CS771: Assignment 1

Panshul Rastogi 190586
Varun Soni 190949
Parth Bhardwaj 190589
Sanket Dhananjay Kale 190760
Mirge Saurabh Arun 190498
Pulak Gautam 210791

Abstract

The following document comprises of submission of the group **The Questionable Bunch** for Assignment 1 of the course CS771 involving cracking advanced XORRO PUF. In essence, we have used $\binom{n}{2}$ models and hyperparameter tuning to achieve about 97% accuracy (where n is the total number of XORROs involved).
Every section number corresponds to that respective question of the assignment

1

By giving a detailed mathematical derivation (as given in the lecture slides), show how a simple XORRO PUF can be broken by a single linear model. Recall that the simple XORRO PUF has just two XORROs and has no select bits and no multiplexers (see above figure and discussion on Simple XORRO PUF). Thus, the challenge to a simple XORRO PUF has just R bits. More specifically, give derivations for a map $\phi : \{0,1\}^R \mapsto \mathbb{R}$ mapping R-bit 0/1-valued challenge vectors to D-dimensional feature vectors (for some D > 0) and show that for any simple XORRO PUF, there exists a linear model.
Let us first consider the case of a simple XORRO PUF case.

The (oscillating) outputs of the two XORROs is fed into a counter which can find out which XORRO has a higher frequency. If the upper XORRO (in this case XORRO 0) has a higher frequency, the counter outputs 1 else if the lower XORRO (in this case XORRO 1) has a higher frequency, the counter outputs 0.
Frequency of a XORRO PUF is defined as:

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

where, t_0 and t_1 are the times the XORROs take flip the value of O/P while being at equal to 0 and 1 respectively. Since, frequency is inversely proportional to time difference. We determine the sign of time difference (between two XORROs), and hence determine the sign of frequencies (which would be the opposite).
After careful observation, we concluded that the time delay in passing of signal from i^{th} XOR gate can be expressed in terms of r_i (first input bit of i_{th} XOR gate) and a_i (second input bit of i^{th} XOR gate **i.e. challenge bit** c_i) as follows:

$\Delta D(i) = a_i(r_i{\delta_{11}}^i + (1-r_i){\delta_{01}}^i) + (1-a_i)((r_i{\delta_{10}}^i + (1-r_i){\delta_{00}}^i)$

$\Rightarrow \Delta D(i) = a_i r_i({\delta_{11}}^i + {\delta_{00}}^i - {\delta_{01}}^i - {\delta_{10}}^i) + a_i({\delta_{01}}^i - {\delta_{00}}^i) + r_i({\delta_{10}}^i - {\delta_{00}}^i) + {\delta_{00}}^i$
In two consecutive oscillations i.e. in time $t_0 + t_1$, every XOR gate involved in the XORRO gets the opposite first input. More formally, one can say that if r_i is 1 then in the next oscillation when O/P values inverts, r_i would be zero, and vice versa.

Preprint. Under review.

So the total delay in a complete cycle for in passing through i^{th} gate, would be adding the ΔD with $r_i = 0$ and $r_i = 1$ which gives:

$\Delta D(i) = a_i(\delta_{11}{}^i + \delta_{01}{}^i - \delta_{10}{}^i - \delta_{00}{}^i) + \delta_{00}{}^i + \delta_{10}{}^i$
We transform the input from $\{0, 1\} \mapsto \{-1, 1\}$ by putting $a_i = \frac{x_i+1}{2}$ which simplifies the equation as:

$\Delta D(i) = \frac{x_i+1}{2}(\delta_{11}{}^i + \delta_{01}{}^i - \delta_{10}{}^i - \delta_{00}{}^i) + \delta_{00}{}^i + \delta_{10}{}^i = \frac{1+x_i}{2}(\delta_{11}{}^i + \delta_{01}{}^i) + \frac{1-x_i}{2}(\delta_{10}{}^i + \delta_{00}{}^i)$
The total delays for both upper and lower XORROs will be (for two consecutive oscillations):

$$\Delta D_{upper} = \sum_{i=0}^{R-1} [\frac{1+x_i}{2}(\delta_{U11}{}^i + \delta_{U01}{}^i) + \frac{1-x_i}{2}(\delta_{U10}{}^i + \delta_{U00}{}^i)]$$

$$\Delta D_{lower} = \sum_{i=0}^{R-1} [\frac{1+x_i}{2}(\delta_{L11}{}^i + \delta_{L01}{}^i) + \frac{1-x_i}{2}(\delta_{L10}{}^i + \delta_{L00}{}^i)]$$

The XORRO PUF response is generated by the comparison of ΔD_{upper} and ΔD_{lower}. Let us denote the difference by ΔD, then

$$\Delta D = \sum_{i=0}^{R-1} [\frac{1+x_i}{2}(\delta_{U11}{}^i + \delta_{U01}{}^i) + \frac{1-x_i}{2}(\delta_{U10}{}^i + \delta_{U00}{}^i)] - \sum_{i=0}^{R-1} [\frac{1+x_i}{2}(\delta_{L11}{}^i + \delta_{L01}{}^i) + \frac{1-x_i}{2}(\delta_{L10}{}^i + \delta_{L00}{}^i)]$$

$$= \sum_{i=0}^{R-1} [\frac{1+x_i}{2}(\delta_{U11}{}^i + \delta_{U01}{}^i - \delta_{L11}{}^i + \delta_{L01}{}^i) + \frac{1-x_i}{2}(\delta_{U10}{}^i + \delta_{U00}{}^i - \delta_{L10}{}^i + \delta_{L00}{}^i)]$$

Let us assume $\alpha_i = \delta_{U11}{}^i + \delta_{U01}{}^i - \delta_{L11}{}^i + \delta_{L01}{}^i$ and $\beta_i = \delta_{U10}{}^i + \delta_{U00}{}^i - \delta_{L10}{}^i + \delta_{L00}{}^i$

$$\Delta D = \sum_{i=0}^{R-1} [\frac{1+x_i}{2}(\alpha_i) + \frac{1-x_i}{2}(\beta_i)]$$

Let us accumulate the terms using vector notations and accumulating the terms

$$\Delta D = \overrightarrow{\mathbf{w}}^T \overrightarrow{\mathbf{x}}$$

where,

$\overrightarrow{\mathbf{x}} = \{\frac{1+x_0}{2}, \frac{1+x_1}{2}, ..., \frac{1+x_{n-1}}{2}, \frac{1-x_0}{2}, \frac{1-x_1}{2}, ..., \frac{1-x_{n-1}}{2}\} = \{a_0, a_1, ..., a_{n-1}, 1-a_0, 1-a_1, ..., 1-a_{n-1}\} = \phi(\mathbf{c})$

$\overrightarrow{\mathbf{w}} = \{\alpha_0, \alpha_1, ..., \alpha_{n-1}, \beta_0, \beta_1, ..., \beta_{n-1}\}$

We argue that response of the XORRO PUF can be represented as,

Response = $\frac{1+sign(-\Delta D)}{2}$ = $\frac{1+sign(\overrightarrow{-\mathbf{w}}^T \overrightarrow{\mathbf{x}})}{2} = \frac{1+sign(\overrightarrow{\mathbf{w}'}^T \overrightarrow{\mathbf{x}})}{2}$
where $\mathbf{w}'$ = -$\mathbf{w}$

To further verify this argument let us take all the possible cases:

Case 1: When $\Delta D > 0$

This would imply that the time taken by upper XORRO in one complete oscillation (i.e. getting same O/P value consecutively) is more than the lower XORRO.
Since, frequency is inversely proportional to time taken, the frequency of upper XORRO (XORRO 0) would be smaller than lower XORRO (XORRO 1). Hence, the counter would output 0 which matches with our output too.
Case 2: When $\Delta D < 0$

Similar argument can be given for this case too. The time taken by upper XORRO in one complete oscillation (i.e. getting same O/P value consecutively) is less than the lower XORRO.
So, the frequency of upper XORRO (XORRO 0) would be greater than lower XORRO (XORRO 1). Hence, the counter would output 1 which also matches with our output too.

Hence, we conclude that our expression for response is correct.

2

Show how to extend the above linear model to crack an Advanced XORRO PUF. Do this by treating an advanced XORRO PUF as a collection of multiple simple XORRO PUFs. For example, you may use M = 2^{S-1} $(2S-1)$ linear models, one for each pair of XORROs, to crack the advanced XORRO PUF.

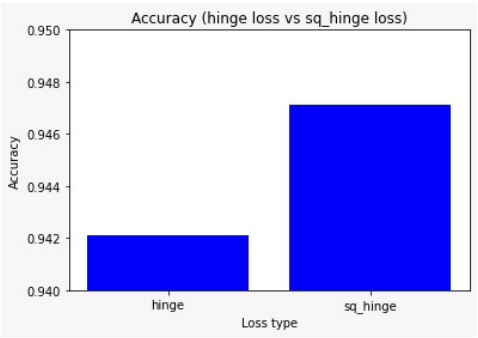
We can extend the described notion of a simple XORRO PUF to advanced XORRO PUF by building multiple linear models for each selected pair XORROs (i,j) where i < j. This would result in M = $2^{S-1}(2S-1)$ linear models which in our case is 120 models. (i.e. $\binom{16}{2}$ models). We used the provided data to estimate the input for each model. That turned out to be very similar, which highlights the accuracy of our strategy.
We will transform any input with selected XORRO pair (i,j) wherein i > j to our desired input (i.e. a pair of (i',j') where i' < j')
To achieve this, we invert our selected pair (i,j) whenever we encounter such a pair with i > j, and for we invert the corresponding output column (i.e. $r_i \mapsto$(i > j)$\oplus r_i$).
To **further upon improve accuracy** we exploit the fact that:

$$\alpha_i{}^{(m',n')} + \alpha_i{}^{(n',k')} = \alpha_i{}^{(m',k')}$$

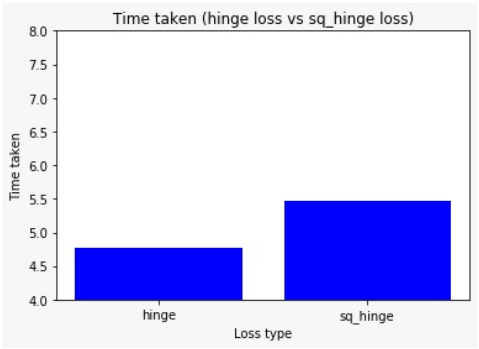
where, $\alpha_i{}^{(a,b)}$ corresponds to difference i^{th} XOR in the selected XORRO pair as (a,b) (i.e. a^{th} and b^{th} XORROs are selected to act as upper and lower XORRO respectively)
This can easily shown using the expression of α^i we derived for simple XOR case. Same relation holds true for β^i as well. Thus, this condition can be naturally extended to the whole weight vector $\overrightarrow{\mathbf{w}}$.
This condition enables us to derive the fact that the set $\{\mathbf{w}^{(i,j)} : j = i+1\}$ where $i, j \in \mathbb{N}$ and $\{\mathbf{w}^{(0,i)} : i \in \mathbb{N}\}$ forms a basis of the space of **w**. That is, if we somehow are able to predict these values, we can use them to refine the other weights we got using linear classifier models such as Linear SVC, Logistic Regression, etc.
This gives us scope to run a Linear Regression to find these basis using the other weights we got from classifier. Later, we can again use this basis vector we got (i.e. coefficients of Linear Regression) to fine tune values of **w's**.

4

Report outcomes of experiments with both the sklearn.svm.LinearSVC and sklearn.linear model.LogisticRegression methods when used to learn the ensemble linear model. In particular, report how various hyperparameters affected training time and test accuracy using tables, charts.

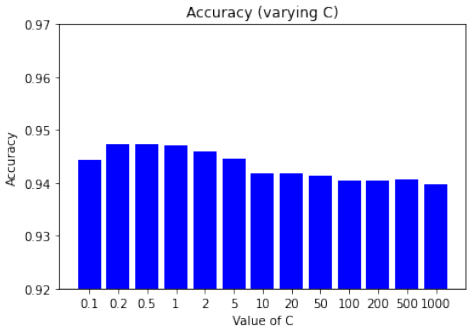


(a) It is clearly seen that squared_hinge loss function gives a very slightly better (by about 0.04%) accuracy

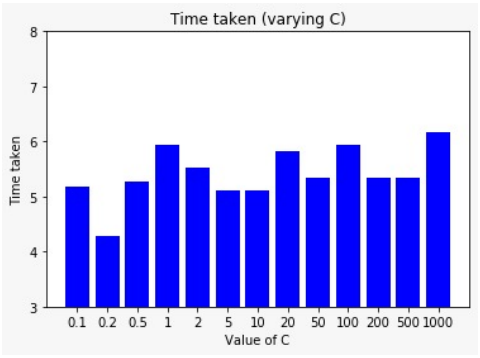


(b) Although accuracy is greater with squared_hinge but at the cost of a much greater increase training time (increase of about 1 sec).

Figure 1: changing the **loss hyperparameter** in LinearSVC (hinge vs squared hinge)

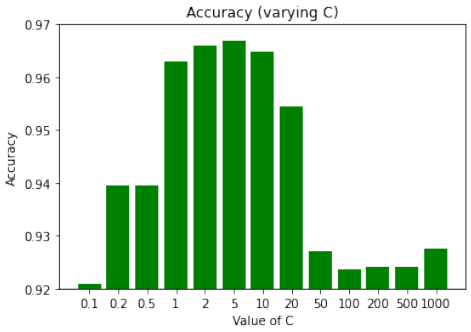


(a) Lowering values of C tend to give better accuracy, but lowering its value beyond 0.2 lowers the accuracy as well

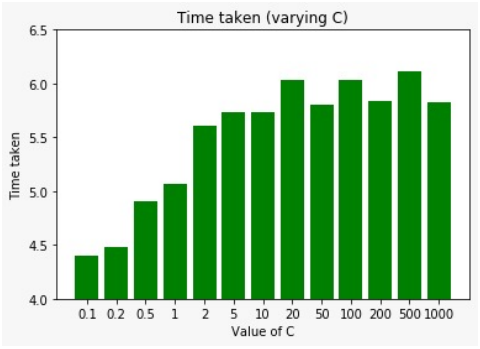


(b) The trend in training time taken with change in C hyperparameter is somewhat erratic, but on an average the training time lowers on decreasing C value

Figure 2: setting **C hyperparameter in LinearSVC** to high/low/medium values

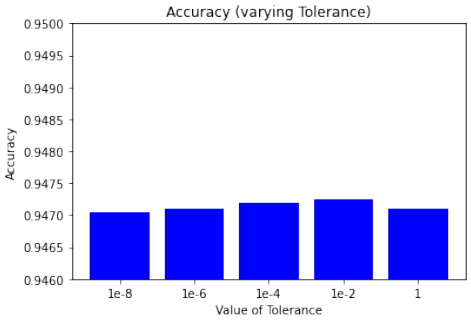


(a) The accuracy with changing values of C achieves its maxima around C = 5

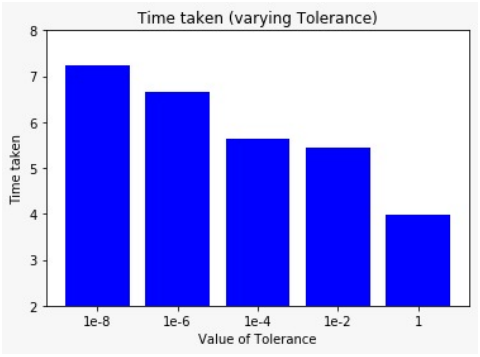


(b) Training time decreases with decreasing value of C, a C value of 5 seems to be the best choice with moderately high training time while giving the best accuracy

Figure 3: setting **C hyperparameter in LogisticRegression** to high/low/medium values

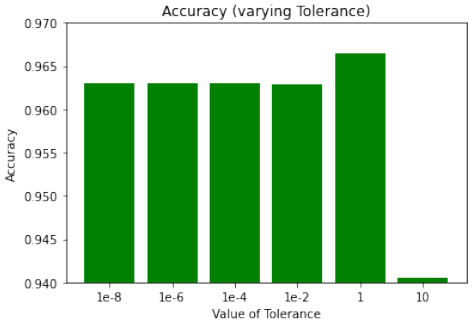


(a) Increasing values of tol increases accuracy until 1e-2, beyond that it starts dropping. The changes are although very insignificant.

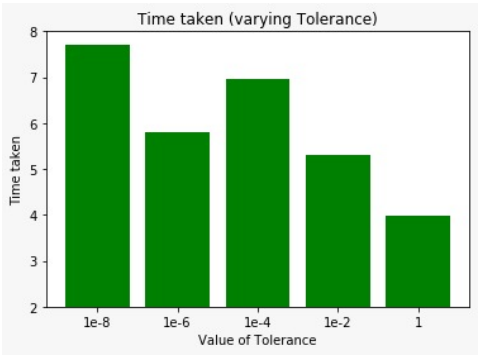


(b) Training time taken monotically decreases with decreasing values of tol

Figure 4: changing the **tol hyperparameter in LinearSVC** to high/low/medium values

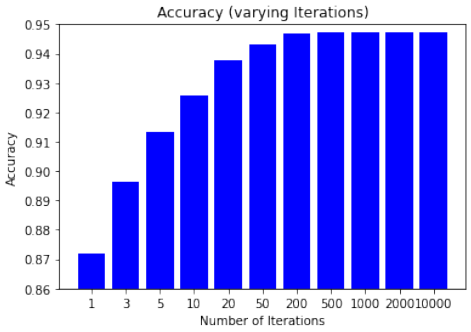


(a) Accuracy increases with increasing value of tol initially but beyond a value of 1, the accuracy drops drastically

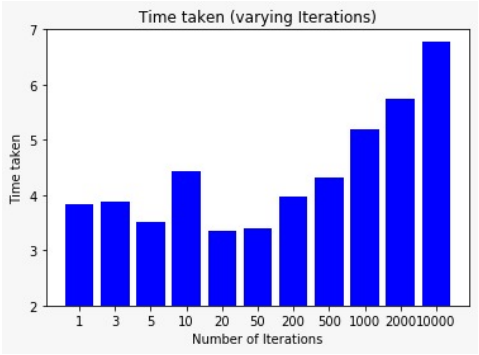


(b) Training time on an average decreases with decreasing value of tol, a value around 1 turns out to be the most sought after since it balances the accuracy vs training time tradeoff

Figure 5: changing the **tol hyperparameter in LogisticRegression** to high/low/medium values

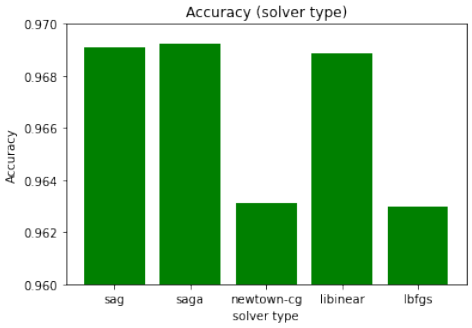


(a) Increasing max_iter greatly increases accuracy initially, but beyond a value of 500, the change is insignificant

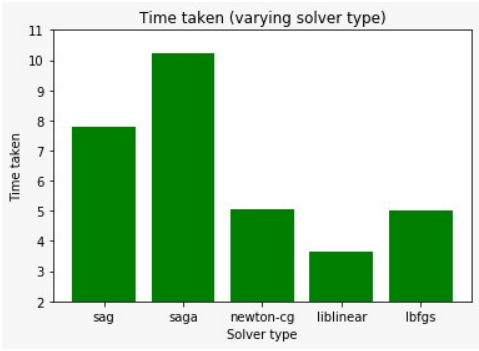


(b) Quite obviously, increasing the value of max_iter increases training time, a value around 200-500 is most suitable balancing this tradeoff

Figure 6: changing the **max_iter** in LinearSVC to high/low/medium values



(a) Order of accuracy with different solvers is clearly seen, with 'saga' and 'sag' giving the greatest accuracy



(b) 'saga' and 'sag' do offer much greater accuracy but with a tradeoff of a great increase in time taken as well

Figure 7: changing the **solver** in Logistic Regression to high/low/medium values

References

[1] Batina Lejla, Picek Stjepan and Mondal Mainack (2022) *Security, Privacy, and Applied Cryptography* Engineering. Springer 12th International Conference, SPACE 2022