# Question 1 Solution

**Saranya Pal**
Department of Chemistry
210930
210930@iitk.ac.in

**Vaibhav Shaily**
Department of Aerospace Enginerring
211139
211139@iitk.ac.in

**Manish Sharma**
Department of Material Science Engineering
210585
210585@iitk.ac.in

**Prince Kumar**
Department of Material Science Engineering
210786
210786@iitk.ac.in

**Nirupam Das**
Department of Aerospace Engineering
231010048
231010048@iitk.ac.in

## Abstract

This study challenges the notion that Companion ArbiteR Physical Unclonable Functions (CAR-PUFs) are immune to prediction by linear machine learning models. CAR-PUFs, composed of working and reference PUFs, employ a secret threshold to determine responses based on timing differences. We demonstrate that despite Melbo's belief in the complexity of predicting CAR-PUF responses, there exists a linear model capable of perfect prediction given sufficient challenge-response pairs (CRPs). Leveraging data from a CAR-PUF with 32-bit challenges, our analysis reveals that by transforming challenge vectors, a linear model can accurately predict CAR-PUF responses. This research underscores the importance of scrutinizing purported security measures in PUF design and highlights the vulnerability of CAR-PUFs to linear model prediction.

## The Classifier Model for the Companion PUF:-

As per the question:-

If $|\Delta_w - \Delta_r| > \tau$, the model evaluates to 1; otherwise if $|\Delta_w - \Delta_r| \leq \tau$, the model evaluates to $-1$

Therefore, the classifier model has to be:-

$$\text{Model} = \frac{1 + \text{sign}(|\Delta_w - \Delta_r| - \tau)}{2}$$

We all know the nature of the **sign** function.

$$\text{sign}(x) = \begin{cases} -1 & \text{if } x < 0 \\ 0 & \text{if } x = 0 \\ 1 & \text{if } x > 0 \end{cases}$$

From this, we can conclude that the decision boundary of our model should be:-

$$|\Delta_w - \Delta_r| - \tau = 0$$

Preprint. Under review.

So, $|\Delta_w - \Delta_r| = \tau$. Squaring both sides:

$$(\Delta_w - \Delta_r)^2 - \tau^2 = 0$$

Expanding the terms, we get:

$$\Delta_w^2 - 2\Delta_w\Delta_r + \Delta_r^2 - \tau^2 = 0$$

Let **X** denote the feature vector provided to us in the form of training data.

$$\mathbf{X} \in \mathbb{R}^{32}$$

In this context, $\Delta_w$ and $\Delta_r$ can be effectively modeled using linear techniques since they adhere to the characteristics of standard arbiter PUFs.

$$\Delta_w = \tilde{W}_w^T\mathbf{X}$$
$$\Delta_r = \tilde{W}_r^T\mathbf{X}$$

Now let us represent the Linear model $\Delta_w$ with the parameters (**u**, p) and $\Delta_r$ with parameters (**v**, q). We must note that **u** and **v** are parameters of 32 dimensions whereas p and q are scalars.

$$\mathbf{u}, \mathbf{v} \in \mathbb{R}^{32}$$
$$p, q \in \mathbb{R}$$

Substituting in equation for the decision boundary, we get:

$$(\tilde{W}_w^T\mathbf{X})^2 - 2 \cdot (\tilde{W}_w^T\mathbf{X}) \cdot (\tilde{W}_r^T\mathbf{X}) + (\tilde{W}_r^T\mathbf{X})^2 - \tau^2 = 0$$

In order to further simplify it, we can represent the 32 dimensional parameters **u** and **v** as:

$$\mathbf{u} = (u_1, u_2, \ldots, u_{32})$$
$$\mathbf{v} = (v_1, v_2, \ldots, v_{32})$$
$$\mathbf{X} = (x_1, x_2, \ldots, x_{32})$$

So, expanding all of the terms, we get:-

$$(\tilde{W}_w^T\mathbf{X})^2 = \sum_{i=1}^{32} u_i^2x_i^2 + p^2 + \sum_{i=1}^{32}\sum_{j=i+1}^{32} u_iu_jx_ix_j + 2p\sum_{i=1}^{32} u_ix_i$$

$$(\tilde{W}_r^T\mathbf{X})^2 = \sum_{i=1}^{32} v_i^2x_i^2 + q^2 + \sum_{i=1}^{32}\sum_{j=i+1}^{32} v_iv_jx_ix_j + 2q\sum_{i=1}^{32} v_ix_i$$

$$\begin{aligned}(\tilde{W}_w^T\mathbf{X})(\tilde{W}_r^T\mathbf{X}) = \sum_{i=1}^{32} u_iv_ix_i^2 + 2pq + 2\sum_{i=1}^{32}\sum_{j=i+1}^{32}(u_iv_j + v_iu_j)x_ix_j \\ + 2p\sum_{i=1}^{32} v_ix_i + 2q\sum_{i=1}^{32} u_ix_i\end{aligned}$$

Thus, the decision boundary equation becomes:

$$\begin{aligned}&\sum_{i=1}^{32}(u_i^2 - 2u_iv_i + v_i^2)x_i^2 + (p^2 - 2pq + q^2) \\ &\quad + \sum_{i=1}^{32}\sum_{j=i+1}^{32}(4u_iu_j + 4v_iv_j - 2u_iv_j - 2u_jv_i)x_ix_j \\ &\quad + 2(p-q)\sum_{i=1}^{32}(u_i - v_i)x_i - \tau^2 = 0\end{aligned}$$

$$\Rightarrow \sum_{i=1}^{32}(u_i-v_i)^2x_i^2+\sum_{i=1}^{32}\sum_{j=i+1}^{32}(u_i+v_i)(2u_j-v_j)x_ix_j+2(p-q)\sum_{i=1}^{32}(u_i-v_i)x_i+[(p-q)^2-\tau^2]=0$$

Now, $x_i$ = 1 or $x_i$ = -1 are the possible values. So, when we use $x_i^2$, we actually convert all these values to 1. Thus the 32 terms of $x_i^2$ are all added to the bias term and becomes the new bias term. The new decision boundary thus becomes:

$$\sum_{i=1}^{32}\sum_{j=i+1}^{32}(u_i+v_i)(2u_j-v_j)x_ix_j+2(p-q)\sum_{i=1}^{32}(u_i-v_i)x_i+[(p-q)^2-\tau^2+c]=0$$

where c represents $\Rightarrow \sum_{i=1}^{32}(u_i-v_i)^2x_i^2$

Thus, We can represent the above decision boundary with the help of a linear model in 528 dimensions.

## Conclusion:-

So for a 32 dimensional vector, **c**, $\phi(\mathbf{c})$ is given as-

$$\phi(\mathbf{c}) = \{c_ic_j \mid c_i, c_j \in \mathbf{c}\ and\ i<j\} \cup \mathbf{c}$$

# Question 3 Solution


## Abstract

This study investigates the outcomes of utilizing the $sklearn.svm.LinearSVC$ and $sklearn.linear\_model.LogisticRegression$ methods for learning the behaviour of CAR-PUFs given some data of challenge-response pairs. Impacts of various hyperparameters, including the loss function, C value, tolerance, and penalty type, on accuracy and training time were explored. Results are presented through tables and charts, representing the effects of these hyperparameters on model performance across both SVM-LinearSVC and Logistic Regression methods.
All the tests were done on a private CPU with following specifications-
Processor- 12th Gen Intel(R) Core(TM) i5-1235U 1.30 GHz
RAM- 8.00 GB


## 1 Results obtained from Linear SVC

In this section, we performed training on our dataset utilizing the $sklearn.svm.LinearSVC()$ method. Optimal parameters for least accuracy loss and training time can be easily inferred from the following data.

### 1.1 Performance

Modelling the data using **SVM-LinearSVC**, resulted in prediction accuracy of 99.3% on the "test" dataset and an average training time of about 2.53 seconds on the "train" dataset.

### 1.2 Hyperparameter Tuning

99.3% accuracy was achieved for the default hyperparameters. However, improvement in accuracy was observed for some non-default values. A clear trend can be observed between the performance and the values of hyperparameters of the models. The plots below provides the details of these:-

- **Changing the *loss* hyperparameter**

  $sklearn.SVM$ provides two types of losses- "$hinge$" and "$squared\_hinge$"(default value). The $hinge$ loss can only be used for dual problem. However, $hinge$ fails to converge for the data even for 1000 iterations and any reasonable tolerance value.
  Following are the performance characteristics of LinearSVC for both loss values-

| Loss function | Accuracy | Training Time |
|---|---|---|
| **Hinge loss** | 88.46 | 6.7092147 |
| **Squared hinge loss** | 99.34 | 2.88874742 |

  It is easy to observe that $squared\_hinge$ performs better than $hinge$ loss function. The high training time for $hinge$ loss is because the model failed to converge for $hinge$ loss.

  Note that the data is for regularization parameter set to its optimal value (C=2.7).

Preprint. Under review.

- **Setting C to high/low/medium values**
  The regularization parameter, C controls the trade-off between maximizing the margin and minimizing the classification error. The value of C is set to 1 by default for LinearSVC. The following trend is observed in the performance of the model for varying values of C-

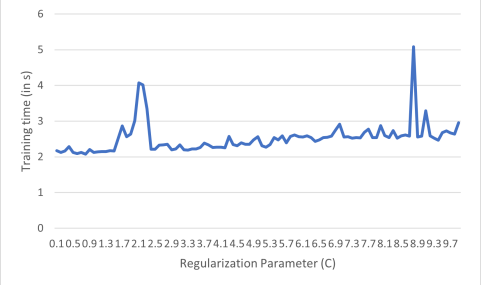


Figure 1: Training Time vs C

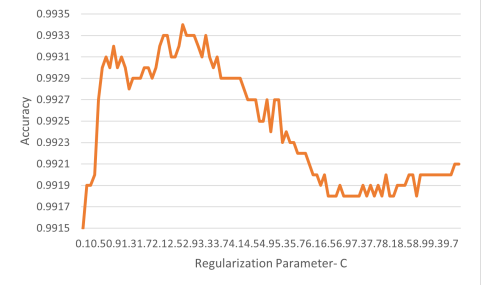


Figure 2: Accuracy vs C

From the above graph it is readily observed that the accuracy of the model first increases with the value of C, peaks around at about C=2.7 with accuracy of 99.34% and then drops down.
On the other hand, there is only a small increase in training time for increasing values of C. The random peaks in the graph might be due to the randomness of the system and not indicative of the model's reaction to change in C value.

- **Changing tol to high/low/medium values**
  The tolerance value signifies the threshold at which the convergence iteration ceases, indicating the stage where we ascertain that our classifier has converged. The value of tolerance is set to $10^{-4}$ by default. The Training time and Accuracy with **negative logarithm of Tolerance** values are plotted below.

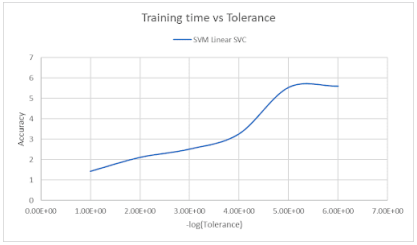


Figure 3: tolerance vs training time

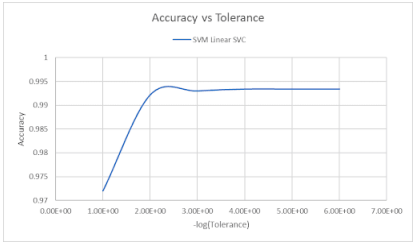


Figure 4: tolerance vs Accuracy

The above plots clearly show that the optimised value with respect to accuracy of tolerance is the default value which is $10^{-4}$. Also, it is clear that the training time increases as well as we keep on reducing the tolerance value.

- **Changing the penalty (l2 vs l1)**
  $svm.LinearSVC$ provides two types of penalties- $l1$, which is absolute loss, and $l2$, which is squared loss. $l2$ penalty is usually set as the default penalty parameter for $sklearn$ models. The model's convergence was unsuccessful when applying the l1 penalty, but switching to the l2 penalty facilitated successful convergence.

| Penalty | Accuracy (in %) | Training Time |
|---|---|---|
| **l1 Penalty** | 99.16 | 95.64494492 |
| **l2 Penalty** | 99.34 | 2.72156618 |

Using l1 penalty drastically increases the training time of the model without any significant increase in the accuracy. l2 penalty, on the other hand, had decidedly smaller training time and greater accuracy. Therefore, l2 penalty is undeniably the better choice of the two.

The above plots and tables explain the variation in our results when the hyperparameters of the LinearSVC are tuned.

## 2 Results obtained from Logistic Regression

Repeating all the above steps, but with $LogisticRegression$ instead of $svm.LinearSVC$ yields the following results. The accuracy obtained on the "$test$" after fitting to the "$train$" data was 99.28% with the default hyperparameters. This result is comparable with the one obtaied with LinearSVC classifier. This means now our model was able to classify 9928 data points correctly out of the given 10000 points.

### 2.1 Accuracy Score

Again as previous, the obtained 99.28% accuracy is on the hyperparameters set as default, which is comparable to the accuracy achieved by using svm model. However, the training time required by logistic regression is significantly less than that of svm at around 0.95 seconds.

### 2.2 Hyperparameter Tuning

The achieved 99.3% accuracy was based on the default hyperparameters. The plots below provides the details of these:-

- **Setting C to high/low/medium values**

  At default tolerance value, we set C to various range of values. However, the highest possible accuracy was reached when C was set to 6.0. It is observed that the accuracy of the model increases with increase in value of C at first and then reaches maximum at around C=6.0 after which it remains constant/

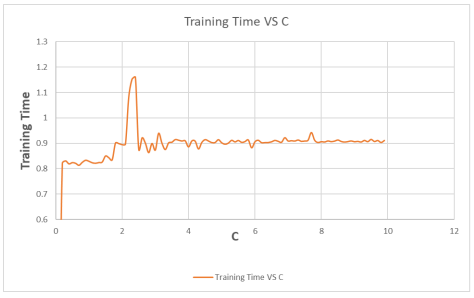


Figure 5: Training time vs C

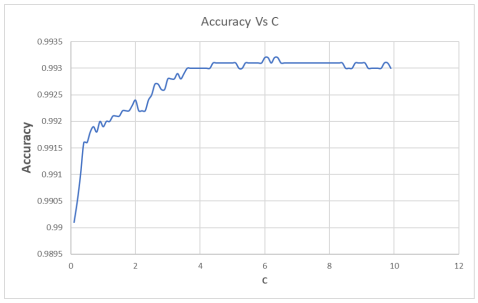


Figure 6: Accuracy vs C

- **Changing tol to high/low/medium values**

  The tolerance value was set to different values and its training time and accuracy was plotted. For a better visualisation purpose, we plotted these on the negative logarithmic of tolerance scale.The highest accuracy was obtained when tol was set to $10^{-4}$


Figure 7: log Tolerance vs Training Time

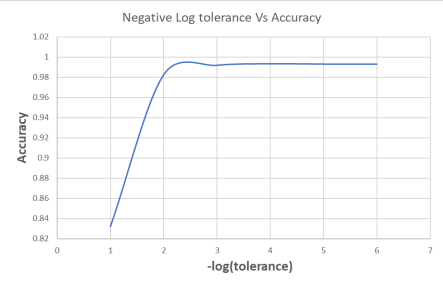


Figure 8: log Tolerance vs Accuracy

- **Changing the regularization hyperparameter in (l2 vs l1)**
  When we used the l1 regularization parameter, the model did not converge. It converged only when we used the l2 regularization parameter. Also, the standard sklearn library, only allows the use of l1 penalty with "saga" solver.

| **Penalty** | **Accuracy (in %)** | **Training Time** |
|---|---|---|
| **l1 Penalty** | 99.04 | 20.9712754 |
| **l2 Penalty** | 99.32 | 0.9552411 |

  It is evident that l1 penalty requires significantly more time and also performs poorly in accuracy in comparison to l2 penalty. Thus, l2 penalty is decidedly better for Logistic Regression also.

The above plots explain the variation in our results when the hyperparameters of the LogisticRegression are tuned

## 3 Comparison of the Hyperparameters from both the Classifiers

In this section, we will compare the results, namely the accuracy and the training time from both of our classifiers.

- **Modifying the C Hyperparameter**

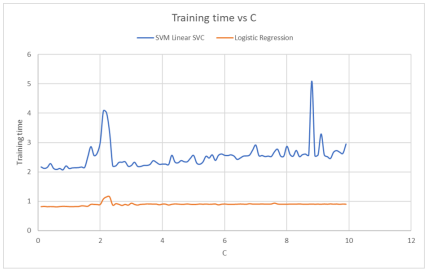


Figure 9: Training Time vs C

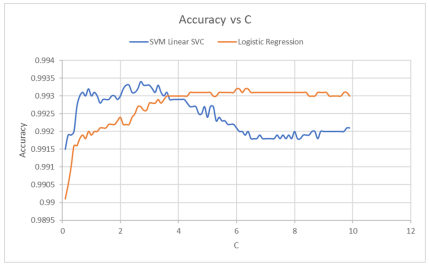


Figure 10: Accuracy vs C

- **Modifying the tolerance value** (Negative log tolerance is plotted on the x-axis)


Figure 11: Training Time vs log Tolerance

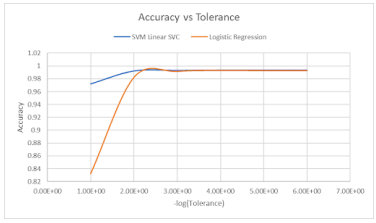


Figure 12: Accuracy vs log Tolerance

## 4 Conclusion

After comparing both of our models, we've observed that their performance is comparable in terms of accuracy. However, when considering training time, we find that Logistic Regression outperforms Linear SVC in this scenario.