A Systematic Evaluation of Profiling through Focused Feature Selection

Stjepan Picek[1], Annelie Heuser[2], Alan Jovic[3], and Lejla Batina[4]

[1] Delft University of Technology, The Netherlands, e-mail: S.Picek@tudelft.nl,
[2] CNRS, IRISA, France, e-mail: annelie.heuser@irisa.fr,
[3] University of Zagreb Faculty of Electrical Engineering and Computing, Croatia, e-mail: alan.jovic@fer.hr,
[4] L. Batina is with Radboud University, The Netherlands, e-mail: lejla@cs.ru.nl

Abstract. Profiled side-channel attacks consist of several steps one needs to take. An important, but sometimes ignored, step is a selection of the points of interest (features) within side-channel measurement traces. A large majority of the related works start the analyses with an assumption that the features are preselected. Contrary to this assumption, here we concentrate on the feature selection step. We investigate how advanced feature selection techniques stemming from the machine learning domain can be used to improve the attack efficiency. To this end, we provide a systematic evaluation of the methods of interest. The experiments are performed on several real-world datasets containing software and hardware implementations of AES, including the random delay countermeasure. Our results show that Wrapper and Hybrid feature selection methods perform extremely well over a wide range of test scenarios and a number of features selected. We emphasize L1 regularization (Wrapper approach) and Linear SVM with recursive feature elimination used after chi square filter (Hybrid approach) that perform well in both accuracy and guessing entropy. Finally, we show that the use of appropriate feature selection techniques is more important for an attack on the high-noise datasets, including those with countermeasures than on the low-noise ones.

1 Introduction

Profiled side-channel attacks (SCAs) have received a lot of attention in recent years because this type of attacks defines the worst case security assumptions. Besides the more traditional choice of template attack, a number of machine learning (ML) techniques have been investigated in this context [?, ?, ?]. The common knowledge from those results suggests that profiled side-channel analysis can be extremely powerful for key recovery, with machine learning being a highly viable choice. Contrary, feature selection, in particular, the usage of ML-based techniques, did not receive significant attention. Early works on template attacks introduced SOST/SOSD [?] as feature selection methods and consequently, most of the follow-up works assume that this step has somehow been performed in a satisfactory, if not optimal, manner. A common strategy often also suggests using Pearson correlation for this purpose, see e.g., [?, ?].

First, we ask a question on the importance of the number of features in a dataset. For a fixed number of training samples, the predictive power of a classifier algorithm eventually reduces as the dimensionality (the number of features) of the problem increases. Consequently, for scenarios with a large number of features, we need to use more training examples, where that number increases exponentially with the number of dimensions. This results in the so-called curse of dimensionality (and the closely related Hughes effect). When discussing features (also known as points of interest, points in time, variables, attributes), we can distinguish among relevant, irrelevant, and redundant features. A meaningful separation in these categories is very important when optimizing the attack strategy and can be divided into the following general directions:

1. feature selection – where the most important subsets of features are selected,
2. dimensionality reduction – where methods like PCA transform the original features into new features, and
3. deep learning techniques like convolutional neural networks that perform implicit feature selection.

The last two techniques can be very successful but they do not provide information about the original features. Such techniques either completely transform the features or use them in a manner too complicated to be understood by human experts. Moreover, deep learning could often have no performance advantage against "standard" machine learning if the number of measurements is not very large [**?**]. Note that in this paper we do not consider comparisons with deep learning techniques, but we refer interested readers to [**?**, **?**, **?**].

There are many papers considering profiled SCA, where the number of features is fixed and the analysis is conducted by considering only the changes in the number of traces or by selecting a more powerful classifier, see e.g., [**?**, **?**]. It is indeed somewhat surprising that the SCA community (until now) did not take a closer look at the feature selection part of the classification process. Similar to the powerful classification methods coming from the ML domain, there are also feature selection techniques one could utilize. To the best of our knowledge, there exists one work that focuses on the feature selection for profiled SCA, but it does not consider ML techniques and it compares only methods known for side-channel analysis either as feature selection techniques or as distinguishers [**?**]. Note that, in leakage detection (see e.g., [**?**]), one is identifying data-dependent but not necessarily model-agnostic leakage information. Consequently, detecting features (points in the power trace) is a task orthogonal to leakage detection, as leakage detection (according to, e.g., TVLA) may not necessarily lead to a successful key recovery. One approach could be as follows: first, use leakage detection to identify possible leakages in the trace, then analyze the corresponding operation, in particular, determine if the model is key sensitive, and finally use feature selection in combination with the underlying model for a profiled distinguisher.

In this paper, we concentrate on feature selection techniques but we also investigate PCA to give insights into performance differences between feature selection and dimensionality reduction techniques. More precisely, we investigate how the efficiency of SCA distinguishers can increase due to feature selection techniques. For this, we employ several feature selection techniques ranging from "simple" ones like the Pearson correlation, which is a de-facto standard in the side-channel community, to more complex approaches such as various Wrapper and Hybrid methods used in ML. To the best of our knowledge, the use of such advanced techniques has never been reported in the context of SCA before.

We show that feature selection is an important step in profiled attacks. We give insights on its use for the following goals: 1) faster training of models, 2) reducing model complexity, 3) improving model performance (when suitable features are selected), 4) reducing overfitting, 5) "correcting" the covariance matrix in template attack when the number of features is too large with respect to the number of traces.

1.1 Our contributions

1. We introduce a novel approach of using ML techniques for the important problem of feature selection in SCA.

2. We demonstrate the potential of Wrapper and Hybrid methods in SCA as they often perform the best for feature selection on the examined datasets.
3. We show how to overcome some previously identified shortcomings of template attacks by the ML techniques, which not just solves the problems but also improves upon the performance of templates as well.
4. We show that our feature selection methods may also be used for dimensionality reduction, having similar or better results than PCA in most cases.
5. All our results are verified on the real-world datasets in different settings. The analysis is very detailed. In total, we consider and run more than 600 experimental scenarios in this work.

1.2 Previous work

Ever since the seminal work of Chari et al. introducing template attacks [?], efforts were put into optimizing those and enlarging their scope. The observation on the profiling, i.e., training phase in template attacks, has naturally led to machine learning techniques and their potential impact on the key recovery phase.

With that respect, a number of ML techniques have been investigated, see e.g., [?, ?, ?]. The results suggested the unquestionable potential of ML techniques for templates and, as such, they were stimulating for further research. However, the limitations of ML approaches were unveiled and their full potential remained unclear. The work of Lerman et al. [?] in particular compared template attacks and machine learning on dimensionality reduction. They concluded that template attacks are the method of choice as long as a limited number of features can be identified in leakage traces containing most of the relevant information. Accordingly, an increase in the number of points of interest favors ML methods. Our results show that the answer is not so simple, i.e., it depends on several factors, such as the number of features, classifiers, implementation details, etc.

Regarding the feature selection problem in SCA, there were very few attempts and works devoted to this topic, as some simple techniques were considered sufficient. Early works introduced SOST/SOSD [?] as feature selection methods and the majority of follow-up papers skipped this step completely. One strategy also suggested using Pearson correlation for this purpose e.g., [?, ?, ?], which is an obvious solution, but does not answer the question on whether we can do better.

Some authors noticed the importance of finding adequate time points in other scenarios. Reparaz et al. [?] introduced a technique to identify tuples of time samples before key recovery for multivariate DPA attacks. Here, typically, the attacker is confronted with a masked implementation, requiring higher-order attacks (hence multiple features corresponding to the right time moments, e.g., when a mask is generated and manipulated). Zheng et al. looked into this specific feature selection question but left ML techniques aside [?]. Picek et al. considered several ML techniques for profiling attacks and investigated the influence of the number of features in the process by applying Information Gain feature selection [?]. Finally, we also question the previous results on dimensionality reduction as our comparison of ML feature selection and PCA [?] (which is feature extraction) favors the former.

2 Background

2.1 Notation

Calligraphic letters (e.g., $\mathcal{X}$) denote sets, capital letters (e.g., X) denote random variables taking values in those sets, and the corresponding lowercase letters (e.g., x) denote their realizations. Let k^*

be the fixed secret cryptographic key (byte) and the random variable T the plaintext or ciphertext of the cryptographic algorithm which is uniformly chosen. The measured leakage is denoted as X and we are particularly interested in multivariate leakage $\boldsymbol{X} = X_1, \ldots, X_D$, where D is the number of time samples or features (attributes) in machine learning terminology.

Considering a powerful attacker who has a device and the knowledge on the secret key implemented, a set of N profiling traces $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$ is used to estimate the leakage model beforehand. Note that this set is multi-dimensional (i.e., it has dimension $D \times N$). In the attack phase, the attacker then measures additional traces $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_Q$ from the device under attack to recover the unknown secret key k^*.

2.2 Datasets

We use three datasets that we consider to be a representative sample of commonly encountered scenarios. More precisely, one dataset is without countermeasures and with a small amount of noise, which is a relatively easy scenario for profiled attack when the number of measurements is sufficient. Next, we consider one dataset without countermeasures but with a large amount of noise. There, we are approaching more realistic scenarios where profiled techniques have problems in reaching high performance. Finally, the last dataset has a countermeasure in the form of random delays, which represents a realistic scenario for evaluation. We do not consider datasets with masked implementations since we assume that the mask is different in the training and testing phase, which makes feature selection more complex.

DPAcontest v4 Dataset [?] The 4th version of the DPAcontest dataset provides measurements of a masked AES software implementation. As the mask is known, one can easily turn it into an unprotected scenario. As this is a software implementation, the most leaking operation is not the register writing but the processing of the S-box operation, and thus the attack targets the first round. Hence, the leakage model is

$$Y(k^*) = \texttt{Sbox}[P_{b_1} \oplus k^*] \oplus \underbrace{M}_{\text{known mask}}, \qquad (1)$$

where P_{b_1} is a plaintext byte and we choose $b_1 = 1$. Compared to the measurements from the 2nd version of the dataset, the SNR is much higher with a maximum value of 5.8577. For our experiments, we start with a preselected window of 3 500 features from the original trace (we simply preselect all features around the S-box operation).

AES_HD Dataset This dataset is chosen in order to target an unprotected implementation of AES-128 encryption specification. The core of AES-128 was written in VHDL in a round based architecture, taking 11 clock cycles for each encryption. A UART module is wrapped around the core to enable external communication. The module is designed to allow accelerated measurements so avoid any DC shift due to environmental variation over prolonged measurements. The total area footprint of the design contains 1 850 LUT and 742 flip-flops. Xilinx Virtex-5 FPGA of a SASEBO GII evaluation board was used to implement the design. Side-channel traces were measured using a high sensitivity near-field EM probe, which was placed over a decoupling capacitor on the power line. Measurements were sampled on the Teledyne LeCroy Waverunner 610zi oscilloscope. A suitable and commonly used (HD) leakage model, when attacking the last round of an unprotected hardware implementation, is the register writing in the last round [?], i.e.,

$$Y(k^*) = HW(\underbrace{\mathtt{Sbox}^{-1}[C_{b_1} \oplus k^*]}_{\text{previous register value}} \oplus \underbrace{C_{b_2}}_{\text{ciphertext byte}}), \quad (2)$$

where C_{b_1} and C_{b_2} are two ciphertext bytes, and the relation between b_1 and b_2 is given through the inverse ShiftRows operation of AES. $b_1 = 12$ was chosen, which resulted in $b_2 = 8$, as it is one of the easiest bytes to attack. The obtained measurements that form the dataset are relatively noisy and the resulting model-based SNR (signal-to-noise ratio), i.e., $\frac{var(signal)}{var(noise)} = \frac{var(y(t,k^*))}{var(x-y(t,k^*))}$ has a maximum value of 0.0096. In total, 500 000 traces were captured corresponding to 500 000 randomly generated plaintexts, each trace with 1 250 features. However, not all the traces were used for training and testing the model. The evaluations details are given in Section 4. As this implementation leaks in the HD model, we denote this implementation as AES_HD. The dataset is publicly available at https://github.com/AESHD/AES_HD_Dataset.

Random Delay Dataset [?] As our third use case, we use an actual protected implementation to prove the potential of our approach. Our target is a software implementation of AES on an 8-bit Atmel AVR microcontroller with implemented random delay countermeasure, as described by Coron and Kyzhvatov in [?]. We mounted our attacks against the first AES key byte by targeting the first S-box operation. The dataset consists of 50 000 traces of 3 500 features each. For this dataset, the SNR has a maximum value of 0.0556. This dataset is publicly available at https://github.com/ikizhvatov/randomdelays-traces.

2.3 Profiled Attacks and Guessing Entropy

In this section, we introduce the methods we use in the classification tasks. Note that we opted to work with only a small set of techniques, since we aim to explore how to find the best possible subset of features, while the classification task should be considered as just a means of comparison among feature selection methods. Consequently, we try to be as "method-agnostic" as possible and we note that for each set of features, one could probably find a classification algorithm performing slightly better. As noted in [?], there is no need to include many classifiers to obtain the best solutions. Usually, one of the best classifiers suffices, which is certainly the Random Forest algorithm. We use Random Forest for classification in all the experiments since it provides stable and accurate results [?,?]. Also, linear kernel Support Vector Machine is used because of its efficiency and accuracy for Wrapper and Hybrid based feature selection, as explained in continuation. As mentioned in Section 1.2, the template attack (TA) (i.e., template attack classifier) is the traditional method of choice in SCA, especially when the number of features is small. Consequently, we use TA classifier and its pooled version [?] for comparison with Random Forest.

Random Forest Random Forest (RF) is a well-known ensemble decision tree learner [?]. Decision trees choose their splitting attributes from a random subset of k attributes at each internal node. The best split is taken among these randomly chosen attributes and the trees are built without pruning. RF is a stochastic algorithm because of its two sources of randomness: bootstrap sampling and attribute selection at node splitting. Learning time complexity for RF is approximately $\mathrm{O}\big(I \cdot$

$k \cdot N \cdot logN\big)$, where I is the number of trees in the forest, k is the number of features considered at each node in each tree (usually $k = \sqrt{D}$, D being the total number of features) and N is the number of samples. We use RF as the classifier of choice for multiclass classification in our work. This is mainly in line with available research [**?**], where it is expected that RF will perform among the best classifiers. RF is used in all evaluations of the reduced sized feature sets.

Support Vector Machines Support Vector Machine (SVM) is a kernel-based machine learning family of methods used to accurately classify both linearly separable and linearly inseparable data [**?**]. The basic idea when the data are not linearly separable is to transform them to a higher dimensional space by using a transformation kernel function. In this new space, the samples can usually be classified with higher accuracy. We use SVM with the linear kernel as the classification algorithm for Wrapper and Hybrid based feature selection (see Sections 3.2 and 3.3). Linear kernel SVM is used instead of a polynomial or radial based SVM, because advanced feature selection approaches require the construction of many models, which is computationally intensive and therefore unsuitable for nonlinear kernel function based SVM. The time complexity range for linear kernel SVM is $\mathrm{O}\big(DN\big)$, which is significantly less than $\mathrm{O}\big(DN^3\big)$ for the time complexity of radial kernel SVM. We note that utilizing a linear kernel is an efficient choice when the number of dimensions is high (as in our case) or when we can assume there is a linear separation between data.

Template Attack The template attack relies on the Bayes theorem and considers the features as dependent. In the state-of-the-art, template attack relies mostly on a normal distribution. Accordingly, template attack assumes that each $P(\boldsymbol{X} = \boldsymbol{x}|Y = y)$ follows a (multivariate) Gaussian distribution that is parameterized by its mean and covariance matrix for each class Y. The authors of [**?**] propose to use only one pooled covariance matrix averaged over all classes Y to cope with statistical difficulties and thus a lower efficiency. Besides the standard approach, we additionally use this version of the template attack in our experiments. The time complexity for TA is $\mathrm{O}\big(ND^2\big)$ in the training phase and $\mathrm{O}\big(|\mathcal{Y}|D^2\big)$ in the testing phase ($|\mathcal{Y}|$ is the number of classes).

2.4 Guessing Entropy

After running profiled attacks, we obtain accuracy as the measure of performance for our classifiers. Since this measure can be often misleading in SCA, especially in the Hamming weight scenario [**?**], we also use the guessing entropy to properly assess the performance of our feature selection and classification techniques [**?**]. A side-channel adversary $A_{E_K,L}$ conducts experiment $\mathsf{Exp}_{A_{E_K,L}}$, with time-complexity τ, memory complexity m, and making Q queries to the target implementation of the cryptographic algorithm. The attack outputs a guessing vector g of length o, and is considered a success if g contains the correct key k^*. o is also known as the order of the success rate.

Guessing entropy (GE) measures the average number of key candidates to test after the attack. The Guessing entropy of the adversary $A_{E_k,L}$ against a key class variable S is defined as:

$$GE_{A_{E_K,L}}(\tau, m, k^*) = \mathsf{E}[\mathrm{Exp}_{A_{E_K,L}}].$$

3 Feature Selection Techniques

A successful feature selection algorithm should output an optimal or near-optimal subset of features while ignoring the rest. Such algorithms can be classified into three broad classes of feature selection techniques: Filter methods, Wrapper methods, and Hybrid methods [?]. The Wrapper and Hybrid classes of methods are known to either increase or retain the accuracy of the Filter methods [?, ?].

Only the first three presented Filter methods (Pearson correlation coefficient, SOSD, SOST) have been used as feature selection techniques for side-channel analysis in previous works, whereas the remaining methods, to the best of our knowledge, have never been studied to find the most important features in SCA traces. We consider methods from all three classes of feature selection techniques in order to cover a wide set of feature selection cases. The choice of individual methods from these classes is based on our previous experience and the fact that all the methods are well-established in the field of feature selection, as noted in the corresponding subsections below. We also consider in this section the Principal Component Analysis. While PCA is, strictly speaking, dimensionality reduction and not feature selection technique, we compare it with the feature selection methods, because it is often used in SCA attacks.

3.1 Filter Selection Methods

The selection of features using Filter methods is independent of the classifier method. Features are selected based on their scores obtained after running various types of statistical tests. We depict the Filter methods principle in Figure 1, with methods and numbers pertaining to our work.

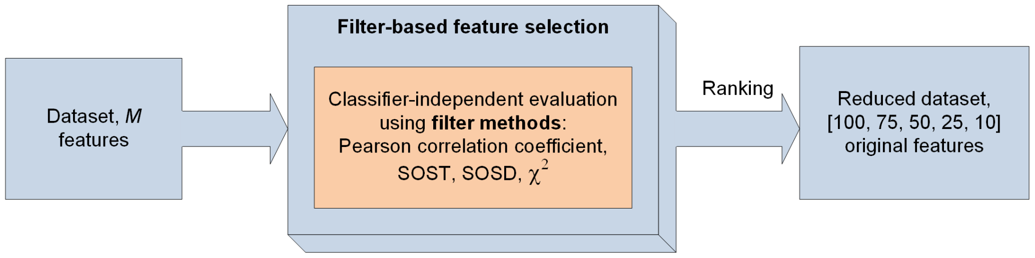

Dataset, *M* features
Filter-based feature selection
Classifier-independent evaluation using **filter methods**: Pearson correlation coefficient, SOST, SOSD, χ^2
Ranking
Reduced dataset, [100, 75, 50, 25, 10] original features

Fig. 1: Filter methods

Pearson Correlation Coefficient Pearson correlation coefficient measures linear dependence between two variables, x and y, in the range $[-1, 1]$, where 1 is the total positive linear correlation, 0 is no linear correlation, and -1 is the total negative linear correlation. Pearson correlation for a sample of the entire population is defined by [?]:

$$Pearson(x, y) = \frac{\sum_{i=1}^{N}((x_i - \bar{x})(y_i - \bar{y}))}{\sqrt{\sum_{i=1}^{N}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{N}(y_i - \bar{y})^2}}. \tag{3}$$

We calculate Pearson correlation for the target class variables HW and intermediate value, which consists of categorical values that are interpreted as numerical values. The features are ranked in descending order of the coefficient.

SOSD

In [?], the authors proposed the sum of squared differences as a selection method, simply as:

$$SOSD(x, y) = \sum_{i,j>i}(\bar{x}_{y_i} - \bar{x}_{y_j})^2, \tag{4}$$

where $\bar{x}_{y_i}$ is the mean of the traces where the model equals y_i. Because of the square, SOSD is always positive. Another advantage of using it is to emphasize big differences in means.

SOST SOST is the normalized version of SOSD [?] and is thus equivalent to the pairwise Student's t-test:

$$SOST(x,y) = \sum_{i,j>i} \left((\bar{x}_{y_i} - \bar{x}_{y_j}) / \sqrt{\frac{\sigma^2_{y_i}}{n_{y_i}} + \frac{\sigma^2_{y_j}}{n_{y_j}}} \right)^2 \tag{5}$$

with n_{y_i} and n_{y_j} being the number of traces where the model equals to y_i and y_j, respectively.

Chi Square Chi square (χ^2) is a measure of dependence between two stochastic variables. It is a cumulative test statistic, which asymptotically approaches a χ^2 distribution. In the general case, χ^2 distribution may be obtained from the sum of squares of the set of k standard normal random variables, where k are the degrees of freedom. χ^2 test statistic for each feature-class pair may be calculated using the expression:

$$\chi^2 = \sum_{i=1}^{n} \frac{(x_{y_i} - E_{y_i})^2}{E_{y_i}}. \tag{6}$$

Here, n is the number of discrete categories, x_{y_i} is the observed value of category y_i, and E_{y_i} is the expected (theoretical) frequency of category y_i. Note that, for numerical features, the values need to be discretized to obtain categories before calculation of the statistic. By using the statistic, we proceed to remove the features that are the most likely to be independent of class attribute and therefore irrelevant for classification. Finally, since this measure works only for non-negative values, before using it, we normalize the data into $[0,1]$ range. The complexity of calculating the measure is $O(N \cdot D)$.

3.2 Wrapper Selection Methods

In Wrapper methods, there is a feature selection algorithm implemented as a wrapper around a classifier [?]. The feature selection algorithm searches for a good subset by using a classifier algorithm as a part of the function evaluating feature subsets, as depicted in Figure 2. Here, the classifier algorithm is considered as a black box and is run on the dataset with different sets of features removed from the data. The subset of features with the highest evaluation is chosen as the final set on which to run the classifier [?]. Note that, since Wrapper methods check many different subsets, the feature selection process is often treated as a high-dimensional problem. L1 regularization with linear SVM is used for Wrapper based feature selection in all the experiments, because the combination is sufficiently fast, accurate, and memory-undemanding. The other potential candidates that could have been used are naive Bayes, linear SVM and k-nearest neighbors. However, although the use of solely these classifiers may be faster compared to L1 regularization with linear SVM, they may not be as accurate in estimating the accuracy of feature subsets. On the other hand, methods such as random forest, neural network, non-linear SVM, etc. are more complex and are not typically used as wrappers, since they exhibit non-linear complexity dependence on the number of instances.

L1-based Feature Selection In general, regularization encompasses methods that add a penalty term to the model, which then reduces the overfitting and improves generalizations. L1 regularization works by adding a regularization term $\alpha \cdot R(\theta)$, where θ represents the parameters of the model that is used to penalize large weights/parameters. For a D-dimensional input (i.e., the number of features equal to D), $R(\theta)$ is equal to $\sum_{i=1}^{D} |\theta_i|$. In the regularization term, α controls

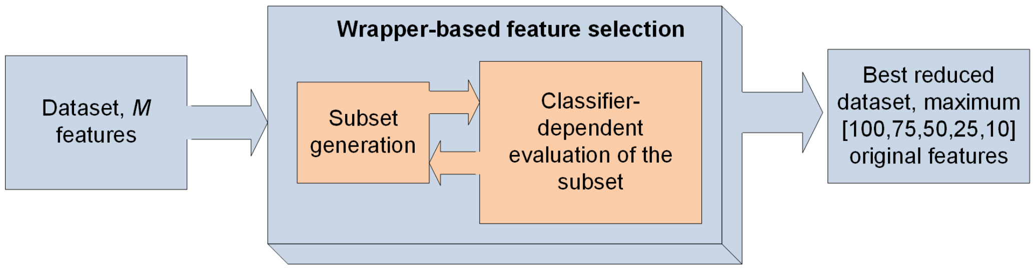

Wrapper-based feature selection
Dataset, *M* features
Subset generation
Classifier-dependent evaluation of the subset
Best reduced dataset, maximum [100,75,50,25,10] original features

Fig. 2: Wrapper methods

the trade-off between fitting the data and having small parameters. By adding a penalty for each non-zero coefficient, the expression forces weak features to have zero as coefficients, where a zero value means that the feature is omitted from the set. The usage of L1 regularization as a tool for feature selection is well known, for example, the linear least-squares regression with L1 regularization (Lasso) algorithm [**?**]. There can be certain effects with L1 regularization when used for feature selection: most notably, out of a group of highly correlated features, L1 regularization will tend to select an individual feature [**?**].

3.3 Hybrid Selection Methods

Hybrid methods combine Filter and Wrapper techniques. First, a filter method is used in order to reduce the feature space dimension space. Then, a wrapper method is utilized to find the best candidate subset. Hybrid methods usually achieve high accuracy that is characteristic to wrappers and high efficiency characteristic to filters. We depict a diagram for Hybrid methods, as used in this paper, in Figure 3. In our experiments, we first use χ^2 to reduce the number of features to 250 in order to further reduce the runtime of Hybrid selection techniques. Then, we apply either the Linear SVM selection or the Stability selection technique.

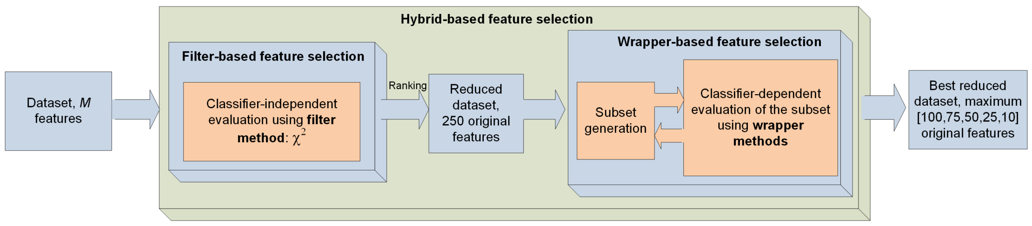

Hybrid-based feature selection
Filter-based feature selection
Wrapper-based feature selection
Dataset, *M* features
Classifier-independent evaluation using **filter method**: χ^2
Ranking
Reduced dataset, 250 original features
Subset generation
Classifier-dependent evaluation of the subset using **wrapper methods**
Best reduced dataset, maximum [100,75,50,25,10] original features

Fig. 3: Hybrid methods

Linear SVM Based Hybrid Selection We use a recursive feature elimination approach with linear SVM Wrapper to obtain the target reduced feature sets. The method was first described in [**?**]. Here, the "best-first" backward direction search method is used. This strategy uses greedy hill climbing, starting from the full feature subset and inspecting how the elimination of a feature or a set of features from the starting set influences the output of the classifier. The feature(s) whose removal influences the accuracy the least are eliminated from the set.

Stability Selection Stability selection is a method based on subsampling in combination with some classification algorithm (that can work with high-dimensional data) [**?**]. The key concept of stability selection is the stability paths, which is the probability for each feature to be selected when randomly resampling from the data. In other words, a subsample of the data is fitted to the L1 regularization model, where the penalty of a random subset of coefficients has been scaled. By repeating this procedure n times, the method will assign high scores to the features that are repeatedly selected. We use multinomial logistic regression for this task and we set the number of randomized models n to 25. Multinomial logistic regression uses a linear predictor function $f(k,i)$ to predict the probability that observation i has the outcome k, of the form $f(k,i) = \beta_{0,k} + \beta_{1,k}x_{1,i} + \ldots + \beta_{M,k}x_{M,i}$ where $\beta_{M,k}x_{M,i}$ is a regression coefficient of the mth variable and the kth outcome. The β coefficients are estimated using the maximum likelihood estimation, which requires finding a set of parameters for which the probability of the observed data is the greatest.

3.4 Principal Component Analysis

Principal component analysis (PCA) is a well-known linear dimensionality reduction method that may use Singular Value Decomposition (SVD) of the data matrix to project it to a lower dimensional space [**?**]. PCA creates a new set of features (called principal components) that are linearly uncorrelated, orthogonal, and form a new coordinate system. The number of components equals the number of original features. The components are arranged in a way that the first component covers the largest variance by a projection of the original data and the subsequent components cover less and less of the remaining data variance. The number of kept components, designated with L, maximizes the variance in the original data and minimizes the reconstruction error of the data transformation. The Python implementation of PCA uses either the LAPACK implementation of the full SVD or a randomized truncated SVD by the method of Halko et al. [**?**], depending on the shape of the input data and the number of components selected to extract. We experiment with L values in the range $[10, 25, 50, 75, 100]$.

4 Experimental Evaluation

In our experiments, we are interested in supervised (profiled) problems that have a large number of features (sample points from power traces) D but where there could exist a small subset D' of features that is sufficient to classify the features X according to the classes Y. We use the previously described Filter, Wrapper, and Hybrid methods to reduce the number of features found in the original datasets to the smaller subsets of sizes $[10, 25, 50, 75, 100]$. The investigated subset sizes are selected based on the usual number of features considered in related work (see Section 1.2). We have also tried increasing the number of features, inspecting up to 200 features. The results were not better and the analysis was prolonged. Specifically, the features in range 101-200 lead to no improvement in accuracy or guessing entropy with respect to only the first 100 included features, for all methods.

Once the best feature subsets are selected, we run three profiled attacks: Random Forest, TA, and TA pooled (TA$_p$) for each feature selection technique to evaluate its efficiency. We use multiple profiled attacks to avoid potential effects that a certain feature selection technique could have on a specific attack. We emphasize that the goal here is not to compare the efficiency of attacks and, consequently, we do not give such an analysis. Finally, we note that for the Wrapper methods, selecting the exact number of features can be difficult (since the methods can simply discard multiple features) and, consequently, subset sizes of $[10, 25, 50, 75, 100]$ represent an upper bound on the number of actually selected features.

From the initial datasets, we randomly select 10 000 power traces for training and another 25 000 randomly selected traces for testing. We opted to have a larger test set to obtain meaningful results with guessing entropy. For evaluation on the training set, we conduct 5-fold cross-validation and use the averaged results of individual folds to select the best classifier parameters. We report the results from testing phase only and we present them as the accuracy (%) of the classifier, where the accuracy is the number of correctly classified traces divided by the total number of traces. All experiments are done with MATLAB and Python (scikit-learn library) tools. For the L1 regularization with linear SVM Wrapper, Hybrid Linear SVM, and Hybrid Stability selection, we tune the parameter C for each subset size. For Linear SVM, we further select the step equal to 5 to remove features – in each iteration of the algorithm, we discard 5 least important features from the feature set. For RF, we experiment with $I = [10, 50, 100, 200, 500, 1\,000]$ trees in the tuning phase, with no limit to tree size. Based on the tuning phase, we select 500 trees for the HW model and 100 trees for the intermediate value model.

4.1 Results

We give results for test set accuracy in Tables 1– 7 and for guessing entropy in Figures 4– 6. Due to the lack of space, we do not show GE results for all tested scenarios, but only for a representative subset of them. For each size of the feature subset in Tables 1– 6, we give the best obtained solution in a cell with the gray background color. For Table 7, the gray background of a cell indicates a better result for PCA than for all feature selection methods.

DPAcontest v4 Dataset Tables 1 and 2 display the results for DPAcontest v4 with the HW model and intermediate value model, respectively. For the HW model, we observe that Linear SVM Hybrid method is, by far, the best performing feature selection method when considering accuracy, comparable or outperformed only slightly by PCA for a larger number of features (see the first row of Table 7). Linear SVM works very well for the low-noise scenario and when the number of classes is rather low (9 for the HW model). Note that the results for Linear SVM are comparable to the results for L1 and stability selection for the intermediate value model (256 classes), thus suggesting that the method is more appropriate for the smaller number of classes.

Figure 4 shows that, for GE, the changes between the different techniques are rather small with an advantage of Linear SVM, L1, and Correlation using 10 features. When considering 100 features, all techniques perform almost equivalently, except for PCA, which performs the worst. Due to the low noise present in this scenario, all the feature selection methods have found highly similar features, see Figure 7 later in the paper. Comparing the results for 100 and 10 features, it is shown that when the number of features is large (i.e., 100), there is a higher chance that most of the informative features are included by all methods than when the number of features is small (i.e., 10). For 10 features, there is a larger difference between the methods, indicating that some important features are omitted by some methods.

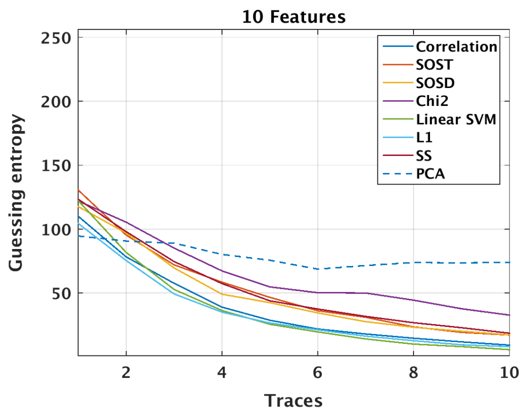
When considering the intermediate value model (see Table 2), we observe that the Wrapper and Hybrid methods have the highest accuracy, outperforming filters and PCA. Here, even accuracy for 100 features varies significantly.

Table 1: Accuracy for DPAcontest v4 - HW model

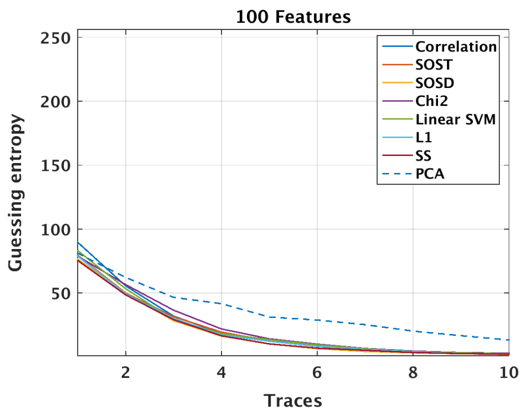
Pearson correlation					
Classifier	10	25	50	75	100
TA	69.8	72.5	0.3	2.8	41.8
TA (pooled)	68.5	71.7	80.9	81.7	91.4
RF	74.5	81.4	84.6	84.1	85.8
SOST					
Classifier	10	25	50	75	100
TA	71.5	73.9	0.5	20	0.2
TA (pooled)	69.4	73.4	80.6	86.6	91.6
RF	74.2	81.4	84.3	84.7	86
SOSD					
Classifier	10	25	50	75	100
TA	72.2	74.7	8.7	8.7	3.8
TA (pooled)	69.8	74.4	77.3	84.5	89.6
RF	75.9	81.6	82.5	83.7	84.4
χ^2					
Classifier	10	25	50	75	100
TA	72.2	74.3	36.9	30.5	0.5
TA (pooled)	69.8	74.3	81.1	84.9	91.6
RF	76.2	81.4	84.7	84.5	86.4
Linear SVM wrapper					
Classifier	10	25	50	75	100
TA	19.9	51.6	1.5	4.6	1.3
TA (pooled)	14	49.7	85.3	98.1	98.1
RF	89.7	91.9	92.3	91.6	91.2
L1 regularization					
Classifier	10	25	50	75	100
TA	8.4	32.1	1.3	91.5	11
TA (pooled)	9.6	27.4	90.1	97	97.3
RF	80.4	86.7	88.8	89.8	89.4
Stability selection					
Classifier	10	25	50	75	100
TA	20.7	31.3	0	86.2	30.3
TA (pooled)	16.3	28.9	92.2	97.4	98
RF	75.3	91.4	91.5	91.2	90.7

For GE in the intermediate value model, we observe the same phenomena as for the HW model: all the techniques are differing only slightly when considering a low number of features and become closer when more features are considered.

AES_HD Dataset For AES_HD dataset, we give results in Tables 3 and 4 for HW model and intermediate value model, respectively. For HW model, some observations made for DPAcontest v4 also apply for AES_HD. We see that having more features also, in general, results in higher accuracy. Still, in some scenarios, accuracy for the smaller feature set size is even higher than for larger feature set sizes but those differences are rather small. Differing from DPAcontest v4, for AES_HD, we do



(a) 10 features, HW, RF.



(b) 100 features, HW, RF.

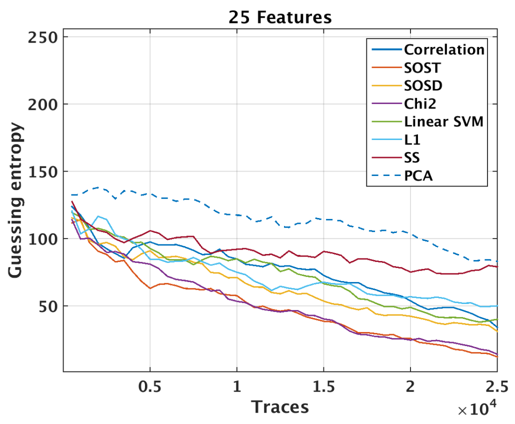
Fig. 4: Guessing entropy, DPAcontest v4 dataset

not observe a significant drop in performance when using only 10 features. PCA performs well for this case, slightly outperforming feature selection methods with respect to accuracy (see the third row of Table 7).

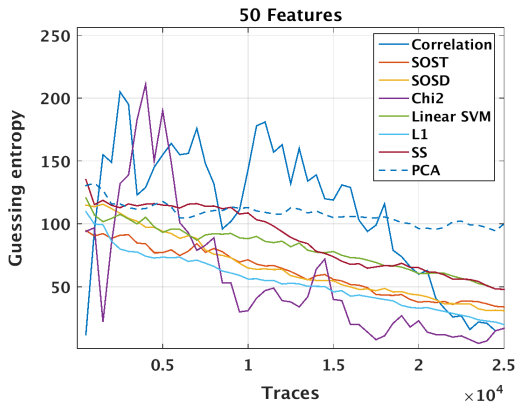
For the intermediate value model, the accuracy is very low and even looks like random guessing (1/256, see Table 4). The results show that there is no significant difference in behavior for any technique. This is expected, since there are 256 classes and only 10 000 measurements in the training phase, which is barely enough to have results better than random guessing when dealing with such difficult datasets.

We are able to reach a low guessing entropy (i.e., retrieve the secret key), as Figure 5 clearly illustrates. More specifically, Figures 5a until 5f depict GE results for the AES_HD dataset for HW and intermediate value model ranging between 25 and 100 features. In this high-noise scenario, we observe a more distinct behavior for different techniques. Generally, PCA-based attack mostly performs comparable or worse than the feature selection techniques. In Figures 5b and 5d, one can observe that Correlation for 50 features or Correlation and SOSD for 100 features only become stable when using a large number of measurements in the attacking phase with RF. Figures 5e and 5f show that, despite approximately even accuracy for the intermediate model, there are marked differences among some methods with respect to GE. In these cases, when using TA pooled classifier, PCA, Linear SVM, and Chi2 underperform with respect to other methods.

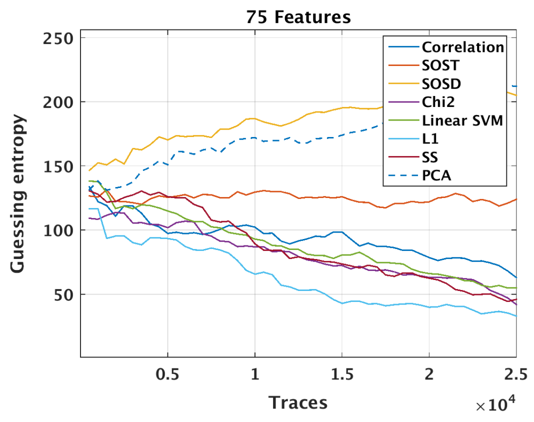
Random Delay Dataset Finally, Tables 5 and 6 give results for the Random Delay dataset for HW and intermediate value model, respectively. For the HW model, the highest accuracies are spread among the feature selection methods. Namely, for 5 scenarios, we have 4 different techniques reaching the highest accuracies. PCA performs slightly worse than feature selection methods for HW model. Figure 6 shows that GE results are also widely spread. For HW, as well as for intermediate value model, Linear SVM and L1 usually perform well, while in some rare cases, SOST also performs well, while Linear SVM underperforms. (Figure 6f). We can observe that, again, Linear SVM is suitable when a small amount of features is selected (see Figures 6a and 6c). Comparing the results of RF



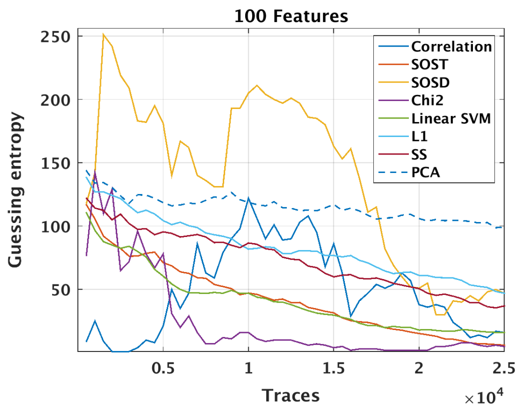
(a) 25 features, HW, TA.



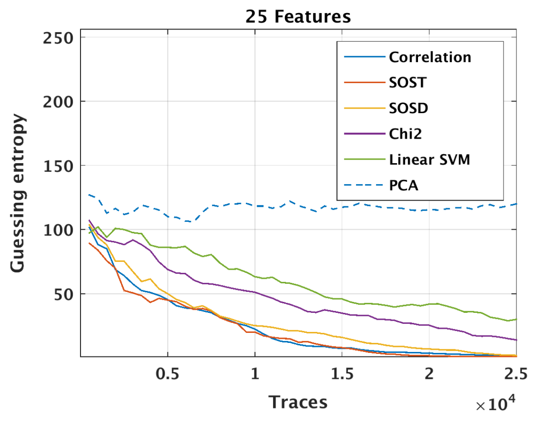
(b) 50 features, HW, RF.



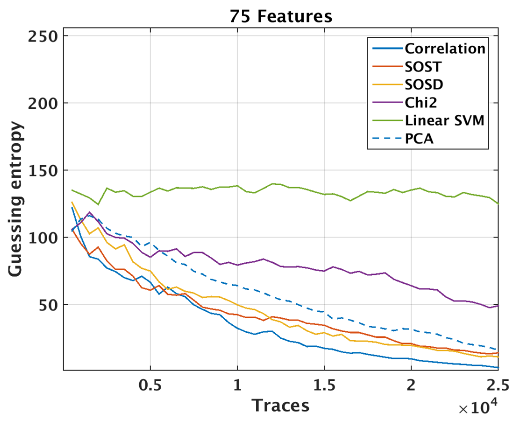
(c) 75 features, HW, TA.



(d) 100 features, HW, RF.



(e) 25 features, intermediate value, TA pooled.



(f) 75 features, intermediate value, TA pooled.

Fig. 5: Guessing entropy, AES_HD

Table 2: Accuracy for DPAcontest v4 - intermediate value model

Pearson correlation

Classifier	10	25	50	75	100
TA	11.4	0.4	0.4	0.2	0.4
TA (pooled)	15.8	18.0	20.5	31.1	53
RF	13	20.4	25	29.8	36.2

SOST

Classifier	10	25	50	75	100
TA	11.5	0.1	0.1	0.1	0
TA (pooled)	16.2	32.6	51.7	62.3	64.3
RF	15.3	32.5	38.4	42.2	43.1

SOSD

Classifier	10	25	50	75	100
TA	17.9	0.3	0.1	0	0.1
TA (pooled)	23.2	38	56.1	64.4	65.7
RF	18	30.1	39.9	41.2	42.1

χ^2

Classifier	10	25	50	75	100
TA	1.6	0.3	0.1	0.2	0.2
TA (pooled)	1.6	3.7	28.4	57.1	69
RF	23.9	34.7	41.2	44	45.8

Linear SVM wrapper

Classifier	10	25	50	75	100
TA	26.8	20.2	0.	0.1	0
TA (pooled)	24.3	43.9	64.9	71	74.3
RF	24.6	44.5	70.8	74.2	75.5

L1 regularization

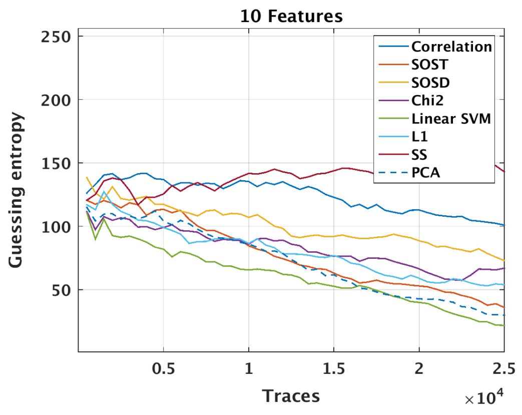
Classifier	10	25	50	75	100
TA	28.7	0	0.2	0	0
TA (pooled)	26.1	51.8	66.9	74	75.6
RF	28.8	53.1	73.9	74.4	75.3

Stability selection

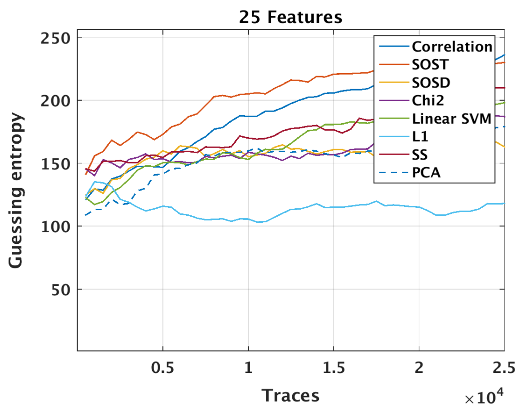
Classifier	10	25	50	75	100
TA	25.2	0.2	0	0.3	0.1
TA (pooled)	21.4	47.4	64.3	73.1	75.7
RF	24.8	46.9	65.6	71	75.2

and TA pooled classifiers for the intermediate value model, RF was shown to provide significantly more stable GE results. PCA-based attacks perform comparably to most feature selection methods on this dataset.

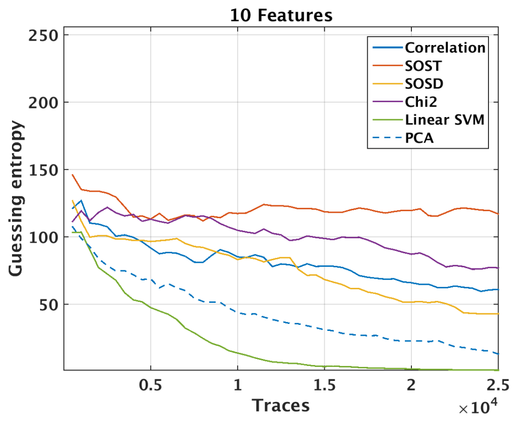
Feature Illustration In Figures 7a and 7b, we depict 100 selected features for all datasets, HW and intermediate value models, respectively. The visualization allows a more detailed inspection in the behavior of feature selection methods. Namely, if different methods find similar features, then the selected features are probably globally more relevant than the others for the classification problem (assuming that not all the methods are wrong). If different methods find different features, while



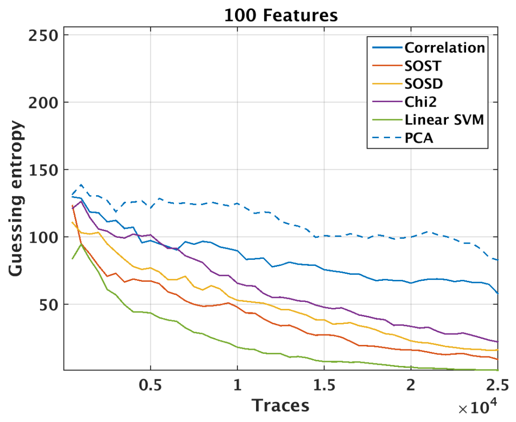
(a) 10 features, HW, RF.



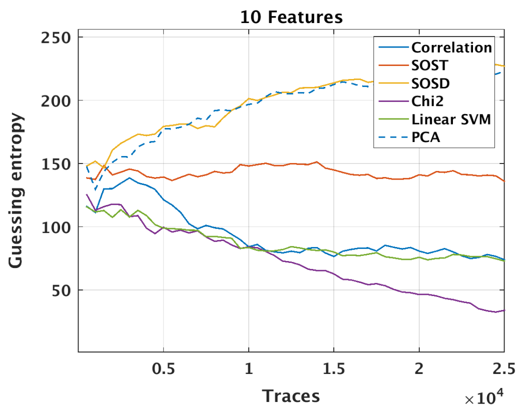
(b) 25 features, HW, TA.



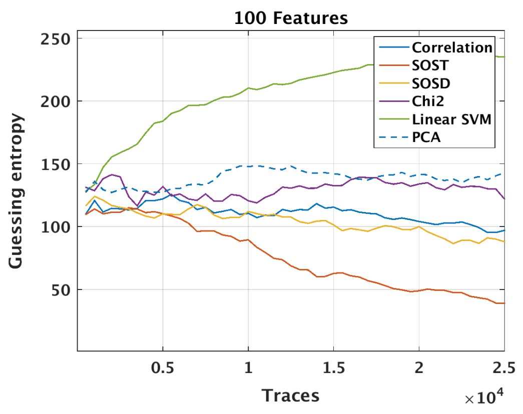
(c) 10 features, intermediate value, RF.



(d) 100 features, intermediate value, RF.



(e) 10 features, intermediate value, TA pooled.



(f) 100 features, intermediate value, TA pooled.

Fig. 6: Guessing entropy, Random Delay dataset

Table 3: Accuracy for AES_HD - HW model

Pearson correlation

Classifier	10	25	50	75	100
TA	11.4	17.7	5.3	5	1.5
TA (pooled)	4.4	5.6	7.9	8.2	8.9
RF	23.8	24.7	25.2	25.3	25.5

SOST

Classifier	10	25	50	75	100
TA	10.5	17.7	5.4	13	11.1
TA (pooled)	4.3	5.7	7.9	8	9.5
RF	23.7	24.6	24.6	25	25.4

SOSD

Classifier	10	25	50	75	100
TA	10.5	18.4	1.4	0.7	0.6
TA (pooled)	4.3	6	8.2	8.8	9.5
RF	23.7	25.2	25.9	26.4	26.2

χ^2

Classifier	10	25	50	75	100
TA	11.4	18.2	4.1	2.3	1.4
TA (pooled)	4.4	6.4	7.8	9.1	9.5
RF	23.8	25	24.8	25.7	25.3

Linear SVM wrapper

Classifier	10	25	50	75	100
TA	11	16.3	1.9	10.2	4.3
TA (pooled)	5.1	6	8.2	8.9	9.9
RF	24.4	24.9	25.4	25.8	25.8

L1 regularization

Classifier	10	25	50	75	100
TA	10.1	16.3	7.1	3.3	7.9
TA (pooled)	5.5	7.5	8.1	9.3	9.9
RF	23.6	24.9	25.3	26	25.7

Stability selection

Classifier	10	25	50	75	100
TA	11	16.1	8.5	3.8	10.4
TA (pooled)	5.6	5.7	6.9	7.8	8.2
RF	24.7	25.8	25.7	25.8	26

obtaining similarly good classification results, then this suggests that many features are informative enough to produce accurate models. If, however, different methods find different features, while obtaining different classification results (some better than others), then this suggests that some methods perform better selection than the others. For DPAcontest v4 and both considered models, a large part of the selected features for all techniques is the same. Consequently, the obtained results for both accuracy and GE are similar. This indicates that in a low-noise scenario, the choice among the feature selection methods is not crucial. For the AES_HD dataset, we can observe that there are some regions where all the selection techniques find relevant features. Interestingly, for L1 regularization and HW model, the selected features are much less grouped when compared to

Table 4: Accuracy for AES_HD - intermediate value model

Pearson correlation

Classifier	10	25	50	75	100
TA	0.3	0.4	0.4	0.3	0.4
TA (pooled)	0.4	0.4	0.4	0.4	0.4
RF	0.4	0.4	0.4	0.4	0.3

SOST

Classifier	10	25	50	75	100
TA	0.4	0.4	0.4	0.4	0.4
TA (pooled)	0.4	0.5	0.4	0.5	0.5
RF	0.3	0.4	0.4	0.4	0.3

SOSD

Classifier	10	25	50	75	100
TA	0.3	0.4	0.3	0.4	0.4
TA (pooled)	0.4	0.4	0.4	0.4	0.4
RF	0.3	0.4	0.4	0.4	0.5

χ^2

Classifier	10	25	50	75	100
TA	0.4	0.4	0.4	0.4	0.4
TA (pooled)	0.4	0.4	0.4	0.4	0.4
RF	0.4	0.4	0.4	0.5	0.4

Linear SVM wrapper

Classifier	10	25	50	75	100
TA	0.4	0.4	0.4	0.4	0.4
TA (pooled)	0.4	0.4	0.4	0.4	0.5
RF	0.4	0.4	0.4	0.4	0.4

L1 regularization

Classifier	10	25	50	75	100
TA	0.3	0.4	0.4	0.5	0.5
TA (pooled)	0.4	0.4	0.4	0.4	0.4
RF	0.4	0.4	0.4	0.4	0.4

Stability selection

Classifier	10	25	50	75	100
TA	0.4	0.4	0.5	0.4	0.4
TA (pooled)	0.4	0.4	0.3	0.4	0.5
RF	0.4	0.4	0.4	0.4	0.5

the other selection techniques. The similarity in the selected features is reduced compared to the DPAcontest v4 dataset. This indicates that, for the high-noise scenario, the choice of the methods is more important than for the low-noise one. Finally, for the Random Delay dataset, all techniques select quite different features, which results in a significantly different performance, as seen in the GE results. This suggests that, for the high-noise with countermeasures scenarios, the choice of the feature selection method is very important, however, the overall results are still lower when compared to the less difficult scenarios.

Table 5: Accuracy for Random Delay - HW model

Pearson correlation

Classifier	10	25	50	75	100
TA	11.4	17.4	5.5	0.8	6.5
TA (pooled)	4.4	5.3	6.8	7.5	8.1
RF	25.3	26.1	26	26.2	26

SOST

Classifier	10	25	50	75	100
TA	9.4	15.7	5.5	5.7	4.5
TA (pooled)	5.9	6.2	8.5	9.3	9.8
RF	25.5	26	26.6	26.4	26.4

SOSD

Classifier	10	25	50	75	100
TA	10.1	17	0.5	8.2	14.6
TA (pooled)	6.6	7.9	8.8	9.5	9.9
RF	25.2	25.8	26.7	26.3	26.2

χ^2

Classifier	10	25	50	75	100
TA	9.6	16.6	2.5	8.5	10.6
TA (pooled)	5.9	6.9	8.3	9.1	9.5
RF	25	25.4	25.9	26	26.1

Linear SVM wrapper

Classifier	10	25	50	75	100
TA	7.2	15.5	1.7	1.9	7.6
TA (pooled)	4.7	5.9	7.0	7.5	8.3
RF	25.6	25.7	26.1	26.1	26.1

L1 regularization

Classifier	10	25	50	75	100
TA	10.7	15.9	1.3	7.3	6.3
TA (pooled)	6.3	6.5	7.8	8.7	9
RF	24.9	25.6	25.9	25.7	26.2

Stability selection

Classifier	10	25	50	75	100
TA	13.7	16.8	1.5	7.5	2.7
TA (pooled)	8	6.8	8.8	9.6	10
RF	24.8	25.5	25.8	25.8	26.1

4.2 General Observations

After presenting the results for different considered scenarios, we now concentrate on more general findings pertaining to feature selection in SCA.

1. Different feature selection techniques can result in a radically different classifier behavior, which is especially evident from the presented GE results. Consequently, one should devote the same amount of attention to feature selection as to classification. This is in line with the "No Free Lunch" theorem, which states that there is no single best algorithm for all problems [?].

Table 6: Accuracy for Random Delay - intermediate value model

Pearson correlation

Classifier	10	25	50	75	100
TA	0.4	0.4	0.4	0.4	0.4
TA (pooled)	0.4	0.4	0.4	0.4	0.4
RF	0.4	0.4	0.3	0.5	0.4

SOST

Classifier	10	25	50	75	100
TA	0.4	0.4	0.4	0.4	0.3
TA (pooled)	0.4	0.4	0.4	0.4	0.4
RF	0.4	0.5	0.4	0.4	0.3

SOSD

Classifier	10	25	50	75	100
TA	0.3	0.4	0.4	0.4	0.5
TA (pooled)	0.3	0.4	0.4	0.4	0.5
RF	0.4	0.4	0.4	0.4	0.5

χ^2

Classifier	10	25	50	75	100
TA	0.4	0.4	0.4	0.4	0.4
TA (pooled)	0.4	0.4	0.4	0.4	0.5
RF	0.4	0.4	0.4	0.4	0.4

Linear SVM wrapper

Classifier	10	25	50	75	100
TA	0.3	0.3	0.4	0.4	0.4
TA (pooled)	0.4	0.4	0.4	0.4	0.4
RF	0.4	0.4	0.4	0.4	0.4

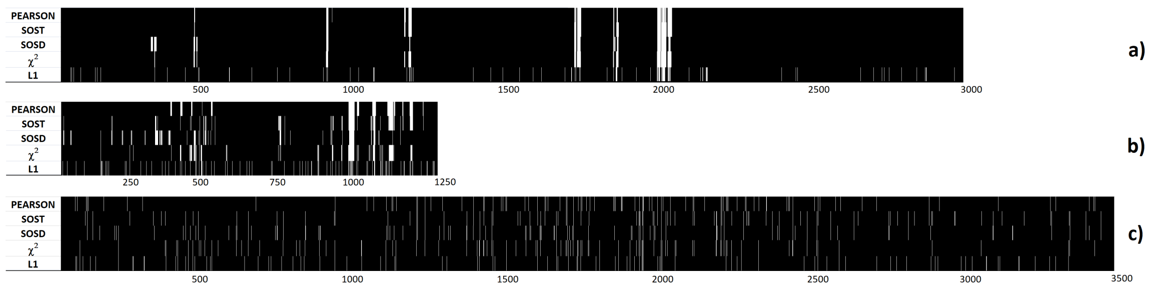
L1 regularization

Classifier	10	25	50	75	100
TA	0.3	0.4	0.4	0.4	0.5
TA (pooled)	0.4	0.4	0.5	0.4	0.4
RF	0.4	0.3	0.4	0.5	0.4

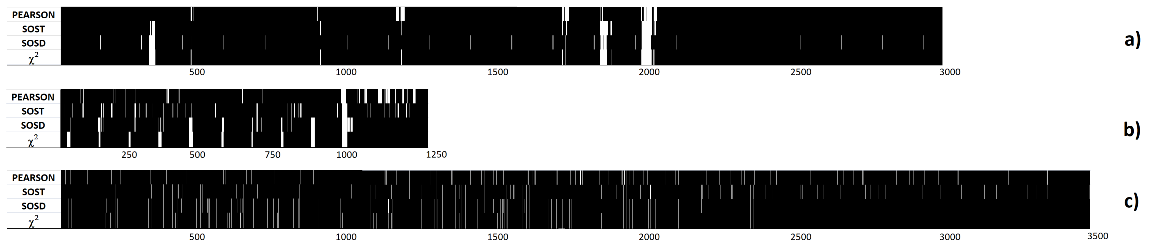
Stability selection

Classifier	10	25	50	75	100
TA	0.4	0.4	0.4	0.5	0.4
TA (pooled)	0.4	0.4	0.4	0.4	0.4
RF	0.4	0.4	0.4	0.4	0.4

2. It is important to conduct feature selection individually for each model considered. For instance, we show that, if feature selection is done for the Hamming weight model, then, in general, one should not use the same features when considering the intermediate value model.
3. We confirm that having a higher number of features than the number of traces per class results in template attack becoming unstable, as also indicated by previous works (e.g., [**?**]), which is an observation that does not hold for machine learning techniques. In particular, for TA, we observe the effect of instability in the estimation of the covariance matrix when using the intermediate value model (= 256 classes) and if the number of features is > 10. The pooled version tries to circumvent instabilities by reducing the number of covariance matrices to be



(a) HW model



(b) Intermediate value model

Fig. 7: 100 selected features for a) DPAcontest v4, b) AES_HD, c) Random Delay

Table 7: PCA classification results.

	Accuracy, % (best classifier)				
Dataset	10	25	50	75	100
DPA v4, HW	26.4(RF)	38.1(RF)	93.8(TA$_p$)	97.6(TA$_p$)	98.3(TA$_p$)
DPA v4, int.	0.7(RF)	4.0(RF)	40.1(TA$_p$)	61.3(TA$_p$)	74.7(TA$_p$)
AES_HD, HW	25.2(RF)	25.9(RF)	26.7(RF)	26.7(RF)	26.8(RF)
AES_HD, int.	0.4(all)	0.4(all)	0.5(RF)	0.4(all)	0.4(all)
Rand. D., HW	24.3(RF)	25.1(RF)	25.5(RF)	25.7(RF)	26.2(RF)
Rand. D., int.	0.4(all)	0.4(all)	0.4(all)	0.4(all)	0.4(all)

estimated to a single one, which may include information loss. We show an alternative to increasing the number of traces or using only one pooled covariance matrix as suggested by [**?**]. More precisely, an alternative approach is to use one of the Wrapper or Hybrid techniques, which may result in improved performance of template attack.

4. We show that even a very small subset of features, if selected properly, can result in better performance than a superset obtained with other selection techniques (that may contain redundant or incorrect features).
5. We show that it is possible to conduct feature selection even in the presence of a random delay countermeasure. There, although some important features are moved in the time domain, the

amount of information obtained from traces is sufficient for a reliable feature selection, resulting in efficient attacks.

6. Datasets with large amounts of noise are difficult for classification as well as for feature selection. This is expected, especially for Wrapper and Hybrid methods, since there we use ML classifiers for feature selection.
7. When considering datasets with a large amount of noise or countermeasures, it is possible to conduct a successful attack even in extremely constrained scenarios where we have only 10 features, if they are well-chosen.

5 Conclusions and Perspectives

In this paper, we addressed the following questions: how to select the most informative features from raw data and what is the influence of the feature selection step in the performance of the classification algorithm? Our results show that the proper selection of features has a tremendous impact on the final classification results. We notice that often with a small number of features when using a proper selection technique, one can achieve approximately the same results as some other method using a much larger number of features.

We demonstrated how state-of-the-art techniques for feature selection from the ML area behave for profiling in side-channel analysis. We observe that much more powerful techniques than those currently used in the SCA community are applicable and achieve higher accuracies. Unfortunately, our results do not reveal a single method as the best performing one. Still, this is to be expected, since the "No Free Lunch" theorem also holds for feature selection. We emphasize that the Pearson correlation is rarely the most successful technique for feature subset selection, which is a common choice for feature selection in the SCA community. When considering guessing entropy results, we emphasize Linear SVM Hybrid method and L1 regularization Wrapper that performed consistently well for all datasets. This is especially interesting, since L1 regularization did not perform the best when considering accuracy and the Random Delay and AES_HD datasets. Naturally, feature selection in the case of "easy" scenarios (e.g., DPAcontest v4) is not the most important and effective task, but in scenarios with high noise and even countermeasures (Random Delay dataset), our techniques may bring significant improvements.

The obtained accuracy results in most cases favor ML-based feature selection techniques when compared to PCA-based feature extraction. At the same time, when considering guessing entropy, we see that PCA is never the best technique. Future work may compare ML-based feature selection with other dimensionality reduction methods, e.g. SNR metrics [**?**], in detail and determine the superiority in specific contexts.

Acknowledgement This work was supported in part by the Technology Foundation STW (Project 13499 TYPHOON - VIDI) from the Dutch government.