Improving CEMA using Correlation Optimization

Pieter Robyns[1], Peter Quax[2] and Wim Lamotte[1]

[1] Hasselt University - tUL, Expertise centre for Digital Media
Martelarenlaan 42, 3500 Hasselt, Belgium
{pieter.robyns,wim.lamotte}@uhasselt.be
[2] Hasselt University - tUL - Flanders Make, Expertise centre for Digital Media
Martelarenlaan 42, 3500 Hasselt, Belgium
peter.quax@uhasselt.be

Abstract. Sensitive cryptographic information, e.g. AES secret keys, can be extracted from the electromagnetic (EM) leakages unintentionally emitted by a device using techniques such as Correlation Electromagnetic Analysis (CEMA). In this paper, we introduce Correlation Optimization (CO), a novel approach that improves CEMA attacks by formulating the selection of useful EM leakage samples in a trace as a machine learning optimization problem. To this end, we propose the correlation loss function, which aims to maximize the Pearson correlation between a set of EM traces and the true AES key during training. We show that CO works with high-dimensional and noisy traces, regardless of time-domain trace alignment and without requiring prior knowledge of the power consumption characteristics of the cryptographic hardware. We evaluate our approach using the ASCAD benchmark dataset and a custom dataset of EM leakages from an Arduino Duemilanove, captured with a USRP B200 SDR. Our results indicate that the masked AES implementation used in all three ASCAD datasets can be broken with a shallow Multilayer Perceptron model, whilst requiring only 1,000 test traces on average. A similar methodology was employed to break the unprotected AES implementation from our custom dataset, using 22,000 unaligned and unfiltered test traces.

Keywords: Correlation Optimization · Software Defined Radio · Correlation Electromagnetic Analysis · correlation loss · machine learning

1 Introduction

In present-day communication systems, the confidentiality and integrity of information is primarily ensured through the use of cryptographic algorithms. At their core, these algorithms rely on secret pieces of information, i.e. keys, known only to the communicating parties in order to obtain a *computational advantage* over the adversary. That is, given that the cryptographic algorithm is not theoretically broken, it should be computationally infeasible for an adversary to modify or eavesdrop on the communication without having access to the secret information. However, as demonstrated in numerous previous works (see [ZF05] and the references therein), an adversary can infer secret information by statistically analyzing physical properties of the hardware implementation during the execution of a cipher. These physical properties, named side channels, can thus unintentionally "leak" information to an adversary.

In the literature, various types of side channels have been identified that can be used to exfiltrate secret information, including power consumption [KJJ99], temperature [BKMN09, HS13], acoustic [BDG+10, GST14, ST04], and electromagnetic (EM) side channels [GMO01, GHT05, GW08, GPT15, Mon13, MBTO13, PSB+18, QS01, VP09]. Out of those, the EM side channel is particularly interesting from the perspective of an

IACR Transactions on Cryptographic Hardware and Embedded Systems ISSN 2569-2925, Vol. 2019, No. 1, pp. 1–24
DOI:10.13154/tches.v2019.i1.1-24

adversary, for a number of reasons. First, EM waves can be captured without requiring physical contact with the hardware [Mon13, OC16]. This is in contrast to power consumption analysis, where the device under attack must typically be modified to obtain accurate measurements [Tiu05]. Second, EM leakage can originate from various components of a circuit due to coupling effects and circuit geometry [AARR03, MRG+10, Mon13] and is therefore hard to mitigate completely. Third, EM waves can potentially travel long distances, depending on the power and wavelength of the leakage signal. For example, Vuagnoux and Pasini demonstrated that EM emanations of PS/2 keyboards can be captured at a distance of up to 20 meters, even through walls [VP09]. Another example can be found in the work of Guri et al., where the invocation of specific memory-related instructions was used to transfer data via the resulting EM leakage, over a distance of up to 30 meters [GKH+15]. For these reasons, we will exclusively focus on the EM side channel in this paper, although the concepts described could be applied to other side channels as well.

A known methodology to perform an EM-based side-channel attack on a device is Correlation Electromagnetic Analysis (CEMA) [DLM+09], which is based on the Correlation Power Analysis (CPA) attack for power side channels introduced by Brier et al. in [BCO04]. In a CEMA attack, the adversary assumes that the power consumption of the hardware is related to the Hamming distance or Hamming weight leakage model. Since consuming power produces electromagnetic radiation [KJJ99], the resulting EM emissions can be captured by an adversary and correlated with the leakage model in order to determine the secret information being processed.

In *template or profiling attacks* on the other hand, the adversary makes no preliminary assumption about the leakage model [OC16]. Instead, the adversary is assumed to be in possession of a "training" device identical to the device being attacked [CRR02]. The attack is then performed in a two-phase process. In the training phase, the adversary estimates the probability distribution of the secret information in function of the measured EM leakage of the training device. Then, in the attack phase, the EM leakage of the device under attack is matched with the most probable template from the training phase [CRR02, PSB+18].

Recently, several works have indicated that profiling attacks can be interpreted as classification problems in the domains of Machine Learning (ML) and Deep Learning (DL) [CDP17, HGDM+11, LPMS17, MPP16, PSH+18]. In this paper, we introduce a novel profiling attack that, unlike these previous approaches, does not rely on classification for determining the secret key. More specifically, we do not directly *classify* individual EM traces into secret-key, intermediate, or Hamming values, but rather learn an *encoding*[1] of the EM traces that maximizes the Pearson correlation with the correct secret key. To this end, we introduce the "correlation loss function", which allows us to optimize the encodings for use in a CEMA attack using conventional ML optimization algorithms such as gradient descent. Furthermore, we show that our approach kan be applied in the frequency domain, which removes the requirement of aligning the EM traces [GHT05, GW08, Mon13, Tiu05]. We evaluate our methodology both on the ASCAD dataset [PSB+18] as well as a custom dataset comprised of Advanced Encryption Standard (AES) operations performed by an ATmega 328. The EM traces of the custom dataset were recorded using a Software Defined Radio (SDR) and commodity EM probe, which shows that our approach can be used to perform a CEMA even with low-cost EM measurement hardware.

The structure of this paper is as follows: in Section 2, we will first discuss a number of concepts that are necessary for understanding the remainder of the paper. Section 3 then describes our Correlation Optimization technique, which is the main contribution of this paper. In this section, the technique will also be discussed and evaluated on both the ASCAD dataset, as well as the custom dataset. Related works are described in Section 4,

[1]Encodings are widely used for applications such as face verification and face recognition [SKP15]. Some works in the machine learning literature refer to encodings as "embeddings".

followed by conclusions and directions for future work in Sections 5 and 6.

2 Background

2.1 Notation and terminology

In both the Side-Channel Analysis (SCA) and ML literature, the notational conventions and terminology vary depending on the authors of the work. Furthermore, some of the notations used in these domains overlap. As a concrete example, the output of the Hamming Weight (HW) power consumption model in the work of Brier et al. is denoted as W [BCO04], which is also a common notation for the weight matrix in the ML domain. In the interest of avoiding ambiguities, we will now specify the notations and terminology used in this paper.

We define a *dataset* as a collection of *traces*, where a trace is equivalent to a capture or measurement of the EM signal generated during one execution of an algorithm. Each trace consists of a number of *samples* of the instantaneous amplitude of the EM signal. A *training example* is a single trace that is used as the input to a ML model during its training phase. In this context, the samples of a trace (or a transformation thereof) are the *features* or *inputs* to the model. The *labels* or *classes* are then the outputs of the model. In this paper, we will often refer to the outputs of the model for a given input trace as the *encodings* of that trace.

Formally, we denote a training example with n_x features as a tensor $x = \{x_1, x_2, \dots x_{n_x}\}, x \in \mathbb{R}^{n_x}$. All n_m training examples can be stacked together in a matrix X as follows:

$$X = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \dots & x_{n_x}^{(1)} \\ x_1^{(2)} & \ddots & \ddots & x_{n_x}^{(2)} \\ \vdots & \ddots & \ddots & \vdots \\ x_1^{(n_m)} & x_2^{(n_m)} & \dots & x_{n_x}^{(n_m)} \end{bmatrix}$$

Note that individual training examples are referenced using a superscript index between parentheses, e.g. $x^{(4)}$ represents the 4th training example. A summary of all variables and notations used in this paper can be found in Table 1.

2.2 Advanced Encryption Standard

The AES cipher is widely used for providing confidentiality and integrity in protocols such as Wi-Fi (802.11) [IEE12], Bluetooth [Blu16], TLS [DR08], and many others. The cipher has been extensively studied in previous works. For a detailed overview of the inner workings of AES, we refer the reader to the design document of AES [DR13]. This paper focuses exclusively on side-channel attacks against the first round of AES. In this round, the `RoundKey` is initialized to the secret key $k = \{k_1, k_2, \dots, k_{16}\}$ [DR13]. We will henceforth refer to the elements of k as "key bytes". Given a plaintext $p = \{p_1, p_2, \dots p_{16}\}$, the intermediate value v after the `SubBytes` operation is equal to:

$$v = \text{SBox}(p \oplus k) \tag{1}$$

Because the secret key k is processed directly during the first round of the cipher, performing a side-channel attack is trivial. Several countermeasures have been developed to increase the difficulty of successfully performing an attack. A first countermeasure is called "masking", which attempts to change the power consumption characteristics of a device by randomizing the processed intermediate values [Mon13]. An example of such an approach is the "table recomputation" method [PR07].

Table 1: Overview of the notations used in this paper.

Symbol	Description	Symbol	Description
α	Scaling factor.	n_b	Number of bits.
$A^{[l]}$	Activations matrix of layer l.	n_l	Number of layers.
b	Bias tensor.	n_m	Number of training examples.
D	Current state.	n_x	Number of features.
d	Key byte guess score tensor.	ϕ	Encoding function.
ϵ	Value of 10^{-15}.	p	Plaintext tensor.
η	Noise tensor.	ρ	Pearson correlation coefficient.
g	Activation function.	R	Reference state.
H	Hypothesis matrix.	t	Time unit.
h	Tensor of model hypothesis values.	v	Intermediate value.
h_t	Power consumption model at time t.	$W^{[l]}$	Weight matrix of layer l.
$J(\hat{y}, y)$	Cost function.	X	Matrix of training examples.
k	True key tensor.	$x_j^{(i)}$	Feature j of training example i.
$\hat{k}$	Key guess tensor.	$\hat{Y}$	Matrix of model output values.
k_s	Byte s of k.	Y	Matrix of true values.
$\mathcal{L}(\hat{y}, y)$	Loss function.	$\hat{y}$	Model output values (encodings).
l	Layer index.	y	True model output.

A second commonly implemented countermeasure is “hiding”, which, as the name implies attempts to hide the hardware power consumption characteristics from an adversary. This can be done in a number of ways, for example by introducing jitter to the clock of the device, introducing dummy instructions, adding random interrupts, etc. [Mon13].

2.3 Correlation Electromagnetic Analysis of AES

The classic CPA technique, as first described in the work of Brier et al. [BCO04], utilizes the Hamming Distance (HD) to model the data leakage through a power side channel. More specifically, the number of bits switching from an n_b-bit reference state R to another state D at a given time t is assumed to be linearly related to the power consumption of the hardware [BPT11, Mon13, TOT+14, Tiu05]. Formally, we define this power consumption model h as follows: [BCO04]

$$h_t = \alpha HW(D_t \oplus R_t) + \eta_t \tag{2}$$

In the above equation, HW is the Hamming weight function, α is a scaling factor and η_t is a noise term at time t. Generally, α and η are unknown, and the adversary's goal is to recover D_t and R_t from h_t. If the underlying hardware implements pre-charged logic or if certain transitions are prohibited due to separated busses for data and addresses, the reference state R_t can be systematically equal to 0 [BCO04, OC15]. In this case, the HD model generalises to the HW model.

When applied to AES, a CEMA using the HW or HD model often targets the output of the `AddRoundKey` and `SubBytes` operations in the first round of the cipher, as the secret key is processed directly in this round [GW08, MPP16, MDM16, Mon13]. If we plug the intermediate value from Equation 1 into the HW leakage model, the EM leakage in function of the key and plaintext becomes:

$$h_t = \alpha HW(\text{SBox}(p \oplus k)) + \eta_t \tag{3}$$

The adversary can now determine the value of h_t by supplying a known plaintext to the device under attack and analyzing traces of EM emanations captured during the first

AES round. Recall that we define a trace as an AM-demodulated signal (i.e. a time series of signal magnitudes) captured using an oscilloscope or signal analyzer, that characterizes the power consumption of the device over time.

In order to determine the unknown k given h_t, the adversary can construct a hypothesis power consumption matrix $H_{ij} = \mathrm{SBox}(p_i \oplus j)$ for each key byte index $i \in \{1, 2, \ldots, 16\}$ and each possible key value $j \in \{0, 1, \ldots, 255\}$. This matrix essentially gives all power consumption values h_t for each possible key, assuming $\alpha = 1$ and $\eta = 0$. The adversary then calculates the Pearson correlation coefficients between the observed values for h_t from the captured EM traces and the hypothesized values from H for each key guess [BCO04, MBTO13]. This process is repeated for each relevant time unit t, resulting in a three-dimensional correlation matrix:

$$\rho_{tij}(h_t, H_{ij}) = \frac{cov(h_t, H_{ij})}{\sigma_{h_t}\sigma_{H_{ij}}} \tag{4}$$

Finally, the best key guess $\hat{k}$ is determined by selecting the key value with the highest absolute value[2] of the correlation:

$$\hat{k}_i = \arg\max_{t,j}(|\rho_{tij}|) \tag{5}$$

2.4 Machine Learning and Deep Learning attacks on AES

In ML and DL side-channel attacks, the objective of finding the secret key k given a collection of traces is formulated as a *supervised classification problem*. Analogous to the classic Template Attacks (TAs), solving this problem involves two phases: a training phase and testing phase.

In the training phase, a “training set” of EM traces is first collected from the reference device. Here, each trace or “training example” is labeled with a corresponding “class label”. The class label is what the algorithm will be trained to predict and can be chosen in several ways. One possibility is to consider each possible key byte value as a separate class, which yields 256 class labels. Another possibility could be to consider the HW of each key byte value, resulting in only 9 class labels (all possible Hamming weight values of a single byte). The ML algorithm is subsequently trained, which means that a model is parameterised such that a certain predetermined loss function (e.g. the cross-entropy loss) is minimized.

After training is complete, a “test set” of EM traces is collected from the targeted device during the testing phase. Since the key bytes are unknown in this case, only the EM traces are available. The ML algorithm will output the probability of each EM trace belonging to a certain class, based on the parameters learned by the model during the training phase. The accuracy of this prediction largely depends on the quality of the training data, but also on the chosen type of ML model, hyperparameters, optimizer, and loss function. In this paper, we will primarily focus on simple ML models such as Multi-Layer Perceptrons (MLPs). We will show that even such simple models outperform the state-of-the-art models from previous works when using our approach, which will be detailed in Section 3.

2.4.1 Multi-Layer Perceptrons

The architecture of MLPs consists of layers of interconnected processing units, called perceptrons or neurons. Such layers are also called “fully connected” layers. Each neuron in a layer has an associated set of input values, weights, a bias term, and activation function. The output of a single neuron is determined by:

[2]The absolute value of the Pearson correlation is considered, since it does not matter whether the correlation between the leakage and a certain key byte is positive or negative.

$$a_1 = g(xw + b_1) \tag{6}$$

In this equation, x is a tensor of input features (e.g. trace samples), w is the tensor of weights associated with each input feature, b_1 is the bias term, and $g(x)$ is the activation function. Neurons can be stacked together to form layers, such that the output becomes:

$$A^{[l]} = g(A^{[l-1]}W^{[l]} + b^{[l]}) \tag{7}$$

$$A^{[0]} = X \tag{8}$$

$$A^{[n_l]} = \hat{Y} \tag{9}$$

where $l \in \{0, 1, \ldots n_l\}$ indicates the layer index[3]. Note that we define the first activations $A^{[0]}$ as simply the input features X, and the final activations as the output predictions of the neural network, denoted as $\hat{Y}$.

2.4.2 Convolutional Neural Networks

In Convolutional Neural Networks (CNNs), two additional types of layers are introduced: convolutional layers and pooling layers [MPP16]. Here, the convolutional layer is a layer that performs a series of convolution operations on its inputs. The weights of the convolution kernels or "filters" are learned by the optimizer algorithm. These filters act as feature detectors for the next layers that can be useful across the entire input [LB95]. At the same time, less weights need to be trained since the size of the filters is typically smaller than the number of input features itself.

Convolutional layers are usually followed by pooling layers, which reduce the resolution of the inputs by performing a local averaging (average pooling) or local maximum (max pooling) and a subsampling operation [LB95]. This operation makes CNNs more robust to small input shifts and deformations compared to MLPs [CDP17, LB95]. Figure 1 shows an example of a CNN with multiple convolutional and pooling layers. Deeper layers are capable of detecting more complex features of the input due to their larger receptive field. At the end of the CNN, a series of fully connected layers follows, which uses the features detected by the convolution filters to determine the output of the model.

2.5 The ASCAD dataset

The ASCAD dataset was introduced by Prouff et al. in [PSB+18] for the purpose of providing a benchmark to evaluate ML and DL techniques in context of side-channel attacks. The dataset consists of three separate HDF5 files: `ASCAD.h5`, `ASCAD_desync50.h5`, and `ASCAD_desync100.h5`. Each file contains 60,000 EM traces (50,000 training / cross-validation traces and 10,000 test traces) captured with a sensor attached to an oscilloscope sampling at 2 GS/s. The traces contain 700 samples of the EM radiation emitted by an ATMega8515 device during the execution of the first round of a software AES implementation. The AES implementation is secured against first-order side-channel attacks with the masking technique (see Section 2.2). Traces in the `ASCAD.h5` file have been time-aligned in a preprocessing step, whereas the traces in `ASCAD_desync50.h5` and `ASCAD_desync100.h5` have been shifted with a maximum jitter window of respectively 50 and 100 samples [PSB+18]. Besides EM measurements, the ASCAD authors have provided the source code of the models used in their work. In Section 3.6.1, we will compare these models to our Correlation Optimization (CO) technique.

[3]The index starts from 0 because by convention, the input layer is not counted as an actual layer.

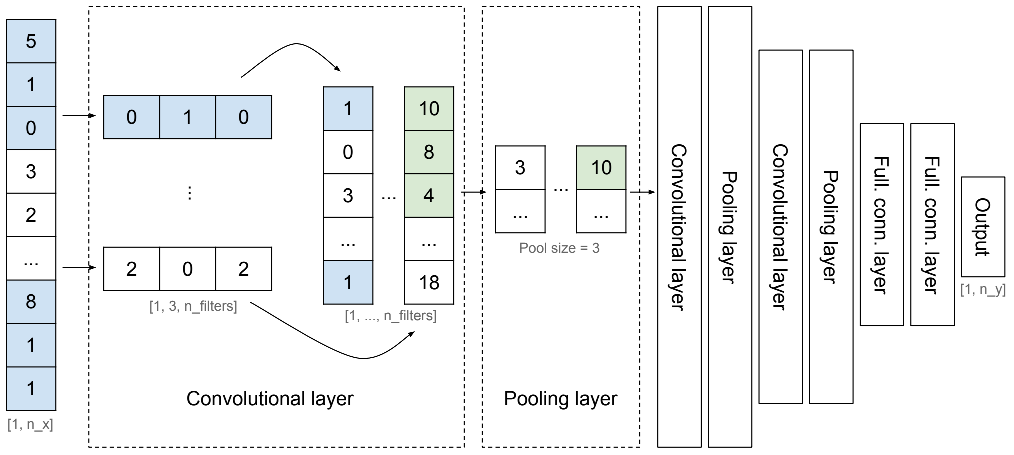


Figure 1: Example of a simple 1D convolutional neural network architecture.

3 Correlation Optimization

When performing a CEMA attack as described in Section 2.3, recall that the adversary constructs the matrix ρ_{tij} which gives the correlation coefficients of all key byte hypotheses for each time unit in the trace. Then, the hypothesis with the highest correlation to the power consumption traces is selected as the most likely true key byte. Note that this correlation is determined by only *one time unit* for each trace, i.e. the samples at other time units are discarded. However, some of these discarded samples are correlated to the correct hypothesis, albeit to a lesser degree.

This observation raises the question of how information from multiple samples can be combined in order to improve the efficiency of CEMA. Since the leakage itself depends on the input plaintext, key, and complex electromagnetic interactions governed by Maxwell's equations in the underlying hardware, the samples that contain useful information may not occur at the same time units over multiple traces. Measurements are also noisy (due to interference and fading effects) and highly dimensional (in terms of samples per trace), which complicates the issue further. Although classic TAs consider multiple time units as well [CRR02], they are not suited for high-dimensional data [CDP17, LPMS17].

We now introduce our approach, called "Correlation Optimization", which aims to address the aforementioned issues by exploiting the information leakage from multiple samples. To this end, we consider the selection of "good" samples as a ML optimization problem. That is, given an input of trace samples $x^{(i)} = \{x_1^{(i)} \dots x_{n_x}^{(i)}\} \in X$ with $i \in \{1, 2, \dots n_m\}$, we would like to determine which *encoding* of samples $\hat{y}^{(i)} \in \hat{Y}$ can be obtained such that $\rho(\hat{Y}, Y)$ is maximal. Here, the term encoding refers to an arbitrary function ϕ of the input features, i.e.:

$$\hat{y}^{(i)} = \phi(x^{(i)}) \tag{10}$$

Observe that the above optimization problem aligns well with the problems from domains such as face recognition and face verification, where the ML model *matches encodings* of faces rather than predicting a class probability for each face[4] [SKP15]. Since in these domains ML models achieve state-of-the-art performance, we will apply similar techniques in our approach.

[4]To understand why, note that when given a database of millions of faces, it would be infeasible to train a model with a one-hot encoded class label assigned to each of the faces.

3.1 The correlation loss function

Recall that we would like to find an encoding of input samples $\hat{y} = \{\hat{y}^{(1)}, \ldots, \hat{y}^{(n_m)}\}$ such that $\rho(\hat{Y}, Y)$ is maximal. To this end, a loss function must be defined that can be evaluated by an optimizer algorithm in order to train the parameters of our ML model. Ideally, this loss function should be large when there is no linear correlation between $\hat{Y}$ and Y, and zero when the correlation is maximal. For a tensor of key bytes $y_k = \{y_k^{(1)}, \ldots, y_k^{(n_m)}\}$ at index $k \in \{1, 2, \ldots n_k\}$ and the corresponding tensor of encodings $\hat{y}_k = \{\hat{y}_k^{(1)}, \ldots, \hat{y}_k^{(n_m)}\}$, we define the loss function as:

$$\mathcal{L}(\hat{y}_k, y_k) = 1 - \frac{cov(\hat{y}_k, y_k)}{\sigma_{\hat{y}_k}\sigma_{y_k} + \epsilon} \tag{11}$$

$$= \frac{\sum_{i=1}^{n_m}[(\hat{y}_k^{(i)} - \bar{\hat{y}}_k)(y_k^{(i)} - \bar{y}_k)]}{\sqrt{\sum_{i=1}^{n_m}(\hat{y}_k^{(i)} - \bar{\hat{y}}_k)^2}\sqrt{\sum_{i=1}^{n_m}(y_k^{(i)} - \bar{y}_k)^2} + \epsilon} \tag{12}$$

Here, ϵ is a small value (e.g. 10^{-15}) introduced to prevent division by zero. If we assume that $\hat{y}_k$ and y_k are mean-normalized, Equation 12 can be converted to a more convenient vector form:

$$\mathcal{L}(\hat{y}_k, y_k) = 1 - \frac{\hat{y}_k \cdot y_k}{\|\hat{y}_k\| \cdot \|y_k\| + \epsilon} \tag{13}$$

In case we want to simultaneously train the ML model for all values of k, a cost function[5] can be defined as follows:

$$J(\hat{y}, y) = \sum_{k=1}^{n_k} \mathcal{L}(\hat{y}_k, y_k) \tag{14}$$

Observe that the maximum value of the cost function J is 32 in the worst case (when all correlations are -1), and 0 in the best case (when all correlations are 1).

3.2 Evaluation methodology

For the evaluation of our CO technique, we will use the same methodology as Prouff et al., so that an objective comparison can be made. As such, our ML models will be trained to attack the third key byte of the masked AES implementation from the ASCAD database (see Section 2.5 for a description of this dataset). The implementation of these models was written in Python, using the library "Keras" [C+15] with a Tensorflow backend [AAB+15]. All source code, scripts, and data used to generate the figures is available on Github at https://github.com/rpp0/correlation-optimization-paper.

3.2.1 t-fold cross-validation

Unless specified otherwise, each ML model was evaluated with a 10-fold cross validation, using a train and test split of respectively 45,000 and 5,000 traces. Hence, we first train a model from scratch with 45,000 random traces, and evaluate the performance of this newly trained model on 5,000 different (unseen) traces. This process is repeated 10 times, and finally the performance metrics are averaged to obtain a conlusion.

[5] In some ML papers, a loss function defines the cost for a single training example, whereas the cost function defines the total cost. In this paper, we define the cost function as the total cost for all bytes of the AES key instead of all training examples.

3.2.2 Performance metrics

In each of our experiments, we use two metrics to assess the performance of our models: "rank" and "confidence". Here, we define the "rank" as the index of the correct key in a tensor d that contains the scores assigned to a key byte k_s by the model, sorted such that $d_i \geq d_j \ \forall \ i, j \in \{1, 2, \ldots, 256\} \mid i < j$. For example, the score tensor can contain key byte probabilities or correlations. More formally, we define the rank as:

$$\text{rank}(d) = \{i - 1 | d_i = \text{score}(k_s)\} \tag{15}$$

Note that the lowest possible rank is 0. If we employ an optimal guessing strategy by iteratively guessing the key byte with the next highest score in d, the rank plus one corresponds to the number of guesses required to find the correct key. The *expected* number of key guesses is called the Guessing Entropy (GE), and is commonly used to evaluate SCA methods [KB07, OC15, Riv08, SMY09]:

$$GE = \sum_{i=1}^{K} iP(\text{rank}(d) = i - 1) \tag{16}$$

We obtain the expected value of the number of key guesses by averaging the rank over the 10 folds of the ML model, which we will denote as the "mean rank" in the coming figures. The "confidence" is then defined as the score difference between rank 0 and rank 1 or equivalently, as the distance between the maximum score in d and the next highest score in d [MBTO13]:

$$\text{confidence}(d) = d_1 - d_2 \tag{17}$$

Analogous to the rank, we average the confidence over all 10 folds to obtain the "mean confidence". Although this metric is less frequently used in related works, we believe it is useful because it gives an insight into how well the model can distinguish a singular key byte guess.

3.2.3 Input and label preprocessing

The inputs and labels to the ML models are preprocessed during the training phase. First, all traces are split into mini-batches of 512 traces before being fed to the network. This removes the requirement of having to load the entire dataset in memory, at the cost of decreased performance since multiple iterations are now needed to process the entire training set. Note that if the mini-batch size is too small, its correlation loss might not be representative for the entire training set, and it may fail to converge as a result. We emperically determined 512 traces to be a good value for the mini-batch size for the ASCAD dataset.

Second, the true labels y are preprocessed such that $y_s^{(i)} = HW(SBox(p_s^{(i)} \oplus k_s^{(i)}))$, where s is the key byte index and i is the training example index. It is important to note that we *do not* supply the variable AES masking values during the training phase: an encoding that correlates the EM radiation directly with the corresponding key byte will automatically be learned by the model. Further, as we will discuss in Section 3.4.4, assuming the Hamming Weight power consumption model is not necessary, but will reduce the required training time and complexity of the model architecture. Finally, the training example input features $x^{(i)}$ are simply equivalent to the raw samples of trace i, unless specified otherwise.

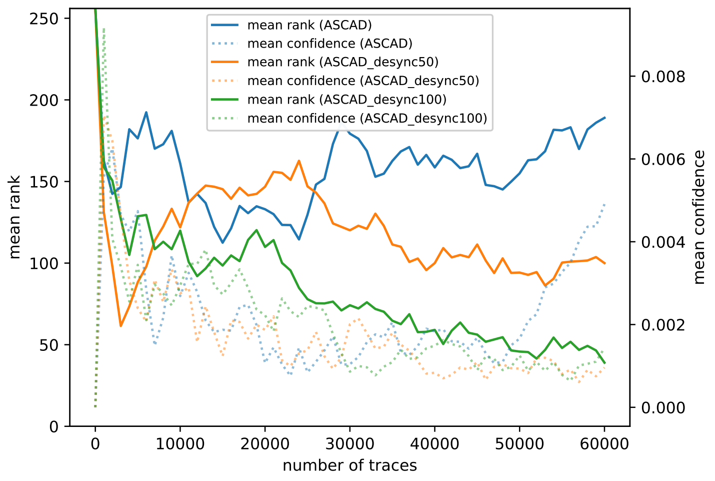


Figure 2: Mean rank based on a 10-fold cross-validation of the time-domain CO identity encoding function in relation to the number of validation traces used for a CEMA attack. The CEMA attack is unsuccessful even if all 60,000 traces are utilized, and the confidence in the guessed key bytes is low, ranging from 0.001 to 0.005.

3.3 Time-domain CO

Using the correlation loss function defined in Section 3.1, an encoding function ϕ that maximizes Equation 4 can be learned by the ML model. In this section, we will discuss the performance of the encoding function if we use time domain samples as its input features. To this end, we first train the ML model for 100 epochs on the three ASCAD datasets. Then, we generate the encodings for each of the traces using the trained model, and perform a CEMA attack on these encodings.

3.3.1 Identity function

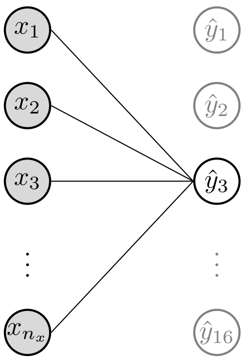
In order to establish a performance base line, we first consider the case where ϕ is the identity function, i.e. $\hat{y} = \{x_1, x_2, \ldots, x_m\}$. This is equivalent to performing a regular CEMA attack on the EM traces as described in Section 2.3. Since there are no weights or other parameters to learn, we use all 60,000 available traces for the evaluation of the model. The results of this evaluation are shown in Figure 2.

As expected, the standard CEMA attack is not successful in guessing the true key due to the masking countermeasure that was implemented (see Section 2.2). Although the mean rank does seem to decrease slightly for `ASCAD_desync50` and `ASCAD_desync100` when more traces are considered, note that the confidence is low for each dataset. This indicates there is no clear difference between the best key guess and the second best key guess, and that more traces would therefore be needed to obtain a reliable result.

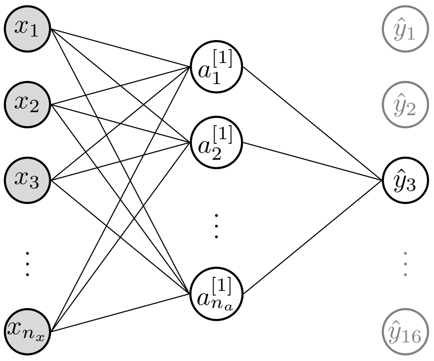
3.3.2 Single-layer MLP

When using a single-layer MLP architecture for CO, we essentially let the learning algorithm determine a linear combination of time-domain samples that, when passed through a nonlinear activation function, results in a maximal correlation with the correct key for the entire training set. The output of the MLP is:

$$\hat{Y} = g(XW + b) \tag{18}$$



(a)



(b)

Figure 3: The single-layer (a) and two-layer (b) MLP architectures used in our paper for CO. Only connections to the output for key byte 3 (y_3) are shown for clarity.

The architecture is visualized in Figure 3 (a). In practice, we add a batch normalization layer before the activation function to speed up the learning algorithm [IS15]. As the activation function, we chose the leaky ReLU activation in order to mitigate the occurrence of zero-gradients [MHN13].

Note that the CEMA attack will be performed on a *single* output encoding sample, determined by the parameters learned by ML model. The mean rank and the confidence after training on 45,000 traces for 100 epochs and evaluating on a validation set of 5,000 different traces is shown in Figure 4.

Clearly, the model has learned an improvement over the identity function: a mean rank of 1 is achieved after 4,960 traces even though we used only 5,000 traces for the CEMA attack instead of 60,000. It should be noted that a mean rank of 0 *can be* achieved if more traces were to be added to the validation set, though we only show the first 5,000 traces for a fair comparison with the other experiments and related works.

3.3.3 Two-layer MLP

Although single-layer MLPs are intuitive in the sense that they essentially learn a single weight for each sample in a trace, they do not allow for learning complex encoding functions. Ideally, we would like to learn dependencies between samples as well. To achieve this, a more complex architecture can be used, such as an MLP with a hidden layer as shown in Figure 3 (b). The output of the model then becomes:

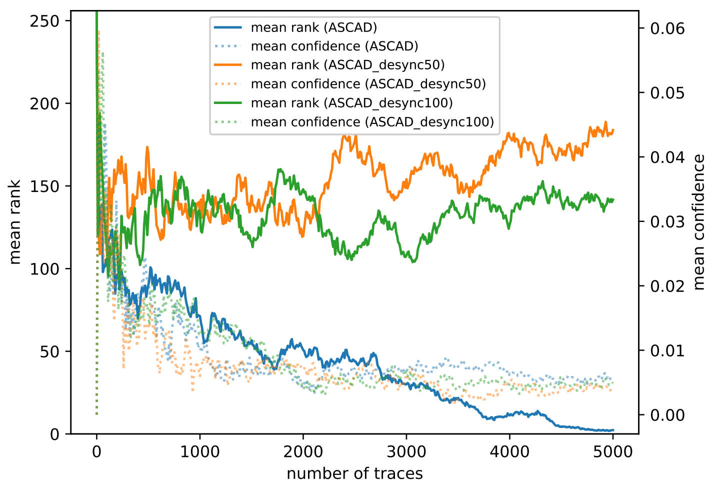


Figure 4: Mean rank based on a 10-fold cross-validation of the time-domain single-layer MLP CO model in relation to the number of validation traces used for a CEMA attack. Compared to the identity function, an improvement can be observed for the `ASCAD` dataset: a minimum mean rank of 1 is obtained after 4,960 traces. However, the mean confidence in the guessed key bytes is still low, ranging from 0.003 to 0.005. For the desynchronized datasets, no improvement is observed.

$$\hat{Y} = g(g(XW^{[0]} + b^{[0]})W^{[1]} + b^{[1]}) \tag{19}$$

Again, we add batch normalization layers to speed up training and perform a CEMA attack on the encodings of 5,000 traces after training on 45,000 traces. The result is shown in Figure 5. With this model, we obtain a very encouraging result for the `ASCAD` dataset: the model has learned to defeat the masked implementation of AES and requires only around 1,000 traces to achieve a mean rank of 0. Furthermore, the confidence after 5,000 traces is an order of magnitude higher compared to the single-layer MLP.

Unfortunately, for the `ASCAD_desync50` and `ASCAD_desync100` dataset, the model is not able to learn a meaningful encoding. This is to be expected, as MLPs are very sensitive to translations of the input features [LB95]. Introducing deliberate clock jitter to the hardware of a device running AES would therefore be an effective countermeasure against time-domain CO.

3.4 Frequency-domain CO

As evidenced in Section 3.3, shifting traces in time to cause misalignment can be an effective mitigation against CEMA attacks. Cagli et al. discuss several methods to mitigate the issue of misalignment in their work: increasing the number of side-channel acquisitions, applying realignment techniques, and using CNNs [CDP17]. However, another possibility is to consider traces in the frequency domain [GHT05, GW08, Mon13], as first proposed for Differential Electromagnetic Analysis (DEMA) attacks by Tiu in [Tiu05]. Inspired by this approach, we studied the effectiveness of CO in the frequency domain. To this end, we first preprocess each of the traces by applying a 700-point Fast Fourier Transform (FFT) and taking the magnitude of the result, discarding the phase information.

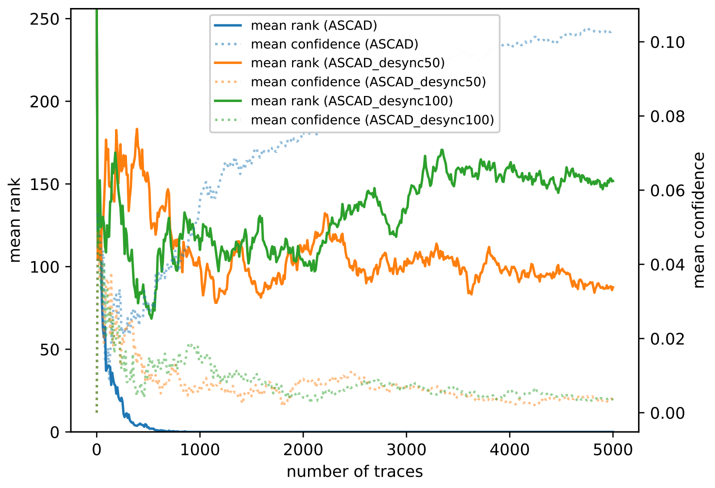


Figure 5: Mean rank based on a 10-fold cross-validation of the time-domain two-layer MLP CO model in relation to the number of validation traces used for a CEMA attack. The CEMA attack successfully finds the correct key for the `ASCAD` dataset after 800 traces with a mean confidence of 0.038, which rises further to 0.10 as more samples are included in the validation set. However, the attack fails on the desynchronized datasets `ASCAD_desync50` and `ASCAD_desync100`.

3.4.1 Identity function

Similarly to our approach for the time-domain CO, we first consider the case where ϕ is the identity function in order to establish a baseline for frequency-domain CO. Thus, we have $\hat{y} = abs(FFT(\{x_1, x_2, \ldots, x_m\}))$ as the output of the model for a single training example.

The result of performing a CEMA attack on all 60,000 traces is shown in Figure 6. Here, the mean rank is better compared to the time-domain CO identity function experiment: the mean rank for `ASCAD_desync100` reaches zero at 51,000 traces. This indicates that there is a single frequency component that, when analysed, allows us to defeat the masked AES implementation without CO. However, the confidence for each dataset is still low, and many traces are required for the CEMA attack to succeed.

3.4.2 Single-layer MLP

Next, we consider linear combinations of the FFT frequency bins passed through an activation function. We use the same encoding function ϕ as for the time-domain single-layer MLP model from Section 3.3.2. Figure 7 shows the mean rank and the confidence of a CEMA attack on the encodings of 5,000 traces, after training on 45,000 traces for 100 epochs.

The improvement over the identity function is similar to what we observed for time-domain CO: only 4,840 traces are required to obtain a mean rank of 0, with a mean confidence of 0.008.

3.4.3 Two-layer MLP

Using an MLP with a hidden layer allows the model to learn more complex relationships between frequency bins of the FFT. Again, we use the same encoding function ϕ as in Section 3.3.3. The result after evaluation of the model is shown in Figure 8. Observe that the model is now successfully able to learn a meaningful encoding function for *all of the*

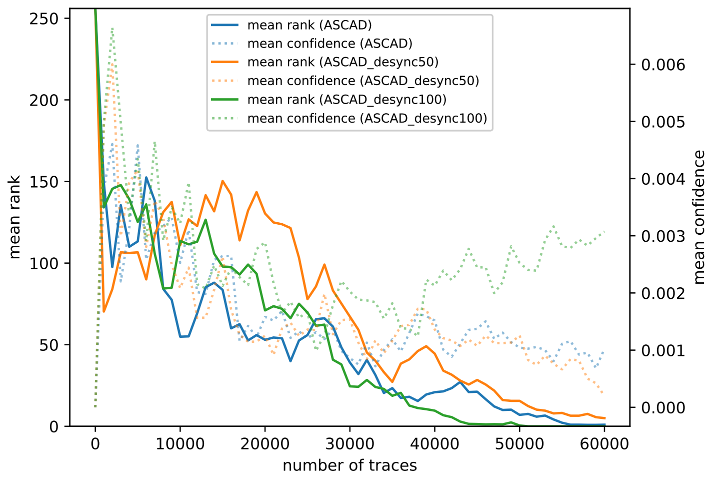


Figure 6: Mean rank based on a 10-fold cross-validation of the frequency-domain CO identity encoding function in relation to the number of validation traces used for a CEMA attack. The CEMA attack successfully finds the correct key for the `ASCAD_desync100` dataset after 51,000 traces, but the mean confidence is low (0.003). The attack is unsuccessful for the other datasets.

ASCAD datasets. After approximately 1,000 traces, a mean rank of 0 is achieved by each model. Furthermore, if we increase the number of traces, the mean confidence in the key guess increases as well. We believe this is a very encouraging result, showing that CO can be used to defeat both the masked AES and clock jitter countermeasures.

3.4.4 No model assumption

As indicated in Section 3.2.3, the true labels $y^{(1)}, y^{(2)}, \ldots, y^{(n_x)}$ are preprocessed such that their key byte values correspond to $\mathrm{HW}(\mathrm{SBox}(p_s^{(i)} \oplus k_s^{(i)}))$. Hence, we assume that the cryptographic device leaks information based on the HW power consumption model. In the following experiment, we determine whether the power consumption model can be learned implicitly by the optimization algorithm. To this end, we label $y_s^{(i)} = \mathrm{SBox}(p_s^{(i)} \oplus k_s^{(i)})$ so that the label values correspond to the intermediate value after the processing of the SBox. Then, we perform CO with the frequency-domain two-layer MLP model on the `ASCAD_desync100` dataset. The results are shown in Figure 9.

Compared to when a HW model was assumed (see Section 3.4.3), the correct key is guessed around 1,200 traces instead of 1,000 traces. Furthermore, the confidence is slightly lower, with 0.045 compared to 0.066. We conjecture that this slight decrease in performance is caused by the added complexity of learning the HW function.

3.5 Low-cost CEMA

The ASCAD dataset was recorded using an expensive oscilloscope and a sample rate of 2 GS/s. Since the frequency-domain CO from Section 3.4 showed promising results for noisy and unaligned data, we also investigated the effectiveness of the technique when using lower-cost hardware such as an SDR. To this end, we recorded a custom dataset of EM traces transmitted by an Arduino Duemilanove running a software AES implementation, using a Universal Software Radio Peripheral (USRP) B200 SDR sampling at 8 MS/s on the 64 MHz band with a TBPS01 EM probe and wideband amplifier. The experimental setup is shown in Figure 10.

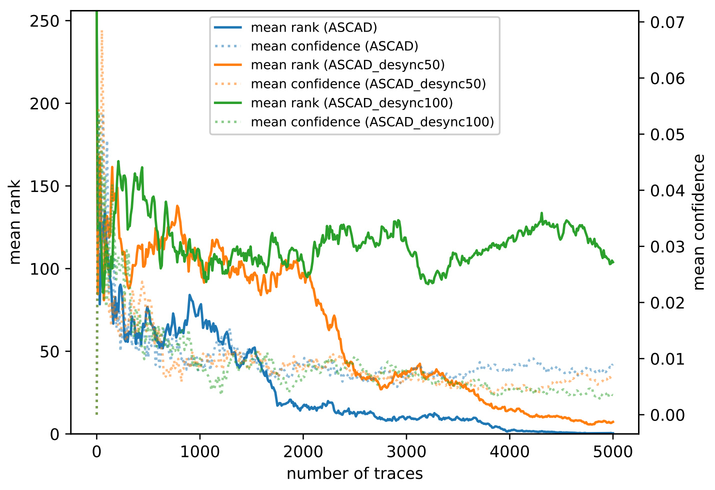


Figure 7: Mean rank based on a 10-fold cross-validation of the frequency-domain single-layer MLP CO model in relation to the number of validation traces used for a CEMA attack. Again, an improvement compared to the identity function can be observed: a mean rank of 0 is obtained for the `ASCAD` dataset after only 4,840 traces, with a mean confidence of 0.008. The attack is unsuccessful on the other datasets.

3.5.1 Dataset properties

The custom dataset consists of two sets of traces: a training set of 51,200 traces and validation set of 32,768 traces. Each trace contains the instantaneous amplitude of the I/Q signal provided by the SDR. No further preprocessing was performed. While recording the training set, the Arduino continuously performed AES encryption with a *random key*, such that one trace contains the EM leakage of one random encryption operation. The validation set contains EM traces of AES encryptions performed with a *fixed key*. This ensures that the ML model will not overfit on one specific key during training, and that a sufficient number of traces is available to perform a CEMA attack during validation. We stress that the fixed key used during the validation step is never encountered during training.

3.5.2 CO results

At first, we directly used the MLP architecture from Section 3.4.3 to train the model on the 51,200 random-key training examples. This approach turned out to be unsuccessful: the model heavily overfits on the noise that is present in the training examples after each subsequent epoch, and does not learn a generalizable relation between the key byte value and EM leakage. As a result, the training set loss is very low, whereas the validation set loss is high.

In order to resolve this issue, we artificially generated more training data by applying the *data augmentation* technique. More specifically, for each of the training examples, we set the starting offset of the sample window for which the FFT is calculated to a random offset between 0 and 500 time units. Indeed, with this approach, a training example will be time-shifted slightly differently for each epoch, which reduces the overfitting of the MLP model. The result after training for 100 epochs on the augmented training set is shown in Figure 11.

Observe that after 22,000 traces, the CEMA attack successfully determines the correct key, without performing any alignment or filtering of the EM traces and with a SDR sample rate of only 8 MS/s.

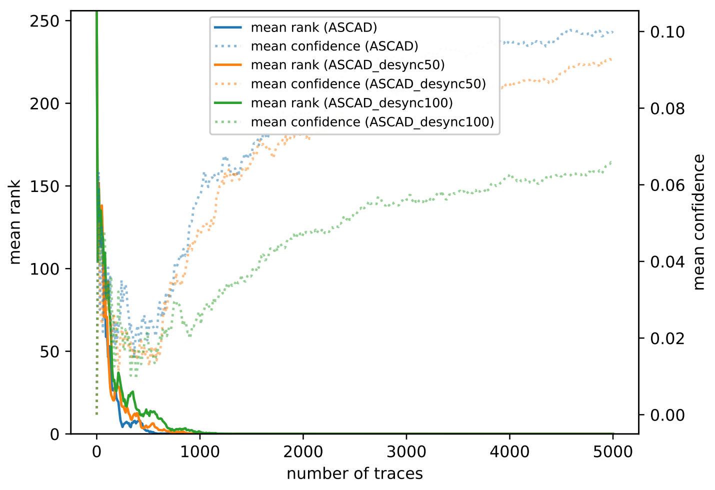


Figure 8: Mean rank based on a 10-fold cross-validation of the frequency-domain two-layer MLP CO model in relation to the number of validation traces used for a CEMA attack. The correct key is guessed for each dataset after approximately 1,000 traces. Adding more traces to the validation set further increases the mean confidence to 0.099, 0.092, and 0.066 for respectively the `ASCAD`, `ASCAD_desync50` and `ASCAD_desync100` datasets.

3.6 Discussion

3.6.1 Comparison to previous approaches

Previous works in context of applying ML to SCA have in common that they all use a model optimized by minimizing the mean cross-entropy loss over the training set. Thus, for each training example, the probability distribution $P(\hat{y}^{(i)} = v \mid x^{(i)})$ is calculated for each intermediate value $v \in \{0, 1, \ldots 255\}$ and its cross-entropy with the one-hot encoding of the true intermediate value $y^{(i)}$ is determined. Intuitively, this can be regarded as a Simple Electromagnetic Analysis (SEMA) attack on a single trace, since no information from other traces is used.

Although the above approach has been demonstrated to achieve reasonable results in context of SCA, we believe it is more suited to image classification applications, where only one image is often available that needs to be classified. In SCA, there is the need for extracting information from *multiple* examples, due to the noisy nature of EM traces. CO achieves this by optimizing the correlation coefficient of a mini-batch with the true key byte value. This may explain why even a simple two-layer MLP architecture performs better than the `best_cnn` model from the ASCAD paper; their CNN is unable to determine the correct key for the `ASCAD_desync100` dataset (see [PSB+18, p. 39]). We confirmed this result by retraining their `best_cnn` model for 100 epochs and performing a 10-fold cross-validation with the ASCAD datasets, analogous to the experiments previously discussed in Sections 3.3 and 3.4. The results of this experiment are shown in Figure 12.

3.6.2 Complexity

The most complex model we considered is the two-layer MLP model from Section 3.4.3. This model contains 180,741 parameters and took 271 seconds to train for 100 epochs on a Dell Latitude laptop with quad-core Intel Core i5-7300U CPU at 2.60 GHz (no GPU). By comparison, the ASCAD `best_cnn` model contains 66,652,544 parameters. In addition, the presence of convolutional layers further increases the complexity of this model. As a result, training the ASCAD CNN models for 100 epochs with this architecture takes 5.21

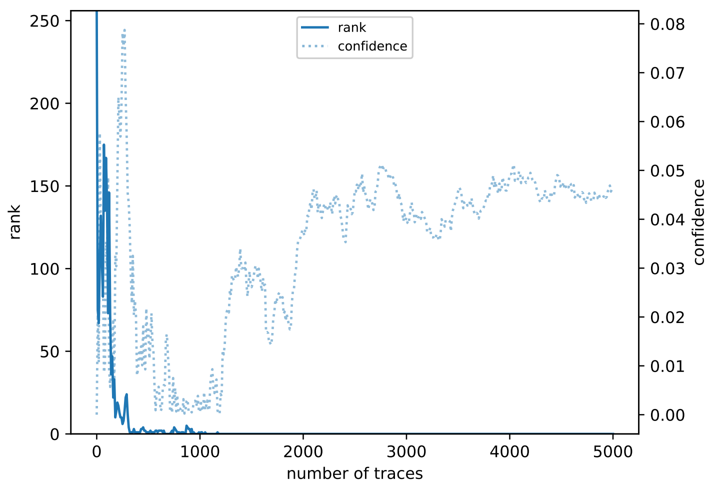


Figure 9: Rank and confidence of the frequency-domain two-layer MLP CO model in relation to the number of validation traces used for a CEMA attack. The power consumption model is not assumed; it is learned by the model. The correct key for the `ASCAD_desync100` dataset is guessed after approximately 1,200 traces with a confidence of 0.001, and increases further to 0.045 after 5,000 traces.

days per model on the same machine.

3.6.3 Hybrid CO and round detection

In our custom dataset containing the Arduino EM traces, each trace contains a variable number of samples, and the only assertion is that precisely one AES encryption is performed within the trace. In Section 3.5, we demonstrated that it is not required to align the traces in a preprocessing step due to the frequency-domain approach. The only requirement is that the first round of the AES encryption is contained within the FFT window. However, the larger the FFT window, the more irrelevant information will be included in the power spectrum and hence, the more traces are required to successfully perform a CEMA attack. This raises the question of how the FFT window can be optimally selected.

We did not consider the optimal selection of FFT windows in this paper, but leave a number of interesting pointers for future work. A first possible approach is to let the neural network determine a boundary of where the first AES round starts. This could be achieved with an algorithm such as YOLO [RDGF16], which automatically determines bounding boxes of certain objects (in this case an AES round). A second approach could be to perform a "hybrid" CO that takes place both in the time domain as well as the frequency domain. For example, by calculating a spectogram with overlapping FFT windows of the entire trace and finding the spectogram section with the highest correlation or by using wavelet transforms.

4 Related works

The first study of using ML in SCA was conducted by Hospodar et al. in 2011, where a power analysis of a software AES implementation without countermeasures was performed using a Support Vector Machine (SVM) [HGDM+11]. In 2015, Lerman et al. compared ML techniques with the classic TA and show that ML are especially interesting when the number of useless samples in a leakage trace increases and/or when the training set size

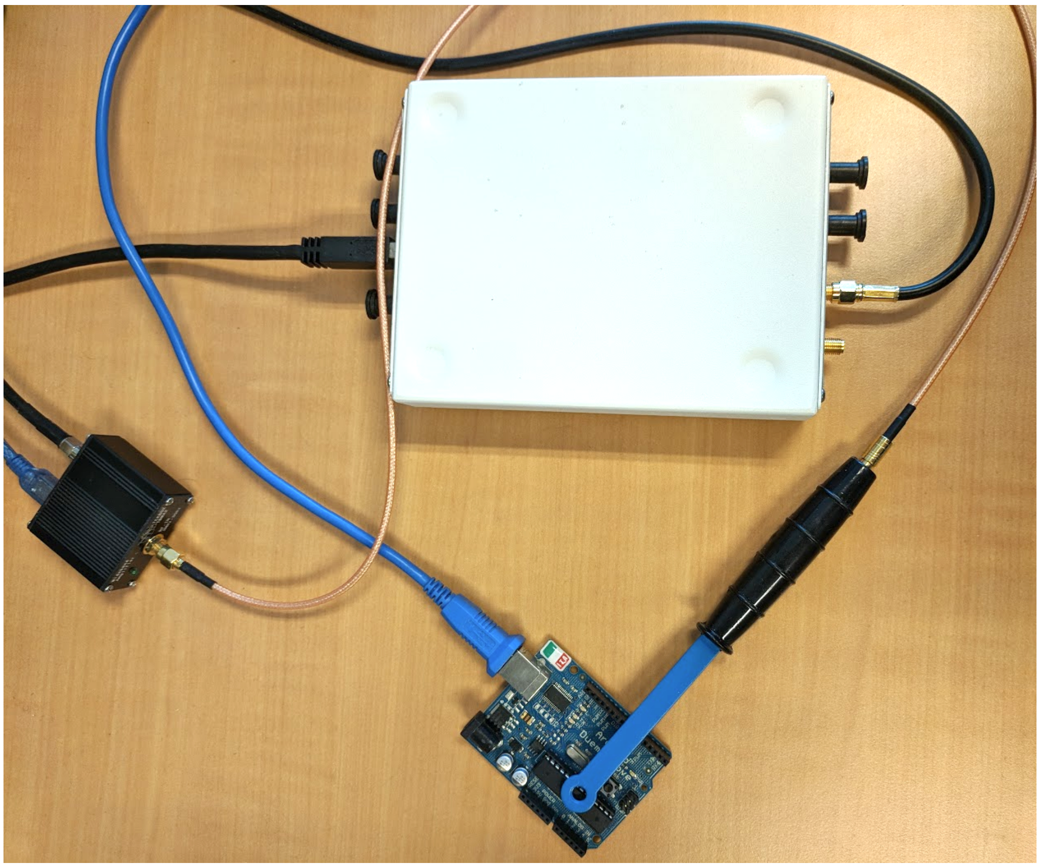


Figure 10: Experimental test setup for the low-cost CEMA attack. A Tekbox TBPS01 EM probe is positioned near the VCC pin of the ATmega 328, and is connected to the USRP B200 SDR through a 40 dB wideband amplifier.

is small. Maghrebi et al. apply DL techniques in side-channel context, and show that DL-based attacks are more efficient than ML-based and template attacks [MPP16]. This observation was recently nuanced by Picek et al. who suggest that ML techniques can perform on a similar level or even outperform DL techniques depending on the level of noise, number of measurements and number of features [PSH$^+$18]. Cagli et al. studied the robustness to misalignment of CNNs for an unmasked AES implementation [CDP17]. A comprehensive study regarding the application of DL algorithms in context of EM side-channel attacks was conducted by Prouff et al. in [PSB$^+$18]. They also introduced the ASCAD benchmark database, which was used in extensively Section 3 to evaluate our approach.

The use of frequency-domain features in SCA was pioneered by Tiu et al. in 2005 [Tiu05]. A similar technique was used in context of CPA in [SDB$^+$10] to mitigate misalignment problems. Barenghi et al. performed CEMA attacks on intervals of the FFT bins of the trace in order to determine leaking frequencies [BPT11]. This technique was further extended and explored in context of performing SCA using SDRs by Montminy et al. [MBTO13]. Other techniques to find the most leaking frequencies are investigated in [MRG$^+$10, TOT$^+$14].

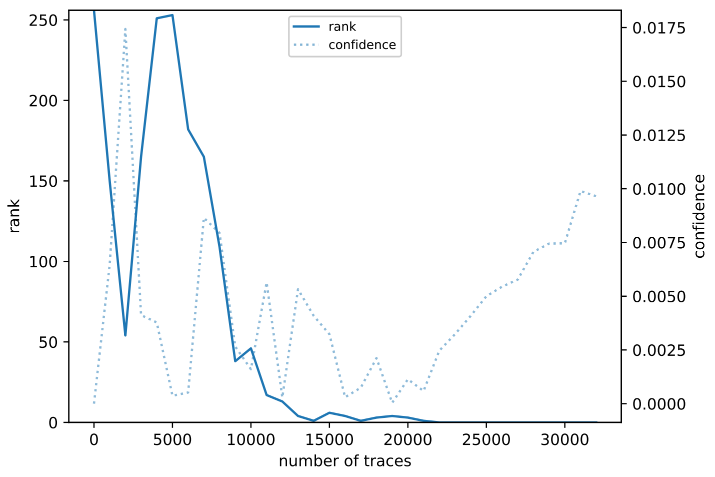


Figure 11: Rank and confidence for a single model trained on the custom dataset, after performing a CEMA attack on its encodings. The correct key is found after 22,000 traces.

5 Conclusion

We introduced a novel approach to improve CEMA attacks, called Correlation Optimization (CO). In this approach, a ML model is trained to learn "encodings" of a set of EM traces, which are subsequently used in a CEMA attack. These encodings are optimized such that their Pearson correlation with the secret key is maximal, by minimizing the correlation loss function defined in this paper. This is in contrast to previous works, where models are trained to classify individual EM traces into secret-key byte, intermediate, or Hamming values by optimizing the mean cross-entropy of the class probabilities.

The identity function (regular CEMA attack), one-layer and two-layer MLP models that we considered in this paper were evaluated on the ASCAD benchmark datasets for SCA [PSB+18] in both the time and frequency domain. Our best model, the frequency-domain two-layer MLP, is on average able to find the correct key byte after considering 1,000 traces from the `ASCAD`, `ASCAD_desync50`, and `ASCAD_desync100` datasets. We believe this is a significant improvement over the work of Prouff et al., where TA attacks, their "MLP best" model and "CNN best" model all failed to find the correct key byte for the `ASCAD_desync100` dataset after considering 5,000 traces.

In addition to evaluating our models on the ASCAD datasets, we examined their performance on a custom dataset as well. The custom dataset contains unprotected AES EM leakage traces of an Arduino Duemilanove recorded with a USRP B200 SDR sampling at 8 MS/s on the 64 MHz band. Even though the traces are highly dimensional and noisy, our frequency-domain two-layer MLP model is able to find the correct key after 22,000 traces, without requiring prior trace alignment. Finally, in order to allow for reproducing the results that were presented in this paper, all used datasets and code have been published to Github at the following location: `https://github.com/rpp0/correlation-optimization-paper`.

6 Future work

In future work, several aspects regarding CO can be investigated further. First, our technique can be evaluated in other contexts (e.g. other benchmark datasets, hardware or side-channels) in order to confirm whether the encouraging results obtained in this paper generalise to these cases as well. Second, we considered only simple ML models

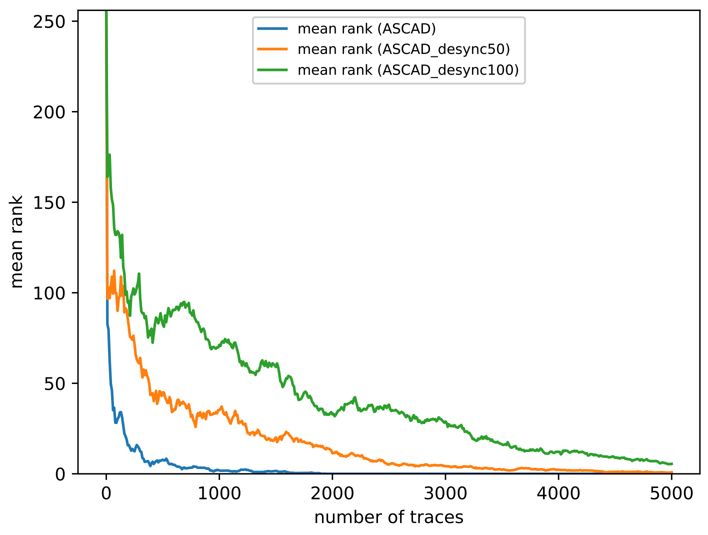


Figure 12: Mean rank based on a 10-fold cross-validation of the `best_cnn` model from the work of Prouff et al. [PSB+18], in relation to the number of validation traces used for a classification attack. Their model is able to determine the correct key for the `ASCAD` dataset after 1,910 traces, but performs poorly on the `ASCAD_desync50` and `ASCAD_desync100` datasets compared to our best CO models.

based on the MLP. Although this is sufficient for the CEMA attack to succeed on the ASCAD datasets and our custom dataset, more advanced models such as deep CNNs may be required in other scenarios. Additionally, the configuration space of architectures and hyperparameters is very large, and a better configuration may be determined in future work to further improve the results presented in this paper.

As mentioned in Section 3.6.3, we believe that both a combination of time and frequency-domain features, as well as automatic detection of an AES round using ML or DL could be interesting avenues for future research. We are currently investigating these ideas, along with the application of CO to other cryptographic algorithms besides AES.

Acknowledgements

This research was funded by a Ph.D. Grant of the Research Foundation Flanders (FWO), grant number 1S14916N. We would like to thank Tom Haber, Balazs Nemeth, Jeroen Put and Nick Michiels for their valuable input and advice.

References

[AAB+15] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[AARR03] Dakshi Agrawal, Bruce Archambeault, Josyula R. Rao, and Pankaj Rohatgi. The EM Side—Channel(s). In Burton S. Kaliski, çetin K. Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES*, pages 29–45, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[BCO04] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 16–29. Springer, 2004.

[BDG+10] Michael Backes, Markus Dürmuth, Sebastian Gerling, Manfred Pinkal, and Caroline Sporleder. Acoustic Side-Channel Attacks on Printers. In *USENIX Security symposium*, pages 307–322, 2010.

[BKMN09] Julien Brouchier, Tom Kean, Carol Marsh, and David Naccache. Temperature attacks. *IEEE Security & Privacy*, 7(2):79–82, 2009.

[Blu16] Bluetooth SIG, Inc. Bluetooth Core Specification v5.0, December 2016.

[BPT11] Alessandro Barenghi, Gerardo Pelosi, and Yannick Teglia. Information leakage discovery techniques to enhance secure chip design. In *IFIP International Workshop on Information Security Theory and Practices*, pages 128–143. Springer, 2011.

[C+15] François Chollet et al. Keras. https://keras.io, 2015.

[CDP17] Eleonora Cagli, Cécile Dumas, and Emmanuel Prouff. Convolutional Neural Networks with Data Augmentation Against Jitter-Based Countermeasures. In *International Conference on Cryptographic Hardware and Embedded Systems*, pages 45–68. Springer, 2017.

[CRR02] Suresh Chari, Josyula R Rao, and Pankaj Rohatgi. Template attacks. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 13–28. Springer, 2002.

[DLM+09] Amine Dehbaoui, Victor Lomne, Philippe Maurine, Lionel Torres, and Michel Robert. Enhancing electromagnetic attacks using spectral coherence based cartography. In *IFIP/IEEE International Conference on Very Large Scale Integration-System on a Chip*, pages 135–155. Springer, 2009.

[DR08] Tim Dierks and Eric Rescorla. RFC 5246: The Transport Layer Security (TLS) Protocol. *The Internet Engineering Task Force*, 2008.

[DR13] Joan Daemen and Vincent Rijmen. *The design of Rijndael: AES-the advanced encryption standard*. Springer Science & Business Media, 2013.

[GHT05] Catherine H Gebotys, Simon Ho, and Chin Chi Tiu. EM analysis of Rijndael and ECC on a wireless Java-based PDA. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 250–264. Springer, 2005.

[GKH+15] Mordechai Guri, Assaf Kachlon, Ofer Hasson, Gabi Kedma, Yisroel Mirsky, and Yuval Elovici. GSMem: Data Exfiltration from Air-Gapped Computers over GSM Frequencies. In *USENIX Security Symposium*, pages 849–864, 2015.

[GMO01] Karine Gandolfi, Christophe Mourtel, and Francis Olivier. Electromagnetic analysis: Concrete results. In Çetin K. Koç, David Naccache, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems — CHES 2001*, pages 251–261, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[GPT15] Daniel Genkin, Itamar Pipman, and Eran Tromer. Get your hands off my laptop: physical side-channel key-extraction attacks on PCs. *Journal of Cryptographic Engineering*, 5(2):95–112, Jun 2015.

[GST14] Daniel Genkin, Adi Shamir, and Eran Tromer. RSA key extraction via low-bandwidth acoustic cryptanalysis. In *International cryptology conference*, pages 444–461. Springer, 2014.

[GW08] Catherine H Gebotys and Brian A White. EM analysis of a wireless Java-based PDA. *ACM Transactions on Embedded Computing Systems (TECS)*, 7(4):44, 2008.

[HGDM$^+$11] Gabriel Hospodar, Benedikt Gierlichs, Elke De Mulder, Ingrid Verbauwhede, and Joos Vandewalle. Machine learning in side-channel analysis: a first study. *Journal of Cryptographic Engineering*, 1(4):293, 2011.

[HS13] Michael Hutter and Jörn-Marc Schmidt. The temperature side channel and heating fault attacks. In *International Conference on Smart Card Research and Advanced Applications*. Springer, 2013.

[IEE12] IEEE Computer Society. Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. IEEE 802.11 Standard, September 2012.

[IS15] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

[KB07] Boris Köpf and David Basin. An information-theoretic model for adaptive side-channel attacks. In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 286–296. ACM, 2007.

[KJJ99] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael Wiener, editor, *Advances in Cryptology — CRYPTO' 99*, pages 388–397, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[LB95] Yann LeCun and Yoshua Bengio. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, pages 255–258, 1995.

[LPMS17] Liran Lerman, Romain Poussier, Olivier Markowitch, and François-Xavier Standaert. Template attacks versus machine learning revisited and the curse of dimensionality in side-channel analysis: extended version. *Journal of Cryptographic Engineering*, Apr 2017.

[MBTO13] David P Montminy, Rusty O Baldwin, Michael A Temple, and Mark E Oxley. Differential electromagnetic attacks on a 32-bit microprocessor using software defined radios. *IEEE Transactions on Information Forensics and Security*, 8(12):2101–2114, 2013.

[MDM16] Zdenek Martinasek, Petr Dzurenda, and Lukas Malina. Profiling power analysis attack based on MLP in DPA contest V4.2. In *39th International Conference on Telecommunications and Signal Processing (TSP)*, pages 223–226. IEEE, 2016.

[MHN13] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, 2013.

[Mon13] David P Montminy. *Enhancing Electromagnetic Side-Channel Analysis in an Operational Environment*. PhD thesis, Air Force Institute of Technology, 2013.

[MPP16] Houssem Maghrebi, Thibault Portigliatti, and Emmanuel Prouff. Breaking cryptographic implementations using deep learning techniques. In *International Conference on Security, Privacy, and Applied Cryptography Engineering*. Springer, 2016.

[MRG$^+$10] Olivier Meynard, Denis Réal, Sylvain Guilley, Florent Flament, Jean-Luc Danger, and Frédéric Valette. Characterization of the electromagnetic side channel in frequency domain. In *International Conference on Information Security and Cryptology*, pages 471–486. Springer, 2010.

[OC15] Colin O'Flynn and Zhizhang David Chen. Side channel power analysis of an AES-256 bootloader. In *28th Canadian Conference on Electrical and Computer Engineering (CCECE)*, pages 750–755. IEEE, 2015.

[OC16] Colin O'Flynn and Zhizhang Chen. Power analysis attacks against IEEE 802.15.4 nodes. In *International Workshop on Constructive Side-Channel Analysis and Secure Design*, pages 55–70. Springer, 2016.

[PR07] Emmanuel Prouff and Matthieu Rivain. A Generic Method for Secure SBox Implementation. In *International Workshop on Information Security Applications*, pages 227–244. Springer, 2007.

[PSB$^+$18] Emmanuel Prouff, Remi Strullu, Ryad Benadjila, Eleonora Cagli, and Cécile Dumas. Study of deep learning techniques for side-channel analysis and introduction to ASCAD database. *IACR Cryptology ePrint Archive*, page 53, 2018.

[PSH$^+$18] Stjepan Picek, Ioannis Petros Samiotis, Annelie Heuser, Jaehun Kim, Shivam Bhasin, and Axel Legay. On the performance of convolutional neural networks for side-channel analysis. Cryptology ePrint Archive, Report 2018/004, 2018. https://eprint.iacr.org/2018/004.

[QS01] Jean-Jacques Quisquater and David Samyde. ElectroMagnetic Analysis (EMA): Measures and Counter-Measures for Smart Cards. In *Proceedings of the International Conference on Research in Smart Cards: Smart Card Programming and Security*, E-SMART '01, pages 200–210, London, UK, UK, 2001. Springer-Verlag.

[RDGF16] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You Only Look Once: Unified, Real-time Object Detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[Riv08] Matthieu Rivain. On the exact success rate of side channel analysis in the Gaussian model. In *International Workshop on Selected Areas in Cryptography*, pages 165–183. Springer, 2008.

[SDB$^+$10] Oliver Schimmel, Paul Duplys, Eberhard Boehl, Jan Hayek, Robert Bosch, and Wolfgang Rosenstiel. Correlation power analysis in frequency domain. In *International Workshop on Constructive SideChannel Analysis and Secure Design (COSADE)*, 2010.

[SKP15] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[SMY09] François-Xavier Standaert, Tal G Malkin, and Moti Yung. A unified framework for the analysis of side-channel key recovery attacks. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 443–461. Springer, 2009.

[ST04] Adi Shamir and Eran Tromer. Acoustic cryptanalysis. *presentation available from* `http://www.wisdom.weizmann.ac.il/~tromer`, 2004.

[Tiu05] C Tiu. A New Frequency-Based Side Channel Attack for Embedded Systems. Master's thesis, University of Waterloo, 2005.

[TOT$^+$14] Sébastien Tiran, Sébastien Ordas, Yannick Teglia, Michel Agoyan, and Philippe Maurine. A model of the leakage in the frequency domain and its application to CA and DA. *Journal of Cryptographic Engineering*, 4(3):197–212, 2014.

[VP09] Martin Vuagnoux and Sylvain Pasini. Compromising Electromagnetic Emanations of Wired and Wireless Keyboards. In *USENIX security symposium*, pages 1–16, 2009.

[ZF05] YongBin Zhou and DengGuo Feng. Side-Channel Attacks: Ten Years After Its Publication and the Impacts on Cryptographic Module Security Testing. *IACR Cryptology ePrint Archive*, page 388, 2005.