CS771 Assignment 1

The Brainiacs

Aishwarya Srivastava (200064)

Akanksha Singh (200070)

Anushka Panda (200174)

Kshiteesh Bhardwaj (200525)

Raj Vardhan Singh (200760)

1 The One with the Simple XORRO PUF

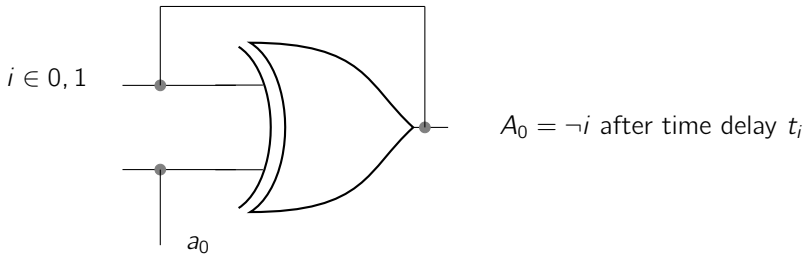
1.1 Notations

For the kth XOR gate in the XORRO 0, we take δ_{ij}^{k} as the delay for the case where we wish to flip j and a_k, $i, j \in \{0, 1\}$ and a_k meaning the other input to the XOR gate. A_k stands for the output of the kth XOR gate. For the XORRO 1, the delay is denoted as $\delta_{ij}^{k\prime}$.

The total delay for flipping the value of $i \in \{0, 1\}$ in the XORRO 0, if it contains $k + 1$ XORs, is denoted as t_i^{k+1}, and for the XORRO 1 is $t_i^{k+1\prime}$. The subsequent frequencies are f^{k+1} and $f^{k+1\prime}$

1.2 Proof of the required statement

1.2.1 Case 1: XORRO with only one XOR gate



Preprint. Under review.

In this case the time delay values t_0 and t_1 can be easily calculated for both the XORROs in question as follows.

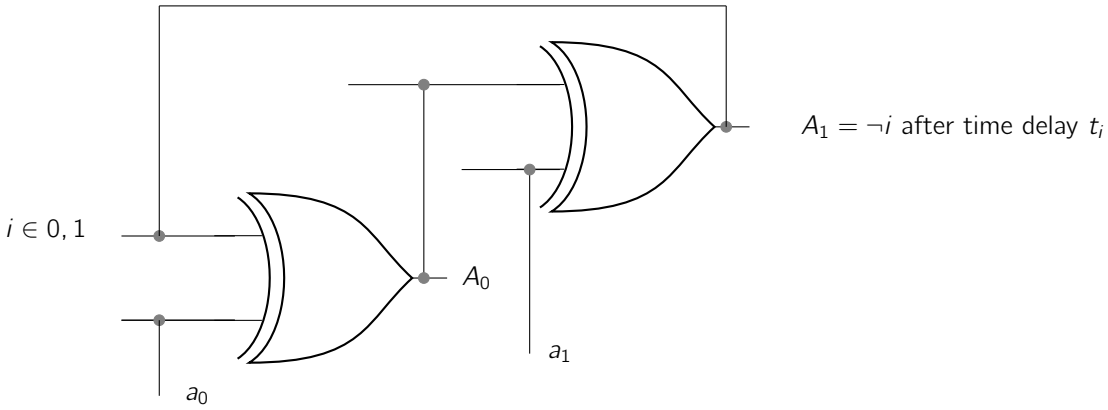
$$t_0^1 = (1 - a_0)\delta_{00}^0 + a_0\delta_{01}^0 \tag{1}$$

$$t_1^1 = a_0\delta_{11}^0 + (1 - a_0)\delta_{10}^0 \tag{2}$$

From (1) and (2),

$$\begin{aligned} t_0^1 + t_1^1 &= (1 - a_0)\delta_1^1 + a_0\delta_1^1 \\ &\quad + a_0\delta_1^1 + (1 - a_0)\delta_1^1 \\ &= \delta_1^1 + \delta_1^1 + a_0(\delta_1^1 + \delta_1^1 - \delta_1^1 - \delta_1^1) \end{aligned} \tag{3}$$

1.2.2 Case 2: XORRO with two XOR gates



We notice that the value of A_0 is a_0 when $i = 0$ and $\neg a_0$ when $i = 1$.

A_1, of course, depends on A_0 and a_1. So in this case,

$$\begin{aligned} t_0^2 &= (1 - a_0)\delta_{00}^0 + a_0\delta_{01}^0 \\ &\quad + (1 - a_1)(1 - a_0)\delta_{00}^1 + (1 - a_0)a_1\delta_{01}^1 \\ &\quad + (1 - a_1)a_0\delta_{10}^1 + a_0a_1\delta_{11}^1 \end{aligned} \tag{4}$$

$$\begin{aligned} t_1^2 &= a_0\delta_{11}^0 + (1 - a_0)\delta_{10}^0 \\ &\quad + (1 - a_1)a_0\delta_{00}^1 + a_0a_1\delta_{01}^1 \\ &\quad + (1 - a_1)(1 - a_0)\delta_{10}^1 + (1 - a_0)a_1\delta_{11}^1 \end{aligned} \tag{5}$$

From (4) and (5),

$$
\begin{aligned}
t_0^2 + t_1^2 &= (1-a_0)\delta_{00}^0 + a_0\delta_{01}^0 \\
&\quad + (1-a_1)(1-a_0)\delta_{00}^1 + (1-a_0)a_1\delta_{01}^1 \\
&\quad + (1-a_1)a_0\delta_{10}^1 + a_0a_1\delta_{11}^1 \\
&\quad + a_0\delta_{11}^0 + (1-a_0)\delta_{10}^0 \\
&\quad + (1-a_1)(a_0)\delta_{00}^1 + (a_0)a_1\delta_{01}^1 \\
&\quad + (1-a_1)(1-a_0)\delta_{10}^1 + (1-a_0)a_1\delta_{11}^1 \\
&= t_0^1 + t_1^1 \\
&\quad + (1-a_1)(1-a_0)\delta_{00}^1 + (1-a_0)a_1\delta_{01}^1 \\
&\quad + (1-a_1)a_0\delta_{10}^1 + a_0a_1\delta_{11}^1 \\
&\quad + (1-a_1)a_0\delta_{00}^1 + a_0a_1\delta_{01}^1 \\
&\quad + (1-a_1)(1-a_0)\delta_{10}^1 + (1-a_0)a_1\delta_{11}^1 \\
&= t_0^1 + t_1^1 \\
&\quad + \delta_{00}^1 - a_1\delta_{00}^1 - a_0\delta_{00}^1 + a_0a_1\delta_{00}^1 \\
&\quad + a_1\delta_{01}^1 - a_0a_1\delta_{01}^1 + a_0\delta_{10}^1 - a_0a_1\delta_{10}^1 + a_0a_1\delta_{11}^1 \\
&\quad + a_0\delta_{00}^1 - a_0a_1\delta_{00}^1 + a_0a_1\delta_{01}^1 \\
&\quad + \delta_{10}^1 - a_1\delta_{10}^1 - a_0\delta_{10}^1 + a_0a_1\delta_{10}^1 + a_1\delta_{11}^1 - a_0a_1\delta_{11}^1 \\
&= t_0^1 + t_1^1 \\
&\quad + \delta_{00}^1 + \delta_{10}^1 + a_1(\delta_{01}^1 + \delta_{11}^1 - \delta_{11}^1 - \delta_{10}^1) \\
&= \sum_{i=0}^{1}(\delta_{00}^i + \delta_{10}^i + a_i(\delta_{01}^i + \delta_{11}^i - \delta_{10}^i - \delta_{00}^i))
\end{aligned}
\tag{6}
$$

1.2.3 Case 3: XORRO with three XOR gates

We notice that A_0 remains the same, and A_2 depends on a_2 and A_1, and A_1 is equal to $a_0 + a_1 - 2a_0a_1$.

Continuing in a similar fashion, we obtain the following values of t_0, t_1 and $t_0 + t_1$:

$$\begin{aligned} t_0^3 &= (1-a_0)\delta_{00}^0 + a_0\delta_{01}^0 \\ &+ (1-a_1)(1-a_0)\delta_{00}^1 + (1-a_0)a_1\delta_{01}^1 \\ &+ (1-a_1)a_0\delta_{10}^1 + a_0a_1\delta_{11}^1 \\ &+ (1-a_0-a_1+2a_0a_1)(1-a_2)\delta_{00}^2 \\ &+ (1-a_0-a_1+2a_0a_1)a_2\delta_{01}^2 \\ &+ (a_0+a_1-2a_0a_1)(1-a_2)\delta_{10}^2 \\ &+ (a_0+a_1-2a_0a_1)a_2\delta_{11}^2 \end{aligned} \tag{7}$$

$$\begin{aligned} t_1^3 &= a_0\delta_{11}^0 + (1-a_0)\delta_{10}^0 \\ &+ (1-a_1)(a_0)\delta_{00}^1 + (a_0)a_1\delta_{01}^1 \\ &+ (1-a_1)(1-a_0)\delta_{10}^1 + (1-a_0)a_1\delta_{11}^1 \\ &+ (a_0+a_1-2a_0a_1)(1-a_2)\delta_{00}^2 \\ &+ (a_0+a_1-2a_0a_1)a_2\delta_{01}^2 \\ &+ (1-a_0-a_1+2a_0a_1)(1-a_2)\delta_{10}^2 \\ &+ (1-a_0-a_1+2a_0a_1)a_2\delta_{11}^2 \end{aligned} \tag{8}$$

And we get $t_0^3 + t_1^3$ by summing (7) + (8), opening the brackets and cancelling the terms.

$$t_0^3 + t_1^3 = \sum_{i=0}^{2}(\delta_{00}^i + \delta_{10}^i + a_i(\delta_{01}^i + \delta_{11}^i - \delta_{10}^i - \delta_{00}^i)) \tag{9}$$

1.2.4 Generalization: XORRO with R XOR gates

Finally we start seeing a pattern for the XORRO time delay. If an arbitrary PUF contains R number of XOR gates, then the time delay $t_0^R + t_1^R$ is given by ... (drumrolls)

$$\begin{aligned} t_0^R + t_1^R &= \sum_{i=0}^{R-1}(\delta_{00}^i + \delta_{10}^i + a_i(\delta_{01}^i + \delta_{11}^i + \delta_{10}^i + \delta_{00}^i)) \\ &= \sum_{i=0}^{R-1}(\delta_{00}^i + \delta_{10}^i) + \sum_{i=0}^{R-1}(\delta_{01}^i + \delta_{11}^i - \delta_{10}^i - \delta_{00}^i)a_i) \\ &= b' + \mathbf{w'}^T\mathbf{c} \end{aligned} \tag{10}$$

where $b' = \sum_{i=0}^{R-1}(\delta_{00}^i + \delta_{10}^i)$,

$$\mathbf{w'} \in \mathbb{R}^{R\times 1} = \begin{bmatrix} \delta_{01}^0 + \delta_{11}^0 - \delta_{10}^0 - \delta_{00}^0 \\ \vdots \\ \delta_{01}^{R-1} + \delta_{11}^{R-1} - \delta_{10}^{R-1} - \delta_{00}^{R-1} \end{bmatrix}$$

and

$$\mathbf{c} \in \mathbb{R}^{R\times 1} = \begin{bmatrix} a_0 \\ \vdots \\ a_{R-1} \end{bmatrix}$$

1.2.5 Finale

We now compute the XORRO PUFs challenge-response prediction. The challenge, $\mathbf{c}$, will of course be R bits long, and the response will be 1 if the XORRO 0 reaches first, and 0 if the XORRO 1 makes it before.

The sign of the time difference, will be key in predicting the response. Hence,

$$\begin{aligned}
(t_0 + t_1) - (t'_0 + t'_1) &= (\sum_{i=0}^{R-1}(\delta^i_{00} + \delta^i_{10}) + \sum_{i=0}^{R-1}(\delta^i_{01} + \delta^i_{11} - \delta^i_{10} - \delta^i_{00})a_i)) \\
&\quad - \sum_{i=0}^{R-1}(\delta^{i\prime}_{00} + \delta^{i\prime}_{10}) + \sum_{i=0}^{R-1}(\delta^{i\prime}_{01} + \delta^{i\prime}_{11} - \delta^{i\prime}_{10} - \delta^{i\prime}_{00})a_i) \\
&= \sum_{i=0}^{R-1}(\delta^i_{00} + \delta^i_{10} - \delta^{i\prime}_{00} - \delta^{i\prime}_{10}) \\
&\quad + \sum_{i=0}^{R-1}(\delta^i_{01} + \delta^i_{11} - \delta^i_{10} - \delta^i_{00} \\
&\quad - \delta^{i\prime}_{01} - \delta^{i\prime}_{11} + \delta^{i\prime}_{10} + \delta^{i\prime}_{00})a_i \\
&= b + \mathbf{w}^T\mathbf{c}
\end{aligned} \tag{11}$$

where $b = \sum_{i=0}^{R-1}(\delta^i_{00} + \delta^i_{10} - \delta^{i\prime}_{00} + \delta^{i\prime}_{10})$,

$$\mathbf{w} \in \mathbb{R}^{R\times 1} = \begin{bmatrix} \Delta_0 \\ \vdots \\ \Delta_{R-1} \end{bmatrix}$$

and

$$\Delta_i = \delta^i_{01} + \delta^i_{11} - \delta^i_{10} - \delta^i_{00} - \delta^{i\prime}_{01} - \delta^{i\prime}_{11} + \delta^{i\prime}_{10} + \delta^{i\prime}_{00}, i \in [0, R-1]$$

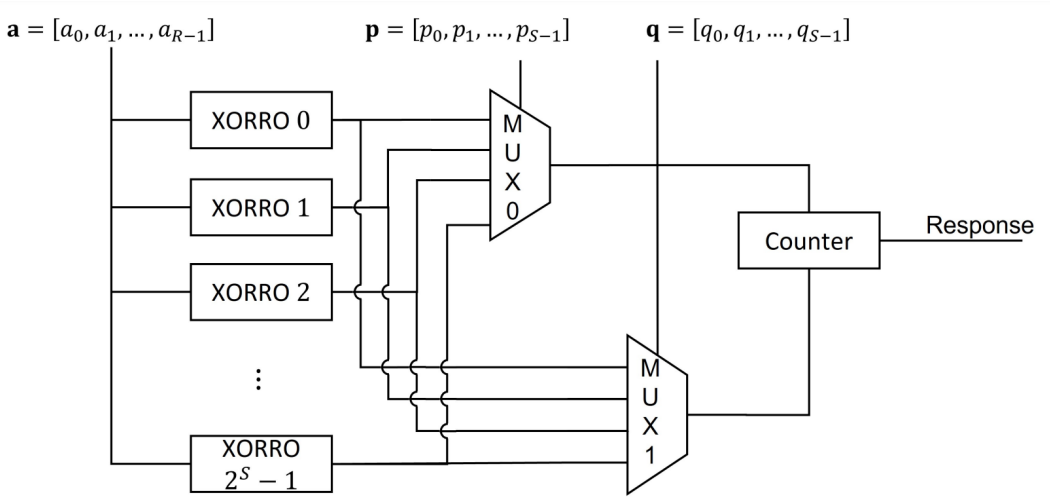
We realise there is no need for any further modification of the challenge vector, since it is giving a linear function as it is. Hence, we take $R = D$ and $\phi(\mathbf{c}) = \mathbf{c}$.

So the signum$(b + \mathbf{w}^T\mathbf{c})$ gives 1 if $t_0 + t_1 > t'_0 + t'_1$, and -1 if $t_0 + t_1 < t'_0 + t'_1$. With the assumption that one XORRO will always be late, we neglect the possibility of $t_0 + t_1 = t'_0 + t'_1$. Hence the function $y(\mathbf{c}) = \frac{1+sign(b+\mathbf{w}^T\mathbf{c})}{2}$ will give 1 if XORRO 0 arrives first and 0 otherwise. Hence proven.

2 The One with the Advanced XORRO PUF

2.1 Description of the Problem

We have 2^S XORROs and $2S$ select bits. The first S of these select bits is interpreted as a number between 0 and $2^S - 1$ and used to select the corresponding XORRO as the first XORRO. This selection is done by using a multiplexer. The next S select bits are used to select another XORRO as the second XORRO. The XORROs are numbered from 0 to $2^S - 1$. The frequencies of the two selected XORROs are compared using a counter. If the XORRO selected by the upper MUX has a higher frequency, the counter outputs 1 else if the XORRO selected by the lower MUX has a higher frequency, the counter outputs 0. The output of the counter is our response to the challenge.



2.2 Our Approach

The data has been provided to us as a table of tuples corresponding to challenge-response pairs, p and q. We store the indices to challenge-response pairs corresponding to the p bits and q bits in a dictionary along with the corresponding decimal values of p and q bits, say P and Q as the key. With training data of each (P, Q) we append a trained model based on the same data to the object associated with the tuple.

Given $S = 4$, we have 16 XORROS and 8 select bits in the advanced XORRO PUF. In the training data matrix, the first 64 columns contain the configuration bits, the next 4 columns contain the select bits for MUX 0 and the next 4 columns contain the select bits for MUX 1. The last column contains the response.

The first 64 + 4 + 4 = 72 columns constitute the challenge. We split the training data into challenges, select bits and responses. Then we create a dictionary, `train_dict` to store the indices of the corresponding training data for each XORRO. While iterating through the training data, we only traverse the upper triangular region of the matrix. It has been provided to us that the two XORRO PUFs chosen will never be the same. Hence if we were to declare a model for each possible pair of XORRO PUFs, we would get total of $\frac{(2^S)^2-(2^S)}{2} = 2^{S-1}(2^S - 1) = 8 \times 15 = 120$ models. We neglect the lower triangular region since it would have the same results, albeit flipped, as the upper triangular region.
We convert the p and q select bits to decimal value p_val and q_val respectively selecting the corresponding XORRO PUFs X and Y as input. We create a challenge response pair, append the `crp` to the training matrix and store index of the training data for each XORRO PUF pair.

Using the index, we take the training data for each XORRO PUF pair, we create a model using `LinearSVC` (using hinge loss function), train the model and store it in the list `models`.

We define `my_predict` function to make predictions on test challenges. We split the test data into challenges: `X_test` and select bits: `p, q` and predict the responses for each challenge.
Since the statement proven in Q1 holds for any arbitrary pair of XORRO PUFs, no matter what is the output of MUX0 and MUX1, a linear model will always exist for the corresponding (XORRO $P-1$, XORRO $Q-1$).

3 The One with the Code

...can be checked out in `submit.py :)`

4 The One with the Results

4.1 LinearSVC

4.1.1 Varying c

c	t_train	t_test	acc
100	1.5605297256000086	3.862231476799957.0	0.9393499999999999
1	1.6651930728000253	3.72529280699996	0.939485
0.01	1.5715244404000033	3.911262119600019	0.9396149999999999

4.1.2 Varying the loss function

LossFunction	t_train	t_test	acc
Hinge loss	1.6651930728000253	3.72529280699996	0.939485
Squared Hinge loss	1.4972170794000248	3.7503795944000102	0.93963

4.1.3 With v.s. Without Penalty

Please note that this calculation was made with the squared-hinge loss function.

Penalty	t_train	t_test	acc
Without	1.4972170794000248	3.7503795944000102	0.93963
With	1.6864013043998967	3.9705344659999353	0.9396850000000001

4.2 LinearRegression

4.2.1 Varying c

c	t_train	t_test	acc
100	1.5778111356000408	3.767631881200077	0.9396800000000001
1	1.6063363011999627	3.7439174388000085	0.939525
0.01	1.7413894152000466	4.079124166799966	0.9396850000000001

4.2.2 Varying the loss function

Loss Function	t_train	t_test	acc
Hinge loss	1.6063363011999627	3.7439174388000085	0.939525
Squared Hinge loss	1.6551792108000427	3.968126361800023	0.93955

4.2.3 With v.s. Without Penalty

Penalty	t_train	t_test	acc
Without	1.6063363011999627	3.7439174388000085	0.939525
With	1.7621921828000269	3.7856324582000527	0.939405