuracy. In contrast, for *LogisticRegression*, higher C values (lower regularization) result in better accuracy for both classes, while the training time remains relatively stable.

(c) Effect of Variation in Tolerance for LinearSVC and LogisticRegression Models

For part (c) of this analysis, we assess the performance of *LinearSVC* and *LogisticRegression* models by varying the `tol` hyperparameter. The `tol` hyperparameter dictates the tolerance level for the optimization algorithm. Lower `tol` values can result in longer training times but may enhance convergence and accuracy.

Table 3: Performance Comparison of LinearSVC and LogisticRegression with Different Tolerance

Model	Tolerance	Total Features	Model Train Time (in sec)	Test Accuracy 0	Test Accuracy 1
LinearSVC	Low	32	10.5112	98.61%	99.44%
LinearSVC	Medium	32	10.6930	98.51%	99.43%
LinearSVC	High	32	10.0257	98.60%	99.44%
LogisticRegression	Low	32	13.7456	98.06%	99.26%
LogisticRegression	Medium	32	13.9469	98.06%	99.26%
LogisticRegression	High	32	13.8676	98.06%	99.26%

LinearSVC:

- **Training Time:** Training time increases slightly as tolerance values rise, from 8.9728 seconds at low tolerance to 9.8234 seconds at high tolerance. This suggests that lower tolerance values, requiring more precise convergence, lead to quicker training times.
- **Test Accuracy for Class 0:** Test accuracy for class 0 remains fairly stable across different tolerance values, with a small variation. Accuracy ranges from 98.54% at medium tolerance to 98.62% at high tolerance, indicating that tolerance has little impact on performance for class 0.
- **Test Accuracy for Class 1:** Test accuracy for class 1 is also consistent across different tolerance values, ranging from 99.42% to 99.45%. This stability implies that the tolerance parameter does not significantly affect the model's ability to accurately classify class 1 instances.

LogisticRegression:

- **Training Time:** Training time varies with tolerance values, from 13.1829 seconds at high tolerance to 14.2537 seconds at medium tolerance. Unlike *LinearSVC*, *LogisticRegression* does not exhibit a clear trend in training time relative to tolerance.
- **Test Accuracy for Class 0:** Test accuracy for class 0 remains consistent at 98.06% across all tolerance values, indicating that tolerance does not affect performance for class 0 in *LogisticRegression*.
- **Test Accuracy for Class 1:** Similarly, test accuracy for class 1 stays constant at 99.26% for all tolerance values, showing that the tolerance parameter does not influence the model's performance for class 1.

In summary, the tolerance parameter has a minor impact on the training time and accuracy of the *LinearSVC* model, with lower tolerance values leading to slightly faster training times. Conversely, the tolerance parameter does not significantly affect the performance of the *LogisticRegression* model, as both training time and accuracy remain stable across different tolerance values.

(d) Effect of Variation in Penalty (Regularization) Hyperparameter for LinearSVC and LogisticRegression Models

In part (d) of this analysis, we assess the performance of the *LinearSVC* and *LogisticRegression* models with varying penalty (regularization) hyperparameters. The penalty hyperparameter determines the type of regularization used to prevent overfitting. We compare the models' performance using both L1 and L2 regularization methods.

Table 4: Performance Comparison of LinearSVC and LogisticRegression with Different Penalty

Model	Penalty	Total Features	Model Train Time (in sec)	Test Accuracy 0	Test Accuracy 1
LinearSVC	L1	32	125.2938	98.54%	99.45%
LinearSVC	L2	32	10.9339	98.58%	99.42%
LogisticRegression	L1	32	92.7126	98.49%	99.39%
LogisticRegression	L2	32	13.7720	98.06%	99.26%

LinearSVC:

- **Training Time:** Training LinearSVC with an L1 penalty takes significantly longer at 121.6582 seconds compared to 12.0927 seconds for an L2 penalty. This difference reflects that the L1 penalty, which encourages sparsity in model coefficients, demands more computational resources and time for convergence.
- **Test Accuracy for Class 0:** The test accuracy for class 0 is 98.61% with an L1 penalty and 98.59% with an L2 penalty. This small variance indicates that both penalties yield comparable performance for class 0.
- **Test Accuracy for Class 1:** Similarly, the test accuracy for class 1 is 99.43% with an L1 penalty and 99.45% with an L2 penalty. Like class 0, the choice of penalty has minimal impact on accuracy for class 1.

LogisticRegression:

- **Training Time:** Training LogisticRegression with an L1 penalty requires 78.0598 seconds, while an L2 penalty takes 13.1829 seconds. This demonstrates that L1 regularization is more computationally intensive than L2 regularization for LogisticRegression.
- **Test Accuracy for Class 0:** The test accuracy for class 0 is 98.48% with an L1 penalty and 98.06% with an L2 penalty. The higher accuracy with the L1 penalty suggests that it may be more effective in preventing overfitting for class 0.
- **Test Accuracy for Class 1:** Similarly, the test accuracy for class 1 is 99.39% with an L1 penalty and 99.26% with an L2 penalty. Again, the L1 penalty shows slightly better performance for class 1, mirroring the results for class 0.

In a nutshell, the choice between L1 and L2 penalties significantly affects training times but has minimal impact on test accuracies for both *LinearSVC* and *LogisticRegression* models. L1 penalties require substantially more computational effort due to their promotion of sparsity in model coefficients. However, they demonstrate slightly higher test accuracies, particularly for *LogisticRegression*, suggesting potential benefits in preventing overfitting.