Keep it Unsupervised: Horizontal Attacks Meet Simple Classifiers

Sana Boussam[2,3] and Ninon Calleja Albillos[1,4]

[1] CEA, LETI, MINATEC Campus, F-38054 Grenoble, France
[2] INRIA and LIX, Institut Polytechnique de Paris, Palaiseau, France
[3] Thales ITSEF, Toulouse, France
[4] Univ. Grenoble Alpes, F-38000, Grenoble, France
sana.boussam@inria.fr
ninon.callejaalbillos@cea.fr

Abstract. In the last years, Deep Learning algorithms have been browsed and applied to Side-Channel Analysis in order to enhance attack's performances. In some cases, the proposals came without an in-depth analysis allowing to understand the tool, its applicability scenarios, its limitations and the advantages it brings with respect to classical statistical tools. As an example, a study presented at CHES 2021 [16] proposed a corrective iterative framework to perform an unsupervised attack which achieves a 100% key bits recovery. In this paper we analyze the iterative framework and the datasets it was applied onto. The analysis suggests a much easier and interpretable way to both implement such an iterative framework and perform the attack using more conventional solutions, without affecting the attack's performances.

1 Introduction

The rise of DL-SCA. Side-Channel Attacks (SCA) are a type of physical attacks in which an attacker exploits implementation weaknesses of a cryptographic algorithm in embedded systems in order to extract secret information, by targeting all types of physical leaks (*e.g.* execution time, power consumption, electromagnetic radiation ...) generated during the execution of this algorithm. More generally, SCA fall into two categories: *profiled attacks* and *non-profiled attacks*. *Profiled attacks* are one of the most powerful SCA and require a device similar to the targeted one in order to build a learning database that models device behavior (*i.e.* leakage model) according to the values of sensitive variables that depend on the secret. One of the recent strategies, actively explored by the side-channel community since 2017, consists in using deep learning techniques to enhance the classical profiled attack scenario while mitigating the impact of some countermeasures (*e.g.* masking) [13]. On the other side, *non-profiled attacks* are less restrictive than *profiled attacks*. In this scenario, the attacker only captures physical traces and has possibly some details about the implementation of the targeted algorithm to carry out the attack. Therefore, one of the first difficulty he encounters consists in finding an adequate approach to extract

sensitive information from the targeted embedded system. While this restrictive scenario is more realistic than the profiled attacks from a practical point of view, the application of machine learning and deep learning techniques are less pronounced.

Motivations of our work. Recently, Perin *et al.* published in CHES 2021 [16] a *non-profiled* horizontal attack where they introduce a deep learning framework to correct errors after their clustering process. Correction of noisy labels is an open problem in machine learning. The methods used in the literature follow different approaches like using robust learning algorithms to mitigate the impact of noisy labels, noise identification and reduction techniques or even noise correcting methods [18]. Perin *et al.* are one of the first to present an attack followed by a deep learning correction framework, used as a black-box, which permits to retrieve 100% of correct bits for at least one key. While such contribution sounds promising for the community, the black-box property provided by the use of deep learning techniques can be problematic to assess the benefits and the limitations of such contribution. This raises the question of more appropriate and less costly solutions which can be beneficial from an interpretation point of view.

Contributions. This paper reduces this issue through the proposal of the following contributions:

- First, we identify that the datasets targeted by Perin *et al.* [16] are linearly separable. Then, we bring out the limitations of the corrective approach by showing that iterative framework's performance is strongly related to the neural network used and thus to the dataset considered. This illustrates the need of understanding how machine learning or deep learning techniques should be refined depending on use-case scenarios. Consequently, we adapt the iterative framework by justifying the design of a dedicated neural network structure. This results in an improvement in terms of network complexity, accuracy and computation time compared to the proposed attack. While the average accuracy is improved by up to 8%, the attack time is divided by 10. This contribution is reported in Section 4.
- To make our observation more consistent with a real use-case scenario, we assess the suitability of the corrective framework under a noisy environment and observe that such restriction highly impacts its performance. This contribution is reported in Section 5.
- As a consequence, we finally question the ability of a more conventional approach to conduct such attack scenario. In that purpose, we propose a *non-profiled* attack which is composed of two steps: a feature extraction based on principal components analysis and a k-means clustering performed onto the projected data. This attack is performed on the same datasets as those introduced in [16] and results are compared to those obtained by Perin *et al.*. Similarly, the average accuracy is improved by up to 8% with this deep learning free approach. This contribution is reported in Section 6.

Sections 2 and 3 provide the background and a succinct overview of the unsupervised attack proposed by Perin *et al.* [16].

2 Background

In this section we recall well-known terms and summarize well-known notions that we will use in the rest of the paper.

2.1 Supervised vs Unsupervised learning

Machine learning is a vast domain. The problems it addresses may be categorized in many ways. Here, we are interested in distinguishing problems addressed by the so-called *supervised* and *unsupervised* learning algorithms. Supervised learning algorithms construct solutions to problems asking to provide predictions on the basis of labeled examples (*i.e. training dataset*), *e.g.* binary or multi-class classification or regression. On the contrary, unsupervised learning algorithms are not provided with labeled examples and are asked to construct a function that highlights certain data structures. Examples of problems addressed by unsupervised learning are density estimation, clustering and dimensionality reduction.

2.2 T-test and Sum of Squared T-statistics (SOST)

The Welch's t-test is an adaptation of the Student's t-test that aims at determining if two populations with unequal variances have equal means. In the context of side-channel analysis, this statistical tool is used to capture the data dependency with the physical signal (*e.g.* EM emanations).
Let T_0 and T_1 be two sets of n_0 and n_1 side-channel traces of d time samples, such that the sensitive variable manipulated in the two groups is different (fixed in both groups or random in one and fixed in the other). Let also $\mu_{0,i}$ (resp. $\mu_{1,i}$) and $\sigma^2_{0,i}$ (resp. $\sigma^2_{1,i}$) be the mean and the variance of the emitted side-channel signal at sample i for the set traces T_0 (resp. T_1). The t-value associated with the ith sample is computed as:

$$t_i = \frac{\mu_{0,i} - \mu_{1,i}}{\sqrt{\frac{\sigma^2_{0,i}}{n_0} + \frac{\sigma^2_{1,i}}{n_1}}}, \qquad i \in \{1, ..., d\}.$$

In side-channel context, a large t-value is interpreted as a difference in behavior in terms of side-channel signal which is induced by the value of the sensitive variable handled during the execution of the algorithm. Samples with large t-value are thus points of interest (PoIs) (*i.e.* samples that potentially contain the most of relevant information related to the secret).
In 2006, Gierlichs *et al.* [8] introduced a PoI selection method based on the so-called Sum of Squared T-statistics (SOST), which is a multi-class generalisation of the t-test. Such a PoI selection method is a supervised dimensionality reduction tool.

2.3 Principal Components Analysis (PCA)

In the context of dimensionality reduction, Principal Components Analysis (PCA) is a well-established unsupervised technique. Instead of performing a point selection, it performs a *feature extraction*, orthogonally projecting the original data onto a lower-dimensional space. PCA requires to determine the Principal Components (PCs), which are the projecting vectors that maximizes the variance of the projected data. Interpreting the side-channel traces as realizations of a d-dimensional zero-mean random vector X, it can be shown that the PCs are the eigenvectors $\alpha_1 ... \alpha_d$ of the covariance matrix of X. As X is an unknown distribution, the empirical covariance matrix estimated from acquisition is used. The estimated variances of each projected component $Y_i = \alpha_i X$ is its corresponding eigenvalue λ_i. Thus, the PCs are typically ordered in decreasing order with respect to their corresponding eigenvalue, such that a natural choice to maximize the variance of the projected data consists in projecting onto the first PCs. In SCA context, it has been discussed that a major issue with PCA application consists in the choice of the PCs so that significant information is conserved. Various techniques and criteria have been proposed to solve this issue. In [4], the authors introduced a method using the Explained Global Variance (EGV). This method recommends using natural principal component order to select components. In [3], an alternative selection method, is proposed: the Explained Local Variance (ELV) selection method consists in computations of local variance for each time sample j and component α_i. The sum of such variance, $ELV(\alpha_i, j)$, over all samples j gives $EGV(\alpha_i)$. Thus, this component selection method, based on the globally-accepted assumption that side-channel information is localised in few PoIs, strives for a balance between the EGV of a given component and the number of samples required to achieve consistent part of it.

Definition 1. *Given a principal component α_i and a sample j, the Explained Local Variance is defined by :*

$$ELV(\alpha_i, j) = \frac{\lambda_i \times \alpha_i[j]^2}{\sum_{k=1}^{r} \lambda_k} = EGV(\alpha_i)\alpha_i[j]^2$$

with $EGV(\alpha_i) = \frac{\lambda_i}{\sum_{k=1}^{r} \lambda_k}$, r the rank of the covariance matrix and λ_i the eigenvalue of the ith principal component.

The ELV can be sorted in descending order for each component and summed in a cumulative way. From this, we can observe trends followed by components to achieve their EGV and deduce a ranking of all PCs.

2.4 Clustering algorithms, Voronoi diagrams and *k*-means

Clustering algorithms are used to find a structure in a unlabeled dataset. It is a partitioning process which gathers similar objects into groups according to defined metrics. There are different types of clustering algorithms. If all clusters

are determined simultaneously, the algorithm is called partitional. Otherwise, if the clusters are established using previous clusters, the algorithm is hierarchical. When the number of clusters is known, which is the case in this work, partitional algorithms are preferred. One of the most well-established partitional clustering algorithm is the k-means. The k-means clustering consists in determining a set of k points in $\mathbb{R}^d$ called centroids that minimize the distance between each data point and its nearest centroid. Finding the optimal solution of a k-means clustering problem is NP-hard even when $k = 2$ (see [1] proof). One of the best known heuristics for solving k-means clustering problem is Lloyd's algorithm [11] and its generalizations [10]. This algorithm gives an approximate solution to this problem by seeking a local minimum instead of the global one. To do so, k-means clustering problem is approximated by an iterative computation of *Voronoï diagrams* (or *tessellations*) of k regions until a convergence criterion is satisfied. This results in a *centroidal Voronoï tessellation* of the input [5]. Let $\mathcal{E}$ be an Euclidean space of finite dimension and $P = \{\mathbf{p}_1, ..., \mathbf{p}_k\}$ be a finite set of k points called *generators* in $\mathcal{E}$.

Definition 2 (Voronoï diagrams). *A Voronoï diagram $\mathcal{V}_P$ is a partition of $\mathcal{E}$ into a set of k regions V_i such that $\bigcup_{i=1}^{k} V_i = \mathcal{E}$ and $\forall\,(i,j) \in \{1, ..., k\}^2$, $i \neq j$, $V_i \cap V_j = \emptyset$. The Voronoï region V_i corresponding to the generator $\mathbf{p}_i$ is defined as:*

$$V_i = \{\mathbf{x} \in \mathcal{E} \quad | \quad |\mathbf{x} - \mathbf{p}_i| < |\mathbf{x} - \mathbf{p}_j| \quad \text{for } j = 1, ..., k,\ j \neq i\}$$

Definition 3 (Centroidal Voronoï tesselation). *A centroidal Voronoï tessellation (CVT) is a particular type of Voronoï diagram in which the generators $\mathbf{p}_i$ of Voronoï regions V_i are their mass centroids μ_i.*

Considering the fact that k-means algorithm generates a CVT, the decision boundaries between the k clusters found by a k-means are therefore linear since they correspond to the boundaries between the k Voronoï regions of the generated CVT. Indeed, given a Voronoï region V_i, $i \in \{1, ..., k\}$, the decision boundary between V_i and $V_j \in \mathcal{A}(V_i) \subset \mathcal{V}_P$, with $\mathcal{A}(V_i)$ the set of the adjacent Voronoï regions of V_i, is an affine hyperplane defined by the set of points equidistant from μ_i and μ_j [7].

2.5 Perceptrons and Deep Neural Networks

A so-called *perceptron* is the simplest neural networks since it consists of an artifical neuron. This learning algorithm is a binary linear classifier. Let consider $\Theta = \{W, b\}$ as a set of parameters. Given an input x, a perceptron learns a parametric linear function $f_\Theta : \mathbb{R}^d \to \mathbb{R}$ of the form $f_\Theta(x) = W^T x + b$ where $W \in \mathbb{R}^d$ is a vector which represents the *weights* associated with x and $b \in \mathbb{R}$ is a *bias*. Its output is determined by the application of a non-linear function σ called *activation function*. Indeed, given a threshold $\tau \in \mathbb{R}$, $y = 1$ if $\sigma(f_\Theta(x)) > \tau$ and 0 otherwise. In binary classification, the *activation function* used is the *Sigmoid function* $\sigma : \mathbb{R} \to [0, 1]$; also named *logistic function*. This

function is defined as $\sigma(f_\Theta(x)) = \frac{1}{1+e^{-f_\Theta(x)}}$ and aims at converting, given the input x, $f_\Theta(x)$ into a probability for x to belong to class 1. Since the output of the *Sigmoid function* is a probability, the threshold τ is set to 0.5 (*i.e.* the case when x lies on the decision boundary). The decision boundary characterized by a perceptron is thus an affine hyperplane defined by $W^T x + b = 0$.

Deep neural networks (DNN) are neural networks with multiple layers of artificial neurons that can learn more complex parametric functions (*i.e.* non linear functions). Among the several typology of DNN existing in literature, Multi-Layer Perceptrons (MLP) and Convolutional Neural Networks (CNN) are the most widely used in SCA domain in order to perform classification task. The MLP consist in concatenations of layers (also called dense layers) comprising several artificial neurons and activation functions. The CNN are similar to MLP, but adds some special layers (convolutional and pooling layers) that allows them to extract geometrically local features and to be robust to geometrical deformations, such as misalignment.

3 Overview of [16]

In this section we report a succinct summary of the unsupervised attack proposed in 2021 by Perin *et al.*

Target. The attack's targets are the protected ECC software implementations on Curve25519 from μNaCl [6], a cryptographic library for ARM Cortex-M[5]. The library includes two regular and constant-time elliptic curve scalar multiplication (ECSM) implementations based on the Montgomery ladder. In [14], the authors strengthen side-channel resistance of these ECSM by adding coordinate re-randomization to both implementations. Although scalar randomization countermeasure is not implemented, Perin *et al.* collect side-channel traces using a random scalar for each ECSM execution, in such a way that the proposed attack could be considered effective also against implementations protected with scalar randomization.

Vulnerability. The vulnerability the attack exploits is the leakage coming from a conditional swap operation (*CSWAP*) [14] during the ECSM. Indeed, during the ith round of the main multiplication loop, the *CSWAP* condition depends directly on the ith bit of the secret scalar.[6] Hence, an attacker that can classify leakages from CSWAP operations with respect to their condition can recover the 255 secret bits of the key bit per bit.

[5] https://munacl.cryptojedi.org/curve25519-cortexm0.shtml

[6] More specifically, the swap condition value at the ith iteration is equal to the XOR between the ith bit and $(i-1)^{th}$ bit of the secret scalar.

Datasets & scenario. Perin *et al.* conducted their attack on two datasets corresponding respectively to *CSWAP*'s implementation using pointer or arithmetic swapping: `cswap_pointer` and `cswap_arith`. Each dataset is composed of $N = 300$ complete traces. Each trace is split in 255 subtraces, corresponding to the execution of a single CSWAP operation. Thus the dataset contains $300 \times 255 = 76,500$ subtraces of $1,000$ time samples for `cswap_pointer` and $8,000$ time samples for `cswap_arith`. The complete attack proposed in [16] comprises two phases, namely a prelabeling and a corrective phase. The **prelabeling phase** is decomposed into two steps. The first one uses an adapted unsupervised version of SOST [15] in order to select a set of POIs. Then, the k-means algorithm is applied on the selected POIs in order to group the traces corresponding to the same value of CSWAP condition. After this prelabeling phase, the resulting accuracies found in [16] are respectively of 52.44% and 52.24% on `cswap_arith` and `cswap_pointer` datasets. The **corrective phase** takes the prelabeling results as input and applies a corrective framework[7] in order to correct wrong labels. Based on `Co-teaching` strategy [9], it consists in iteratively correcting mislabeled traces by training twice[8] a neural network *ANN* (which is a CNN in [16]) to perform a classification task on two disjoints parts $\mathcal{D}_1$ and $\mathcal{D}_2$ of the complete dataset $\mathcal{D}$. After each training, the neural network is used to relabel the subdataset on which it did not train. The label correction provided by this framework relies on the ability of *ANN* to extract discriminative features despite the presence of noise in labels. The interested readers may refer to [16, section 4] for deeper details on iterative framework.

Performance metric. For each dataset, training is done using 250 complete traces (*i.e.* $250 \times 255 = 63,750$ subtraces). Attack is performed using the 50 remaining traces (i.e. $50 \times 255 = 12,750$ subtraces). To estimate iterative framework's performance, the authors defined two metrics called *Average and Maximum single trace accuracy metric*. The maximum (resp. average) single trace accuracy is the maximum (resp. average) accuracy obtained using the iterative framework on these 50 remaining traces. An attack is considered to be successful if a maximum single trace accuracy of 100% is reached by the framework for at least one trace at any epoch.

Experimental results achieved in [16]. Perin *et al.* assess iterative framework's correction capability, using the previous defined metrics, following four different scenarios: (S_1) CNN without regularization[9], (S_2) CNN with data augmentation, (S_3) CNN with dropout, (S_4) CNN with data augmentation and dropout. In this paper, only the results related to (S_1) and (S_4) are reported as

[7] Source code: https://github.com/AISyLab/IterativeDLFramework

[8] At each training phase, trainable paramaters are reset.

[9] Regularization consists in applying a technique to avoid overfitting. Overfitting is a phenomenon that occurs when a neural network fits exactly the training dataset, which causes a lack of generalization (*i.e.* poor performance on unseen data). Data augmentation [17] and dropout [19] are two well-known regularization techniques which reduce this issue.

a particular focus is proposed on those scenarios (see table 1 which summarizes results achieved for fixed CNN hyperparameters and 50 framework iterations[10]).

Table 1: Average and maximum single trace accuracies obtained in [16] for fixed CNN hyperparameters (50 framework iterations).

	`cswap_pointer`		`cswap_arith`	
	Average	Maximum	Average	Maximum
After prelabelling phase	52.24%	59.22%	52.44%	59.22%
Scenario 1: no regularisation	85%	97.64%	52%	76.07%
Scenario 4: dropout + data augmentation	91%	100%	83%	100%

The fact that Perin *et al.* achieve a successful attack only when applying regularization techniques suggests that their neural network overfits. The overfitting phenomenon can be due to several reasons [20] such as a lack of training data or a high model complexity (in terms of number of trainable parameters). For the latter reason, the function induced by the neural network, does not fit with the underlying classification problem. Here, in the case of [16], we assume that overfitting is induced by the model complexity, which was not considered in the original paper. In the following, we propose in section 4.1 to simplify the approach introduced by Perin *et al.* through the analysis of the decision boundary's complexity. Based on the results found, we propose a new neural network that is more suitable for the classification problem inherent in both datasets.

4 Corrective framework fitting the decision boundaries

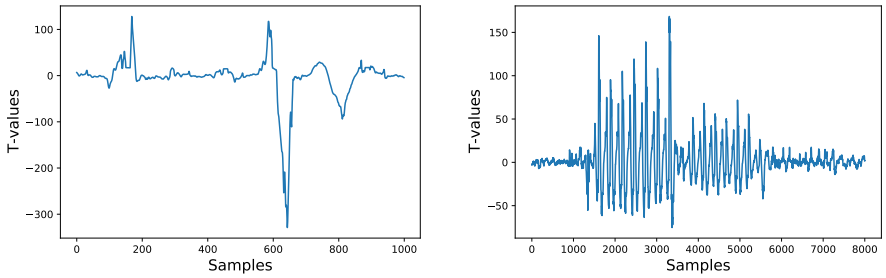
In this section, we aim at providing a better understanding of the attack scenario proposed in [16]. To do so, we conduct an in-depth analysis of this work and propose a new version of this attack with more suitable tools for `cswap_pointer` and `cswap_arith` datasets.

4.1 Simplification of the neural network

As mentioned above, we assume that overfitting is due to a high model complexity. To confirm our assumption, we first conduct additional investigations in order to characterize the decision boundaries of both datasets. Then, based on the results found, we simplify the new neural network such that we obtain a classifier more suitable for the classification problems.

[10] These results are those reported in [16] as we were not able to reproduce it using the source code.

Linear decision boundaries. The characterization of the decision boundaries is done in two steps. First, we conduct a (supervised) PoIs selection. Then, we determine the shape of the decision boundary by applying simple tools such as k-means and examining their performance (*i.e.* accuracy obtained). Fig. 1 shows Welch's t-tests (recalled in Sec. 2.2) obtained on both datasets, using their true labels. As we can see, both datasets have significant side-channel leakages. Samples that maximize the t-value seem to be sampled around 600-650 for `cswap_pointer` and around $3,300$-$3,350$ for `cswap_arith`.



(a) `cswap_pointer` dataset (b) `cswap_arith` dataset

Fig. 1: Welch's t-tests obtained on the two datasets using true labels.

By applying the same methodology as Perin *et al.*, we first compute a k-means clustering on the PoIs of the two datasets (*i.e.* samples 600-650 for `cswap_pointer` and samples $3,300$-$3,350$ for `cswap_arith`) using 250 of the 300 complete traces (*i.e.* $63,750$ subtraces). Then we predict cluster index for the $50 \times 255 = 12,750$ remaining subtraces. We obtain an maximum and average single trace accuracy of 96.1% and 98.03% (resp. 80.26% and 87.05%) for `cswap_pointer` dataset (resp. `cswap_arith` dataset). These results highlight that a k-means is sufficient to discriminate subtraces processing bit 0 and bit 1. Therefore, we deduce that the classification problem is not so complex to solve as both classes can be separable by a k-means decision boundary. According to Sec. 2.4, a hyperplane seems to be sufficient to discriminate the two clusters (*i.e.* subtraces processing bit 0 and bit 1). This means that a linear classifier would be more suitable for the underlying classification problem of both datasets.

Linear classifier. Starting from the observations above, we simplify the neural network proposed in [16] in order to adapt it to a linear classification problem. In fact, a linear classifier that models a linear function, such as a perceptron (see Sec. 2.5), should be sufficient to approximate the true unknown decision boundary. To verify this assumption, we reduce the complexity of the CNN architecture used in [16] (see Tab. 2a) until we obtain a neural network composed of a single layer of 1 neuron (see Tab. 2b).

Table 2: Comparison of neural network architectures.

	Layers	`cswap_pointer`	`cswap_arith`
	Inputs	1,000 samples	8,000 samples
Feature learning	AvgPooling1D	×	✓
	Conv1D_1	8 filters	8 filters
	MaxPooling1D	×	✓
	Conv1D_2	16 filters	16 filters
	Conv1D_3	32 filters	32 filters
Classification	Dense_1	100 neurons	100 neurons
	Dense_2	100 neurons	100 neurons
	Softmax	2 neurons	2 neurons

(a) CNN architectures considered in [16]

Layers	`cswap_pointer`	`cswap_arith`	
Inputs	1,000 samples	8,000 samples	
Sigmoid	1 neuron	1 neuron	Classification

(b) Proposed neural network architecture

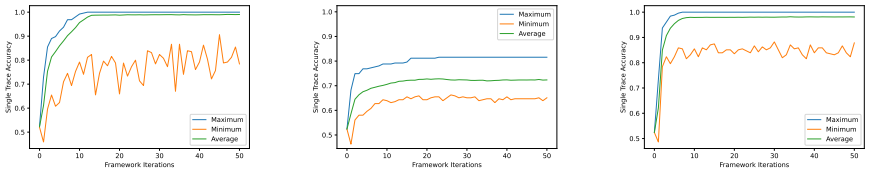
4.2 Experimental results obtained on attacked datasets

This section presents the results achieved on both datasets with this new neural network according to the setting (supervised/unsupervised context) considered.

Supervised learning. In order to confirm our hypothesis that the data are linearly separable and to assess the suitability and effectiveness of our proposed architecture, we first try this neural network in a supervised context (*i.e.* training is done using the true labels). To do so, for both datasets, we define a single layer of one neuron and use a *sigmoid* activation function. We use *RMSprop* optimizer and set the learning rate to 10^{-4}. Training is done for 20 epochs with a batch size of 100. No regularization technique is applied. We use the same methodology as Perin *et al.*, namely we first train our neural network with 250 complete traces = 63,750 subtraces and then we attack the remaining 50 traces = 12,750 subtraces. After training our neural newtwork, we achieve for both dataset a maximum single trace accuracy of 100%. The average single trace accuracies are respectively 99.77% and 98.33% for `cswap_pointer` and `cswap_arith` datasets. These results appear to clearly confirm our previous hypothesis: both datasets are linearly separable.

Unsupervised learning. Once we validate our hypothesis in a supervised learning context, we assess the ability of our new neural network to correct the mislabeled data by following the methodology provided in [16]. As a remainder, after the prelabeling phase, Perin *et al.* retrieves 52.24% (resp. 52.44%) of the private key bits when the `cswap_pointer` (resp. `cswap_arith`) dataset is considered. To configure our neural network in the iterative framework, we use the same hyperparameters as those configured in the supervised context. Only the

batch size and the number of epochs differ, we set them to 250 and 10 respectively. Furthermore, we keep the same methodology as the one adopted in [16] (*i.e.* training on 250 complete traces/attack on the 50 remaining traces).
Fig. 2 shows the results we obtain using the same inputs (*i.e.* subtraces and labels) as Perin *et al.* and without regularization. As we can see, after 50 iterations, the framework achieved a maximum (resp. average) single trace accuracy of 100% (resp. 98.9%) for `cswap_pointer` dataset (see Fig. 2a). Hence, for this dataset, we conduct a successful attack using a perceptron. However, it seems to be more complicated for `cswap_arith` dataset. Indeed, for this dataset, the framework reached a maximum (resp. average) single trace accuracy of 79.22% (resp. 70.93%) (see Fig. 2b). These results are better than those obtained by Perin *et al.* (scenario without regularization) but still not sufficient to consider the attack as successful. As in [16], we decide to reduce the size of subtraces of `cswap_arith` dataset in order to see if when considering a smaller number of samples, the framework manages to correct mislabeled subtraces. Thus, we narrow down as in [16] the subtraces of `cswap_arith` from $8,000$ to $2,000$ samples by considering only the interval of samples $1,500$-$3,500$. We call this dataset `cswap_arith_reduced`. Fig. 2c shows the results achieved by the framework on this reduced dataset. By narrowing down the attacking sample interval, we now obtain a successful attack with a perceptron for `cswap_arith`. Indeed, when considering only the sample interval $1,500$-$3,500$, the framework reached a maximum (resp. average) single trace accuracy of 100% (resp. 98%). Thus, these results, obtained without regularization and with a very simple neural network, highlight the fact that even in an unprofiled context, a linear classifier is sufficient to correct mislabeled subtraces.

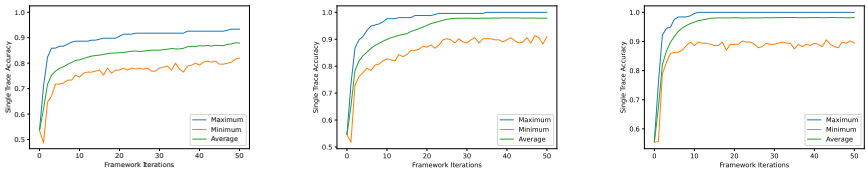


(a) `cswap_pointer` dataset (b) `cswap_arith` dataset (c) `cswap_arith_reduced` dataset

Fig. 2: Results obtained using iterative framework with our neural network after 50 iterations (no regularization).

Although we managed to carry out successful attacks for both datasets, we do not explain why, when considering the $8,000$ samples (*i.e.* considering entire subtraces and not a restricted sample interval), we are unable to conduct a successful attack even though the neural network we consider is adapted to the classification problem inherent to `cswap_arith` dataset.
Since changing the hyperparameters (number of epochs and batch size) did not solve the problem, we decide to investigate the impact of increasing the initial

accuracy on the framework to see if it is still possible to achieve a successful attack on `cswap_arith` dataset. To do so, we create new labels by applying a random uniform noise on the true labels. Fig. 3 shows the results achieved by the iterative framework for `cswap_arith` dataset according to the accuracy related to the prelabeling phase. This accuracy will be called the "initial accuracy" in the rest of the paper. Indeed, when we increase the initial accuracy by a small amount *i.e.* from 52.44% (initial accuracy found by Perin *et al.*) to 53.66%, we still have the same behavior as in the previous experiments (see figure 3a). However, in figure 3b, we can notice that from an initial accuracy of about 54.57%, we can finally achieve a successful attack for `cswap_arith` dataset. With an even higher accuracy, the iterative framework seems to find more quickly all the bits of at least one trace, as we can see in Fig. 3c. The obtained results illustrate that the performance of the iterative framework highly depends on the quality of the prelabeling process. Even if the decision boundary to approximate is simple (*e.g.* linear decision boundary), the inappropriate use of the prelabeling phase can highly impact the ability of the iterative framework to correct the mislabeled subtraces.



(a) Initial accuracy: 53.66% (b) Initial accuracy: 54.57% (c) Initial accuracy: 55.45%

Fig. 3: Results achieved for `cswap_arith` dataset using iterative framework with our neural network after 50 iterations (no regularization).

4.3 Comparison of the two approaches and discussion

Tab. 3 compares the results achieved for the same inputs and methodology according to the neural network considered in this work and in [16]. We observe that our proposition achieves better results than the CNN on both datasets.
For the `cswap_pointer` dataset, our perceptron without regularization retrieves on average 98.9% of the bits while the CNN used in [16] reaches an average accuracy of 91% when regularization techniques are applied. This enhancement allows us, on average, to complete the attack using brute force regardless of the trace considered. Indeed, an average single trace accuracy of 98.9% means that there are on average 3 remaining bits to correct in order to recover the secret scalar. This correction can be done by performing $n = \sum_{i=0}^{3} \binom{255}{i} \sim 2^{22}$ tests . In comparison with the results reported in [16], an average single trace accuracy of 91% corresponds to a 23-bit correction *i.e.* $\sim 2^{109}$ tests to recover

the secret key. In this case, the use of brute force to complete the attack is not feasible. For `cswap_arith_reduced` dataset, our neural network without regularization obtains similar results as [16] when they use regularization techniques. Here again, on average, the secret scalar can be retrieved using brute force (correction of 6 bits $\sim 2^{39}$ tests). Although our proposition does not obtain better results than the scenario with regularization for `cswap_arith` dataset, we still achieve better results than the CNN used without regularization. More generally, in a scenario without regularization, an improvement of average accuracies of 13.9% (`cswap_pointer` dataset), 18.93% (`cswap_arith` dataset) and 40% (`cswap_arith_reduced` dataset) can be observed by finding a neural network that suitably approximates the linear decision boundary.

Table 3: Comparison of average and maximum single trace accuracy obtained after 50 framework iterations for both datasets.

	`cswap_pointer`		`cswap_arith`		`cswap_arith_reduced`	
	Average	Maximum	Average	Maximum	Average	Maximum
1 neuron, no regularisation	98.9%	100%	70.93%	79.22%	98%	100%
CNN [16], no regularisation	85%	97.64%	52%	76.07%	58%	86.27%
CNN [16], dropout + data augmentation	91%	100%	83%	100%	98%	100%

Moreover, using a suitable neural network allows us to reduce the network complexity. Indeed, through the simplification of the neural network, the number of trainable parameters decreased from respectively $45,978$ (`cswap_pointer` dataset), $3,069,820$ (`cswap_arith` dataset) and $669,820$ (`cswap_arith_reduced` dataset) to $1,001$, $8,001$ and $2,001$. This enhancement highlights the high complexity of the neural network used by Perin *et al.* and thus confirms the assumption we made about the overfitting that occurs in [16]. Finally, this enhancement allows us also to significantly reduce the computational time. Using a 1.8 GHz Intel Core i7-10510U processor, the attack (*i.e.* 50 framework iterations) duration drops from respectively 1 hour and 15 minutes (`cswap_pointer` dataset), 18 hours (`cswap_arith` dataset) and 5 hours (`cswap_arith_reduced` dataset) down to respectively 8 minutes, 35 minutes and 10 minutes.

Finally, through this section, we saw that the performance of the iterative framework is strongly related to the neural network used, which is itself very dependent on the dataset considered. As a general fact, deep-learning techniques provide advantages with respect to conventional techniques when one has to tackle with noisy data or complex problems, *e.g.* non-linear decision boudaries. Datasets used in [16] do not have such characteristic, thus do not allow to assess the suitability of the iterative corrective framework in interesting cases. To get a

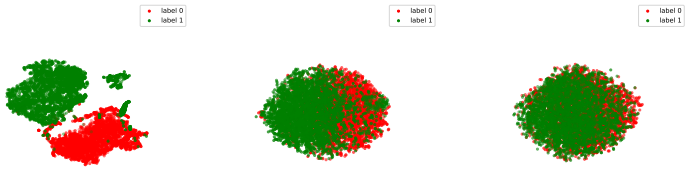
full overview of the corrective framework capability, next section identifies its benefits and its limitations under a noisy environment.

5 Correction capability of the iterative framework

In this section, we question the ability of the corrective approach to perform correctly when the clusters are entangled. To do so, we detail our methodology in Sec. 5.1 and introduce the results we obtained in Sec. 5.2.

5.1 Experiments

In order to assess the correction capability of the iterative framework on more noisy datasets we consider `cswap_pointer` and `cswap_arith_reduced` datasets , with an additional artificial Gaussian noise following $\mathcal{N}(0, \sigma^2)$, with $\sigma \in \{15, 25, 30, 50, 75, 90\}$. This is beneficial to generate different clusters entanglement levels. Fig. 4 shows examples of clusters entanglement on `cswap_pointer` traces for different σ values. As expected, the higher the noise level, the more entangled the clusters become. Therefore, they can no longer be separated by a linear decision boundary.



(a) Without noise (b) Adding a Gaussian noise ($\sigma = 30$) (c) Adding a Gaussian noise ($\sigma = 90$)

Fig. 4: Visualization of `cswap_pointer` dataset's clusters using t-SNE [12]

For each noise configuration, we consider two kind of neural networks architectures, namely perceptron and CNNs. Those two structures will produce different hypothesis space $\mathcal{F}$. We perform a supervised training in order to first ensure the presence of leakages when we entangle the clusters, and then, to obtain a neural network $f_{\text{sup}} \in \mathcal{F}$ that we considered as a correction upper bound. Then we apply the unsupervised iterative corrective strategy (using the same prelabels provided in [16]) and compare the obtained neural network f to f_{sup}. If f performs similarly to f_{sup}, we deduce that the unsupervised property of the iterative framework does not affect the attack performance, and thus that the iterative approach is suitable to deal with entangled clusters. If f has poorer performance than f_{sup}, we deduce that the iterative framework is not able to extract an efficient predictive model under the assumption of entangled clusters.

5.2 Results

Supervised learning. For both perceptron and CNN, training is done for 20 epochs and a batch size of 100. As in [16], we use *RMSprop* as optimizer with a learning rate set to 10^{-4}. As we can see in Tab. 4, adding Gaussian noise with standard deviation up to $\sigma = 50$ on traces results in non-linear separable clusters and leads the performance of the perceptron to be slightly lower than the CNN. Beyond $\sigma = 50$, performances of both neural networks are deteriorated by noise for both datasets.

Table 4: Average and maximum single trace accuracy achieved for `cswap_pointer` (1) and `cswap_arith_reduced` (2) datasets in a supervised setting.

		$\sigma = 15$		$\sigma = 25$		$\sigma = 30$		$\sigma = 50$		$\sigma = 75$		$\sigma = 90$	
		Average	Maximum	Average	Maximum	Average	Maximum	Average	Maximum	Average	Maximum	Average	Maximum
(1)	CNN [16]	99.87%	100%	99.32%	100%	98.83%	100%	93.11%	96.47%	84.66%	89.41%	79.43%	87.84%
	Perceptron	98.87%	100%	97.8%	99.6%	96.66%	98.82%	90.19%	95.29%	79.62%	84.7%	77.1%	83.13%
(2)	CNN [16]	97.46%	99.21%	92.13%	96.07%	89.84%	94.12%	73.52%	81.96%	67.78%	75.29%	64.56%	72.94%
	Perceptron	93.29%	96.86%	87.45%	91.76%	84.83%	89.01%	72.76%	84.31%	67.63%	74.11%	65.3%	74.11%

Unsupervised learning. For the unsupervised setting, we adopt for CNN the configuration in which regularization techniques are employed (see Tab. 1). Regarding hyperparameters for training, we pick the same ones used in supervised context. The only difference lies in choosing the batch size for the perceptron and the number of epochs for both neural networks. We set them respectively to 250 and 10. For all σ values, we summarize in Tab. 5 accuracies obtained after 50 framework iterations for `cswap_pointer` and `cswap_arith_reduced` datasets. As we can see in Tab. 5, for both neural networks the behavior of the iterative framework differs depending on noise level. As expected, the higher the noise, the more the decision boundaries are non-linear. Therefore, using the iterative framework with a CNN seems more suitable when σ increases. Indeed, when σ equals up to 30 (resp. 50) for `cswap_pointer` (resp. `cswap_arith_reduced`) dataset, the results obtained with a CNN are better than those obtained with a perceptron. However, when considering $\sigma \geq 50$ (resp. $\sigma \geq 75$) for `cswap_pointer` (resp. `cswap_arith_reduced`) dataset, CNN performances are on average lower than perceptron performances. This phenomenon could be explained by the fact that deep neural networks have a high capacity to memorize noisy labels (*i.e.* they can easily fit random labels) [21,2]. Thus it seems to appear that, in an unsupervised setting, the more the clusters are entangled, the less correction is provided by the framework.

Table 5: Average and maximum single trace accuracy achieved for `cswap_pointer` (1) and `cswap_arith_reduced` (2) datasets after 50 framework iterations.

		$\sigma = 15$		$\sigma = 25$		$\sigma = 30$		$\sigma = 50$		$\sigma = 75$		$\sigma = 90$	
		Average	Maximum	Average	Maximum	Average	Maximum	Average	Maximum	Average	Maximum	Average	Maximum
(1)	CNN [16]	97.11%	99.22%	98.62%	100%	71.25%	81.96%	50.73%	65.49%	50.68%	63.92%	50.78%	60.78%
	Perceptron	98.21%	100%	64.03%	78.43%	67.15%	76.08%	62.73%	72.55%	57.99%	71.76%	56.43%	66.27%
(2)	CNN [16]	90.26%	97.65%	80.6%	89.8%	66.28%	80.39%	71.87%	82.75%	50.26%	61.96%	50.13%	63.92%
	Perceptron	88.07%	95.67%	68.81%	81.96%	63.03%	75.29%	58.84%	70.2%	54.74%	67.06%	54.22%	65.88%

Indeed, for `cswap_pointer` dataset, average single trace accuracy decreases from 97.11% (resp. 98.21%) when $\sigma = 15$ for CNN (resp. perceptron) to 50.78% (resp. 56.43%) when $\sigma = 90$. Based on the results obtained from Tab. 4, note that a huge performance gap is observed between the supervised and the unsupervised settings. While the predictive model f_{sup} can retrieve at least 90% of the private key bits when the standard deviation of the applied Gaussian noise is up to $\sigma = 50$, the unsupervised approach affects the selection of the predictive model $f \in \mathcal{F}$ by reducing for CNN (resp. perceptron) the correction process by 43% (resp. almost 30%) when $\sigma = 50$. Therefore, the noisy configuration blurred the iterative framework to correct the wrong bit predictions even if the prelabeling phase retrieves 52.44% of the private key bits (same prelabeling configuration as in [16]). This observation suggests that some improvement can be provided to the iterative framework in order to reduce this performance gap while preserving its unsupervised property. In addition, depending on the noise level, if the prelabeling phase retrieves 52.44% of the private key bits, the use of the iterative framework does not improve the bit correction such that, in the worst use-case scenario (*i.e.* $\sigma = 90$), the iterative framework degrades the attack performance. This observation illustrates the limitation of this machine learning tools. In particular, it cannot extract a (non-)linear separation between the entangled clusters if it does not exist in the first place.

The same observations hold true for `cswap_arith_reduced` dataset: the higher the noise level, the harder it becomes for the framework to correct wrong bits regardless of the neural network used. As mentioned in Sec. 5.1, this suggests that the unsupervised property of the iterative framework affects the selection of the predictive model $f \in \mathcal{F}$. Therefore, even if the prelabeling phase retrieves 52% of the private key bits for both datasets, the suitability of the tool proposed in [16] should be shaded when it deals with noisy traces which is a classical setting in side-channel context.

Through this section, we identified that the efficiency of the proposed framework can be altered under a noisy environment. Indeed, this experiment illustrates the limitation of the iterative approach to correctly deal with entangled clusters in unsupervised setting as the performance gap between profiled and un-

profiled setting increases with the amount of noise. Therefore, we question the need of using advanced statistical tools against the one classically introduced in the side-channel community (*e.g.* dimensionality reduction techniques, clustering algorithms, ...). In that purpose, the following section proposes an easier alternative and introduces a new strategy based on the combination of PCA and k-means.

6 Deep learning free approach

Now that we have highlighted the limitations of the methodology proposed in [16], we want to show that nor correction techniques nor deep learning are required for this attack to be effective. Indeed, this section addresses a more suitable unsupervised attack on `cswap_pointer` and `cswap_arith` datasets using classical algorithms and techniques.

6.1 Alternative attack: overview

The alternative DL-free strategy that we propose consists in two steps:

- PCA feature extraction: as we have seen in Sec. 4.1, a linear combination of time samples allows to extract an interesting feature and perform a successful attack. As PCA is the most common way to find projecting vectors to extract information from data (see Sec. 2.3), we choose to apply PCA to extract a single PC as Sec. 4.1 suggests that such solution exist.
- Clustering with k-means algorithm to directly conclude the attack.

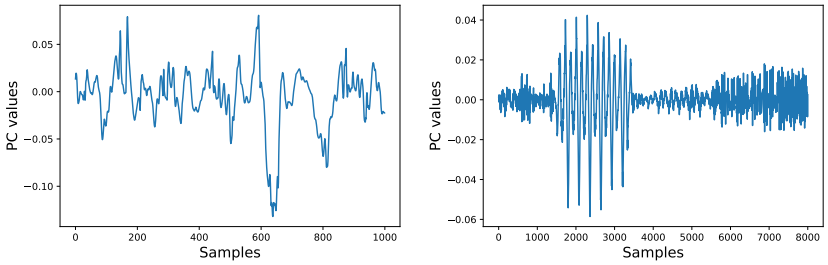
In other words, we propose to settle for the prelabeling phase of the attack proposed in [16], simply performing it in a more opportune way.

PCA is performed on all the $76,500$ subtraces of each dataset with 20 components. As said above, we aim at select a single component to project data onto. We propose to select such a PC in two ways: by visual selection or by Explained Local Variance (ELV, recalled in Sec. 2.3).

6.2 Alternative attack: experimental results

Visual component selection methodology

By visually observing the PCs, we made the following observation on both datasets: at least one PC is very similar to the t-test traces depicted in Fig. 1. For the `cswap_pointer` dataset, exactly one component, the 8th, depicted in Fig. 5a, presents similarities with such t-test trace depicted in Fig. 1a. We observe, on both traces, peaks at the same samples. Based on this observation, we would have chosen this component for our attack.
Similarly, in `cswap_arith`, several components are similar to the t-test analysis but one in particular, the 6th, depicted in Fig. 5b, seems to be less impacted by



(a) 8th PC of the Principal Component Analysis for the `cswap_pointer`

(b) 6th PC of the Principal Component Analysis for `cswap_arith`

Fig. 5: PCs visually identified as similar to t-test outcomes (see Fig. 1).

noise. It presents the main of the two groups of peaks that can be observed in the t-test outcome depicted in Fig .1b. Thus, this component would visually be the best candidate for our attack.

When the two datasets are projected onto the chosen PCs, each subtrace only consists in one sample. These datasets are then used as inputs of the k-means algorithm. We obtain a satisfying result on `cswap_pointer` for k-means on the 8th component. Indeed, the algorithm results in a 94.1% (resp. 97.7%) average (resp. maximal) single trace accuracy. However, for `cswap_arith`, the accuracy is lower than expected. k-means results in a 60.8% (resp. 72.2%) average (resp. maximal) single trace accuracy for 6th component. We suppose the k-means algorithm did not perform well onto `cswap_arith` because significant information is spread within multiple components and the chosen one has a higher noise ratio.

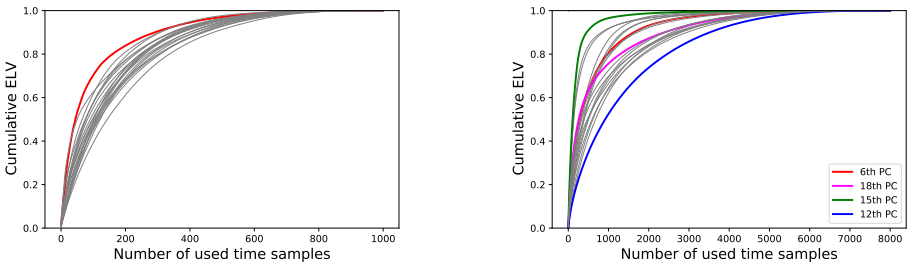
These results are nonetheless more satisfying than those presented in [16] and reported in the first row of Tab. 1. Nevertheless, this methodology requires an access to the t-test results which is in contradiction with the unsupervised context. One possibility is to relax this hypothesis such that the attacker can conduct leakage assessment analysis. In this case, the t-test results can also be used to delete non significant intervals which might improve PCA results. For example, for `cswap_arith`, we could conduct the same attack on samples $1,000$ to $6,000$ of traces and obtain 80% as the average single trace accuracy on the 6th PC.

If we do not want to alter unsupervised context, an alternative would be to test the 20 PCs independently with k-means. For `cswap_pointer`, only the 8th PC gives an accuracy far from random clustering. In this case, visual selection was more efficient but the success rate is equivalent. However, in `cswap_arith`, several components give an average single trace accuracy over 60% with k-means. As presented above, the 6th component results in a 60.8% average single trace accuracy and the 12th and 18th PC respectively result in a 61.4% and 76.1% average single trace accuracy. In this case, brute force is slower than visual selection but more effective because the final success rate is higher. This increase in time complexity might be mitigated by the use of a component selection

method. In the following section, we use the Explained Local Variance in order to limit the brute force complexity of the principal component's choice.

Explained Local Variance technique

After computing ELV values for each sample and component, we obtain 20 traces showing significant samples for each component. For the analysis, samples of ELV traces are sorted in descending order and cumulative traces are computed.
Fig. 6 presents the cumulative ELV trends individually normalized such that their cumulative ELV values are between 0 and 1. The goal is to select the component which requires the least number of samples to achieve its global variance. We can clearly identify that the 8th principal component requires the least number of samples to achieve its global variance. For this dataset, both the visual observations and the ELV criteria resulted into the same conclusion.



(a) `cswap_pointer` (b) `cswap_arith`

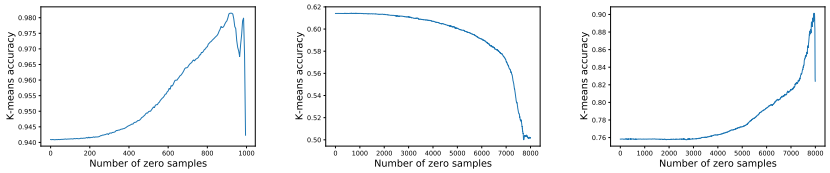
Fig. 6: Normalized cumulative ELV trend of principal components. In red, the components that we would have visually selected. In blue and magenta, for the `cswap_arith` dataset, the 2 PCs which gave k-means accuracies higher than 60%, and in green the PC that would be chosen by the ELV criterion.

Fig. 6b presents the cumulative ELV trends for `cswap_arith`. Here, the component suggested by the ELV criterion is not the same as in the visual selection method.
In this case scenario, the k-means results for both datasets are the following:

- For `cswap_pointer`, the selected PC is the same as the visually selected. Thus, 97.7% of maximal single trace accuracy are also achieved. Here, the ELV method permits to avoid the brute force onto the component selection.
- For `cswap_arith`, the clustering algorithm on the 15th PC, the one selected by ELV criterion, results in a 50.2% average single trace accuracy which is really close to the accuracy achieved by a random clustering. Observing Fig. 6b we realize that the most performing PCs would be select almost as fast by the ELV criterion than by their natural order. In fact, ELV would select the 12th PC as the last one. In that particular case the natural order

is a better choice. Thus, brute force is still necessary and testing components in the natural order would require similar effort as using ELV criteria.

In [3] the ELV concept is also derived in an attempt to denoise components. In the case where ELV selection method is used, we also apply the described denoising method on the selected component.
For `cswap_pointer`, if we only keep the 80 samples where the 8th PC presents the highest ELV, we fix the others to zero and we use the k-means algorithm after projection, we obtain an average (resp. maximal) single trace accuracy of 98.0% (resp. 100%). Fig. 7a presents the evolution of the k-means accuracy for the 8th component depending on the estimated number of POIs. It shows that there is a wide range of number of points that would bring an advantage to be fixed to zero.



(a) 8th PC (`cswap_pointer`) (b) 12th PC (`cswap_arith`) (c) 18th PC (`cswap_arith`)

Fig. 7: k-means accuracy depending on the number of samples fixed to zero in the selected component.

For `cswap_arith`, if we only keep the 55 samples where the 18th PC presents the highest ELV, we fix the others to zero and we use the k-means algorithm after projection, we obtain an average (resp. maximal) single trace accuracy of 90.6% (resp. 96.1%). Fig. 7c presents the evolution of the k-means accuracy for the 18th component depending on the estimated number of POIs. Here again, an attacker does not necessarily have to guess that 55 samples is the optimum. Indeed the trend of the accuracy raises continuously to such a maximum, while fixing to zero samples presenting increasing values of ELV.
Interestingly, the behaviour of the 12th component is different. Indeed, the more we "denoise" by fixing to zero samples which does not contribute a lot to the global variance, the more the accuracy decreases. We can suppose that its global variance is mostly due to noisy signals and that, all in all, this PC is not that interesting.

Results summary and conclusion

In this section, we presented an unsupervised attack scenario which makes use of both PCA and k-means algorithms. Several solutions have been proposed to select a satisfying component depending on whether the primary objective is to

keep it unsupervised or simple. In addition to that, a denoising method has been detailed, that allowed to significantly raise the attack performances.
The best results we could obtain for both datasets are summarized in Tab. 6, and compared with results obtained in [16].

Table 6: Comparison of the average and maximal single trace accuracy and time complexity between the different methods used to extract significant information. The symbol A represents the k-means algorithm time complexity.

	`cswap_pointer`			`cswap_arith`			Time Complexity
	Average	Maximum	PC/PoIs	Average	Maximum	PC/PoIs	
PCA + Visual	94.1%	97.7%	8th	60.8%	72.2%	6th	A
PCA + Natural order	94.1%	97.7%	8th	76.1%	82.4%	18th	$r \times A$
PCA + ELV + Denoising	98%	100%	8th	90.6%	96.1%	18th	$O(d \times c) + r \times A$
SOST [16]	52.24%	55.22%	20 PoIs	52.44%	55.22%	20 PoIs	$n \times d \times A$
SOST + Framework [16]	91%	100%	20 PoIs	83%	100%	20 PoIs	$n \times d \times A$+ Framework

In conclusion, we proposed an alternative unsupervised approach (namely PCA+ELV+Denoising) which is simply based on classical statistical tools for dimensionality reduction and clustering and which still outperforms on average the solution proposed in [16] in all tested cases when comparing average single trace accuracy, with a lower time complexity.
We remark that this simple strategy provides, complete attacks, since the remaining entropy is sufficiently low to allow using brute force to retrieve the all key. Indeed, when considering the PCA equipped with ELV selection and denoising, an average single trace accuracy of 98% on the `cswap_pointer` dataset, means that there are on average 5 remaining bits to correct in order to recover the secret scalar. This correction can be done by performing 2^{33} tests . Concerning the `cswap_arith` dataset, the average residual complexity is too high to be fulfilled (*i.e.* 2^{111} tests), but we remark that, even though we do not reach as in [16] a 100% maximal single trace accuracy, we can still conduct a successful attack since we obtain a minimal residual complexity of 2^{58} (*i.e.* corresponding to the maximal single trace accuracy obtained), that allows a complete attack as well.
Moreover, the obtained results are in some cases close to the ones claimed in [16] after the application of the recursive neural-network-based correcting framework. In this section, we provide simple state-of-the-art solutions to defeat the implementations targeted in [16]. Thus, through those propositions, we question the real benefit of using deep-learning based solutions (*i.e.* iterative approach) in the side-channel attacks context when datasets that are linearly separable are targeted.

7 Conclusion

Dealing with noisy labels is a generic machine learning problem which has a direct application in the side-channel domain. Many techniques have been proposed in the machine learning community to attempt to address this problem: [18]. In particular, one of the approaches studied is based on a co-teaching strategy: iterative correction of noisy labels.

Perin *et al.* applied such an approach in SCA domain [16]. In this paper, we provide an in-depth analysis of their iterative framework. We observed that the study was particularly biased by the simplicity of the used datasets. Indeed, we observed that the used datasets presented a simple linear separability. Anyway, this particularity of the datasets was hidden by the huge complexity of the proposed deep-learning-based solution. This fact should raise a warning about the pertinence of using very complex machine-learning solution for general problems. Indeed, in the considered case, we show that a deep-learning-free solution, *i.e.* the use of a simple perceptron, or the combination of PCA and k-means, results in a faster and more efficient attack than the CNN proposed in [16] while being much more interpretable. Through this contribution, we believe that focusing on the simplest possible attacks also provides a better insight on the actual threats and attack scenarios that one should take into account when implementing or designing cryptosystems. In particular, we consider that it is important to warn that the implementation studied is not only vulnerable to a somewhat involved deep learning framework but also to simpler and faster approaches. We acknowledge that machine-learning solution are more suitable in more complex context *e.g.* noisy or non-linearly separable data or datasets with desynchronization effects (for instance, a future work could investigate the property of convolutional layer to mitigate the desynchronization effect) but they should be sized carefully.

Finally, we also question in this paper the relevance of this corrective approach when considering noisy traces. From these experiments, we conclude that this approach does not solve the problem of handling noisy labels, which implies that this problem is still an open problem in both side-channel and machine learning communities.

Acknowledgments

The authors would like to thank Guilherme Perin and the anonymous reviewers for their valuable comments which helped to improve this work. We would also like to thank Simon Abelard, Eleonora Cagli, Julien Eynard, Antoine Loiseau, Ange Martinelli, Guénaël Renault and Gabriel Zaid for fruitful discussions about this work. This work was financially supported by the defense innovation agency (AID) from the French ministry of armed forces.

References

1. Aloise, D., Deshpande, A., Hansen, P., Popat, P.: Np-hardness of euclidean sum-of-squares clustering. Machine Learning **75**, 245–248 (05 2009). https://doi.org/10.1007/s10994-009-5103-0
2. Arpit, D., Jastrzebski, S., Ballas, N., Krueger, D., Bengio, E., Kanwal, M.S., Maharaj, T., Fischer, A., Courville, A., Bengio, Y., Lacoste-Julien, S.: A closer look at memorization in deep networks (2017)
3. Cagli, E., Dumas, C., Prouff, E.: Enhancing dimensionality reduction methods for side-channel attacks. In: Smart Card Research and Advanced Applications: 14th International Conference, CARDIS 2015, Bochum, Germany, November 4-6, 2015. Revised Selected Papers 14. pp. 15–33. Springer (2016)
4. Choudary, O., Kuhn, M.G.: Efficient template attacks. In: International Conference on Smart Card Research and Advanced Applications. pp. 253–270. Springer (2014)
5. Du, Q., Faber, V., Gunzburger, M.: Centroidal voronoi tessellations: Applications and algorithms. SIAM review **41**(4), 637–676 (1999)
6. Düll, M., Haase, B., Hinterwälder, G., Hutter, M., Paar, C., Sánchez, A., Schwabe, P.: High-speed curve25519 on 8-bit, 16-bit, and 32-bit microcontrollers. Designs, Codes and Cryptography **77** (05 2015). https://doi.org/10.1007/s10623-015-0087-1
7. Gallier, J.: Dirichlet-Voronoi Diagrams and Delaunay Triangulations, pp. 301–319. Springer New York, New York, NY (2011). https://doi.org/10.1007/978-1-4419-9961-0$_1$0, https://doi.org/10.1007/978-1-4419-9961-0_10
8. Gierlichs, B., Lemke-Rust, K., Paar, C.: Templates vs. stochastic methods. In: Goubin, L., Matsui, M. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2006, 8th International Workshop, Yokohama, Japan, October 10-13, 2006, Proceedings. Lecture Notes in Computer Science, vol. 4249, pp. 15–29. Springer (2006). https://doi.org/10.1007/11894063_2, https://doi.org/10.1007/11894063_2
9. Han, B., Yao, Q., Yu, X., Niu, G., Xu, M., Hu, W., Tsang, I., Sugiyama, M.: Co-teaching: Robust training of deep neural networks with extremely noisy labels (2018)
10. Linde, Y., Buzo, A., Gray, R.: An algorithm for vector quantizer design. IEEE Transactions on Communications **28**(1), 84–95 (1980). https://doi.org/10.1109/TCOM.1980.1094577
11. Lloyd, S.: Least squares quantization in pcm. IEEE Transactions on Information Theory **28**(2), 129–137 (1982). https://doi.org/10.1109/TIT.1982.1056489
12. van der Maaten, L., Hinton, G.: Viualizing data using t-SNE. Journal of Machine Learning Research **9**, 2579–2605 (11 2008)
13. Maghrebi, H., Portigliatti, T., Prouff, E.: Breaking cryptographic implementations using deep learning techniques. In: Carlet, C., Hasan, M.A., Saraswat, V. (eds.) Security, Privacy, and Applied Cryptography Engineering. pp. 3–26. Springer International Publishing, Cham (2016)
14. Nascimento, E., Chmielewski, L.: Applying horizontal clustering side-channel attacks on embedded ecc implementations. In: Eisenbarth, T., Teglia, Y. (eds.) Smart Card Research and Advanced Applications - 16th International Conference, CARDIS 2017, Lugano, Switzerland, November 13-15, 2017, Revised Selected Papers. Lecture Notes in Computer Science, vol. 10728, pp. 213–231. Springer (2017). https://doi.org/10.1007/978-3-319-75208-2$_1$3
15. Perin, G., Chmielewski, Ł.: A semi-parametric approach for side-channel attacks on protected rsa implementations. In: International Conference on Smart Card Research and Advanced Applications. pp. 34–53. Springer (2016)

16. Perin, G., Chmielewski, , Batina, L., Picek, S.: Keep it unsupervised: Horizontal attacks meet deep learning. IACR Transactions on Cryptographic Hardware and Embedded Systems **2021**(1), 343–372 (Dec 2020). https://doi.org/10.46586/tches.v2021.i1.343-372, https://tches.iacr.org/index.php/TCHES/article/view/8737
17. Shorten, C., Khoshgoftaar, T.M.: A survey on image data augmentation for deep learning. Journal of big data **6**(1), 1–48 (2019)
18. Song, H., Kim, M., Park, D., Shin, Y., Lee, J.G.: Learning from noisy labels with deep neural networks: A survey (2022)
19. Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**, 1929–1958 (2014)
20. Ying, X.: An overview of overfitting and its solutions. Journal of Physics: Conference Series **1168**(2), 022022 (feb 2019). https://doi.org/10.1088/1742-6596/1168/2/022022, https://dx.doi.org/10.1088/1742-6596/1168/2/022022
21. Zhang, C., Bengio, S., Hardt, M., Recht, B., Vinyals, O.: Understanding deep learning requires rethinking generalization (2017)