CS771 ASSIGNMENT 1

Team : Veg_Korma

Ashutosh Agrawal (210219)

Hitesh Anand (200449)

Pratham Jain (200712)

Priya Gole (200727)

Shambhavi Sabharwal (200914)

Tanush Kumar (201043)

1 Problem - 1

Let the XOR gates in $XORRO_0$ be denoted as $\chi_1, \chi_2, ...\chi_R$. Then for χ_i, we have

S_i : i^{th} select bit
I_i : Input to the i^{th} XOR Gate (output of the $i-1^{th}$ gate for $i \neq 1$)
O_i : Output of the i^{th} XOR Gate

Then the time delay of the different cases can be tabulated as follows:

S_i	I_i	O_i	Time delay
1	1	0	p_i
0	1	1	q_i
1	0	1	r_i
0	0	0	s_i

Therefore the total time delay Δt_i across χ_i for given I_i, S_i can be given by

$$\Delta t_i = S_i I_i P_i + (1-S_i) I_i q_i + S_i(1-I_i) r_i + (1-S_i)(1-I_i) s_i \tag{1}$$

Now it is given that $\sum_{i=1}^{R} S_i$ is odd. Therefore, each of the inputs to every XOR gate χ_i gets flipped after one pass of the input through the $XORRO$.
Therefore, the total time passed to one cycle of $XORRO_0$ is given by

$$\begin{aligned} T &= \sum_{i=1}^{R} \Delta t_i|_{I_1=0} + \sum_{i=1}^{R} \Delta t_i|_{I_1=1} \\ &= \sum_{i=1}^{R} (\Delta t_i|_{I_1=0} + \Delta t_i|_{I_1=1}) \\ &= \sum_{i=1}^{R} (\Delta t_i|_{I_i=0} + \Delta t_i|_{I_i=1}) \end{aligned}$$

From (1)

$$\begin{aligned}\Delta t_i|_{I_i=0} + \Delta t_i|_{I_i=1} &= s_i r_i + (1-S_i)s_i + S_i P_i + (1-S_i)q_i \\ &= S_i(r_i - s_i + p_i - qi) + s_i + q_i \\ \implies T &= S_i(r_i - s_i + p_i - qi) + s_i + q_i\end{aligned}$$

Let $\alpha_i = r_i - s_i + p_i - q_i$ and $\beta_i = s_i + q_i$
Hence,

$$\begin{aligned}\Delta t_i|_{I_i=0} + \Delta t_i|_{I_i=1} &= S_i\alpha_i + \beta_i \\ \implies T &= \sum_{i=1}^{R}(S_i\alpha_i + \beta_i)\end{aligned} \tag{2}$$

α_i and β_i will be different for different $XORRO$. For $XORRO_0$, let them be α_{i_0} and β_{i_0} and for $XORRO_1$, let them be α_{i_1} and β_{i_1} for $0 \le i \le R$. Therefore the time periods of the two $XORRO$ are:

$$\begin{aligned}T_{XORRO_0} &= \sum_{i=1}^{R}(S_i\alpha_{i_0} + \beta_{i_0}) \\ T_{XORRO_1} &= \sum_{i=1}^{R}(S_i\alpha_{i_1} + \beta_{i_1})\end{aligned} \tag{3}$$

Let the response of the counter be denoted by y. Then we can model y as follows:

$$\implies y = \begin{cases} 0 & if \quad T_{Xorro_0} > T_{Xorro_1} \\ 1 & if \quad T_{Xorro_0} < T_{Xorro_1} \end{cases} \implies y = \begin{cases} 0 & if \quad T_{Xorro_1} - T_{Xorro_0} < 0 \\ 1 & if \quad T_{Xorro_1} - T_{Xorro_0} > 0 \end{cases}$$

$$\implies y = \frac{1 + sign(T_{Xorro_1} - T_{Xorro_0})}{2} \tag{4}$$

Now,

$$T_{Xorro_1} - T_{Xorro_0} = \sum_{i=1}^{R}(S_i\alpha_{i_1} + \beta_{i_1}) - \sum_{i=1}^{R}(S_i\alpha_{i_0} + \beta_{i_0}) = \sum_{i=1}^{R} S_i\alpha_{i_1} + b_1 - \sum_{i=1}^{R} S_i\alpha_{i_0} - b_0 \tag{5}$$

Let **w** be a $2(R+1) \times 1$ dimensional vector given by:

$$\mathbf{w} = [\alpha_{1_1}, \alpha_{2_1}, ...\alpha_{R_1}, b_1, \alpha_{1_0}, \alpha_{2_0}, ...\alpha_{R_0}, b_0]^T \tag{6}$$

Then **w** is our parameter vector that is to be learnt .
Defining the map:

$$\begin{aligned}\phi : \{0,1\}^R \to \{-1,0,1\}^{2(R+1)} \\ \phi(\mathbf{c}) = [\phi_1(\mathbf{c}), \phi_2(\mathbf{c}), ...\phi_R(\mathbf{c}), \phi_{R+1}(\mathbf{c}), ...\phi_{2R+2}(\mathbf{c})]\end{aligned}$$

where **c** is the challenge vector. The component wise functions can be defined as:

$$\phi_i(c) = \begin{cases} c_i & i \le R \\ 1 & i = R+1 \\ -c_i & R+1 \le i \le 2R+1 \\ -1 & i = 2R+2 \end{cases} \tag{7}$$

Then by (5), (6) and (7):

$$\mathbf{w}^T\phi(\mathbf{c}) = T_{XORRO_1} - T_{XORRO0}$$

and hence;

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

2 Problem - 2

We can extend the the definition of the above function to crack the advanced $XORRO$ PUF as follows:
Since there are S machines, there are $(R+1)*S$ parameters involved.
For each machine i, we can define the function ϕ_i as follows:

$$\phi_i : \{0,1\}^R \rightarrow \{-1,0,1\}^{R+1}$$
$$\phi_i(\mathbf{c}) = [\phi_{i_1}(\mathbf{c}), \phi_{i_2}(\mathbf{c}), ...\phi_{i_R}(\mathbf{c}), \phi_{i_R+1}(\mathbf{c})]$$

$$\phi_{i_j}(\mathbf{c}) = \begin{cases} m_i(\mathbf{X})c_j & 1 \le j \le R \\ m_i(\mathbf{X}) & j = R+1 \end{cases}$$

where $m_i(\mathbf{X})$ is defined as :

$$m_i : \{0,1\}^4 \rightarrow \{-1,0,1\}$$

$m_i(\mathbf{X})$ takes in the input to the multiplexers and outputs either 1, -1 or 0 according to the following definition:

$$m_i(\mathbf{X}) = \begin{cases} 1 & \text{if machine } i \text{ is chosen by multiplexer 0} \\ -1 & \text{if machine } i \text{ is chosen by multiplexer 1} \\ 0 & \text{otherwise} \end{cases}$$

Thus we can define ϕi in a similar fashion $\forall i \le$ S. Then we can concatenate these functions into one single function Φ:

$$\Phi : \{0,1\}^R \rightarrow \{-1,0,1\}^{(R+1)*S}$$
$$\Phi = [\phi_{1_1}(\mathbf{c}), \phi_{1_2}(\mathbf{c}), ...\phi_{1_R+1}(\mathbf{c}), \phi_{2_1}(\mathbf{c}), ...\phi_{S_(R+1)}(\mathbf{c})]$$

We can extend the definition of the **w** from the previous part to this case. **w** is a $(R+1)*S \times 1$ dimensional vector of the following form:

$$\mathbf{w} = [\alpha_{1_1}, \alpha_{2_1}, ...\alpha_{R_1}, b_1, \alpha_{1_2}, \alpha_{2_2}, ...\alpha_{R_2}, b_2, ...\alpha_{R_{S+1}}, b_S]^T$$

Like in the previous part, $\mathbf{w}^T\ \Phi(\mathbf{c})$ models the difference in time periods of the lower $XORRO$ and upper $XORRO$. Therefore, the response y can be predicted using a linear model of the following form:

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

3 Problem- 4

(a) Effect of changing the loss hyper-parameter in LinearSVC (hinge vs squared hinge)

LinearSVC		
Loss Hyper-parameter Value	Training Time (in s)	Testing Accuracy (out of 1)
hinge	5.166872372	0.99075
squared hinge	5.166872372	0.99075

(b) Effect of setting C hyper-parameter in LinearSVC and LogisticRegression to high/low/medium values

LinearSVC		
C Hyper-parameter	Training Time (in s)	Testing Accuracy (out of 1)
0.1	3.481858236999983	0.9872
1	4.854951405000008	0.9907
10	4.706705581999984	0.99165
50	3.852456726000014	0.99125
100	4.110484468999971	0.9908
500	3.7851457189999564	0.990325
1000	4.507158651000054	0.988875

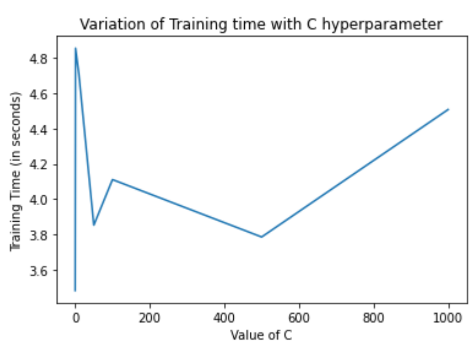

Figure 1: Effect of C hyper-parameter on training time in LinearSVC

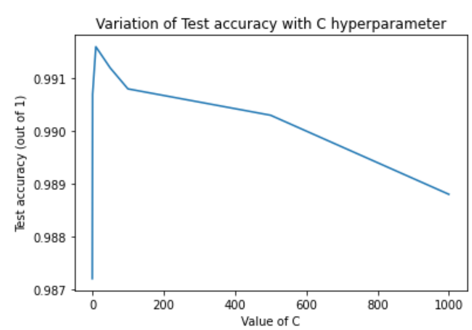

Figure 2: Effect of C hyper-parameter on test accuracy in LinearSVC

Logistic Regression		
C Hyper-parameter	Training Time (in s)	Testing Accuracy (out of 1)
0.1	10.309898443000066	0.987525
1	10.309898443000066	0.987525
10	11.053653662999977	0.991475
50	10.977767763999964	0.992325
100	11.88431051100008	0.99265
500	10.7822257790001	0.99285
1000	11.49991173400008	0.993

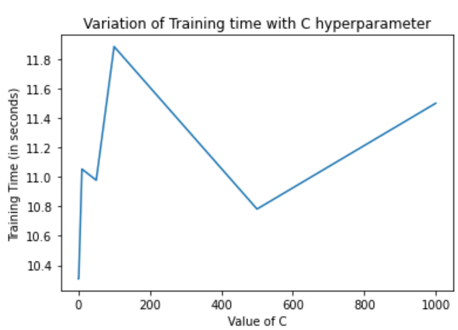

Figure 3: Effect of C hyper-parameter on training time in logistic regression

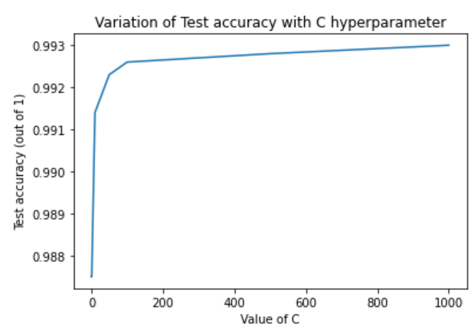

Figure 4: Effect of C hyper-parameter on test accuracy in logistic regression

(c) Effect of changing the penalty (regularization) hyperparameter in LinearSVC and LogisticRegression (L2 vs L1)

LinearSVC (C=10,dual = False)		
Penalty Hyperparameter	Training Time (in s)	Testing Accuracy (out of 1)
L1	32.27752847800002	0.99255
L2	2.6447267930007	0.99245

Since dual=True doesn't support L1 Loss in LinearSVC, we have dual=false for comparison

Logistic Regression (C=1000)		
Penalty Hyperparameter	Training Time (in s)	Testing Accuracy (out of 1)
L1 (solver='saga')	161.513066106	0.990325
L2	11.49991173400008	0.993

Since L1 penalty doesn't work with the default solver in Logistic Regression, we have used saga solver for L1

Observation : LinearSVC on average takes about 10 seconds less to train. However, the maximum accuracy we could achieve using LinearSVC was about 99.16%. LogisticRegression, on the other hand, took more training time, but the maximum accuracy we could get was around 99.3%. Hence, there was a slight tradeoff in the training time and testing accuracy on switching from one model to another. Finally, we chose the LogisticRegression model in our final submission giving more weightage to the test accuracy.