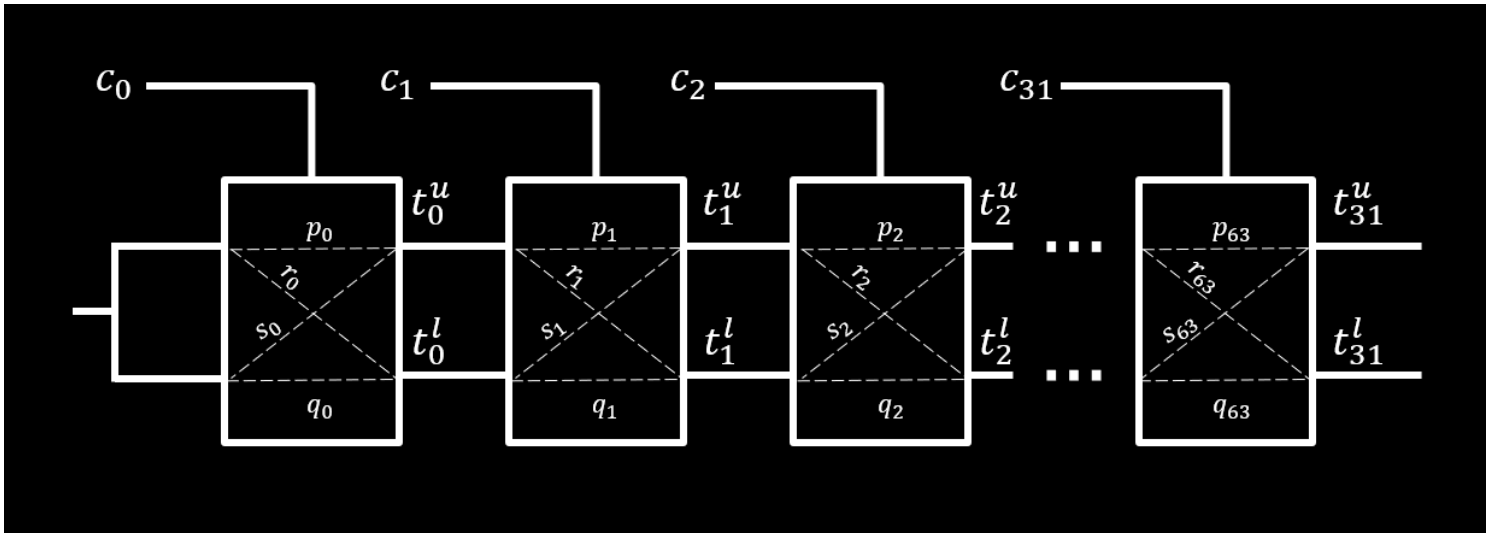


Figure 1: **Working PUF**

For working case:

$$\begin{aligned}
\Delta w_i &= t_i^u - t_i^l \quad \text{– denotes the lag for working PUF} \\
\Delta w_o &= t_o^u - t_o^l \quad \text{– denotes the lag for first } c = c_o \\
\Delta w_1 &= (1 - c_1) \cdot (t_o^u + p_1 - t_o^l - q_1) + c_1 \cdot (t_o^l + s_1 - t_o^u - r_1) \\
&= (1 - 2c_1) \cdot \Delta w_o + c_1 \cdot (t_o^l + s_1 - t_o^u - r_1) + (p_1 - q_1)
\end{aligned}$$

To make the notation simpler, let $d_i = (1 - 2c_i)\Delta w_1 = d_1 \cdot \Delta w_o + d_1 \cdot \alpha_{w_1} + \beta_{w_1}$

$$\begin{aligned}
\text{where,} \quad \alpha_{w_1} &= (p_1 - q_1 + r_1 - s_1)/2 \\
\beta_{w_1} &= (p_1 - q_1 - r_1 + s_1)/2
\end{aligned}$$

Note that a similar relation holds for any stage

$$\begin{aligned}
\Delta w_i &= d_i \cdot \Delta w_{i-1} + d_i \cdot \alpha_{w_i} + \beta_{w_i} \\
\text{where, } \alpha_{w_i} &= (p_i - q_i + r_i - s_i)/2 \\
\beta_{w_i} &= (p_i - q_i - r_i + s_i)/2
\end{aligned}$$

We can safely take $\Delta w_{-1} = 0$ (absorb initial delays into p_o, q_o, r_o, s_o).

We can keep going recursively,

$$\begin{aligned}\Delta w_o &= d_o \cdot \alpha_{w_o} + \beta_{w_o} \quad (\text{since } \Delta w_{-1} = 0)\\ \Delta w_1 &= d_1 \cdot \Delta w_o + d_1 \cdot \alpha_{w_1} + \beta_{w_1} \quad \text{– now plugin value of } \Delta w_o \text{ to get}\\ &= d_1 \cdot d_o \cdot \alpha_{w_o} + d_1 \cdot (\alpha_{w_1} + \beta_{w_o}) + \beta_{w_1}\end{aligned}$$

Similarly,

$$\Delta w_2 = d_2 \cdot d_1 \cdot d_o \cdot \alpha_{w_o} + d_2 \cdot d_1 \cdot (\alpha_{w_1} + \beta_{w_o}) + d_2 \cdot (\alpha_{w_2} + \beta_{w_1}) + \beta_{w_2}$$

We can begin to see a pattern here.

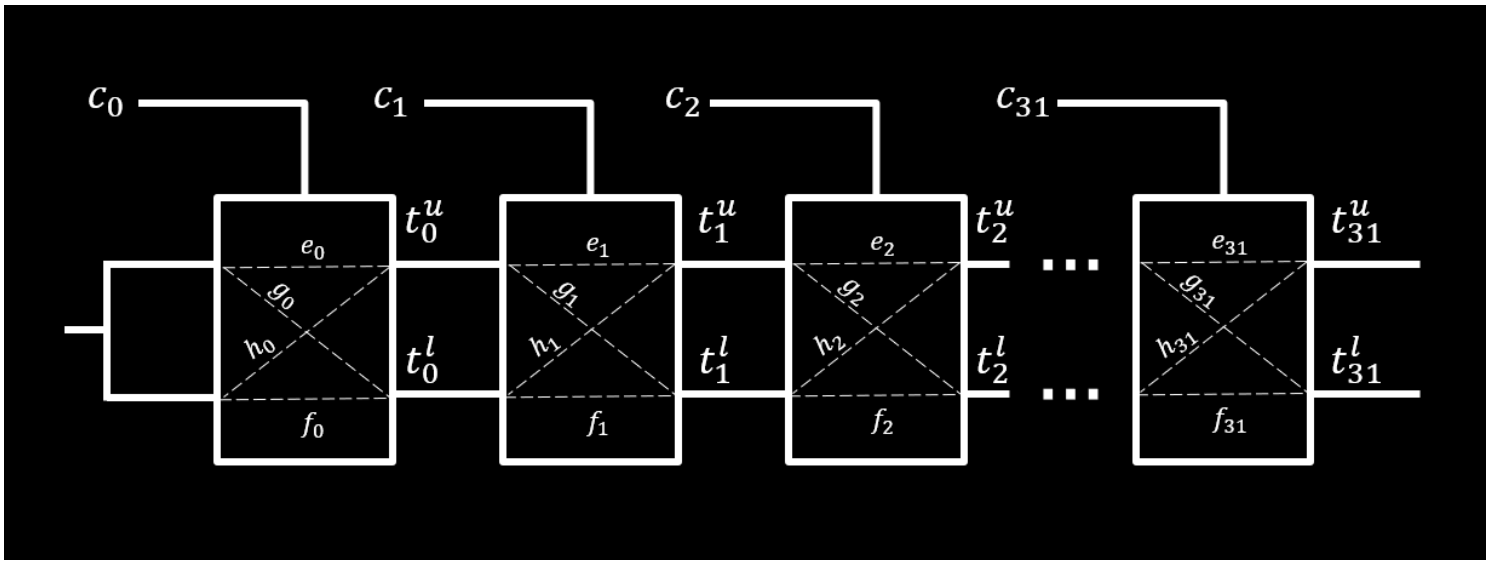
We have,

$$\Delta w_{31} = x_o \cdot W_{w_o} + x_1 \cdot W_{w_1} + ... + x_{31} \cdot W_{w_{31}} + \beta_{w_{31}}$$

$$\Delta w_{31} = W_w^T \cdot X_w + b_w$$

where, $X_{w_i} = d_i \cdot d_{i+1} \cdot d_{i+2}.....d_{31} \quad 0 \le i \le 31$

Similarly, For reference case:



Reference PUF

$$\Delta r_i = t_i^u - t_i^l \quad \text{– denotes the lag for working PUF}$$
$$\Delta r_o = t_o^u - t_o^l \quad \text{– denotes the lag for first } c = c_o$$
$$\begin{aligned}\Delta r_1 &= (1-c_1)\cdot(t_o^u + e_1 - t_o^l - f_1) + c_1 \cdot (t_o^l + h_1 - t_o^u - g_1) \\ &= (1-2c_1)\cdot \Delta g_o + c_1 \cdot (t_o^l + h_1 - t_o^u - g_1) + (e_1 - f_1)\end{aligned}$$

To make the notation simpler, let $d_i = (1-2c_i)$

$$\Delta r_1 = d_1 \cdot \Delta r_o + d_1 \cdot \alpha_{r_1} + \beta_{r_1}$$

$$\begin{aligned}\text{where,} \quad \alpha_{r_1} &= (p_1 - q_1 + r_1 - s_1)/2 \\ \beta_{r_1} &= (e_1 - f_1 - g_1 + h_1)/2\end{aligned}$$

Note that a similar relation holds for any stage,

$$\begin{aligned}\Delta r_i &= d_i \cdot \Delta r_{i-1} + d_i \cdot \alpha_{r_i} + \beta_{r_i} \\ \text{where, } \alpha_{r_i} &= (p_i - q_i + r_i - s_i)/2 \\ \beta_{r_i} &= (p_i - q_i - r_i + s_i)/2\end{aligned}$$

We can safely take $\Delta r_{-1} = 0$ (absorb initial delays into p_o, q_o, r_o, s_o).
We can keep going recursively,

$$\begin{aligned}\Delta r_o &= d_o \cdot \alpha_{r_o} + \beta_{r_o} \quad (\text{since } \Delta r_{-1} = 0) \\ \Delta r_1 &= d_1 \cdot \Delta r_o + d_1 \cdot \alpha_{r_1} + \beta_{r_1} \quad \text{– now plugin value of } \Delta r_o \text{ to get} \\ &= d_1 \cdot d_o \cdot \alpha_{r_o} + d_1 \cdot (\alpha_{r_1} + \beta_{r_o}) + \beta_{r_1}\end{aligned}$$

Similarly,

$$\Delta r_2 = d_2 \cdot d_1 \cdot d_o \cdot \alpha_{r_o} + d_2 \cdot d_1 \cdot (\alpha_{r_1} + \beta_{r_o}) + d_2 \cdot (\alpha_{r_2} + \beta_{r_1}) + \beta_{r_2}$$

We can begin to see a pattern here.
We have,

$$\Delta r_{31} = x_o \cdot W_{r_o} + x_1 \cdot W_{r_1} + ... + x_{31} \cdot W_{r_{31}} + \beta_{r_{31}}$$

$$\Delta r_{31} = W_r^T \cdot X_r + b_r$$

$$\begin{aligned}\text{where,} \quad X_{r_i} &= d_i \cdot d_{i+1} \cdot d_{i+2} ..... d_{31} \qquad 0 \le i \le 31 \\ W_{r_o} &= x_o \\ W_{r_i} &= \alpha_{r_i} + \beta_{r_{i-1}} \qquad i > 0\end{aligned}$$

Now, we will take
$\Delta_i = \Delta w_i - \Delta r_1 - - >$ denote the lag difference between working and reference PUFs

$$\Delta_i = d_i \cdot (\Delta w_{i-1} - \Delta r_{i-1}) + d_i \cdot (\alpha_{w_i} - \alpha_{r_i}) + (\beta_{w_i} - \beta_{r_i})$$

Take, $\alpha_i = \alpha_{w_i} - \alpha_{r_i}$
$\beta_i = \beta_{w_i} - \beta_{r_i}$

Now we have,

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

We can safely take $\Delta_{-1} = 0$
We can keep going recursively,

$$\begin{aligned}\Delta_o &= d_o \cdot \alpha_o + \beta_o \quad (\text{since } \Delta_{-1} = 0)\\ \Delta_1 &= d_1 \cdot \Delta_o + d_1 \cdot \alpha_1 + \beta_1 \quad - \text{now plugin value of } \Delta_o \text{ to get}\\ &= d_1 \cdot d_o \cdot \alpha_o + d_1 \cdot (\alpha_1 + \beta_o) + \beta_1\end{aligned}$$

Similarly,

$$\Delta_2 = d_2 \cdot d_1 \cdot d_o \cdot \alpha_o + d_2 \cdot d_1 \cdot (\alpha_1 + \beta_o) + d_2 \cdot (\alpha_2 + \beta_1) + \beta_2$$

We can begin to see a pattern here.
We have,

$$\Delta_{31} = x_o \cdot W_o + x_1 \cdot W_1 + ... + x_{31} \cdot W_{31} + \beta_{31}$$

$$\Delta_{31} = W^T \cdot X + b$$

$$\text{where,} \quad X_i = d_i \cdot d_{i+1} \cdot d_{i+2} .....d_{31} \qquad 0 \le i \le 31$$
$$W_o = W_{w_o} - W_{r_o}$$
$$W_i = W_{w_i} - W_{r_i} \qquad i > 0$$

Now we are given that,
Response to the challenge is **0** if, $|(\Delta w_i - \Delta r_i)| \le \tau$ where, $\tau > 0$
& Response to the challenge is **1** if, $|(\Delta w_i - \Delta r_i)| > \tau$ where, $\tau > 0$
We are also given that we can let $(\mathbf{u}, p)$, $(\mathbf{v}, q)$ be the two linear models that can exactly predict the outputs of the two arbiter PUFs sitting inside the CAR-PUF
Now as we have assumed,

$$\Delta_i = \Delta w_i - \Delta r_1$$

So in,

$$\Delta_i = \mathbf{W^T} \cdot \mathbf{X} + b \qquad \text{we have,} \quad \mathbf{W^T} = \mathbf{u} - \mathbf{v} \text{ and b = p-q}$$

Now in our response condition, we have

Response to the challenge is **0** if, $|\Delta_i| \leq \tau$ where, $\tau > 0$

& Response to the challenge is **1** if, $|\Delta_i| > \tau$ where, $\tau > 0$

On squaring both sides we will get,

Response to the challenge is **0** if, $(\mathbf{W^T} \cdot \mathbf{X} + b)^2 - (\tau)^2 \leq 0$

Response to the challenge is **1** if, $(\mathbf{W^T} \cdot \mathbf{X} + b)^2 - (\tau)^2 > 0$

Let's open the square, we will get

$$(\mathbf{W^T} \cdot \mathbf{X}) * (\mathbf{W^T} \cdot \mathbf{X}) + 2 * (\mathbf{W^T} \cdot \mathbf{X}) * b + b^2 - \tau^2$$

From this equation, we can see that there will be

(32*33)/2 = 528 unique terms from the first term,

32 terms from the second term and a constant from last term

We are getting 560 terms in our feature map $\phi(\mathbf{c})$, but what we know is there will be terms of $(X_i)^2$ which we know will be constant because either X_i will be -1 or 1 for both the cases $(X_i)^2$= 1. Hence these terms are constant which will be added to the constant term that we got earlier.

Thus finally we have 560-32 = 528 Dimensional feature map, D($\phi(\mathbf{c})$)=528

$\phi(\mathbf{c})$=((X_o*X_1),(X_o*X_2),(X_o*X_3).....(X_{30}*X_{31})),

we know that $X_i = d_i \cdot d_{i+1} \cdot d_{i+2}.....d_{31}$ and $d_i = (1 - 2c_i)$

Hence, $\phi(\mathbf{c})$ is a function of c only.

and $\mathbf{W} = (\mathbf{uv}, \mathbf{u}q + \mathbf{v}p, qp, f(\tau))$

1 Hyperparameter Tuning

1.1 LinearSVC

1.1.1 Loss Function

Loss	Accuracy (%)	Time (sec)
hinge	99.04	4.1
squared_hinge	99.2	4.57

Even though squared_hinge takes slightly more training time, it gives a better accuracy than hinge loss function.

1.1.2 c: Regularization Parameter

c	Accuracy (%)	Time (sec)
0.001	95.6	1.04
0.01	98.36	2.04
0.1	99.01	5.65
1	99.32	4.88
10	99.06	4.38
100	99.05	4.44

The accuracy increases till the value of c is 10 and begins decreasing from there. Training time, on the other hand, increases till c = 0.1 and then begins decreasing.

1.1.3 Penalty Term

Penalty	Accuracy (%)	Time (sec)
l1 (dual = False)	99.31	67.58
l2 (dual = False)	99.31	1.85
l2 (dual = True)	99.2	4.59

l1 and l2 give similar accuracies when dual = False, but the training time of l2 in this case is 36.5 times less than l1.

1.1.4 Tolerance Criteria

Tol	Accuracy (%)	Time (sec)
0.0001	99.23	4.37
0.001	99.18	4.77
0.01	99.36	4.47
0.1	99.22	5
1	99.21	4.54
10	93.35	0.44

The training time as well as the accuracy decrease drastically when tol is changed from 1 to 10. We get the best accuracy at tol = 0.01 while the time taken remains comparable.

1.1.5 Maximum Number of Iterations

Max_iter	Accuracy (%)	Time (sec)
100	97.9	1.78
300	98.85	2.53
200	98.41	2.13
500	99.18	3.24
700	99.19	3.63
850	99.23	4.18
1000	99.26	4.78

It can be seen that from 500 iterations onwards, the accuracy remains consistent while training time increases. linearly with the number of iterations.

1.2 LogisticRegression

1.2.1 C: Inverse of Regularization Strength

C	Accuracy (%)	Time (sec)
0.001	90	0.45
0.01	96.16	0.51
0.1	98.48	0.58
1	99.15	0.69
10	99.31	0.87
100	99.31	1.55

The accuracy increases till C = 10 and then grows consistent. The training time increases proportionally with C.

1.2.2 Solver

Solver	Accuracy (%)	Time (sec)
lbfgs	99.15	0.73
liblinear	99.12	2.86
sag	99.15	13.42
saga	99.1	13.5

There is no noticeable change in the accuracy, but the training time is lowest for 'lbfgs' and highest for 'saga'.

1.2.3 Penalty Term

Penalty	Accuracy (%)	Time (sec)
l1 (solver = 'liblinear')	99.27	71.10
l1 (solver = 'saga')	99.12	21.0
l2 (solver = 'lbfgs')	99.15	0.86
l2 (solver = 'liblinear')	99.12	2.92

Once again, the accuracies are comparable but l1 takes a minimum of 7.19 and a maximum of 82.7 times the time taken by l2.

1.2.4 Tolerance Criteria

Tol	Accuracy (%)	Time (sec)
0.0001	99.15	0.75
0.001	99.15	0.75
0.01	99.15	0.69
0.1	99.15	0.62
1	99.13	0.56
10	93.11	0.52

The accuracy remains constant and then decreases very slightly from tol = 1 while the training time decreases as the tolerance criteria increases.

1.2.5 Maximum Number of Iterations

Max_iter	Accuracy (%)	Time (sec)
100	99.15	0.7
300	99.15	0.69
200	99.15	0.7
500	99.15	0.69
700	99.15	0.73
850	99.15	0.73
1000	99.15	0.72

The maximum number of iterations does not have an effect on the accuracy in this case, and the training time shows very minute deflections as well.

2 Conclusion

Based on our analysis, we come to the conclusion that LogisticRegression gives a better accuracy and takes lesser time in order to train the model when compared to LinearSVC. The hyperparameters for LogisticRegression we found to work best were as follows:

- solver = 'lbfgs'
- penalty = 'l2'
- C = 10
- tol = 0.1
- max_iter = 300