Journal of Hardware and Systems Security (2020) 4:314–328
https://doi.org/10.1007/s41635-020-00106-w

Systematic Side-Channel Analysis of Curve25519 with Machine Learning

Léo Weissbart[1] · Łukasz Chmielewski[1,2] · Stjepan Picek[3] · Lejla Batina[1]

Received: 25 May 2020 / Accepted: 15 September 2020 / Published online: 16 October 2020

Abstract

Profiling attacks, especially those based on machine learning, proved to be very successful techniques in recent years when considering the side-channel analysis of symmetric-key crypto implementations. At the same time, the results for implementations of asymmetric-key cryptosystems are very sparse. This paper considers several machine learning techniques to mount side-channel attacks on two implementations of scalar multiplication on the elliptic curve Curve25519. The first implementation follows the baseline implementation with complete formulae as used for EdDSA in WolfSSl, where we exploit power consumption as a side-channel. The second implementation features several countermeasures, and in this case, we analyze electromagnetic emanations to find side-channel leakage. Most techniques considered in this work result in potent attacks, and especially the method of choice appears to be convolutional neural networks (CNNs), which can break the first implementation with only a single measurement in the attack phase. The same convolutional neural network demonstrated excellent performance for attacking AES cipher implementations. Our results show that some common grounds can be established when using deep learning for profiling attacks on very different cryptographic algorithms and their corresponding implementations.

Keywords Side-channel analysis · Machine learning · Deep learning · Public-key cryptography · Curve25519

1 Introduction

Various cyber-physical devices have become integral parts of our lives. They provide basic services, and as such, also need to fulfill appropriate security requirements. Designing such secure devices is not easy due to limited resources available for implementations, and the need to provide resilience against various attacks. In the last decades, implementation attacks emerged as real threats and the most potent attacks. In implementation attacks, the attacker does not aim at the weaknesses of an algorithm, but the weaknesses in implementations [23]. One powerful category of implementations attacks is the profiled side-channel analysis (SCA) where the attacker has access to a profiling device she uses to learn about the leakage from the device under attack. Profiled SCA uses a broad set of methods to conduct the attack.

In the last few years, attacks based on the machine learning classification task have proved to be very successful when attacking symmetric-key cryptography [20–22, 35, 39]. On the other hand, profiled SCAs on public-key cryptography implementations are much more scarce [8, 25, 38].

While the current state-of-the-art results on profiled SCA and public-key cryptography suggest breaking targets with relatively small effort, many questions remain unanswered. For instance, it is not yet clearwhat are the benefits of countermeasures against machine learning–based attacks. What is more, public-key cryptography has

✉ Léo Weissbart
l.weissbart@cs.ru.nl

Łukasz Chmielewski
l.chmielewski@cs.ru.nl; chmielewski@riscure.com

Stjepan Picek
s.picek@tudelft.nl

Lejla Batina
lejla@cs.ru.nl

[1] Digital Security Group, Radboud University, Nijmegen, Netherlands

[2] Riscure, The Netherlands

[3] Intelligent Systems Department, Cyber Security Group, Delft University of Technology, Delft, Netherlands

different use cases and parameters that also result in classification problems with a significantly different number of classes one commonly encounters when attacking, e.g., block ciphers. Finally, in profiled SCA on symmetric ciphers, we are slowly moving away from scenarios where the only interesting aspect is the attack performance. Indeed, the SCA community is now becoming interested in not only questions like interpretability [24, 32, 45] and explainability [46] of deep learning attacks, but also building methodologies [50] and frameworks [33, 34] for objective analysis.

This paper considers profiled side-channel attacks on two implementations of scalar multiplication on one of the most popular elliptic curves for applications, i.e., Curve25519. The first implementation is the baseline implementation with the complete formulae as used for EdDSA in WolfSSl. The second implementation also includes several countermeasures. To evaluate the security of those implementations, we consider seven different profiled methods. Additionally, we investigate the influence of the dimensionality reduction technique. By doing this, we aim at filling the knowledge gap and give insights into the performance of different profiled methods. Finally, we compare the differences in the attack performance when considering protected and non-protected implementations.

This paper is based on the work "One Trace Is All It Takes: Machine Learning-Based Side-Channel Attack on EdDSA" [48]. The main differences are:

1. We provide results for an additional target, protected with countermeasures.
2. We provide results for several more profiled methods and different dimensionality reduction steps.
3. We investigate the applicability of one visualization technique for deep learning when attacking public-key implementations.

The rest of this paper is organized as follows. In Section 2, we give details about EdDSA and scalar multiplication procedure. Afterwards, we discuss the profiled methods we use in our experiments. Section 3 provides details about the attacker model, the datasets we use, hyperparameter tuning, and dimensionality reduction. In Section 4, we provide experimental results for both targets. In Section 5, we discuss related works. Finally, in Section 6, we conclude the paper and offer some potential future research directions.

2 Background

In this section, we start by introducing the elliptic curve scalar multiplication operation and the EdDSA algorithm. After that, we discuss profiling attacks that we use in our experiments.

2.1 Elliptic Curve Digital Signature Algorithm

In the context of public-key cryptography, one important feature is the (entity) authentication between two parties. This feature ensures to party B that party A has sent a message M and that this message is original and unaltered. Authentication can be performed by the Digital Signature Algorithm (DSA). Nowadays, public-key cryptography for constrained devices typically implies Elliptic Curves cryptography (ECC) as the successor of RSA because it achieves a higher security level with smaller key lengths saving the resources such as memory, power, and energy. The security of ECC algorithms is based on the difficulty of Elliptic Curve Discrete Logarithm Problem (ECDLP), which states that while it is easy and efficient to compute $Q = k \cdot P$, it is "difficult" to find k with knowledge of Q and P.

EdDSA [4] is a variant of the Schnorr digital signature scheme [42] using Twisted Edward Curves, a subgroup of elliptic curves that uses unified formulas, enabling speedups for specific curve parameters. This algorithm proposes a deterministic generation of the ephemeral key, different for every message, to prevent flaws from a biased random number generator. The ephemeral key r is made of the hash value of the message M and the auxiliary key b, generating a unique ephemeral public key R for every message.

EdDSA, with the parameters of Curve25519, is referred to as Ed25519 [3]. EdDSA scheme for signature generation and verification is described in Algorithm 1, where the notation $(x, \ldots, y)$ denotes the concatenation of the elements. The hash function H is SHA-512 [29]. The key length is of size $u = 256$. We denote the private key with k, the private scalar a is the first part of the private key's hashed value, and the auxiliary key b is the second part. We denote the ephemeral key with r and M is the message.

After the signature generation, party A sends (M, R, S), i.e., the message along with the signature pair (R, S) to B. The verification of the signature is done by B with steps 10 to 11. If the last equation is verified, it represents a point on the elliptic curve, and the signature is correct, ensuring that the message can be trusted as an authentic message from A.

2.2 Elliptic Curve Scalar Multiplication

We focus on two types of implementations of EC scalar multiplication. The first implementation is of EdDSA using Ed25519 as in WolfSSL. This implementation is based on the work of Bernstein et al. [4] and is a window-based method with radix-16, making use of a precomputed table

containing results of the scalar multiplication of $16^i|r_i| \cdot G$, where $r_i \in [-8, 7] \cap \mathbb{Z}$ and G is the base point of Curve25519. This method is popular because of its trade-off between memory usage and computation speed, but also because the implementation is time-constant and does not feature any branch condition nor array indices and hence is presumably secure against timing attacks.

Leaking information from the corresponding value loaded from memory with a function *ge_select* is here used to recover e and hence can be used to connect to the ephemeral key r easily. More details are given in the remainder of this paper. We can attack this implementation and extract the ephemeral key r from Step 5 in Algorithm 1.

Algorithm 1 EdDSA Signature generating and verification.

Keypair Generation: (Used once, first time private key is used.)
Input: k, **Output:** a, b, P
1: Hash k such that $H(k) = (h_0, h_1, \ldots, h_{2u-1}) = (a, b)$
2: $a = (h_0, \ldots, h_{u-1})$, interpret as integer in little-endian notation
3: $b = (h_u, \ldots, h_{2u-1})$
4: Compute public key: $P = aG$.

Signature Generation:
Input: M, a, b, P **Output:** R, S
5: Compute ephemeral private key $r = H(b, M)$.
6: Compute ephemeral public key $R = rG$.
7: Compute $h = H(R, P, M) \mod l$.
8: Compute: $S = (r + ha) \mod l$.
9: Signature pair (R, S)

Signature Verification:
Input: M, P, R, S, **Output:** {True, False}
10: Compute $h = H(R, P, M)$
11: Verify if $8SG = 8R + 8hP$ holds in E

The second implementation we focus on is the Montgomery Ladder scalar multiplication as used in μNaCl [14]. The implementation employs arithmetic-based conditional swap and is additionally protected with projective coordinate re-randomization and scalar randomization. The traces used to analyze this implementation are obtained from a publicly available dataset [11]. All details on this implementation, including the additional countermeasures, are described in [27].

2.3 Profiling Attacks

2.3.1 Random Forest (RF)

Random forest is an ensemble learning method that consists of a number of decision trees [6]. Decision trees consist of combinations of Boolean decisions on a different random subset of attributes of input data (called bootstrap sampling). For each node of each tree, the best split is taken among these randomly chosen attributes. Random forest is a stochastic algorithm since it has two sources of randomness: bootstrap sampling and attribute selection at node splitting. While the random forest has several hyperparameters to tune, we investigate the influence of the number of trees in the forest, where we do not pose any limits on the tree size.

2.3.2 Support Vector Machines (SVM)

Support vector machine is a kernel-based machine learning family of methods used to classify linearly separable and linearly inseparable data [47]. The idea for linearly inseparable data is to transform them into a higher dimensional space using a kernel function, wherein the data can usually be classified with higher accuracy. The scikit-learn implementation we use considers libsvm's C-SVC classifier [31] that implements SMO-type algorithm [16]. This implementation of SVM learning is widely used because it is simpler and faster compared to older methods. The multi-class support is handled according to a one-vs-one scheme. We investigate two variations of SVM: with a linear kernel and with a radial kernel. Linear kernel-based SVM has the penalty hyperparameter C of the error term. Radial kernel-based SVM has two significant hyperparameters to tune: the cost of the margin C and the kernel γ.

2.3.3 Convolutional Neural Networks (CNNs)

Convolutional neural networks, like other types of neural networks, have several layers where each layer is made up of neurons, as depicted in Fig. 1. Every neuron in a layer computes a weighted combination of an input set by a net input function (e.g., the sum function in neurons of a fully connected layer) from which a nonlinear activation function produces an output. When the output is different from zero,

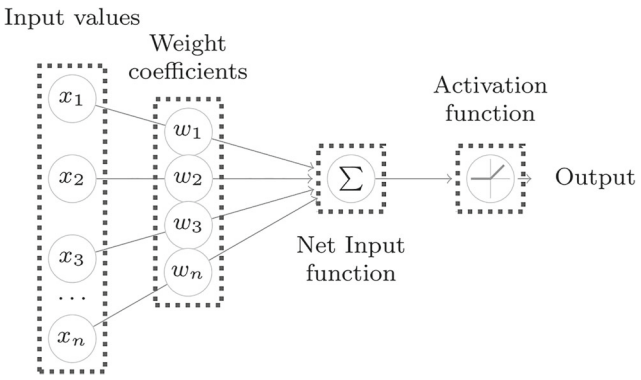


Fig. 1 Anatomy of a neuron

we say that the neuron activation feeds the next layer as its input. Layers with a convolution function as the net input function are referred to as convolutional layers and are the core building blocks in a CNN. Pooling layers are commonly used after a convolution layer to sample down local regions and create spatial regions of interest. The last fully connected layers of a CNN behave as a classifier for the extracted features from the inputs.

In this work, we start from the VGG-16 architecture introduced in [43] for image recognition. This architecture was also recently applied for SCA on AES [20] and EdDSA [48]. This CNN architecture also uses the following elements:

1. Batch normalization to normalize the input layer by applying standard scaling on the activations of the previous layer.
2. Flatten layer to transform input data of rank greater than two into a one-dimensional feature vector used in the fully connected layer.
3. Dropout (randomly dropping out units (both hidden and visible) in a neural network with a certain probability at each batch) as a regularization technique for reducing overfitting by preventing complex co-adaptations on the training data.

The architecture of a CNN depends on a large number of hyperparameters, so choosing hyperparameters for each different application is an engineering challenge. The choices made in this paper are discussed in Section 4.

2.3.4 Gradient Boosting (XGB)

Gradient boosting for classification is an algorithm that trains several weak learners (i.e., decision trees that perform poorly considering the classification problem) and combines their predictions to make one stronger learner. Gradient boosting differs from the random forest in the way the decision trees are built. While in random forest classifier, each tree is trained independently using random samples of the data, and decision trees in gradient boosting depend on the previously trained tree's prediction to correct its errors. Gradient tree boosting is composed of a concatenation of several smaller decision trees. We used the extreme gradient boosting (XGB) implementation of gradient boosting, designed by Chen and Guestrin [10], which uses a sparsity-aware algorithm for handling sparse data and a theoretically justified weighted quantile sketch for approximate learning.

2.3.5 Naive Bayes (NB)

Gaussian Naive Bayes classifier is one of the classification algorithms that applies Bayes's theorem with the "naive" assumption. The naive assumption describes the conditional independence between every pair of features in a given class sample. The Gaussian assumption is assumed as the features probability distribution. The Naive Bayes method is highly scalable with the number of features and requires only a few representative features per class to achieve a satisfying performance.

2.3.6 Template Attack (TA)

The template attack relies on the Bayes theorem and considers the features to be dependent. Commonly, template attack relies on a normal distribution [9] and it assumes that each $P(\vec{X} = \vec{x}|Y = y)$ follows a (multivariate) Gaussian distribution parameterized by its mean and covariance matrix for each class Y. Choudary and Kuhn proposed using one pooled covariance matrix averaged over all classes Y to cope with statistical difficulties and thus lower efficiency [12]. In our experiments, we use this version of the attack.

3 Experimental Setup

3.1 Attacker Model

The general recommendation for EdDSA, as well as other ECDSA implementations, is to select different ephemeral private keys r for each different signature. When this is not applied and the same r is used for different messages, the two resulting signature pairs (R, S) and (R, S') for messages M and M', respectively, can be used to recover r as $r = (z - z')(S - S')^{-1}$, where z and z' represent a majority of leftmost bits of $H(M)$ and $H(M')$ interpreted as integers.[1] Finally, the private scalar a is exposed as $a = R^{-1}(Sr - z)$ and can be misused by the attacker to forge new signatures.[2]

The attacker's aim is the same as for every ECDSA attack: recover the secret scalar a. The difference is that the attacker cannot acquire two signatures with the same random r, but can still recover the secret scalar in two different ways. The first method consists of attacking the hash function's implementation to recover b from the computation of ephemeral private key [40]. The second one attacks the implementation of the scalar multiplication during the ephemeral public key's computation to infer it in a single trace [48]. In this paper, we consider only the profiled attacks, i.e., those based on the supervised machine

[1] To be precise: z and z' correspond to l leftmost bits of $H(M)$ and $H(M')$, respectively, where l denotes the bit length of the group order.

[2] For details, we refer the reader to the presentation about a real-world application of this attack:
https://wikileaks.org/sony/docs/05/docs/Hacks/PS3%20timeline.pdf

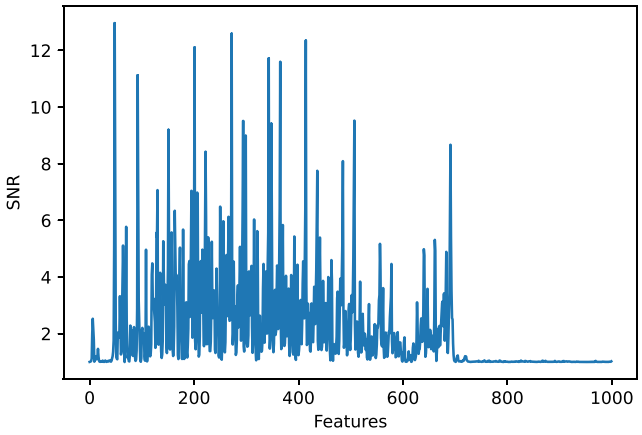


Fig. 2 Signal-to-noise ratio for the baseline implementation dataset

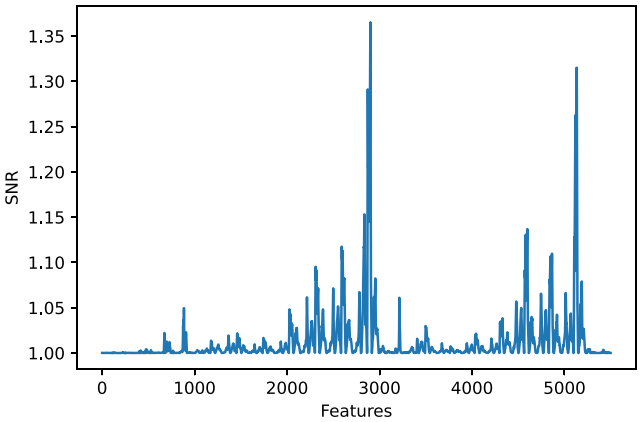


Fig. 3 Signal-to-noise ratio for the protected implementation dataset

learning paradigm, where the task is the classification (learning how to assign a class label to examples). As side-channels, we consider the power and electromagnetic (EM) leakage.

3.2 SCA Datasets

We analyze two publicly available datasets targeting elliptic curve scalar multiplication on Curve25519 for microcontrollers. The first dataset consists of power traces of a baseline implementation, and the second dataset consists of electromagnetic traces of a more protected implementation.

3.2.1 Baseline Implementation Dataset

We consider a dataset of scalar multiplication on Curve25519. The implementation follows the baseline implementation of the scalar multiplication algorithm as in [48]. The traces contain power measurements collected from a Piñata development board[1] based on a 32-bit STM32F4 microcontroller with an ARM-based architecture, running at the clock frequency of 168 MHz. The device is running the Ed25519 implementation of WolfSSL 3.10.2. The target is the EC scalar multiplication of the ephemeral key and the base point of curve Ed25519 (as explained in Section 3.1). Because of the chosen implementation, it is possible to profile the full scalar by nibble in a horizontal fashion. The dataset is thus composed of multiple separate nibble computations.

The dataset has 6400 labeled traces of 1000 features each, with associated nibble value. In Fig. 2, we give the signal-to-noise ratio of this dataset. The SNR is high and reaches a maximum value of 12.9. Such a high SNR is the consequence of dealing with power leakages that are less noisy than usual EM leakages. The leakage is essentially located between points 50 and 700, where several features seem to leak information about the handled nibble.

3.2.2 Protected Implementation Dataset

The traces in the protected dataset are taken from a publicly available dataset [11]. This set contains electromagnetic traces coming from 5997 executions of Curve25519 μNaCl Montgomery Ladder scalar multiplication[3] running on the Piñata target, the same as in Section 3.2.1. The implementation employs an arithmetic-based conditional swap and is additionally protected with the projective coordinate re-randomization and scalar randomization. Each trace from the dataset represents a single iteration of the Montgomery Ladder scalar multiplication that is cut from the whole execution trace; such trace is labeled with the corresponding cswap condition bit.[4] Furthermore, all these cut traces ($5997 \times 255 = 1,529,235$) are aligned to exploit the leakage efficiently. Details about the implementation and how the traces are aligned are in [27].

Figure 3 represents the SNR of the dataset for the bit model. This SNR is relatively flat except for two peaks where the leakage of the data is stronger. One is located before feature 3000 and the second after feature 5000. The noise level is high for an EM dataset but is smaller than the other dataset based on power traces.

3.3 Evaluation Metrics

To examine the feasibility and performance of our attack, we use two different metrics. We first compare the performance using the accuracy metric since it is a standard metric

[1] Pinata Board: https://www.riscure.com/product/pinata-training-target/

[3] http://munacl.cryptojedi.org/curve25519-cortexm0.shtml

[4] Observe that a full scalar can be trivially recovered from the cswap condition bits used in the 255 Montgomery Ladder iterations.

in machine learning. The accuracy metric represents the fraction of the measurements that are classified correctly. The second metric we use is the success rate as it is an SCA metric that gives a more concrete idea on the power of the attacker [44]. Let us consider the settings where we have A attack traces. As the result of an attack, we output a key guessing vector $v = [v_1, v_2, \ldots, v_{|\mathcal{K}|}]$ in decreasing order of probability with $|\mathcal{K}|$ being the size of the keyspace. Then, the success rate is the average empirical probability that v_1 is equal to the correct key.

3.4 Dimensionality Reduction

For computational reasons, one may want to analyze only the most informative features from the dataset's traces. Consequently, we explore several different settings where we use all the features in a trace or conduct dimensionality reduction. For dimensionality reduction, we use a method called principal component analysis. Principal component analysis (PCA) is a linear dimensionality reduction method that uses Singular Value Decomposition (SVD) of the data matrix to project it to a lower dimensional space [5]. PCA creates a new set of features (called principal components) that form a new orthogonal coordinate system that is linearly uncorrelated. The number of components is the same as the number of original features. The components are arranged so that the first component covers the largest variance by a projection of the original data, and the following components cover less and less of the remaining data variance. The projection contains (weighted) contributions from all the original features. Not all principal components need to be kept in the transformed dataset. Since the components are sorted by decreasing covered variance, the number of kept components, designated by L, maximizes the original data variance and minimizes the data transformation's reconstruction error. While PCA is meant to select the principal information from data, there is no guarantee that the reduced data form will give better results for profiling attacks than its complete form.

3.5 Hyperparameter Tuning

Most machine learning methods are parametric and require some hyperparameters to be tuned before the training phase. Depending on this pre-tuning, the trained classifier will potentially have a different outcome. The different classification methods we used are trained with a wide set of hyperparameters as detailed in this section. The exact used hyperparameters are listed in Tables 1 and 4.

TA We use the Template Attack with a pooled covariance matrix [12]. This method has no hyperparameters to tune.

Table 1 Best hyperparameters found for the baseline implementation dataset

Algorithm	Number of features	Best hyperparameters
SVM linear	1 000	C=1 000
	500	C=23.1
	100	C=284.8
	10	C=1 333
SVM rbf	1 000	C=1 000, γ=1
	500	C=12.3, γ=0.65
	100	C=81.1, γ=0.65
	10	C=1 000, γ=1.23
RF	1 000, 500, 100, 10	n_tree=500
XGB	1 000, 500, 100, 10	n_tree=300, max_depth=3

NB We do not conduct hyperparameter tuning as the method is non-parametric (i.e., there are no hyperparameters to tune).

RF We tune the number of decision trees. We consider the following number of trees: 50, 100, 500.

SVM For the linear kernel, the hyperparameter to optimize is the penalty parameter C. We search for the best C in the range $[1, 10^5]$ in logarithmic space. For the radial basis function (RBF) kernel, we have two hyperparameters to tune: the penalty C and the kernel coefficient γ. The search for best hyperparameters is done within $C = [1, 10^5]$ and $\gamma = [-5, 2]$ in logarithmic spaces.

XGB In the same fashion as the random forest classifier, we set the hyperparameter exploration for the number of trees to 50, 100, and 300. We impose a maximum depth for each tree from 1 to 3 nodes, to force each tree to be a weak learner.

CNN The chosen hyperparameters for *VGG-16* follow several rules that have been adapted for SCA in [20] or [39] and that we describe here:

1. The model is composed of several convolution blocks and ends with a dropout layer followed by a fully connected layer and an output layer with the Softmax activation function.
2. Convolutional and fully connected layers use the ReLU activation function ($max(0, x)$).
3. A convolution block is composed of one convolution layer followed by a pooling layer.
4. An additional batch normalization layer is applied for every odd-numbered convolution block and is preceding the pooling layer.
5. The chosen filter size for convolution layers is set to the size 3.

6. The number of filters $n_{filters,i}$ in a convolution block i increases according to the following rule: $n_{filters,i} = max(2^i \cdot n_{filters,1}, 512)$ for every layer $i \geq 0$ and we choose $n_{filters,1} = 8$.
7. The stride of the pooling layers equals two and halves the input data for each block.
8. Convolution blocks follow each other until the size of the input data is reduced to 1.

4 Results

In this section, we first present results for the baseline implementation and the protected implementation afterward. We finish the section with results on visualization and discussion. The best results in Tables 2 and 5 are given in italics.

4.1 Baseline Implementation

After the conducted training phase of all the different classifiers with their hyperparameters, we list in Table 1 the best hyperparameter combinations for each machine learning model.

The resulting CNN architecture for a 1000-feature input is depicted in Fig. 4. Other architectures will have a different number of convolutional blocks and a number of weights depending on the number of features of the input.

In Table 2, we give the accuracy score for different profiling methods when considering the recovery of a single nibble of the key. We can see that all profiling techniques reach excellent performance with accuracy above 95%. When considering all available features (1000), CNN performs the best and achieves an accuracy of 100%. Both SVM (linear and RBF) and RF have the same accuracy. SVM's performance is interesting since the same value for linear and RBF kernel indicates there is no advantage of using higher dimensional space, which means that the classes are linearly separable. Finally, NB, XGB, and TA still perform well, but we conclude they reach the worst results compared with other methods.

Table 2 Accuracy results for the baseline implementation dataset

Algorithm	1000 features	500 PCA	100 PCA	10 PCA
TA	0.9977	0.9992	*0.9992*	0.9830
RF	0.9992	0.9909	0.9921	0.9937
SVM (linear)	0.9992	0.9995	0.9990	*0.995*
SVM (rbf)	0.9992	*0.9996*	0.9989	*0.995*
CNN	*1.00*	0.9796	0.9968	0.96
XGB	0.9965	0.9794	0.9807	0.9901
NB	0.9837	0.9475	0.9731	0.9823

PCA results in lower accuracy scores for most of the considered techniques. When considering 500 or 100 PCA components, the TA's results slightly improve, while RF and CNN results slightly decrease. SVM with both kernels can reach minimally higher accuracy when considering 500 PCA components. When considering the scenario with only the ten most important PCA components, all the results deteriorate compared with the results with 1000 features, and SVM performs the best.

To conclude, all techniques exhibit strong performance, but CNN is the best if no dimensionality reduction is applied. There, the maximum accuracy is obtained after only a few epochs (see Figs. 6 and 7). If dimensionality reduction is applied, CNN shows a progressive performance deterioration. This behavior should not come as a surprise since CNNs are usually used with the raw features (i.e., no pre-processing). Applying such techniques could reduce the performance due to a loss of information and changes in the spatial representation of features. Interestingly, TA and SVM are very stable methods, regardless of the number of used features (components), and those methods show the best performance for a reduced number of features settings.

In Fig. 5, we present a success rate with orders up to 10 for all profiling methods on the dataset without applying PCA. Recall that a success rate of order o is the probability that the correct subkey is ranked among the first o candidates of the guessing vector. While CNN has a 100% success rate of order 1, other methods achieve the perfect score only for orders greater than 6.

The results for all methods are similar in the recovery of a single nibble from the key. To have an idea of how good these methods perform for the recovery of a full 256-bit key, we apply classification on the successive 64 nibbles. We obtain an intuition of the resulting accuracy by considering the cumulative probability P_c of the probabilities of recovery of one nibble P_s : $P_c = \Pi_{64} P_s$ (see Table 3). The cumulative accuracy obtained in such a way can be interpreted as the predictive first-order success rate of a full key for the different methods in terms of a security metric.

From these results, the best result is obtained with CNN when no dimensionality reduction is applied. Other methods are nonetheless powerful profiling attacks with up to 95% performance to recover the full key on the first guess with the best choice of hyperparameters and dimensionality reduction. When considering the results after dimensionality reduction, SVM is the best performing technique when using 500 PCA components.

As can be observed from Figs. 6 and 7, both the scenarios without dimensionality reduction and dimensionality reduction to 100 and 500 components reach the maximal performance very fast. On the other hand, the scenario with

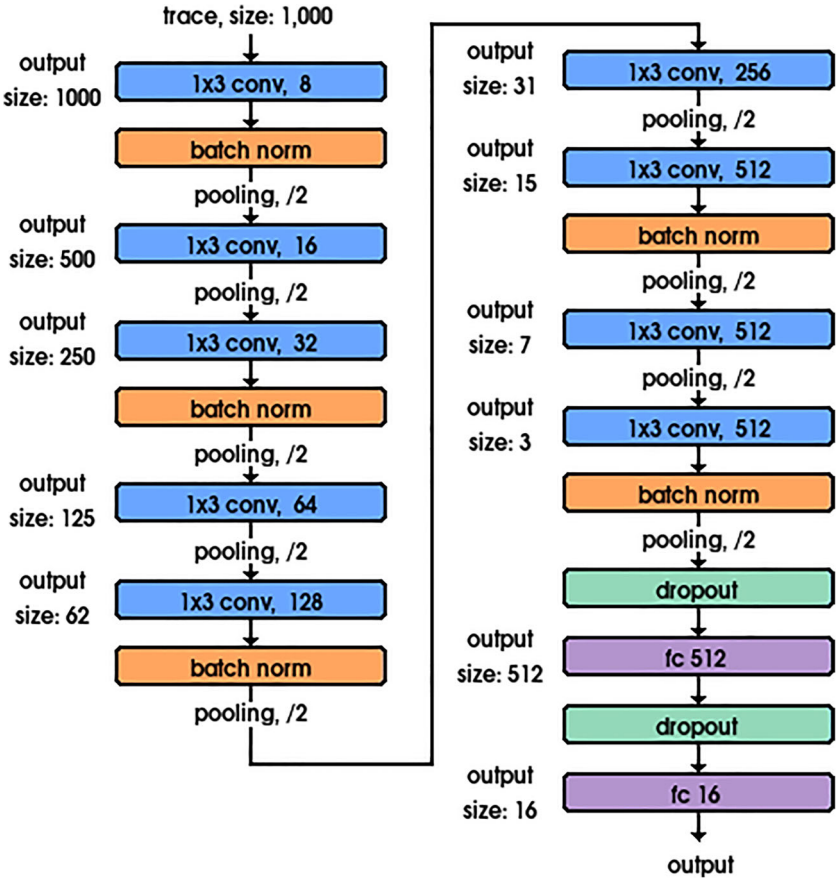


Fig. 4 CNN architecture, as implemented in Keras. This architecture takes a 1000-feature input and consists of nine convolutional layers followed by max pooling layers. For each odd convolutional layer, there is a batch normalization layer before the pooling layer. At the end of the network, there is one fully connected layer

10 PCA components does not reach the maximal performance within 100 epochs since the validation accuracy does not start to decrease. Still, even longer experiments do not show further improvement in the performance, which indicates that the network simply learned all that is possible and that there is no more information that can be used to increase the performance further. Finally, the fast increase in training and validation accuracy, and the stable behavior of profiling methods clearly indicate that attacking the implementation without countermeasures is easy.

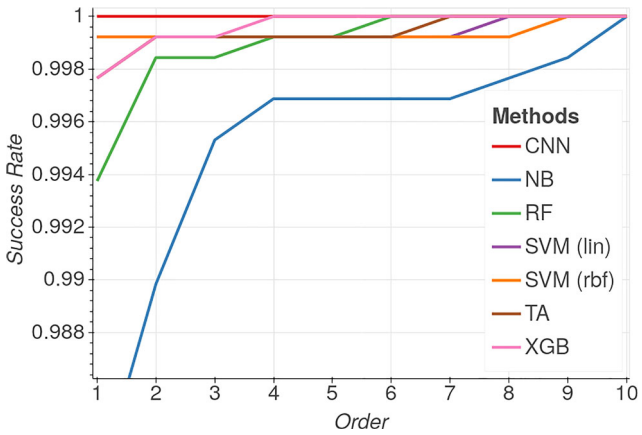


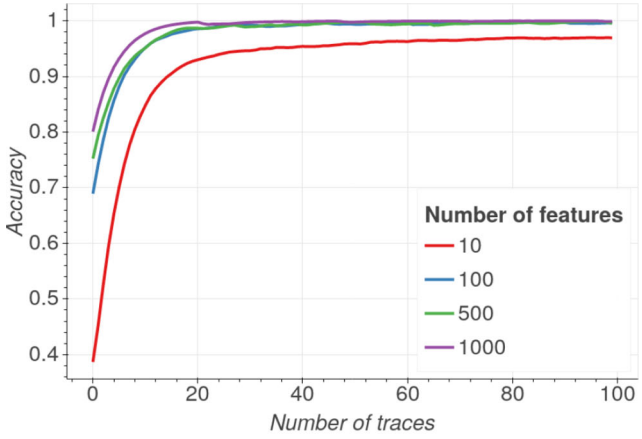
Fig. 5 Success rate results for the baseline implementation dataset

4.2 Protected Implementation

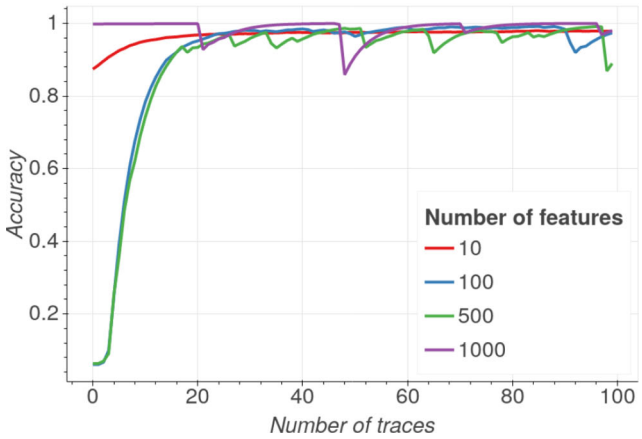
We list the selected hyperparameters for the protected implementation in Table 4. The protected implementation dataset contains more features per trace than the other dataset. Therefore, the number of trainable parameters for machine learning methods greatly increases, increasing the models' training load. We experimented with RF, NB, and

Table 3 Cumulative probabilities for the profiling methods

Algorithm	1000 features	500 PCA	100 PCA	10 PCA
TA	0.86	0.95	0.95	0.33
RF	0.95	0.56	0.61	0.67
SVM (linear)	0.95	0.97	0.94	0.73
SVM (rbf)	0.95	0.98	0.93	0.73
CNN	1.00	0.27	0.82	0.04
XGB	0.80	0.27	0.29	0.53
NB	0.35	0.03	0.18	0.32

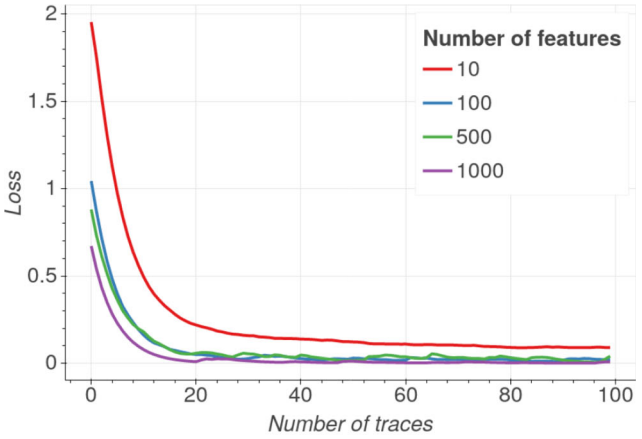


(a) Training Accuracy

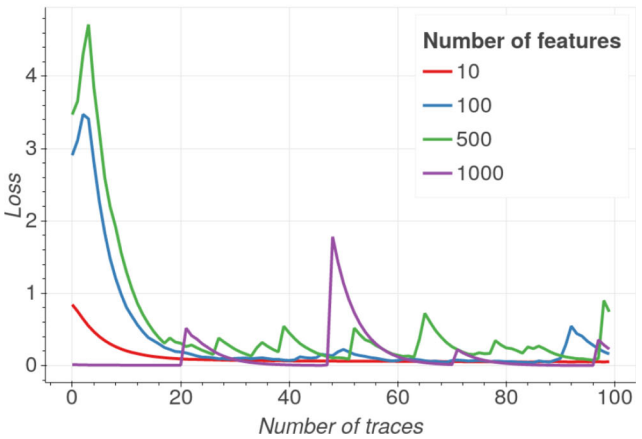


(b) Validation Accuracy

Fig. 6 Accuracy of the CNN method over 100 epochs for the baseline implementation dataset



(a) Training Loss



(b) Validation Loss

Fig. 7 Loss of the CNN method over 100 epochs for the baseline implementation dataset

XGB and left out SVM (both with linear and RBF kernel) as this method's training becomes too expensive.

We show the accuracy results for all tested methods on the protected implementation dataset in Table 5. Notice that, contrary to the previously considered dataset, not all profiling techniques have good performance, and most of them are even close to random guessing. Still, some profiling methods can reach above 99% accuracy, where the best results are obtained with CNN. When PCA is applied, random forest performs poorly with 50.2% accuracy for ten and 1000 components, which is not better than one could expect from random guessing. However, this method turns out to be quite efficient on the raw features and reaches an accuracy of 93% for one bit recovery.

Naive Bayes and XGB perform poorly regardless of the hyperparameters explored and if dimensionality reduction is applied. The accuracy stays around random guessing when PCA is applied with ten and 1000 components, and does not go above 60% in the best case. Naive Bayes and XGB are simple classifiers and, considering their accuracy score on this dataset, are not powerful enough to defeat a protected EC scalar multiplication implementation.

The template attack is performing well, where the more features are taken, the better the results. The best accuracy score for template attack is obtained when all features are kept, and it reaches 99% accuracy. When PCA is applied and 1000 components are selected, the accuracy falls to 89% (which is, in fact, the best result for all considered techniques). Finally, when the number of selected components is reduced to 10, the accuracy falls to 52%.

Table 4 Best hyperparameters found for the protected implementation dataset

Algorithm	Number of features	Best hyperparameters
RF	5 500, 1 000, 10	n_tree=500
XGB	5 500, 1 000	n_tree=300, max_depth=3
	10	n_tree=300, max_depth=2

Table 5 Accuracy results for the protected implementation dataset

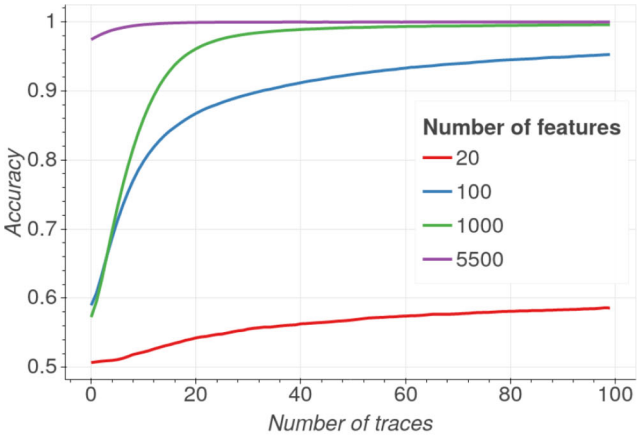
Algorithm	5500 features	1000 PCA	10 PCA
RF	0.9903	0.5022	0.5023
NB	0.6058	0.4971	0.5018
XGB	0.6058	0.4945	0.5019
TA	0.9908	*0.8954*	0.5238
CNN	*0.9999*	0.5014	*0.5572*

CNN is a highly efficient method only when considering the dataset without applying the PCA method, where it reaches an accuracy above 99%. As we can see in Figs. 8 and 9, when PCA is applied, while the training loss and accuracy seems to fit the training set, the model fails to generalize and converge on the validation set given the chosen number of traces and epochs.

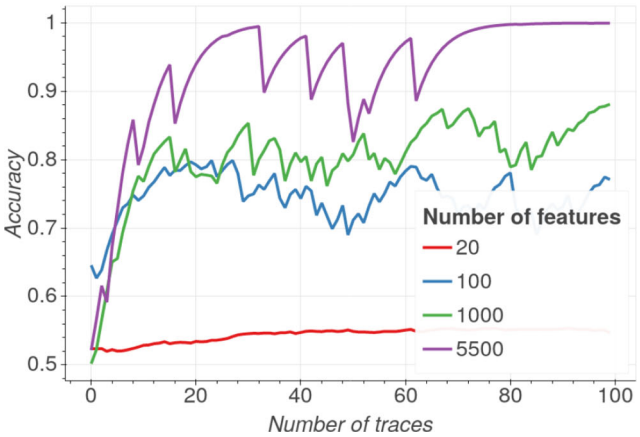
We can evaluate the accuracy of the different methods to predict a 256-bit scalar by computing the cumulative probability of success of a single bit over 256 attempts. The cumulative probability p_c for a 256-bit key considering a single bit probability recovery P_s is $P_c = \prod_{256} P_s$. Here, only the methods with a single accuracy above 99% are worth considering as the other methods have a cumulative probability close to 0. For example, the cumulative accuracy for the random forest with 5500 features is 8%, and CNN with 5500 features is 98%.

4.3 Visualization of the Integrated Gradient

For CNNs, various visualization techniques have been developed to help researchers understand what input features influence the neural network predictions. These tools are interesting in side-channel analysis to evaluate if a network bases its prediction on the part of the trace where the leakage is the strongest. We note that visualization techniques proved to be a helpful tool when considering profiled SCA and block ciphers [17, 24]. We use here the integrated gradient method [30]. In this method, the higher

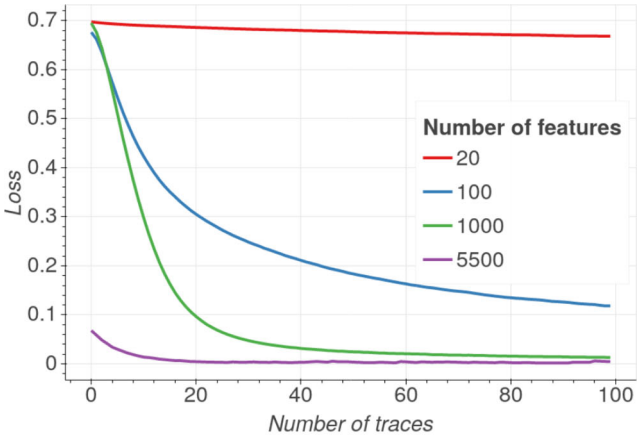


(a) Training Accuracy

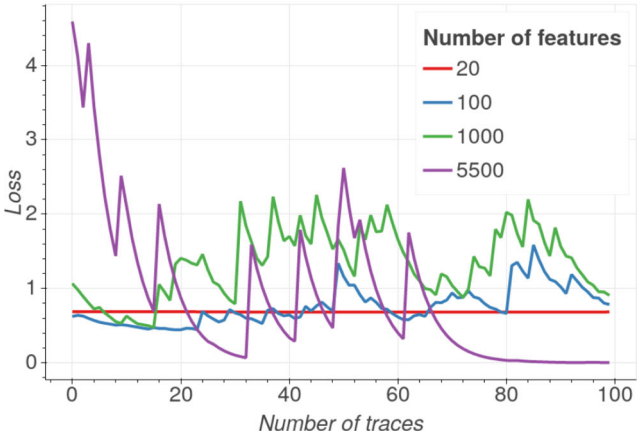


(b) Validation Accuracy

Fig. 8 Accuracy of the CNN method over 100 epochs on the protected implementation dataset



(a) Training Loss



(b) Validation Loss

Fig. 9 Loss of the CNN method over 100 epochs on the protected implementation dataset

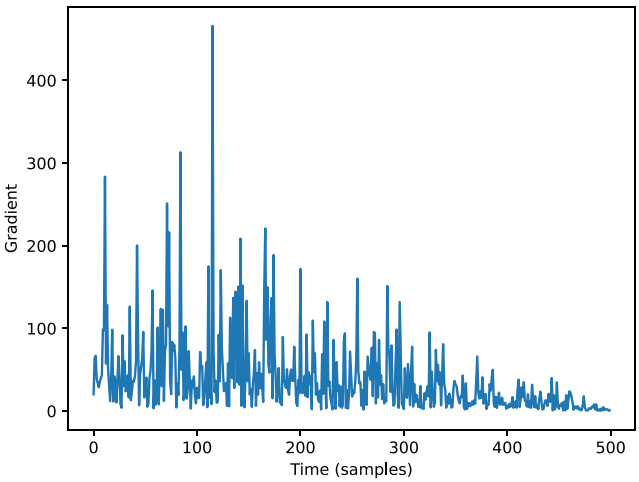
is the gradient value, the more important the feature is for the model's prediction.

From Figs. 10 and 11, we can notice that when we apply principal component analysis, the network tends to rely more on the first features. After applying PCA, the features are reorganized and ranked from the most important to the least important feature. When considering the dataset without applying PCA, the features' order is the same as those sampled with the oscilloscope. We can notice interesting similarities between the SNR of the unprotected implementation (Fig. 2) and the integrated gradient of the CNN. The interpretation of the integrated gradient obtained for the CNN trained on the protected implementation dataset is less evident as the high peaks do not correspond to the leaking features indicated by the SNR (see Fig. 3). When comparing the visualization results for both datasets, the similarity between the baseline results for the full number of features and after dimensionality reduction indicates that the performance should be similar, which is confirmed by the accuracy results. On the other hand, we see striking differences between two visualizations for the protected implementation, where the one with 1000 features cannot concentrate on the most important elements, which is again evident from the accuracy results.

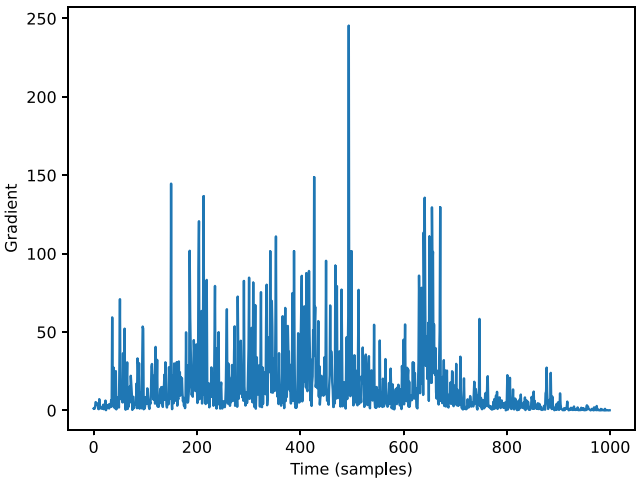
4.4 General Remarks

The obtained results allow us to infer some more general recommendations one could follow one attacking ECC with profiled SCAs:

1. When attacking unprotected implementations, most of the considered methods work well. While CNN

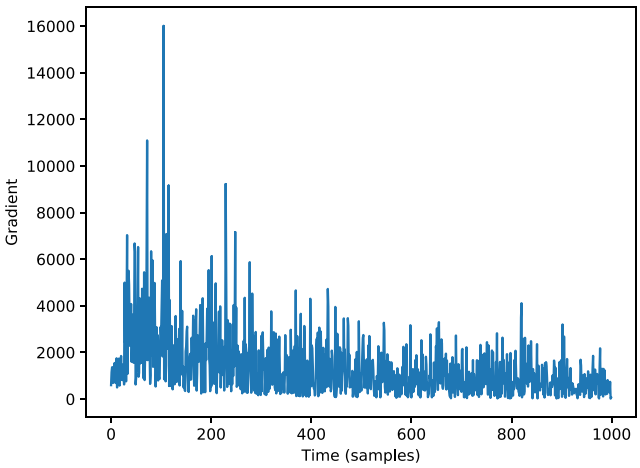


(a) 500 POI

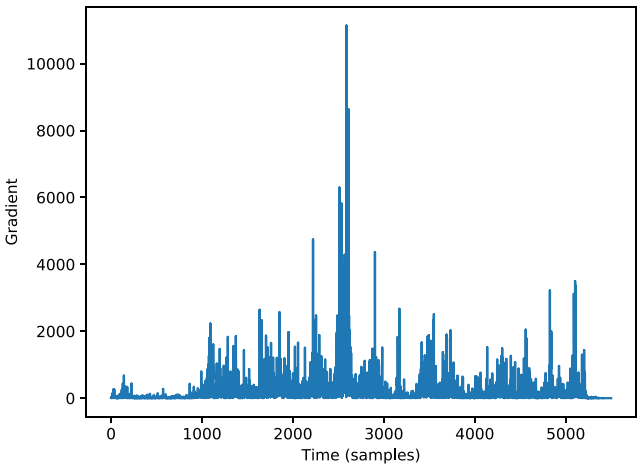


(b) 1 000 features

Fig. 10 Integrated gradient method applied to CNN trained on the baseline implementation dataset



(a) 1 000 features



(b) 5 500 features

Fig. 11 Integrated gradient method applied to CNN trained on the protected implementation dataset

performs the best, computationally simpler methods represent an interesting alternative.
2. For protected implementation, deep learning performs significantly better than other considered methods.
3. For the protected implementation, all the methods perform worse when principal component analysis is applied to reduce the number of features.
4. Template attack should be an interesting option in cases when one cannot use all the features.
5. There is not much difference in the attack performance concerning hyperparameter tuning, which indicates that coarse-grained tuning should be enough.
6. Visualization techniques offer good indication in the performance of CNNs, as they show on what features CNN concentrates. If CNN cannot concentrate on a smaller number of features, this results in a poor attack performance.

5 Related Work

In 2003, Chari et al. [9] introduced a template attack (TA) as a powerful SCA method in the information-theoretic point of view, which became a standard tool for profiling SCA. As TA's straightforward implementations can lead to computationally intensive computation, one option for more efficient computation is to use only a single covariance matrix, which is referred to as the so-called pooled template attack presented by Choudary and Kuhn [12]. There, the authors were able to template a LOAD instruction and recover all 8 bits treated with a guessing entropy equal to 0.

Several works applied machine learning methods to SCA of block ciphers because they resemble general profiling techniques. Two methods stand out particularly in profiling SCA, namely support vector machines [21, 22, 36, 41] and random forest [18, 35, 41]. Few other works also experienced SCA with naive Bayes [36] and gradient boosting methods [37, 49]. With the general evolution in the field of deep learning, more and more works deal with neural networks for SCA and often show top performance. Most of the research concentrated on either multilayer perceptron or convolutional neural networks [7, 13, 22, 37].

There is a large portion of works considering profiling techniques for symmetric-key ciphers, but there is less for public-key cryptography,[5] especially ECC. Template attacks on ECC trace back to an attack on ECDSA, as demonstrated by Medwed and Oswald in 2009 [26]. That work showed TA to be efficient for attacking SPA-resistant ECDSA with the P192 NIST curve on a 32-bit microcontroller [25]. Heyszl presented another template attack on ECC in [19]. That attack exploited register location-based leakage using a high-resolution inductive EM probe. Another approach to attack ECC is the so-called online template attacks [1, 2, 15, 30]. The first three approaches [1, 2, 15] use correlation to match the template traces to the whole attacked traces while the fourth attack [30] employs instead several machine learning distinguishers.

Lerman et al. considered a template attack and several machine learning techniques to attack RSA. However, the targeted implementation was not secure, making the comparison with non-machine learning techniques less favorable [21]. Nascimento et al. applied a horizontal attack on ECC implementation for AVR ATmega microcontroller targeting the side-channel leakage of cmov operation. Their approach to side-channel is similar to ours, but they do not use deep learning in the analysis [28]. Note that approach was extended to unsupervised settings using clustering [27]. Poussier et al. used horizontal attacks and linear regression to conduct an attack on ECC implementations, but their approach cannot be classified as deep learning [38]. Carbone et al. used deep learning to attack a secure implementation of RSA [8]. The results from that paper show that deep learning can reach strong performance against secure implementations of RSA.

6 Conclusions

In this paper, we consider several profiling methods to attack Curve25519 in both unprotected and protected settings. The results show that unprotected implementation is easy to attack with many techniques, where good results are achieved even after dimensionality reduction. We observe a significantly different behavior for the protected dataset, where only CNN can easily break the target implementation. What is more, most of the other methods perform on the level of random guessing. For this dataset, we also see a strong negative influence of dimensionality reduction. Finally, our results with the integrated gradient visualization indicate such methods useful in evaluating CNN's behavior. Indeed, when there are clear peaks for the integrated gradient, this maps to a simple classification task and, consequently, powerful attack performance.

We plan to investigate whether standard machine learning metrics like accuracy have fewer issues for public-key cryptography implementations than are reported for

[5] We do not consider here the post-quantum schemes because, although they belong to public-key cryptography, they differ significantly from ECC or RSA.

symmetric-key ciphers. As this gap between machine learning and side-channel metrics represents one of the most significant challenges in the SCA community today, insights about public-key particularities are needed.

Acknowledgments We thank anonymous reviewers for the suggestions on how to improve the paper.

Funding Ł. Chmielewski is partially supported by European Commission through the ERC Starting Grant 805031 (EPOQUE) of P. Schwabe. We thank anonymous reviewers for the suggestions on how to improve the paper.

Appendix 1: Cover Letter: Special Issue on SPACE 2019

This paper is based on the work "One Trace Is All It Takes: Machine Learning-Based Side-Channel Attack on EdDSA" [48]. The main differences are:

1. We provide results for an additional target protected with countermeasures.
2. We provide results for several more profiled methods and different dimensionality reduction steps.
3. We investigate the applicability of one visualization technique for deep learning when attacking public-key implementations.

More specifically, we rewrote Section 1 to give more emphasis on the relevance for machine learning–based SCA on ECC. In Section 2, we added information about the new implementation we consider and some info on added profiling methods (Sections 2.2, 2.3.4, 2.3.5). In Section 3, we added information about the new dataset (for protected implementation), and we briefly discuss the metrics we use. We additionally discuss the hyperparameter tuning in more detail and dimensionality reduction, where we do not use anymore heuristics to select the number of components. The main changes are in Sections 3.2.2, 3.3, and 3.4.

Section 4 gives results for both implementations (changes in Section 4.1 and new Section 4.2), with and without PCA, and for all considered profiling methods. In this section, all results are new except for the baseline implementation with 1000 features (we also have some additional results for this scenario). The parts on visualization and general remarks are new (Sections 4.3 and 4.4).

Section 5 has only minor differences from the previous version, where we included a few more related works. Finally, Section 6 now gives a more general outline for comparison between two implementations and a new suggestion for future work (since the one from SPACE we covered in this submission).

References

1. Batina L, Chmielewski Ł, Papachristodoulou L, Schwabe P, Tunstall M (2014) Online template attacks. In: Willi Meier DM (ed) Progress in cryptology - INDOCRYPT 2014 - 15th international conference on cryptology in India, New Delhi, India, December 14-17, 2014, Proceedings. LNCS, vol 8885. Springer, pp 21–36. http://cryptojedi.org/papers/#ota
2. Batina L, Chmielewski Ł, Papachristodoulou L, Schwabe P, Tunstall M (2017) Online template attacks. J Cryptogr Eng. https://doi.org/10.1007/s13389-017-0171-8
3. Bernstein DJ (2016) Curve25519: new diffie-Hellman speed records. http://cr.yp.to/papers.html#curve25519 Citations in this document 1(5)
4. Bernstein DJ, Duif N, Lange T, Schwabe P, Yang BY (2012) High-speed high-security signatures. J Cryptogr Eng 2(2):77–89
5. Bohy L, Neve M, Samyde D, Quisquater JJ (2003) Principal and independent component analysis for crypto-systems with hardware unmasked units. In: Proceedings of e-Smart 2003. Cannes, France
6. Breiman L (2001) Random forests. Mach Learn 45(1):5–32
7. Cagli E, Dumas C, Prouff E (2017) Convolutional Neural Networks with Data Augmentation Against Jitter-Based Countermeasures - Profiling Attacks Without Pre-processing. In: Cryptographic hardware and embedded systems - CHES 2017 - 19th international conference, Taipei, Taiwan, September 25-28, 2017, proceedings, pp 45–68
8. Carbone M, Conin V, Cornélie MA, Dassance F, Dufresne G, Dumas C, Prouff E, Venelli A (2019) Deep learning to evaluate secure RSA implementations. IACR Trans Cryptogr Hardw Embed Syst 2019(2):132–161. https://doi.org/10.13154/tches.v2019.i2.132-161, https://tches.iacr.org/index.php/TCHES/article/view/7388
9. Chari S, Rao JR, Rohatgi P (2002) Template attacks. In: International workshop on cryptographic hardware and embedded systems. Springer, pp 13–28
10. Chen T, Guestrin C (2016) XGBoost: a scalable tree boosting system. arXiv:1603.02754
11. Chmielewski Ł (2020) Reassure (h2020 731591) ecc dataset. https://doi.org/10.5281/zenodo.3609789
12. Choudary O, Kuhn MG (2013) Efficient template attacks. In: Francillon A, Rohatgi P (eds) Smart card research and advanced applications - 12th international conference, CARDIS 2013, Berlin, Germany, November 27-29, 2013. Revised selected papers. LNCS, vol 8419. Springer, pp 253–270
13. Cid C, Jacobson MJ, Michael J (eds) (2019) Selected areas in cryptography - SAC 2018 - 25th International Conference, Calgary, AB, Canada, August 15-17, 2018, Revised Selected

Papers, Lecture Notes in Computer Science, vol 11349. Springer, Berlin
14. Düll M, Haase B, Hinterwälder G, Hutter M, Paar C, Sánchez AH, Schwabe P (2015) High-speed curve25519 on 8-bit, 16-bit, and 32-bit microcontrollers. Des Codes Cryptogr 77(2-3):493–514. http://dblp.uni-trier.de/db/journals/dcc/dcc77.html#DullHHHPSS15
15. Dugardin M, Papachristodoulou L, Najm Z, Batina L, Danger J, Guilley S (2016) Dismantling real-world ECC with horizontal and vertical template attacks. In: Constructive side-channel analysis and secure design - 7th international workshop, COSADE 2016, Graz, Austria, April 14-15, 2016. http://eprint.iacr.org/2015/1001/
16. Fan RE, Chen PH, Lin CJ (2005) Working set selection using second order information for training support vector machines. J Mach Learn Res 6:1889–1918. http://dl.acm.org/citation.cfm?id=1046920.1194907
17. Hettwer B, Gehrer S, Güneysu T (2020) Deep neural network attribution methods for leakage analysis and symmetric key recovery. In: Paterson KG, Stebila D (eds) Selected areas in cryptography – SAC 2019. Springer International Publishing, Cham, pp 645–666
18. Heuser A, Picek S, Guilley S, Mentens N (2017) Lightweight ciphers and their side-channel resilience. IEEE Trans Comput PP(99):1–1. https://doi.org/10.1109/TC.2017.2757921
19. Heyszl J, Mangard S, Heinz B, Stumpf F, Sigl G (2012) Localized electromagnetic analysis of cryptographic implementations. In: Dunkelman O (ed) Topics in cryptology – CT-RSA 2012. LNCS, vol 7178. Springer, pp 231–244
20. Kim J, Picek S, Heuser A, Bhasin S, Hanjalic A (2019) Make some noise. unleashing the power of convolutional neural networks for profiled side-channel analysis. IACR Trans Cryptogr Hardw Embed Syst 2019(3):148–179. https://doi.org/10.13154/tches.v2019.i3.148-179. https://tches.iacr.org/index.php/TCHES/article/view/8292
21. Lerman L, Bontempi G, Markowitch O (2014) Power analysis attack: an approach based on machine learning. Int J Appl Cryptol 3(2):97–115. https://doi.org/10.1504/IJACT.2014.062722
22. Maghrebi H, Portigliatti T, Prouff E (2016) Breaking cryptographic implementations using deep learning techniques. In: Security, privacy, and applied cryptography engineering - 6th international conference, SPACE 2016, hyderabad, india, december 14-18, 2016, proceedings, pp 3–26
23. Mangard S, Oswald E, Popp T (2006) Power analysis attacks: revealing the secrets of smart cards. Springer, Berlin. http://www.dpabook.org/
24. Masure L, Dumas C, Prouff E (2019) Gradient visualization for general characterization in profiling attacks. In: Polian I, Stöttinger M (eds) Constructive side-channel analysis and secure design - 10th international workshop, COSADE 2019, Darmstadt, Germany, April 3-5, 2019, proceedings. Lecture notes in computer Science, vol 11421. Springer, pp 145–167. https://doi.org/10.1007/978-3-030-16350-1_9
25. Medwed M, Oswald E (2008) Template attacks on ECDSA. In: International workshop on information security applications. Springer, pp 14–27
26. Medwed M, Oswald E (2008) Template attacks on ECDSA Chung KI, Sohn K, Yung M (eds), vol 5379, Springer. https://eprint.iacr.org/2008/081/
27. Nascimento E, Chmielewski Ł Horizontal clustering side-channel attacks on embedded ecc implementations (extended version). Cryptology ePrint Archive, Report 2017/1204 (2017). https://eprint.iacr.org/2017/1204
28. Nascimento E, Chmielewski Ł, Oswald D, Schwabe P (2017) Attacking embedded ecc implementations through cmov side channels. In: Avanzi R., Heys H (eds) Selected areas in cryptography – SAC 2016. Springer International Publishing, Cham, pp 99–119
29. NIST F. P. (2015) 180-4 secure hash standard (shs), no. August gaithersburg: National Institute of Standards and Technology
30. Özgen E, Papachristodoulou L, Batina L (2016) Classification algorithms for template matching. In: IEEE International symposium on hardware oriented security and trust, HOST 2016, mclean, VA, USA, 2016 (to appear)
31. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E (2011) Scikit-learn: Machine learning in Python. J Mach Learn Res 12:2825–2830
32. Perin G, Ege B, Chmielewski Ł Neural network model assessment for side-channel analysis. IACR Cryptology ePrint Archive 2019, 722 (2019). https://eprint.iacr.org/2019/722
33. Picek S, Heuser A, Alippi C, Regazzoni F (2018) When theory meets practice: A framework for robust profiled side-channel analysis. Cryptology ePrint Archive, Report 2018/1123. https://eprint.iacr.org/2018/1123
34. Picek S, Heuser A, Guilley S (2019) Profiling side-channel analysis in the restricted attacker framework. Cryptology ePrint Archive, Report 2019/168. https://eprint.iacr.org/2019/168
35. Picek S, Heuser A, Jovic A, Bhasin S, Regazzoni F (2019) The curse of class imbalance and conflicting metrics with machine learning for side-channel evaluations. IACR Trans Cryptogr Hardw Embed Syst 2019(1):209–237. https://doi.org/10.13154/tches.v2019.i1.209-237
36. Picek S, Heuser A, Jovic A, Ludwig SA, Guilley S, Jakobovic D, Mentens N (2017) Side-channel analysis and machine learning: a practical perspective. In: 2017 International joint conference on neural networks, IJCNN 2017, anchorage, AK, USA, May 14-19, 2017, pp 4095–4102
37. Picek S, Samiotis IP, Kim J, Heuser A, Bhasin S, Legay AChattopadhyay A, Rebeiro C, Yarom Y (eds) (2018) On the performance of convolutional neural networks for side-channel analysis. Springer International Publishing, Cham
38. Poussier R, Zhou Y, Standaert FXFischer W, Homma N (eds) (2017) A systematic approach to the side-channel analysis of ECC implementations with worst-case horizontal attacks. Springer International Publishing, Cham
39. Prouff E, Strullu R, Benadjila R, Cagli E, Dumas C (2018) Study of deep learning techniques for side-channel analysis and introduction to ASCAD database. IACR Cryptology ePrint Archive 2018, 53
40. Samwel N, Batina L, Bertoni G, Daemen J, Susella R (2018) Breaking ed25519 in wolfSSL. In: Cryptographers' track at the RSA conference. Springer, pp 1–20
41. Schindler W, Huss SA (eds) (2012) Constructive side-channel analysis and secure design - third international workshop, COSADE 2012, Darmstadt, Germany, May 3-4, 2012. proceedings, LNCS, vol 7275. Springer, Berlin
42. Schnorr CP (1991) Efficient signature generation by smart cards. J Cryptol 4(3):161–174
43. Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. arXiv:1409.1556
44. Standaert FX, Malkin T, Yung M (2009) A unified framework for the analysis of side-channel key recovery attacks. In: EUROCRYPT. LNCS, vol 5479. Springer, Cologne, pp 443–461
45. van der Valk D, Picek S (2019) Bias-variance decomposition in machine learning-based side-channel analysis. Cryptology ePrint Archive, Report 2019/570. https://eprint.iacr.org/2019/570

46. van der Valk D, Picek S, Bhasin S (2019) Kilroy was here: The first step towards explainability of neural networks in profiled side-channel analysis. Cryptology ePrint Archive, Report 2019/1477. https://eprint.iacr.org/2019/1477
47. Vapnik VN (1995) The nature of statistical learning theory. Springer, New York
48. Weissbart L, Picek S, Batina L (2019) One trace is all it takes: machine learning-based side-channel attack on edDSA. In: Bhasin S, Mendelson A, Nandi M (eds) Security, privacy, and applied cryptography engineering. Springer International Publishing, Cham, pp 86–105
49. Xu M, Wu L, Zhang X (2018) Power analysis on SM4 with boosting methods. In: 2018 12th IEEE international conference on anti-counterfeiting, security, and identification (ASID). IEEE, pp 188–191
50. Zaid G, Bossuet L, Habrard A, Venelli A (2019) Methodology for efficient cnn architectures in profiling attacks. IACR Trans Cryptogr Hardw Embed Syst 2020(1):1–36. https://doi.org/10.13154/tches.v2020.i1.1-36. https://tches.iacr.org/index.php/TCHES/article/view/8391