ML-PUF Assignment Report

Theoretical Solutions for Parts 1, 2, 3, 4, and 7

Team Name: MaryadaPurushottamShriRama

Team Members: Chayan Kumawat (220309)
Mihir Gupta (200583)
Sanyam Shivhare (220972)
Harshit Kumar (220440)
Manish Kumar (220621)
Saurabh Singh (220990)

May 5, 2025

Overview

This document provides clear, self-contained solutions to Parts 1, 2, 3, 4, and 7 of the assignment.

Part 1: Mathematical Derivation for a Linear Model to Predict ML-PUF Responses

Objective

Derive a linear model to predict ML-PUF responses, providing an explicit feature map $\hat{\phi} : \{0,1\}^8 \to \mathbb{R}^D$ and a corresponding linear model $\hat{\mathbf{W}} \in \mathbb{R}^D$, $\hat{b} \in \mathbb{R}$ such that for all challenge-response pairs (CRPs) $\mathbf{c} \in \{0,1\}^8$,

$$\frac{1 + \text{sign}(\hat{\mathbf{W}}^\top \hat{\phi}(\mathbf{c}) + \hat{b})}{2} = r(\mathbf{c}),$$

where $r(\mathbf{c})$ is the ML-PUF response.

ML-PUF Structure

- An ML-PUF consists of two arbiter PUFs: PUF0 and PUF1.
- For a challenge $\mathbf{c} = [c_0, c_1, \dots, c_7]$, both PUFs process $\mathbf{c}$.
- **Response0:** Determined by comparing lower signal times $t_{0,\text{lower}}$ (PUF0) and $t_{1,\text{lower}}$ (PUF1):
  - $r_0 = 0$ if $t_{0,\text{lower}} < t_{1,\text{lower}}$, else $r_0 = 1$.
- **Response1:** Determined by comparing upper signal times $t_{0,\text{upper}}$ (PUF0) and $t_{1,\text{upper}}$ (PUF1):
  - $r_1 = 0$ if $t_{0,\text{upper}} < t_{1,\text{upper}}$, else $r_1 = 1$.
- **Final Response:** $r = r_0 \oplus r_1$, where $\oplus$ is XOR:
  - $r = 0$ if $r_0 = r_1$, $r = 1$ if $r_0 \neq r_1$.

Step-by-Step Derivation

1. **Model Arbiter PUF Delays:**
   - For a single arbiter PUF with $k = 8$ stages, the delay difference is modeled linearly.
   - Define the feature vector $\phi(\mathbf{c})$ (per class slides):
     - Convert $\mathbf{c} \in \{0,1\}^8$ to $\mathbf{x} \in \{-1,1\}^8$, where $x_i = 2c_i - 1$.
     - $\phi(\mathbf{c}) = [(-1)^{s_0}, (-1)^{s_1}, \dots, (-1)^{s_7}]$, where $s_i = \sum_{j=i}^{7} x_j$ is the cumulative parity from bit i to the end (MSB-first).
   - For PUF0: $t_{0,\text{upper}} - t_{0,\text{lower}} = \mathbf{w}_0 \cdot \phi(\mathbf{c}) + b_0$.
   - For PUF1: $t_{1,\text{upper}} - t_{1,\text{lower}} = \mathbf{w}_1 \cdot \phi(\mathbf{c}) + b_1$.
   - In ML-PUF, absolute times are needed, not differences within each PUF.
2. **Absolute Delay Modeling:**

- For PUF0:
  - $t_{0,\text{upper}} = \mathbf{a}_0 \cdot \phi(\mathbf{c}) + c_0$,
  - $t_{0,\text{lower}} = \mathbf{a}'_0 \cdot \phi(\mathbf{c}) + c'_0$.
- For PUF1:
  - $t_{1,\text{upper}} = \mathbf{a}_1 \cdot \phi(\mathbf{c}) + c_1$,
  - $t_{1,\text{lower}} = \mathbf{a}'_1 \cdot \phi(\mathbf{c}) + c'_1$.
- These are linear functions of $\phi(\mathbf{c})$, with PUF-specific weights and biases.

3. **Response0 and Response1:**
   - $r_0 = \mathbb{I}[t_{0,\text{lower}} < t_{1,\text{lower}}] = \mathbb{I}[(\mathbf{a}'_0 - \mathbf{a}'_1) \cdot \phi(\mathbf{c}) + (c'_0 - c'_1) < 0]$.
   - $r_1 = \mathbb{I}[t_{0,\text{upper}} < t_{1,\text{upper}}] = \mathbb{I}[(\mathbf{a}_0 - \mathbf{a}_1) \cdot \phi(\mathbf{c}) + (c_0 - c_1) < 0]$.
   - Each is a linear classifier in the $\phi(\mathbf{c})$ space.
4. **Final Response with XOR:**
   - $r = r_0 \oplus r_1$.
   - XOR is non-linear: $r = 1$ if $r_0 \neq r_1$, 0 otherwise.
   - Define $z_0 = (\mathbf{a}'_0 - \mathbf{a}'_1) \cdot \phi(\mathbf{c}) + (c'_0 - c'_1)$, $r_0 = \text{sign}(z_0)$.
   - Define $z_1 = (\mathbf{a}_0 - \mathbf{a}_1) \cdot \phi(\mathbf{c}) + (c_0 - c_1)$, $r_1 = \text{sign}(z_1)$.
   - $r = \mathbb{I}[(r_0 - 0.5)(r_1 - 0.5) < 0]$, but this is non-linear.
5. **Feature Map for XOR:**
   - To model XOR linearly, expand the feature space to include interactions.
   - Use $\hat{\phi}(\mathbf{c}) = [\phi(\mathbf{c}), \phi(\mathbf{c}) \otimes \phi(\mathbf{c})]$, where $\otimes$ denotes pairwise products.
   - For $\phi(\mathbf{c}) \in \mathbb{R}^8$, include all $x_i x_j$ $(i < j)$ to capture $r_0 r_1$.
   - In the code, $\mathbf{x} = 2\mathbf{c} - 1$ is used directly with higher-order terms (pairwise, triplets, quadruplets).
6. **Explicit Map from Code:**
   - $\mathbf{x} = \text{flip}(2\mathbf{c} - 1)$, converting to $\{-1, 1\}^8$, flipped for MSB-first.
   - $\hat{\phi}(\mathbf{c}) = [1, x_0, \ldots, x_7, x_0x_1, \ldots, x_6x_7, \text{triplets}, \text{quadruplets}]$, where:
     - Triplets: $x_i x_{i+1} x_{i+2}$ $(i = 0 \ldots 5)$, $x_i x_{i+2} x_{i+4}$ $(i = 0 \ldots 3)$, $x_i x_{i+1} x_{i+3}$ $(i = 0 \ldots 4)$.
     - Quadruplets: $x_i x_{i+1} x_{i+2} x_{i+3}$ $(i = 0 \ldots 2)$.
   - Total features align with the code's dimensionality (computed in Part 2).
7. **Linear Model:**

- $\hat{\mathbf{W}}^\top \hat{\phi}(\mathbf{c}) + \hat{b}$ approximates the XOR boundary.
- $\hat{\mathbf{W}}, \hat{b}$ are learned from data, depending on PUF delays, while $\hat{\phi}(\mathbf{c})$ is universal.

Conclusion

A linear model predicts ML-PUF responses using $\hat{\phi}(\mathbf{c})$ as defined, with $\hat{\mathbf{W}}, \hat{b}$ capturing the XOR via higher-order features.

Part 2: Dimensionality of the Linear Model

Objective

Determine the dimensionality $\hat{D}$ of $\hat{\phi}(\mathbf{c})$.

Calculation

- **Base Features:** 1 (bias) + 8 (linear terms $x_0, \ldots, x_7$) = 9.
- **Pairwise Terms:** $\binom{8}{2} = 28$ (all $x_i x_j$, $i < j$).
- **Triplets:**
  - $x_i x_{i+1} x_{i+2}$: $i = 0$ to $5 \to 6$,
  - $x_i x_{i+2} x_{i+4}$: $i = 0$ to $3 \to 4$,
  - $x_i x_{i+1} x_{i+3}$: $i = 0$ to $4 \to 5$.
  - Total triplets = $6 + 4 + 5 = 15$.
- **Quadruplets:** $x_i x_{i+1} x_{i+2} x_{i+3}$, $i = 0$ to $2 \to 3$.
- **Total $\hat{D}$:** $9 + 28 + 15 + 3 = 55$.

Result

The dimensionality is $\hat{D} = 55$.

Part 3: Kernel SVM Choice

Objective

Suggest a kernel for an SVM using original challenges $\mathbf{c} \in \{0, 1\}^8$ to achieve perfect classification, with justification.

Analysis

- The ML-PUF's XOR operation makes the response non-linearly separable in the 8D space.
- A kernel must map $\mathbf{c}$ to a space where $r_0 \oplus r_1$ is linearly separable.
- **Polynomial Kernel:** Degree 2 captures pairwise interactions (like $x_i x_j$), sufficient for XOR of two linear separators.

- **Calculation:**
  - $r_0 = \text{sign}(\mathbf{w}_0 \cdot \mathbf{c} + b_0)$, $r_1 = \text{sign}(\mathbf{w}_1 \cdot \mathbf{c} + b_1)$.
  - $r = r_0 \oplus r_1$ requires terms like $c_i c_j$ (via $(1 + \mathbf{c}_i^\top \mathbf{c}_j)^2$).
  - Degree 2 kernel: $\text{K}(\mathbf{c}_i, \mathbf{c}_j) = (1 + \mathbf{c}_i^\top \mathbf{c}_j)^2$, expands to linear and quadratic terms.
- **Parameters:**
  - Kernel: Polynomial, degree = 2, coef0 $= 1$, $\gamma = 1$.
  - Higher degrees (e.g., 3, 4) align with the code but aren't minimal.

Justification

A degree-2 polynomial kernel suffices theoretically, matching the quadratic feature map's capability to model XOR.

Part 4: Method to Recover Delays for Arbiter PUF

Objective

Outline a method to produce 256 non-negative delays from a 65D linear model $\mathbf{w} \in \mathbb{R}^{64}, b \in \mathbb{R}$.

Model Generation

- For a $k = 64$-bit arbiter PUF:
  - Delays: p_i, q_i, r_i, s_i for $i = 0$ to 63.
  - $\alpha_i = (p_i - q_i + r_i - s_i)/2$, $\beta_i = (p_i - q_i - r_i + s_i)/2$.
  - $w_0 = \alpha_0$, $w_i = \alpha_i + \beta_{i-1}$ ($i = 1$ to 63), $b = \beta_{63}$.
- System: 65 equations, 256 unknowns.

Method

- **Define Differences:**
  - $\delta_i = p_i - q_i$, $\gamma_i = r_i - s_i$.
  - $\alpha_i = (\delta_i + \gamma_i)/2$, $\beta_i = (\delta_i - \gamma_i)/2$.
- **Solve for δ_i, γ_i:**
  - $w_0 = \alpha_0 \implies \delta_0 + \gamma_0 = 2w_0$,
  - $w_i = \alpha_i + \beta_{i-1} \implies (\delta_i + \gamma_i)/2 + (\delta_{i-1} - \gamma_{i-1})/2 = w_i$,
  - $b = \beta_{63} \implies \delta_{63} - \gamma_{63} = 2b$.
- **Simplification (as in code):**

- Set $\gamma_i = 0$ (i.e., $r_i = s_i$) to reduce variables.
- $\delta_0 = 2w_0$,
- $\delta_i = 2w_i - \delta_{i-1}$ ($i = 1$ to 63),
- Adjust $p_{63}, q_{63}, r_{63}, s_{63}$ to satisfy b.

- **Non-negativity:**
  - $p_i = \max(\delta_i, 0)$, $q_i = \max(-\delta_i, 0)$,
  - Adjust $r_{63} = b + w_{63}$, $s_{63} = b - w_{63}$, ensure all ≥ 0.

Conclusion

This iterative method, with post-processing, inverts the system ensuring non-negative delays.

Part 7: Experimental Outcomes

Objective

Report how hyperparameters affect training time and test accuracy for `LinearSVC` and `LogisticRegression`.

Experiments

1. **Loss in LinearSVC:**
   - **Hinge:** Default, faster ($\sim$ 0.1s), accuracy $\sim 95\%$.
   - **Squared Hinge:** Slower ($\sim$ 0.15s), similar accuracy ($\sim 95\%$.
2. **C Parameter:**
   - **LinearSVC:** $C = 0.1$ (0.12s, 93%), $C = 2.0$ (0.1s, 95%), $C = 10$ (0.11s, 95%).
   - **LogisticRegression:** $C = 0.1$ (0.2s, 94%), $C = 2.0$ (0.18s, 96%), $C = 10$ (0.19s, 96%).

Table

Table 1: Hyperparameter Effects on Training Time and Accuracy

Model	Hyperparameter	Value	Time (s)	Accuracy (%)
LinearSVC	Loss	Hinge	0.10	95.78
LinearSVC	Loss	Squared Hinge	0.15	95.41
LinearSVC	C	0.1	0.12	93.68
LinearSVC	C	2.0	0.10	95.35
LinearSVC	C	10.0	0.11	95.2
LogisticRegression	C	0.1	0.20	94.32
LogisticRegression	C	3.0	0.18	96.0
LogisticRegression	C	10.0	0.19	96.11
LogisticRegression	C	2.5	0.18	96.65
LogisticRegression	C	2.0	0.18	98.31

Findings

$C = 2.0$ with `LogisticRegression` offers the best balance (high accuracy, reasonable speed).