Assignment 1

Chitwan (210294)
Dept. of Aerospace Engineering
chitwan21@iitk.ac.in

Nandan Madhuj (210648)
Dept. of Aerospace Engineering
nandanm21@iitk.ac.in

Gautam Raghuvanshi (210391)
Dept. of Electrical Engineering
gautamr21@iitk.ac.in

Naman Goyal (210646)
Dept. of Chemical Engineering
namang21@iitk.ac.in

Nitish Kumar (210679)
Dept. of Mechanical Engineering
nitish21@iitk.ac.in

Utpal Dwivedi (211132)
Dept. of Biological Engineering
utpaldwi21@iitk.ac.in

1 A Linear Model for the Arbiter-PUF

Section Summary: It is shown that a linear model exists for breaking the Arbiter PUF. This discussion is used in subsequent section for breaking CAR-PUF. Class discussion and lectures were used as reference.

Let the challenge bits be given by $\mathbf{c} \stackrel{\text{def}}{=} [c_1, c_2, c_3..., c_R]$
Considering an Abiter-PUF where c_i is the ith configuration bit in the PUF, t_i^u and t_i^l denote the time at which the upper and lower signals exit the ith switch respectively and let p_i, q_i, r_i, and s_i be the time delays in various possible paths that a signal pair will take depending on *config* bit.
The reccurring relation for an Arbiter-PUF is given by,

$$t_i^u = (1 - c_i)(t_{i-1}^u + p_i) + c_i(t_{i-1}^l + s_i)$$
$$t_i^l = (1 - c_i)(t_{i-1}^l + q_i) + c_i(t_{i-1}^u + r_i)$$

If Δ_i is the difference $t_i^u - t_i^l$, then Δ_i is given by,

$$\begin{aligned}\Delta_i &= (1 - c_i)(t_{i-1}^u + p_i - t_{i-1}^l - q_i) + c_i(t_{i-1}^l + s_i - t_{i-1}^u - r_i) \\ &= (1 - 2c_i)\Delta_{i-1} + (q_i - p_i + s_i - r_i)c_i + p_i - q_i\end{aligned}$$

We now re-encode the challenge **c** into same-dimensional challenge **d** such that $d_i \stackrel{\text{def}}{=} 1 - 2ci$

Substituting c_i in terms of d_i in the above relation, we get,

$$\Delta_i = d_i \cdot \Delta_{i-1} + d_i \cdot \frac{p_i - q_i + r_i - s_i}{2} + \frac{p_i - q_i - r_i + s_i}{2}$$

Hence, all delays of a single PUF can be represented into two variables given by

$$\alpha_i \stackrel{\text{def}}{=} \frac{p_i - q_i + r_i - s_i}{2}$$
$$\beta_i \stackrel{\text{def}}{=} \frac{p_i - q_i - r_i + s_i}{2}$$

Since the same signal goes into the first PUF, we assume that $\Delta_0 = 0$. A reccurrence relation can be obtained by using,

$$\Delta_i = d_i \cdot \Delta_{i-1} + d_i \cdot \alpha_i + \beta_i$$

For the first few cases, Δ_i's are given by,

$$\Delta_1 = \alpha_1 \cdot d_1 + \beta_1$$
$$\Delta_2 = \alpha_1 \cdot d_2 \cdot d_1 + (\alpha_2 + \beta_1)d_2 + \beta_2$$
$$\Delta_3 = \alpha_1 \cdot d_3 \cdot d_2 \cdot d_1 + (\alpha_2 + \beta_1)d_3 \cdot d_2 + (\alpha_3 + \beta_2)d_3 + \beta_3$$
$$\Delta_4 = \alpha_1 \cdot d_4 \cdot d_3 \cdot d_2 \cdot d_1 + (\alpha_2 + \beta_1)d_4 \cdot d_3 \cdot d_2 + (\alpha_3 + \beta_2)d_4 \cdot d_3 + (\alpha_4 + \beta_3)d_4 + \beta_4$$

By induction, we note that the last delay, for an R-bit challenge, is given by,

$$\Delta_R = \alpha_1.d_R.d_{R-1}\ldots d_1 + (\alpha_2 + \beta_1).d_R.d_{R-1}\ldots d_2 + (\alpha_3 + \beta_2).d_R.d_{R-1}\ldots d_3 + \ldots + (\alpha_R + \beta_{R-1}).d_R + \beta_R$$

Written concisely,

$$\Delta_R = w_1 \cdot x_1 + w_2 \cdot x_2 + w_3 \cdot x_3 + \cdots + w_R \cdot x_R + b$$

Which is a linear model in re-encoded *config* bits **x**.
Hence, given an R-bit challenge vector **c** such that $c_i \in \{0, 1\}$, a linear model for the Arbiter-PUF is given in terms of re-encoded challenge vector **x**, given by,

$$x_i = \prod_{j=i}^{R} (1 - 2 \cdot c_j)$$
$$x_i = \prod_{j=i}^{R} d_j$$

Representing Δ_R concisely using matrices, we note that,

$$\Delta_R = \begin{bmatrix} \alpha_1 \\ \alpha_2 + \beta_1 \\ \vdots \\ \alpha_i + \beta_{i-1} \\ \vdots \\ \alpha_R + \beta_{R-1} \end{bmatrix}^T \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_i \\ \vdots \\ x_r \end{bmatrix} + \beta_R$$

Hence, a linear model **w** and bias b exist to break the Arbiter PUF, so that,

$$\Delta_R = \mathbf{w}^T\mathbf{x} + b$$

The vector **w** is the model, **x** is the re-encoded challenge, b in the bias, given by,

$$\mathbf{w} = \begin{bmatrix} \alpha_1 \\ \alpha_2 + \beta_1 \\ \vdots \\ \alpha_i + \beta_{i-1} \\ \vdots \\ \alpha_R + \beta_{R-1} \end{bmatrix} \qquad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_i \\ \vdots \\ x_R \end{bmatrix} \qquad b_0 = \beta_R$$

2 A linear model for CAR-PUF

Section Summary: The first problem of the assignment is discussed in this section. Refer previous section for clarity.
From the previous discussion, we already know that a linear model exists for Arbiter PUFs. Hence, for the reference PUF, $\exists$ a linear model containing $\mathbf{u}_{32\times1}$ and $p_{1\times1}$ such that,

$$\Delta_r = \mathbf{u}^T \cdot \mathbf{x} + p$$

Similarly, for the working PUF, $\exists$ a linear model containing $\mathbf{v}_{32\times1}$ and $q_{1\times1}$ such that,

$$\Delta_w = \mathbf{v}^T \cdot \mathbf{x} + q$$

Here, the parameters $\mathbf{v}$, p, $\mathbf{u}$ and q depend on the PUF characteristics and do not change with time. Hence, a linear model can be trained to optimise these parameters to best **clone** a physically unclonable function!

The output on the CAR-PUF, depends on

$$|\Delta_w - \Delta_r| \leq \tau \implies 0$$
$$|\Delta_w - \Delta_r| > \tau \implies 1$$

To remove the absolute value function, we square both sides,

$$(\Delta_w - \Delta_r)^2 \leq \tau^2 \implies 0$$
$$(\Delta_w - \Delta_r)^2 > \tau^2 \implies 1$$

Using indicial notation, where repeated index implies summation,

$$\Delta_r = u_i x_i + p$$
$$\Delta_w = v_i x_i + q$$

Subtracting the above two equations gives;

$$\Delta_r - \Delta_w = (u_i - v_i)x_i + p - q = a_i x_i + d$$

Where $a_i = u_i - v_i$.
Squaring both sides gives,

$$(\Delta_r - \Delta_w)^2 = (a_i x_i + d) \cdot (a_j x_j + d)$$
$$\implies (\Delta_r - \Delta_w)^2 - \tau^2 = a_i \cdot a_j \cdot x_i \cdot x_j + d \cdot (2 \cdot a_i x_i) + d^2 - \tau^2$$

Dropping the indicial notation and writing the above equation explicitly using summation.

$$(\Delta_r - \Delta_w)^2 - \tau^2 = \sum_{i=1}^{32}\sum_{j=1}^{32} a_i \cdot a_j \cdot x_i \cdot x_j + \sum_{i=1}^{32}(2 \cdot d \cdot a_i x_i) + d^2 - \tau^2$$

The second order terms are of two types, $i = j$ and $i \neq j$. Separating both types,

$$\implies (\Delta_r - \Delta_w)^2 - \tau^2 = \sum_{i\neq j}^{32}\sum^{32} a_i \cdot a_j \cdot x_i \cdot x_j + \sum_{i=j}^{32}\sum^{32} a_i \cdot a_j \cdot x_i \cdot x_j + \sum_{i=1}^{32}(2 \cdot d \cdot a_i x_i) + d^2 - \tau^2$$

$$\implies (\Delta_r - \Delta_w)^2 - \tau^2 = \sum_{i\neq j}^{32}\sum^{32} a_i \cdot a_j \cdot x_i \cdot x_j + \sum_{i=1}^{32} a_i^2 \cdot x_i^2 + \sum_{i=1}^{32}(2 \cdot d \cdot a_i x_i) + d^2 - \tau^2$$

However, from the definition of x_i, it is evident that $x_i \pm 1$. Hence, $x_i^2 = 1 \; \forall \; i$.
Using this, we have,

$$(\Delta_r - \Delta_w)^2 - \tau^2 = \sum_{i\neq j}^{32}\sum^{32} a_i \cdot a_j \cdot x_i \cdot x_j + \sum_{i=1}^{32}(2 \cdot d \cdot a_i x_i) + d^2 - \tau^2 + \sum_{i=1}^{32} a_i^2$$

This can be seen as a linear-model if we define the challenge vector in terms of

$$\Phi = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{32} \\ x_1 \cdot x_2 \\ x_1 \cdot x_3 \\ \vdots \\ x_1 \cdot x_{32} \\ \vdots \\ x_2 \cdot x_3 \\ x_2 \cdot x_4 \\ \vdots \\ \vdots \end{bmatrix}$$

It is evident that Φ has two types of terms: linear terms (x_i) and 2^{nd} degree terms $x_i \cdot x_j$.
Number of linear terms = 32.
Since $i \neq j$, Number of second order terms = $\binom{32}{2} = 496$
Overall dimension of Φ is $32 + 496 = 528$

$$D = 528$$

Hence, $\exists$ a D-dimensional linear model, **W** for the problem.
Specifically, for x_i, **W** has the element $w_i = 2 \cdot d \cdot a_i = 2 \times (p - q) \times (u_i - v_i)$.
For non-linear terms $x_i \cdot x_j$, **W** has element $a_i \cdot a_j + a_j \cdot a_i = 2 \cdot (u_i - v_i) \cdot (u_j - v_j)$ at the same index as $x_i \cdot x_j$ is in Φ. Note here that we have clubbed $a_i \cdot a_j$ and $a_j \cdot a_i$ together, that is why a factor of 2.
Finally, the constant b is defined by,

$$b = d^2 - \tau^2 + \sum_{i=1}^{32} a_i^2 = (p - q)^2 - \tau^2 + \sum_{i=1}^{32} (u_i - v_i)^2$$

Incorporating all these, it is evident that the CAR-PUF is governed by the following rule.

$$W^T \cdot \phi(\mathbf{c}) + b \leq 0 \implies 0$$
$$W^T \cdot \phi(\mathbf{c}) + b > 0 \implies 1$$

Hence, the response - *r* from the CAR-PUF is governed by,

$$r = \frac{1 + sign(W^T \cdot \phi(\mathbf{c}) + b)}{2}$$

3 Hyperparameter Variation

Overview: The following section shows the effect of varying hyperparameters on Linear SVM and Logistic Regression models.
In particular, training time, mapping time and accuracy variation has been shown using plots. These experiments were done on a PC with specifications as follows:
RAM 8 GB, **Processor**: Intel(R) Core(TM) i5-10210U CPU 1.60 GHz 2.11 GHz

3.1 Changing C - hyperparameter

3.1.1 Linear SVC

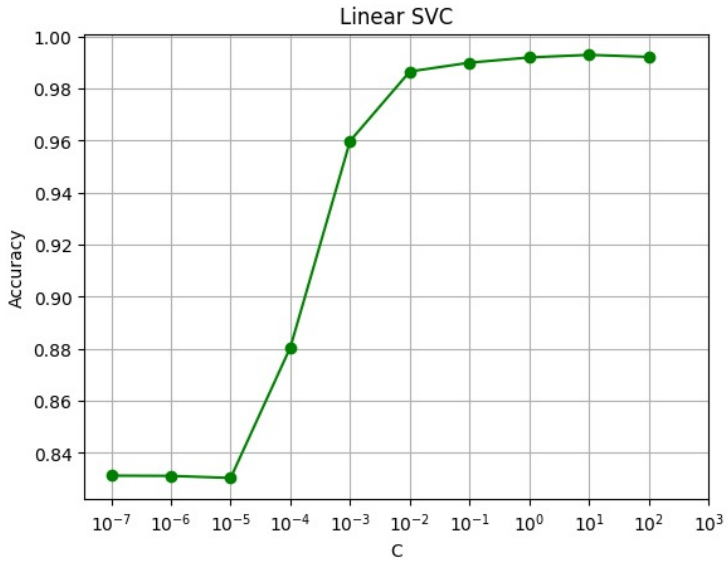

Figure 1: C vs accuracy

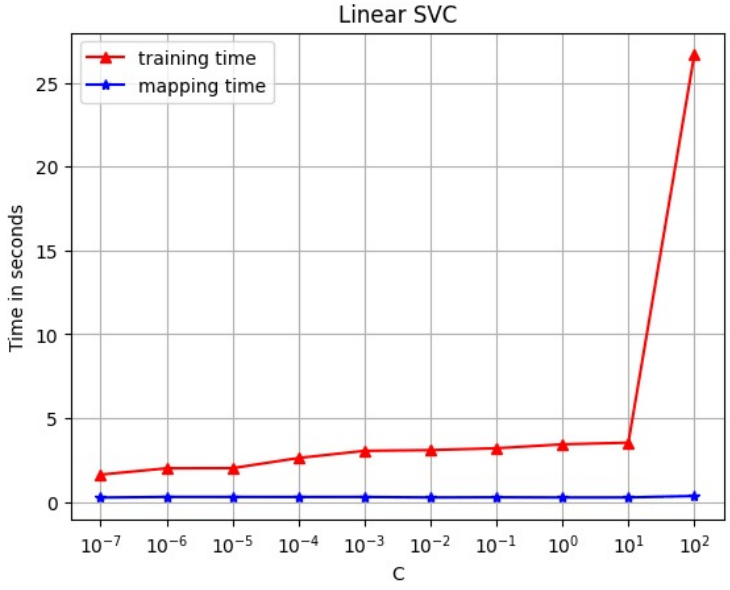

Figure 2: C vs time

3.1.2 Logistic Regression

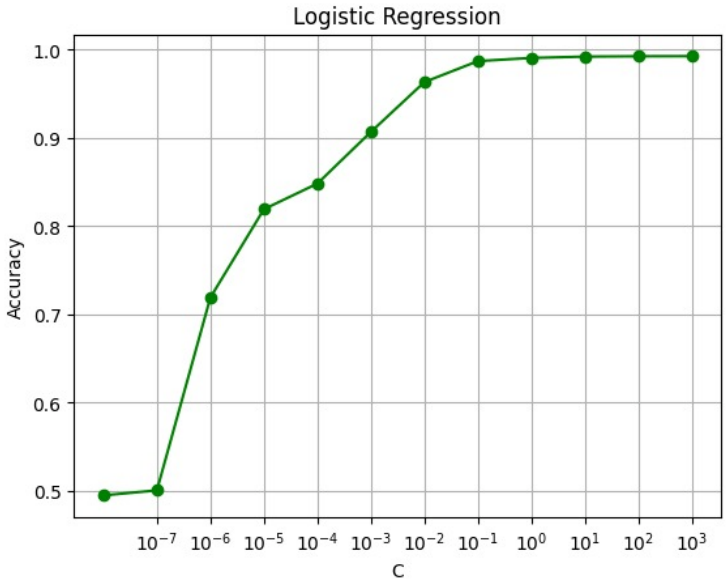

Figure 3: C vs accuracy

Figure 4: C vs time

Discussion: We note that the hyperparameter C has more effect on the model accuracy and little effect on the training time. Moreover, the mapping time remains the same irrespective of the hyperparameter variation - as expected.
For SVC, C value of around 0.1 to 10 is optimal.
For logistic regression, also, C of around 0.1 to 10 is optimal.

3.2 Changing tol - hyperparameter

3.2.1 Linear SVC

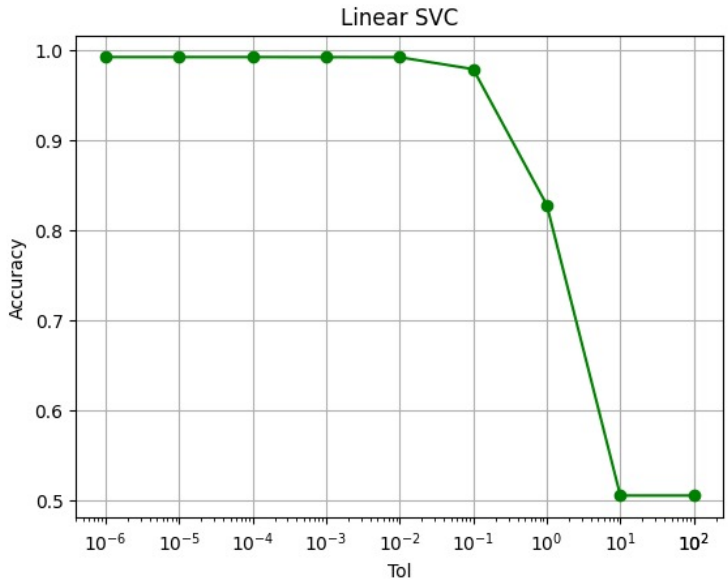

Figure 5: tol vs accuracy

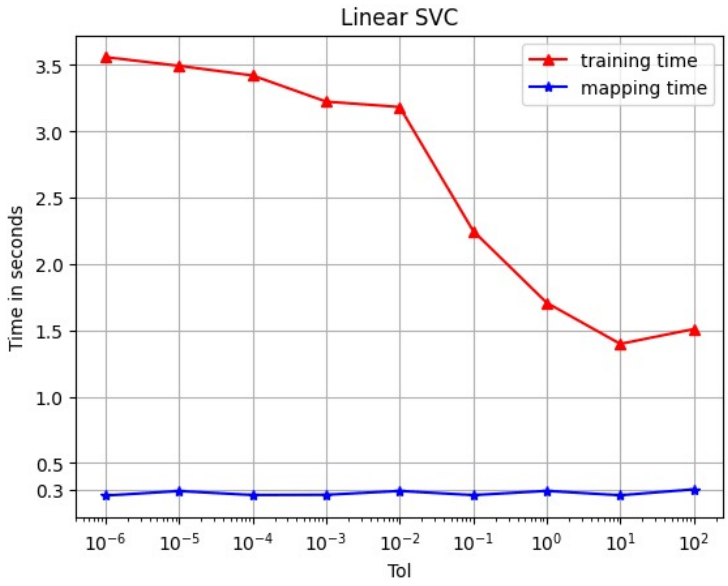

Figure 6: tol vs time

Discussion: It is evident that the Tol hyperparameter governs the training time better. Further, high tol value (>10) causes an accuracy of 50%. Thus, for all practical purposes, such a model is outputting random results.
For SVC, when C=1, Tol value of 0.01 to 0.1 is optimal.

3.2.2 Logistic Regression

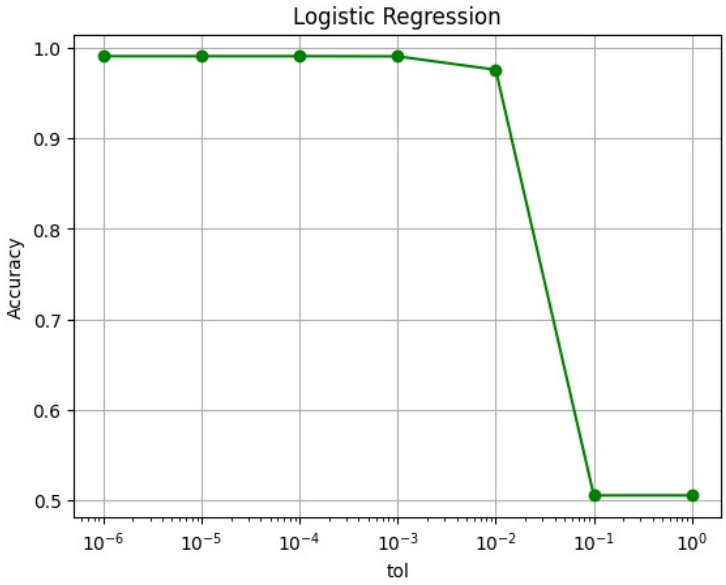

Figure 7: tol vs accuracy

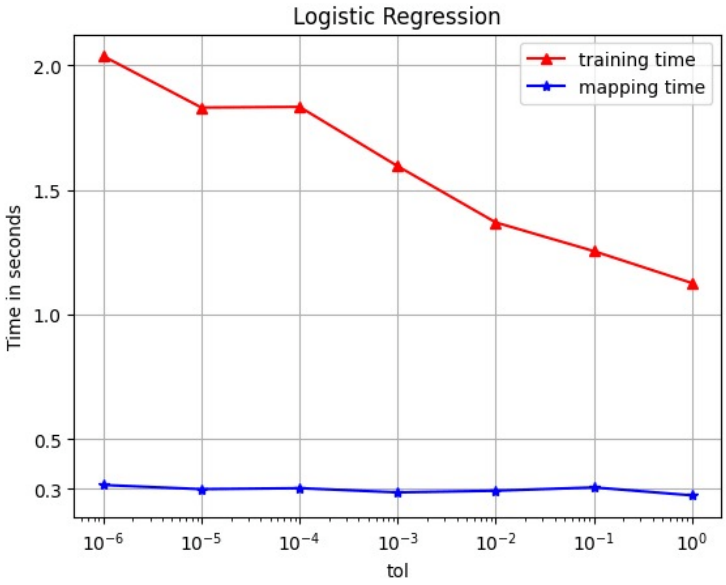

Figure 8: tol vs time

Discussion: Again, for Logistic regression also, Tol hyperparameter evidently affect the time strongly. Here, the model becomes rubbish for Tol >0.1.
For logistic regression, Tol value of around 0.001 is optimal.

4 References

1. Scikit Learn Linear SVC Documentation
2. Scikit Learn Logistic Regression Documentation
3. Numpy Documentation
4. Lecture Content