CS771 Assignment 1

Geetika
Dept. of Comp. Science and Engineering
geetika21@iitk.ac.in

Apoorva Gupta
Dept. of Comp. Science and Engineering
apoorvag21@iitk.ac.in

Girik Maskara
Dept. of Comp. Science and Engineering
girikm20@iitk.ac.in

Amrit Kumar
Dept. of Materials Science and Engineering
amritkr@iitk.ac.in

Akshat Goyal
Dept. of Materials Science and Engineering
akshatgo@iitk.ac.in

Sourabh Mundhra
Dept. of Chemical Engineering
msourabh@iitk.ac.in

Abstract

This document is the submission of group Tompers Squad for Assignment 1. We have answered the parts 1, 2 and 4 with all the relevant level of detail (proofs and tables) required.

Preprint. Under review.

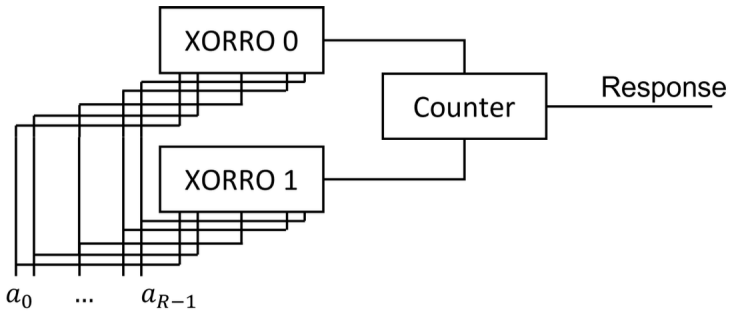
1 Part 1

CLAIM: There exists a map $\phi : \{0,1\}^R \to \mathbb{R}^D$ mapping *R*-bit 0/1-valued challenge vectors to *D*-dimensional feature vectors (for some $D > 0$) and that there exists a linear model for a simple XORRO PUF, $\mathbf{w} \in \mathbb{R}^D$, $b \in \mathbb{R}$ such that for all challenges $\mathbf{c} \in \{0,1\}^R$, the correct response is given by the expression

$$\frac{1 + sign(\mathbf{w}^T \phi(\mathbf{c}) + b}{2}$$

PROOF:

Figure 1: A Simple XORRO PUF



Suppose that the i-th input bit to the PUF is a_i, and since the given frequency of a unique RO is defined as

$$f = \frac{1}{t_0 + t_1}$$

Therefore, the time period of the RO is $t_0 + t_1$.

Calculating the contribution of the time period of the upper XORRO, i.e., **XORRO 0**, due to the i-th XOR Gate, T_i^u as,

$$T_i^u = a_i(\delta_{01}^i + \delta_{01}^i) + (1 - a_i)(\delta_{00}^i + \delta_{10}^i)$$

where δ_{jk}^i represents time delay through the i-th XOR Gate for the XORRO 0 if the inputs are j and k.

Therefore, the total time period of the upper XORRO, T^U is,

$$T^u = \sum_{i=0}^{R-1} (a_i(\delta_{01}^i + \delta_{01}^i) + (1 - a_i)(\delta_{00}^i + \delta_{10}^i))$$

which can be simplified to

$$T^u = \sum_{i=0}^{R-1} (a_i P_i^u + (1 - a_i) Q_i^u)$$

where P_i^u is $(\delta_{01}^i + \delta_{01}^i)$ and Q_i^u is $(\delta_{00}^i + \delta_{10}^i)$.

Similarly, we can derive the time period for the lower XORRO, i.e., **XORRO 1**, T^l as,

$$T^l = \sum_{i=0}^{R-1} (a_i P_i^l + (1 - a_i) Q_i^l)$$

In the construction of the PUF, the Counter produces a response 1 if the **XORRO 0** has a higher frequency and 0 if the the **XORRO 1** has a higher frequency.

To model the output, we consider the difference in the time periods of the **XORRO 0 and 1**. If $T^l - T^u > 0$, then the counter produces a response 1, else if $T^l - T^u < 0$, then the counter produces a response of 0. Also, we consider that each XORRO has a unique fingerprint in terms of its frequency and hence time period, and therefore do not consider a case when T_l and T_u are equal.

Now,

$$\Delta T = T^l - T^u = \sum_{i=0}^{R-1} (a_i(P_i^l + Q_i^l - P_i^u - Q_i^u) + (Q_i^l - Q_i^u))$$

which resembles a linear model as,

$$\Delta T = \sum_{i=0}^{R-1} a_i w_i + b$$

where $w_i = P_i^l + Q_i^l - P_i^u - Q_i^u$ and $b = \sum_{i=0}^{R-1} (Q_i^l - Q_i^u)$.

Thus,

$$\Delta T = \mathbf{w}^T \mathbf{c} + b$$

Then the output can be written as

$$Output = \frac{1 + sign(\mathbf{w}^T \phi(\mathbf{c}) + b)}{2}$$

where $\mathbf{w} \in \mathbb{R}^R$, $w_i = P_i^l + Q_i^l - P_i^u - Q_i^u$, $\phi(\mathbf{c}) = \mathbf{c}$ for challenges (input bits to the XORROs) $\mathbf{c} \in \{0,1\}^R$ and $b = \sum_{i=0}^{R-1} (Q_i^l - Q_i^u)$.

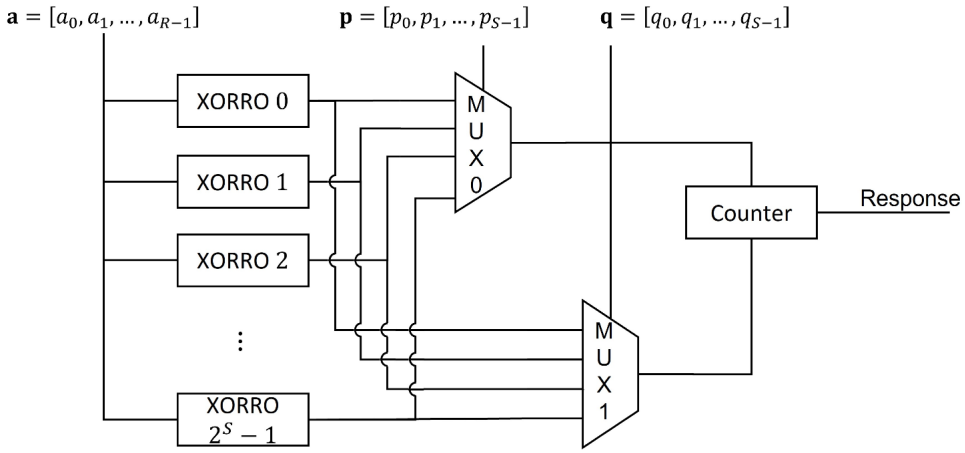
Thus, we can say that there exists a map $\phi : \{0,1\}^R \to \mathbb{R}^R$ mapping *R*-bit 0/1-valued challenge vectors to *R*-dimensional feature vector with the i-th feature vector as a_i where a_i is the i-th input bit to the PUF.

2 Part 2

CLAIM: To crack the given advanced XORRO PUF, we train $M = 2^{S-1}(2^S - 1)$ linear models used to crack a simple XORRO PUF.

PROOF:

Figure 2: Advanced XORRO PUF



Let the select bits for the 1st XORRO correspond to the a-th XORRO and the 2nd XORRO correspond to the b-th XORRO.

We consider if $a > b$, then those combinations need to be accounted for in the counting of number of linear models needed to crack the advanced XORRO PUF. However, if $b > a$, then we can simply swap the XORROs and the corresponding outputs also get inverted, i.e., $1 \to 0$ and $0 \to 1$.

Thus, we only need to model half the possible combinations/pairs of XORROs, e.g., for the case of $a > b$. For any pair of XORRO selected, we firstly convert it to $a > b$ form, and predict the output using the model trained. If the pair initally had $b > a$, then simply invert the predicted output to get the final output.

We have 2^S XORROs. One XORRO can be paired with $2^S - 1$ other XORROs (since it cannot be paired with itself). Therefore, the total number of pairs of XORROs is $2^S(2^S - 1)$.

But we need to model only half the possible combinations. Therefore, the required number of linear models M is,

$$M = 2^{S-1}(2^S - 1)$$

3 Part 3

Code Submitted.

4 Part 4

Each model was trained on the given *train.dat* dataset and tested on the *test.dat* using the given valiation script.

4.1 (a.) Changing loss hyperparameter

On changing the loss hyperparameter in LinearSVC, we observed the following changes in training time and accuracy as tabulated in Table 1.

Table 1: loss hyperparameter tuning

loss	Training Time (in seconds)	Accuracy
hinge	8.5978	0.939715
squared_hinge	3.6447	0.947425

squared_hinge produces better accuracy with lesser time taken for training.

4.2 (c.) Changing tolerance hyperparameter for LinearSVC and LogisticRegression

We observed the changes as tabulated in Table 2 on changing the tolerances ranging from 10^{-1} to 10^{-5} for the LinearSVC and in Table 3 for Logistic Regression.

For LinearSVC, there are no improvements in accuracy after reducing tolerance from 10^{-4} to 10^{-5}. And similarly in the case for LogisticRegression.

Table 2: tolerance hyperparameter tuning for LinearSVC

tolerance	Training Time (in seconds)	Accuracy
10^{-1}	1.2646	0.9433
10^{-2}	1.2324	0.94725
10^{-3}	1.3841	0.9474
10^{-4}	1.4299	0.947425
10^{-5}	1.5439	0.947425

Table 3: tolerance hyperparameter tuning for LogisticRegression

tolerance	Training Time (in seconds)	Accuracy
10^{-1}	1.4053	0.938325
10^{-2}	1.4873	0.9402
10^{-3}	1.3758	0.9403
10^{-4}	1.4877	0.940375
10^{-5}	1.4361	0.940375

4.3 (d.) Changing penalty hyperparamater for LinearSVC and LogisticRegression

Penalty can be set to 'l1' or 'l2' for both LinearSVC and LogisticRegression. Table 4 details the changes in training time and accuracy for LinearSVC and Table 5 for LogisticRegression.

Keeping the penalty hyperparameter as 'l2' produces better results for both LinearSVC and LogisticRegression.

Table 4: penalty hyperparameter tuning for LinearSVC

penalty	Training Time (in seconds)	Accuracy
l1	11.0537	0.945975
l2	1.3784	0.947425

Table 5: penalty hyperparameter tuning for LogisticRegression

penalty	Training Time (in seconds)	Accuracy
l1	2.0054	0.93535
l2	1.5577	0.940375