CS771 Mini Project Report

Group Number: 82

Group Name: The Hyperparameter Heroes

Group Members

1. Keyansh Vaish (Roll No: 220525)
2. Priyanshu Gupta (Roll No: 220826)
3. Keshav Vinod Son (Roll No: 220518)
4. Karan Jangid (Roll No: 220500)
5. Aditya Mishra (Roll No: 220070)

Abstract

This project presents a comprehensive mathematical and algorithmic framework for modeling and classifying responses of Machine Learning-based Physically Unclonable Functions (ML-PUFs). We derive a linear delay model using stage-wise signal propagation, construct a 105-dimensional feature representation, and demonstrate that a polynomial kernel SVM can perfectly classify PUF responses without explicit feature engineering. We also develop a method to recover non-negative delay values compatible with the model and compare performance across logistic regression and linear SVC classifiers, showing consistent 100% classification accuracy under varied hyperparameters.

1 Mathematical Derivation for ML-PUF Linear Model

1.1 Delay Difference Derivation

We define the delay difference at stage i as:

$$\Delta_i = t_i^u - t_i^l$$

The arrival times at the upper and lower paths of stage i are: $t_i^u = (1-c_i)(t_{i-1}^u + p_i) + c_i(t_{i-1}^l + s_i)$
$t_i^l = (1-c_i)(t_{i-1}^l + q_i) + c_i(t_{i-1}^u + r_i)$

Then the difference Δ_i becomes:

$\Delta_i = t_i^u - t_i^l$
$= (1-c_i)(t_{i-1}^u + p_i) + c_i(t_{i-1}^l + s_i)$
$- \left[(1-c_i)(t_{i-1}^l + q_i) + c_i(t_{i-1}^u + r_i)\right]$
$= (1-c_i)(t_{i-1}^u - t_{i-1}^l + p_i - q_i) + c_i(t_{i-1}^l - t_{i-1}^u + s_i - r_i)$
$= (1-c_i)(\Delta_{i-1} + p_i - q_i) + c_i(-\Delta_{i-1} + s_i - r_i)$
$= \Delta_{i-1}(1-2c_i) + (1-c_i)(p_i - q_i) + c_i(s_i - r_i)$

Preprint. Under review.

Define:

$$d_i = 1 - 2c_i, \quad \alpha_i = \frac{p_i - q_i + s_i - r_i}{2}, \quad \beta_i = \frac{p_i - q_i - s_i + r_i}{2}$$

Then:

$$\Delta_i = \Delta_{i-1} d_i + \alpha_i d_i + \beta_i$$

1.2 Sum of Delays

From equations:

$\mathrm{t}_i^u = (1 - c_i)(t_{i-1}^u + p_i) + c_i(t_{i-1}^l + s_i)(1)$

$\mathrm{t}_i^l = (1 - c_i)(t_{i-1}^l + q_i) + c_i(t_{i-1}^u + r_i)(2)$

Add (1) and (2):

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

Define:

$$r_i = s_i + r_i - p_i - q_i$$

Then summing from $i = 0$ to 7:

$$t_7^u + t_7^l = \sum_{i=0}^{7} (p_i + q_i) + \sum_{i=0}^{7} c_i \cdot r_i$$

Using Delay Differences

From previous derivation:

$$t_7^u - t_7^l = \sum_{i=0}^{7} w_i x_i + \beta_7$$

with: $\mathrm{x}_i = \prod_{j=i}^{7} d_j$
$w_0 = \alpha_0, \quad w_i = \alpha_i + \beta_{i-1} for i \geq 1$
$d_i = 1 - 2c_i$
$\alpha_i = \frac{p_i - q_i + s_i - r_i}{2}, \quad \beta_i = \frac{p_i - q_i - s_i + r_i}{2}$

Add and divide:

$$2t_7^u = \sum_{i=0}^{7} c_i r_i + \sum_{i=0}^{7} w_i x_i + \beta_7$$

$$t_7^u = \frac{1}{2} \left(\sum_{i=0}^{7} c_i r_i + \sum_{i=0}^{7} w_i x_i + \beta_7 \right)$$

Rewriting:

$$t_7^u = \frac{1}{2} \left(c_7(r_7 - 2w_7) + \sum_{i=0}^{6} c_i r_i + \sum_{i=0}^{6} w_i x_i + \beta_7 + w_7 \right)$$

In matrix form:

$$t_7^u = \frac{1}{2} \left(W^T \phi(c) + b \right)$$

Where: $\phi(c) = [c_7, c_6, \ldots, c_0, x_0, x_1, \ldots, x_6]^T$
$W = [(r_7 - 2w_7), r_6, \ldots, r_0, w_0, w_1, \ldots, w_6]^T$
$b = \beta_7 + w_7$

1.3 Linear Models for Multiple Responses

Let $t^u_{1,7}$ and $t^u_{0,7}$ be upper path delays for two cases $1, 0$:

$$t^u_{1,7} - t^u_{0,7} = \frac{1}{2}\left(\tilde{W}^T\phi(c) + \tilde{b}\right)$$

Where: $\tilde{W} = [(r_{1,7} - 2w_{1,7}) - (r_{0,7} - 2w_{0,7}), r_{1,6} - r_{0,6}, \ldots, r_{1,0} - r_{0,0}, w_{1,0} - w_{0,0}, \ldots, w_{1,6} - w_{0,6}]^T$
$\tilde{b} = (\beta_{1,7} + w_{1,7}) - (\beta_{0,7} + w_{0,7})$

Similarly for lower path:

$$t^l_{1,7} - t^l_{0,7} = \frac{1}{2}\left(\hat{W}^T\phi(c) + \hat{b}\right)$$

Where: $\hat{W} = [(r_{1,7} + 2w_{1,7}) - (r_{0,7} + 2w_{0,7}), r_{1,6} - r_{0,6}, \ldots, r_{1,0} - r_{0,0}, -(w_{1,0} - w_{0,0}), \ldots, -(w_{1,6} - w_{0,6})]^T$
$\hat{b} = -(\beta_{1,7} + w_{1,7}) + (\beta_{0,7} + w_{0,7})$

XOR Response Feature Map (105-Dimensional)

Let ϕ_0 and ϕ_1 be the feature maps for two different responses. The XOR operation on responses corresponds to the element-wise product of the feature maps as shown in the lecture slides. That is, we consider:

$$\prod_i (\mathbf{w}_i^T\mathbf{x})$$

which implies that our new feature map for breaking ML-PUF is where $\odot$ is the pairwise element-wise multiplication of two vectors:

$$\Phi_{XOR} = \phi_0 \odot \phi_1$$

However, to avoid redundancy, we construct a new feature map $\Psi(\mathbf{r}_0, \mathbf{r}_1)$ of dimension 120 as follows. Also note that $\phi_i(\mathbf{r}_0)$ is the value of ϕ_0 at the index i:

- 15 quadratic terms: $\phi_i(\mathbf{r}_0) \cdot \phi_i(\mathbf{r}_1)$ for $i = 1$ to 15
- 105 unique cross terms: $\phi_i(\mathbf{r}_0) \cdot \phi_j(\mathbf{r}_1)$ for $i < j$

Thus,

$$\tilde{\phi}(\mathbf{r}_0, \mathbf{r}_1) = [\phi_1(\mathbf{r}_0)\phi_1(\mathbf{r}_1), \ldots, \phi_{15}(\mathbf{r}_0)\phi_{15}(\mathbf{r}_1), \phi_1(\mathbf{r}_0)\phi_2(\mathbf{r}_1), \ldots, \phi_{14}(\mathbf{r}_0)\phi_{15}(\mathbf{r}_1)]$$

The final XOR response model is then:

$$y = \tilde{W}^T\tilde{\phi}(\mathbf{r}_0, \mathbf{r}_1) + \tilde{b}$$

where $\tilde{W} \in R^{105}$ and $\tilde{b}$ is the bias term.

2 Dimensionality of $\tilde{D}$

Based on our derivation in Section 1, we see that we require 15 dimensions to find the time in which the upper signal or lower signal of a PUF reaches. We can learn two linear models to compare the upper signals and lower signals of the ML-PUF.

Since there is an XOR block at the end, we need to obtain the pairwise multiplication of these two feature maps, which will have 15 square terms and 210 cross terms, out of which half are repeated. So we are left with 105, which, when added to the original 15, gives us a feature map of 120 dimensions and since the square terms are all 1 based on how we defined a transformation from 0,1 to-1,1 we get reduction of dimensionality by 15 hence giving 105.

3 Kernel SVM Configuration for Perfect Classification

Kernel Choice and Theoretical Justification

To classify ML-PUF responses using original challenges $\mathbf{c} \in \{0,1\}^8$ without explicit feature engineering, we propose a polynomial kernel of degree 2:

$$K(\mathbf{c}, \mathbf{c}') = (\mathbf{c} \cdot \mathbf{c}' + 1)^2,$$

where $\mathbf{c} \cdot \mathbf{c}' = \sum_{i=1}^{8} c_i c_i'$ counts the number of matching 1's between challenges $\mathbf{c}$ and $\mathbf{c}'$.

Mathematical Derivation

The explicit feature map $\tilde{\phi}(\mathbf{c})$ from the code includes:

- Linear terms: $c_1, c_2, \ldots, c_8$,
- Quadratic terms: $c_i c_j \quad (i \leq j)$.

The polynomial kernel $K(\mathbf{c}, \mathbf{c}')$ implicitly computes the inner product in a high-dimensional space spanned by all monomials up to degree 2:

$$K(\mathbf{c}, \mathbf{c}') = \langle \Phi(\mathbf{c}), \Phi(\mathbf{c}') \rangle,$$

where $\Phi(\mathbf{c})$ is the implicit feature map. Expanding $(\mathbf{c} \cdot \mathbf{c}' + 1)^2$ gives:

$$1 + 2(\mathbf{c} \cdot \mathbf{c}') + (\mathbf{c} \cdot \mathbf{c}')^2.$$

This corresponds to:

- Constant term: 1,
- Linear terms: $2 \sum_i c_i c_i'$,
- Quadratic terms: $\sum_{i \leq j} c_i c_j c_i' c_j'$.

Conclusion

The degree-2 polynomial kernel successfully replicates the 105-dimensional feature map derived earlier using linear and quadratic terms, without requiring explicit feature engineering.

Recommended kernel parameters:

- **Kernel type:** Polynomial
- **Degree:** 2
- **Gamma:** 1
- **Coef0:** 1

4 Recovering Non-Negative Delays for Arbiter PUF

4.1 Model Generation as a System of 65 Linear Equations

The model generation process for a 64-bit Arbiter PUF converts 256 delays p_i, q_i, r_i, s_i for $i = 0$ to 63 into a 65-dimensional linear model $(w_0, w_1, \ldots, w_{63}, b)$, where:

$$\begin{gathered} w_0 = \alpha_0 \\ w_i = \alpha_i + \beta_{i-1} \quad for \quad i = 1 to 63 \\ b = \beta_{63} \end{gathered}$$

with

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

This leads to 65 linear equations as previously described, with 256 variables, making it an underdetermined system.

4.2 Method to Recover Non-Negative Delays (Python-Compatible)

To obtain non-negative delays that regenerate the same model, we apply a recursive computation strategy derived from the decoding function:

4.2.1 Step 1: Base Case $i = 0$

Set:

$$p_0 = \max(2w_0, 0), \quad q_0 = \max(-2w_0, 0)$$
$$r_0 = s_0 = 0$$

4.2.2 Step 2: For $i = 1$ to 62

Compute the required intermediate sum from the previous index:

$$prev = p_{i-1} - q_{i-1} - r_{i-1} + s_{i-1}$$

Then, compute:

$$curr = 2w_i - prev$$

Set:

$$p_i = \max(curr, 0), \quad s_i = \max(-curr, 0)$$
$$q_i = r_i = 0$$

4.2.3 Step 3: Final Step $i = 63$

Compute:

$$prev = p_{62} - q_{62} - r_{62} + s_{62}$$
$$curr = w_{63} - \frac{prev}{2}$$

Use bias term $b = w_{64}$ to compute:

$$val1 = b + curr, \quad val2 = curr - b$$

Set:

$$p_{63} = \max(val1, 0), \quad q_{63} = \max(-val1, 0)$$
$$r_{63} = \max(val2, 0), \quad s_{63} = \max(-val2, 0)$$

4.3 Verification

Reconstruct the model as follows:

$$\alpha_i = \frac{p_i - q_i + r_i - s_i}{2}, \quad \beta_i = \frac{p_i - q_i - r_i + s_i}{2}$$
$$w_0 = \alpha_0, \quad w_i = \alpha_i + \beta_{i-1}, \quad b = \beta_{63}$$

This guarantees consistency with the original 65-dimensional linear model, while all delays p_i, q_i, r_i, s_i are non-negative.

5 Solving arbiter PUF inversion problem

Code has been uploaded for this.

6 Solving ML-PUF by learning linear model

Code has been uploaded for this.

7 Comparison between logistic regression and linear SVC

Following tables represents a comparison between the performance of logistic regression model with Linear SVC model:

Model	Loss Type	Training Time(in s)	Accuracy(in %)
Linear SVC	hinge	1	100
Linear SVC	squared hinge	4	100

Table 1: loss parameters

Varying loss hyperparameter across SVC

Varying C value or regularization change

Model	C value	Training Time(in s)	Accuracy(in %)
Linear SVC	2	3	100
Linear SVC	4	4	100
Linear SVC	6	6	100
Linear SVC	8	9	100
Logistic Regression	2	1	100
Logistic Regression	4	0	100
Logistic Regression	6	1	100
Logistic Regression	8	0	100

Table 2: C values

Varying tol or tolerance change

Model	Tolerance Value	Training Time(in s)	Accuracy(in %)
LinearSVC	1e-6	9	100
LinearSVC	1e-5	11	100
LinearSVC	1e-4	1	100
Logistic Regression	1e-6	0	100
Logistic Regression	1e-5	0	100
Logistic Regression	1e-4	0	100

Table 3: tol values

Varying penalty type

Model	penalty	Training Time(in s)	Accuracy(in %)
LinearSVC	l1	59	100
LinearSVC	l2	5	100
Logistic Regression	l1	8	100
Logistic Regression	l2	1	100

Table 4: penalty types