Stacked Ensemble Model for Enhancing the DL based SCA

Anh Tuan Hoang, Neil Hanley, Ayesha Khalid, Dur-e-Shahwar Kundi and Maire O'Neill
Centre for Secure Information Technologies (CSIT), ECIT, Queen's University Belfast, U.K.

Keywords: SCA, AES, Deep Learning, CNN, CNNP, Stacked Ensemble, Key Reveal, Multi Masking.

Abstract: Deep learning (DL) has proven to be very effective for image recognition tasks, with a large body of research on various models for object classification. The application of DL to side-channel analysis (SCA) has already shown promising results, with experimentation on open-source variable key datasets showing that secret keys for block ciphers like Advanced Encryption Standard (AES)-128 can be revealed with 40 traces even in the presence of countermeasures. This paper aims to further improve the application of DL in SCA, by enhancing the power of DL when targeting the secret key of cryptographic algorithms when protected with SCA countermeasures. We propose a stacked ensemble model, which trains the output probabilities and Maximum likelihood score of multiple traces and/or sub-models to improve the performance of Convolutional Neural Network (CNN)-based models. Our model generates state-of-the art results when attacking the ASCAD variable-key database, which has a restricted number of training traces per key, recovering the key within 20 attack traces in comparison to 40 traces as required by the state-of-the-art CNN-based model with Plaintext feature extension (CNNP)-based model. During the profiling stage an attacker needs no additional knowledge of the implementation, such as the masking scheme or random mask values, only the ability to record the power consumption or electromagnetic field traces, plaintext/ciphertext and the key is needed. However, a two step training procedure is required. Additionally, no heuristic pre-processing is required in order to break the multiple masking countermeasures of the target implementation.

1 INTRODUCTION

Since SCA was introduced in 1996 (Kocher, 1996) based on the difference in the power consumption of bit transitions, much research has been conducted on efficient methods to both break and protect cryptographic implementations. Common attack methods such as differential power analysis (DPA) (Kocher et al., 1999), correlation power analysis (CPA) (Brier et al., 2004), or differential frequency-based analysis (DFA) (Gebotys et al., 2005), allow divide and conquer strategies to significantly reduce the computational complexity of key recovery when additional power (or electromagnetic) information is available. For example, in the case of the AES-128, it is reduced from $O(2^{128})$ to $O(16 \times 2^8)$. In order to protect against such attacks a number of countermeasures have been proposed, many of which are now standard in commercial security products such as credit cards *etc.* At the hardware layer techniques such as dual-rail logic (Tiri et al., 2002; Popp and Mangard, 2005; Chen and Zhou, 2006; Hoang and Fujino, 2014) attempt to equalise the power consumption of the underlying algorithm regardless of the data being processed, while at the algorithmic layer techniques such as masking (D and Tymen, 2002; Goubin L., 2011; Nassar et al., 2012) introduce fresh randomness to reduce the useful leakage available to an attacker. All countermeasures techniques come with various trade-offs for the level of protection provided in terms of execution time, randomness required, silicon (or memory) size *etc.*

While statistical and machine learning (ML) have a long history, recent progress in DL in particular, has led to such techniques being applied in the SCA context for key recovery in the presence of countermeasures. These attacks fall under the profiling adversarial model, where it is assumed that the attacker has a similar (or identical) training device(s) to measure a large quantity of traces in order to build an accurate power model, which then allows key recovery from the target device in relatively few traces. Among the DL approaches, convolutional neural network (CNN) based models seem most promising (Maghrebi et al., 2016; Picek et al., 2018; Timon, 2019; Hoang et al., 2020) due to their effectiveness when training with raw data, with the *convolutional* layer acting as a filter to pick out the relevant features for classification.

Hoang, A., Hanley, N., Khalid, A., Kundi, D. and O'Neill, M.
Stacked Ensemble Model for Enhancing the DL based SCA.
DOI: 10.5220/0011139700003283
In *Proceedings of the 19th International Conference on Security and Cryptography (SECRYPT 2022)*, pages 59-68
ISBN: 978-989-758-590-6; ISSN: 2184-7711

1.1 Related Work

There are a large number of available ML and DL algorithms and models, as well as more general statistical learning techniques that can be applied to SCA. Some initial DL based SCA attacks have been proposed by (Markowitch et al., 2013; Gilmore et al., 2015; Martinasek et al., 2016). However, these attacks are based on an assumption that the number of masks are either very limited, or an adversary is able to fully access the internal values of the target devices when profiling, including the values of the random masks, which is generally not feasible in practice.

The effectiveness of using CNN models is shown in (Weissbart et al., 2019), with no dimensionality reduction required when attacking implementations of public-key cryptography when compared to other profiling attack algorithms such as support vector machine (SVM), template attack (TA) or random forest (RF). In efforts to increase the effectiveness of DL performance in the context of SCA, an in-depth analysis in (Prouff et al., 2018; Maghrebi, 2019) noted that the size of filter in a CNN model is an important factor as its length should cover the most interesting points of interest (PoI) to enable the combination of the corresponding leakage while (Kim et al., 2019) shown that adding artificial noise to the source traces was found to greatly reduce over-fitting the model to the training set, improving classification accuracy.

State of the art efforts in profiling attack seen to include SCA domain knowledge into DL architectures as proposed by (Hettwer et al., 2018; Hoang et al., 2020), in which the plaintext was given as an additional input to increase the accuracy when directly training on the key value. Additional efforts to improve CNN models for SCA were presented by (Perin et al., 2020), in which the Maximum likelihood score is applied to a number of simple CNN or multilayer perceptrons (MLPs) models and traces with classification based on Hamming weight. Raw traces are also used by (Lu et al., 2021) to find the leakage information not only from AES computation but also from other execution like the preparation of the masked SBox to attack AES implementation.

While this paper shares a similar approach to inputting domain knowledge as in (Hoang et al., 2020) and the use of multiple models as in (Perin et al., 2020), there is a significant difference with regards to architectural aspects such as the structures and the way multiple models and/or traces are combined for re-training from error , enabling our model to further reduce the number of traces in an attack. Different from those research, our proposed model combines the outputs of multiple traces from multiple CNN-based model with Plaintext feature extension (CNNP) models and their maximum likelihood score (MLS) in stacked ensemble model for re-training from error to increase the accuracy with smaller number of traces.

The main finding of the proposed research are as follows:

- The use of multiple models and trace combination increases the accuracy of a ML model.
- MLS is an efficient method for models and trace combination and should be included in the ensemble model.
- Classification based on byte values should be used instead of hamming weight to avoid many hypothesis keys statistically located in the same high rank (probability).
- The probability outputs of multiple traces and models should be trained (in a stacked ensemble model) to increase the accuracy rather than simply summing up using the MLS method.

Given these findings, we built a complex model with multiple sub-models working together on the same group of traces to generate inputs for the proposed stacked ensemble model. Additionally, the MLS of those outputs are also considered as input features. We label the traces using the output values of the SBox in round 1 so that no projection issues between byte values and hamming weights occurs.

1.2 Our Contributions

This paper shows the limitations of applying MLS when combining sub-models for an ensemble model in the field of SCA (based on traces from a protected AES implementation). We propose stacked ensemble models developed from CNN model with Plaintext extension (CNNP at (Hoang et al., 2020)) that looks to enhance DL from a side-channel aspect, taking the strength of number of models (Perin et al., 2020), MLS and re-training from error into consideration. Our proposed models allow key recovery when targeting an AES implementation protected with multiple masking scheme, outperforming previous research on TAs, MLs or current CNN and CNNP models in terms of required traces for key recovery (24 on the ASCAD (Prouff et al., 2018) variable key database).
Our main contributions include:

1. Demonstrating the advantage of using the stacked ensemble model as a method for combining multiple models and traces over the ordinary MLS and MLS-based ensemble method.
2. Propose a novel stacked ensemble convolutional neural network with Plaintext feature extension

(SECNNP) model architecture for side channel analysis. SECNNP model exploits the advantage combining multiple models and traces over the ordinary MLS and MLS-based ensemble methods.

3. Provide an evaluation and verification of our proposed SECNNP model on the ASCAD database. While the ASCAD masked AES design is an example target of the architecture, it can be applied to any AES implementation. We demonstrate a reduction in the requirements of applying DL in SCA, particularly when attacking implementations with countermeasures, in which thousands of traces can be required to break protected AES implementations using DL. SECNNP enables a key recovery with only 24 traces, which is a half the reduction of 42 traces required by the state of the art models.
4. Discussion on the optimal number of sub-models and traces required in the SECNNP model as well as the trade-off between accuracy and training time for single and multiple *convolutional* filter kernel size models.

2 BACKGROUND

2.1 SCA, Countermeasures and ASCAD Database

Side channel attacks work on the principal that the power consumed by a device is dependent on the operation being performed and the data being processed. This allows an adversary to estimate what the power consumption should be for some intermediate value that is a function of some known data (*e.g.* a plaintext or ciphertext byte), and some unknown data (*e.g.* a secret key byte). In a profiled attack, as in this paper, it is assumed that an attacker has a similar or identical device on which they can record a large number of acquisitions, allowing an accurate profiling of the actual power leakage of the device. Leakage models can be Hamming weight (HW) or Identity (ID). In the HW leakage model, the sensitive variable's Hamming weight of the 8-bit Sboxes is considered as the leakage information. In the ID leakage model, as in this paper, an intermediate value of the cipher is considered as the leakage. It allows the model to learn a broad range of features at different points in time but results in a high volume of classes (256 classes for 8-bit data) in the training and attacking process but the data is more balanced in training and there is a low probability that many hypothesis keys achieve a high-rank even with single trace attacks, and so it is applicable with a small number of traces, *e.g.* less than 100.

While a number of SCA countermeasures in both hardware and software have been proposed in the literature (such as dual-rail logic, dummy operations, threshold implementations *etc.*), here we focus on masking as this is the countermeasure that is applied to the traces under consideration. Masking (including higher-order variants) applies one or several random masks to the sensitive data such as to the input of the S-Box or the S-Box itself, respectively forming 1st-order or higher-order masking schemes (Reparaz et al., 2015).

Multi additive masking for AES implementations can be seen as masking of the plaintext by:

$$\overline{p_i} = p_i \oplus m_i \tag{1}$$

and masking the S-Box for value $i \in [0 \ldots 255]$, which can be prepared in advance:

$$\overline{\text{S-Box}\,(x)} = \text{S-Box}\,(x \oplus m_{i,in}) \oplus m_{i,out} \tag{2}$$

in which, p and m are the plaintext and mask, respectively in equation 1 and x is the masked input for the S-Box, $\oplus m_{i,in}$ and $\oplus m_{i,out}$ are the masks used to unmask the input and mask the output of S-Box in equation 2. Equation 1 ensures that the linear AddRoundKey operation which follows the S-Box works as expected on the masked data, but no unmasked data is processed. Equation 2 ensures that the non-linear S-Box operation will be masked by unknown pair values $m_{i,in}$ and $m_{i,out}$ so that on every execution different data will be processed regardless of the input value.

In our experiment on ASCAD database (Prouff et al., 2018), the target implementation on 8-bit AVR ATMega8515 embedded device has a multi masking scheme, in which the plaintext and SBox are masked by two independent masks as shown in equations 1 and 2. Even though a new ASCAD-v2 implementation is provided with affine masking, this paper aims to ASCAD-v1 due to the availability of dataset and reference models like CNNP (Hoang et al., 2020), TA, MLP and ASCAD-CNN (VGG16 like) models (Prouff et al., 2018). There are two sub-datasets of fixed and variable key in ASCAD-v1 but due to the availability of bijection of $S[(.) \oplus K]$ into fixed K, we will not take the ASCAD fixed key dataset into our consideration. The ASCAD variable-key dataset has a set of $200,000$ traces for training and $100,000$ traces for testing. The training traces have random variable keys, as well as random plaintext and mask values. The $100,000$ testing traces have the same key with random plaintext and mask values. Each trace has $1,400$ features and is again labelled by the output of the 3rd S-Box in the first round, giving ≈ 781 traces per label. Even though a number for reference models of TA, MLP, and CNN are provided as part of the ASCAD database (Prouff et al., 2018), CNNP (Hoang et al., 2020) shows its advanced performance in attacking the database. We utilize CNNP as sub-models in building our proposed SECNNP model.

2.2 Convolutional Neural Networks with Plaintext Feature Extension

2.2.1 Convolutional Neural Networks

CNNs are a DL architecture that incorporates several different types of layers for detecting features for classification as follows.

1. *Convolutional* layers are based on a convolution process, where a small array of small size, known as filter is passed over the input sequence signal (*i.e.* in our case the trace). The filter runs or *strides* over the traces so that if the same feature (*i.e.* leakage) appears in a different position, it can still be detected.*Convolutional* layers can be stacked to find higher abstractions of feature.
2. A pooling layer is often used in conjunction with the *convolutional* layer to reduce the size of the parameters to be learned and the subsequent computational requirements. *MaxPooling* layer considers the maximum value in a region as the most important feature of that region.
3. A *fully-connected* layer flattens the output of the previous layer, combining all previous nodes (or input features) together by multiplying those input nodes with corresponding weights and summing with the bias to calculate each output node.
4. Dropout is known as a simple way to prevent neural networks from over-fitting.Dropout is implemented by randomly disconnecting a percentage of neurons in the network, preventing any particular feature having a disproportionate effect on the model as a whole.
5. Rectified Linear units (ReLu) are used as the activation function in intermediate layers, *i.e.* the *convolutional* and *fully-connected* layers by replacing a value by itself if positive or else zero is negative. This helps mitigate the *vanishing gradients* problem which can happen in large networks.
6. *Softmax* is used for the activation function in the last *fully-connected* layer. This function is also known as a normalised exponential function, which converts the raw logistic scores into corresponding probabilities.

2.2.2 Convolutional Neural Network with Plaintext Feature Extension

It is clearly shown in (Hoang et al., 2020) that the plaintext (or ciphertext) is an important factor in building leakage models as it is XORed with the key prior to the S-Box in the first round. The output of this operation is used for labelling traces when training DL models.

$$y = \text{S-Box}\left(p_i \oplus k_i\right) \tag{3}$$

in which, p and k are the plaintext and key, respectively.

Regardless of the countermeasure utilised, the designer needs to modify the plaintext in some way in order to hide the sensitive value input to the S-Box. Whatever the designers do to protect the sensitive data from side-channel leakage, they will need to modify some signal or variable related to plaintext, and this processing will leave either first or higher-order leakage. The model for Convolutional Neural Network with Plaintext input (CNNP) with a single *convolutional* filter kernel size can be given by the following formula:

$$s \circ [\lambda^3] \circ \beta \circ [\lambda^2] \circ [P_{\text{ext}} \circ [\delta \circ [\alpha \circ \gamma_x]]^3] \tag{4}$$

in which, s, λ, β, P_{ext}, δ, α, and γ are the *softmax*, *fully-connected*, *dropout*, *plaintext feature extension*, *MaxPooling*, *activation*, and *convolutional* layers, respectively. γ_x is a *convolutional* layer with filter kernel size of 3. CNNP with one-hot encoding (Hoang et al., 2020) is used as sub-models in our Stacked Ensemble CNNP (SECNNP) and is shown in figure 1.

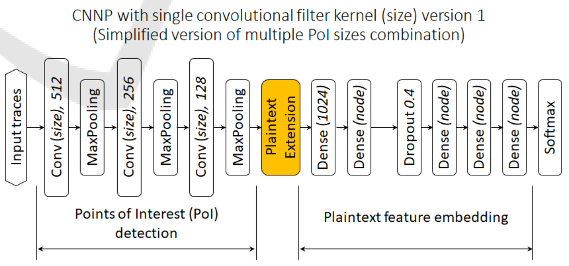

Figure 1: CNNP sub-models.

2.2.3 Maximum Likelihood Scores

Maximum likelihood scores (MLS) can be used for combining results of attacks from multiple different traces (Standaert et al., 2009) and/or different models (Perin et al., 2020) with the same key. It estimates the *likelihood* of each hypothesis key by multiplying the classification probability given by independent traces and/or models to create a *scores vector*. Values in this *scores vector* are then sorted according to probability for ranking. Accuracy is considered as the location of the correct key in the ranked *scores vector*. The closer the rank of the correct key is to 1, the higher the accuracy of the evaluated model.

3 STACKED ENSEMBLE METHOD FOR DL BASED SCA

3.1 Ensemble Methods

Deep learning relies on minimizing some loss function in a given environment, and so, the algorithm will decide the type of patterns it can learn. Hence, for given traces, different algorithms (or different models) will be able to capture different SCA leakage and in many cases, their predictive power complement each other, making a model fails to find the good features in a trace but another succeeds. Ensemble models learn the optimal combination of the base model predictions to increase accuracy.

3.2 CNNP based Stacked Ensemble Model (SECNNP)

3.2.1 Model Development Assumptions

When building a model of the power consumption for the target device, the following assumptions hold:

- The attacker cannot access the design or values used for random number generation but can profile keys on the device as well as access the plaintext and/or ciphertext for a profiling attack.
- The attacker does not know any implementation details except the algorithm being targeted.
- The CNNP model is capable of attacking the AES implementations due to the reduction in the number of unknown factors in training and attacking.
- Multiple models and/or trace combination in a trained model would increase the performance better than that of MLS.

3.2.2 Attack Model

The proposed SECNNP model attacks AES implementations based on recovering the unmasked value of the S-Box output (even where that value is masked) which allows key recovery given knowledge of the plaintext. In particular, we trained the proposed CNNP model by labelling the traces by the 3$^{\text{rd}}$ S-Box byte, which is computed from values known to the attacker during the profiling process as given in equation 3. The trained CNNP models then are used as sub-models in the SECNNP model, in which the same labelling is used but the inputs are the hypothesis key probabilities predicted by those sub-models. This labelling method follows that used by the reference models in the ASCAD databases (v1) and by the CNNP models in (Hoang et al., 2020). In order to attack the remaining key bytes, the traces would have to be relabelled for each S-Box output and the model retrained. However the same power traces can be used to attack the other key bytes.

3.2.3 Two Step Stacked Ensemble Model Training

Different from (Perin et al., 2020), who applied MLS, in which the probability of hypothesis keys corresponding to the hamming weight class are multiplied, we apply a two step training method to build our proposed stacked ensemble model.

- The first training step is the training of a number of general machine learning models, in which a number of CNNP (Hoang et al., 2020) instances are trained. The accuracy of those instances must be diverse (Wang, 2008), and so different numbers of epochs, hyperparameters and *dropout* layers are used in the CNNP sub-models training.
- The use of a second training step is proposed in this paper, in which a number of traces with the same class (from the same plaintext in other words) are provided to the model(s) trained in the first step for their probability of hypothesis key outputs. The MLS of the same trace on multiple models, the MLS of multiple traces on each model and the MLS of all traces on all models are calculated. The output probabilities of each hypothesis key by each sub-model, together with the MLSs calculated above are grouped together as inputs to a MLP. The MLP is trained by the corresponding class given by the trace(s) at the first step. Even though, there is no limitation on the number of traces and models involved in the MLP, we limit the number of models and traces to 3 and 6 due to the computational complexity.

Since the MLSs of traces and models received from the first step are used as an additional input for the MLP, they will be called inner-MLS. The proposed stacked ensemble model will need either multiple traces or multiple sub-models or both for the availability of the inner-MLS. Due to the limitation in the number of traces and sub-models given above, an outer-MLS is used to combine the attacking results from more traces than a input together. Differing from the first and second training step, where the traces are from the same classes, traces in this MLS combination can come from different classes. Hence, we have two kinds of attack: a single plaintext attack, where all traces come from the same plaintext in the SECNNP model and the final MLS and a multiple plaintext attack, where all traces in the SECNNP model come from the same plaintext, but different plaintexts are used in the final MLS.

3.2.4 Multi-trace Multiple CNNP Sub-models Stacked Ensemble Model

The multi-trace multiple CNNP sub-models stacked ensemble model utilizes key probabilities result from multiple CNNP sub-models with multiple trace for hypothesis key probability calculation independently. Since two independent hypothesis key probability outputs are given from different CNNP instances and different traces, two inner MLSs are used as additional inputs for the SECNNP model. One is the MLS from multiple traces predicted by each CNNP submodel and the other is the MLS of all traces and all sub-models.

Figure 2 shows the proposed SECNNP model with n input traces and m CNNP sub-models. The number of CNNP sub-models and traces can change within the limitation of computational power.

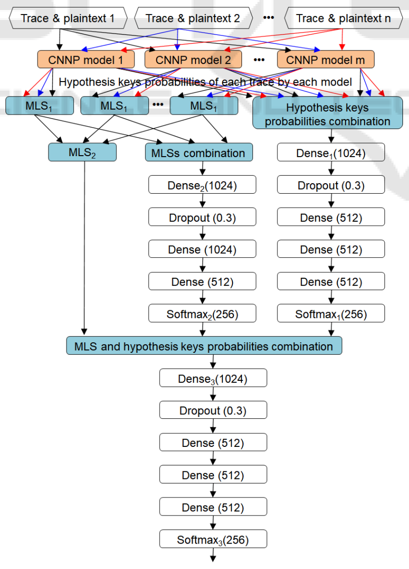

Figure 2: SECNNP model with multiple traces and multiple CNNP sub-models.

The first MLS (MLS$_1$) is used to combine the probabilities of the hypothesis keys predicted by each CNNP sub-model while the second MLS (MLS_2) layer is used to combine the prediction of all n traces predicted by all *CNNP* sub-models. The computation of the MLS$_1$ and MLS$_2$ for a hypothesis key i are given by equations 5 and 6, respectively:

$$MLS_1[i] = \prod_{l=1}^{n}(prob[l][i]); \qquad i \in [0..255] \tag{5}$$

$$MLS_2[i] = \prod_{p=1}^{m}\prod_{l=1}^{n}(prob[p][l][i]); \qquad i \in [0..255] \tag{6}$$

in which, l is the input trace number, p is the CNNP model number, i is the hypothesis key, and so, $prob[p][l][i]$ is the probability of hypothesis key i of the input trace l predicted by the CNNP sub-model p.

The *fully-connected* layer with m CNNP sub-models and n traces receives $m \times n$ probability inputs. The formula for that layer with M nodes is now:

$$\begin{aligned} Dense_1 1024[j] = \sum_{p=1}^{m}\sum_{l=1}^{n}\sum_{i=1}^{256}(b[j] + prob[p][l][i] \\ \times w[j][p][l][i]); \qquad j \in [1..M] \end{aligned} \tag{7}$$

in which $Dense_1 1024[j]$ is the jth output of the *fully-connected* layer, $prob[p][l][i]$ and $w[j][p][l][i]$ are the probability and corresponding weight of the hypothesis key i of the input trace l predicted by the CNNP sub-model p.

The *fully-connected* layer $Dense_2(1024)$ receives m inputs, which are the MLS of the CNNP submodels (MLS$_1$). The formula of that layer with M nodes is now:

$$\begin{aligned} Dense_2 1024[j] = \sum_{p=1}^{m}\sum_{i=1}^{256}(b[j] + MLS[p][i] \\ \times w[j][p][i]); \qquad j \in [1..M] \end{aligned} \tag{8}$$

in which $Dense_2 1024[j]$ is the jth output of the *fully-connected* layer, $MLS[p][i]$ and $w[j][p][i]$ are the probability and corresponding weight of the hypothesis key i of all n input traces predicted by the CNNP submodel p and combined by MLS_1.

The *fully-connected* layer $Dense_3(1024)$ receives 3 inputs, which are the three 256 hypothesis keys probabilities given by $Softmax_1$, $Softmax_2$ and the combined MLS_2 of all n traces predicted by all m CNNP sub-models (MLS_2). The formula of that layer with M nodes is now:

$$\begin{aligned} Dense_3 1024[j] = \sum_{i=1}^{256}(b[j] + MLS_2[i] \times w_{MLS_2}[j][i]) + \\ \sum_{i=1}^{256}(b[j] + prob_{Softmax_2}[i] \times w_{Softmax_2}[j][i]) + \\ \sum_{i=1}^{256}(b[j] + prob_{Softmax_1}[i] \times w_{Softmax_1}[j][i]); \\ j \in [1..M] \end{aligned} \tag{9}$$

in which $Dense_3 1024[j]$ is the jth output of the *fully-connected* layer *Dense*$_3$*(1024)*, $MLS_2[i]$ is the output probability of hypothesis key i from module MLS_2 and $w_{MLS_2}[j][i]$ is the weight of the corresponding input $MLS_2[i]$ for the node j. Similarly, $prob_{Softmax_1}[i]$, $w_{Softmax_1}[j][i]$, $prob_{Softmax_2}[i]$ and $w_{Softmax_2}[j][i]$ are the output probability of hypothesis key i and its weight from the $Softmax_1$ and $Softmax_2$ modules.

The structure given in the figure 2 can be simplified by limiting the number of input or the number of CNNP sub-model to one, making single trace multiple CNNP sub-moldes and multi-trace single CNNP sub-model stacked ensemble models. However, only the multiple CNNP sub-models stacked ensemble model will be discussed due to the significant performance compared with the reference.

4 EXPERIMENTAL RESULTS

Even though (Prouff et al., 2018) provides a number of reference models such as TA, MLPs and a pre-trained CNN, the CNNP models are used for benchmarking and as sub-models for the proposed SECNNP model due to their better performance.

While a number of variants of the SECNNP architecture with different hyperparameter (number of traces, number of sub-models, number of layers and number of nodes) were tested, in the following we present results from SECNNP models with the architectures shown in figure 2. The number of traces is in the range of [1..6] and the number of CNNP sub-models is in the range of [1..3].

SECNNP models develop from CNNP with single *convolutional* filter kernel size of 3 and CNNP with two *convolutional* filter kernel sizes of 3 and 5. The models are trained and evaluated using the ASCAD v1 database traces on a variable-key dataset. Training is performed on a VMware virtual machine, with access to virtual NVIDIA GRID M60-8Q and M40-4Q GPUs with 8GB and 4GB memory, respectively.

In the experiments using the ASCAD database, the number of training epochs and time for each model, together with the rank of the correct key are reported. Our comparison method is straightforward, in that we train the CNNP models with the *training dataset* group. The hyperparameters are taken from (Hoang et al., 2020), in which, we focus on the CNNP models with *convolutional* filter kernel size of 3 and the transfer learning CNNP model with *convolutional* filter kernel sizes of 3 and 5. Three models with different epochs are selected as sub-models for the SECNNP structure in figure 2. The SECNNP structures are then retrained on the same *training dataset* group for the final SECNNP models. We evaluate the trained models on the separate *test dataset* group provided in the ASCAD database.

Traces in the *test dataset* group are sorted by the *plaintext*. Those belonging to the same *plaintext* are then grouped by [1..6] depending on the number of traces required by the SECNNP model. Multiple groups of traces with different *plaintexts* are used if more traces (*e.g.* 42) are required.

At each *run*, a number of trace groups are randomly selected from the test dataset for the attack phase, with the maximum likelihood score (MLS) of each hypothesis key calculated as a function of the number of traces. These maximum likelihood scores are then sorted after each run and the rank of the correct key is recorded. N runs are evaluated and the mean of the correct key rank is computed. We evaluate our models using 100,000 traces from the test dataset. Depending on how fast the key rank converges, the number of traces for each run is different. In the evaluation, we set the number of traces to 42 and the number of runs N is set to 50 for average rank calculation to identify with (Hoang et al., 2020). The better performing models are the ones that have a lower key rank with similar or less traces required.

In our experiments, we consider the number of traces required for a model be able to attack the key as the number where the correct sub-byte key achieves rank 3 or below because a brute force key search for the entire key should then take at least $3^{16} = 43,046,721$ loops, an acceptable computational for modern computers.

4.1 SECNNP Evaluation with Single *Convolutional* Filter Kernel Size

In this section, we will compare the accuracy in attacking traces using the SECNNP models with single *convolutional* filter size, with the CNNP sub-models (reference models) and the MLS of those sub-models.

The evaluation has been done for the SECNNP models with *convolutional* filter kernel size 3 on traces belonging to multiple *plaintexts* groups, in which trace groups are randomly selected. As can be seen in figure 3, the proposed SECNNP model with 1 input trace and 3 CNNP sub-models reduces the required number of traces for attacking the sub-byte key 3 to a half compared with the reference models, in which the MLS of the reference models required 42 traces to achieve key rank 3 while the proposed SECNNP needs just 24 to achieve the same key rank (more than a half).

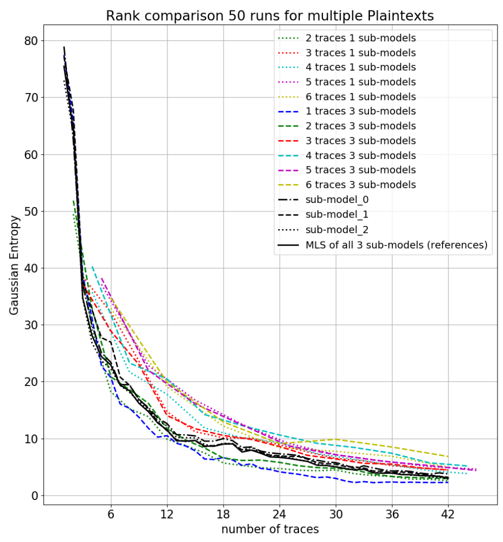

Figure 3: Key rank comparison between proposed SECNNP models with single *convolutional* filter kernel size and reference models and their MLS.

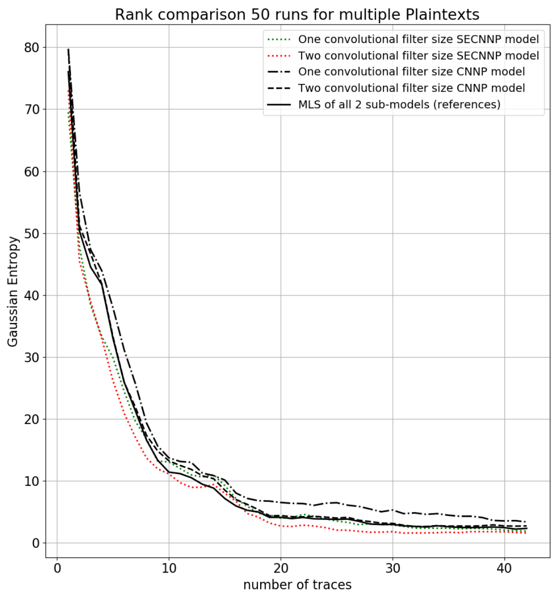

Figure 4: Key rank comparison between proposed SECNNP models with one and two *convolutional* filter kernel sizes and reference models and their MLS.

4.2 SECNNP Evaluation with Multiple *Convolutional* Filter Kernel Size

Due to the better accuracy achieved with the SECNNP model with three CNNP sub-models and a single trace input as shown in 3, in this section, we evaluate a different SECNNP model with the same structure but three 2-*convolutional*-filter-kernel-size CNNP sub-models are used. The proposed model utilizes the CNNP model with *convolutional* filter kernel sizes of 3 and 5 as detailed in (Hoang et al., 2020) as the sub-models.

An evaluation of the SECNNP model using multiple *plaintexts* is given in figure 4. The SECNNP has a single input trace, and so 42 traces are randomly used for each run and 50 runs are tested for average key rank. Even though three CNNP sub-models of each type (single and two *convolutional* filter size) are uses, we use only one CNNP model with the best accuracy among the three of each type as the reference. As can be seen in figure 4, the combined MLS (in black) of the reference models (CNNP with single and two *convolutional* filter kernel sizes) is equivalent with that of the CNNP with two *convolutional* filter sizes and SECNNP with single *convolutional* filter size (in dot green) and achieve key rank 2 after 40 traces. The proposed model (in dot red) reduces the number of required traces from 40 to a half (20 traces) in comparison to the MLS of the two best reference CNNP models.

5 DISCUSSION

The evaluation of the proposed SECNNP models revealed the following:

- Re-training from the mistakes of the classifiers (sub-models) improves the accuracy. The proposed SECNNP is re-trained from the same dataset with the probability of each hypothesis key. This training gives the model a chance to learn from its mistakes by looking at the relationship between the probabilities of the correct and incorrect hypothesis keys. Diversity among the sub-models (Wang, 2008) is considered as a factor for the improvement in key rank.
- It is not necessary to have combination of many different CNN structures to achieve higher accuracy. Instead, the same structure trained with a different number of epochs so that they show diversity in attacking traces are good to combine in an stacked ensembles model.
- A deeper structure which combines models with different input data improves the accuracy. Even though a deeper CNNP on its own does not improve the accuracy, making the model deeper together with changing the input data from traces to hypothesis key probabilities and combining models can increase the accuracy of the SECNNP model. It reduces the number of required traces

to a half, from 40 to 20 making a brute force attack possible.

- A trade-off in time and accuracy. In our experiment, the SECNNP model with a single *convolutional* kernel size requires double the training time compared with the equivalent CNNP model. Consequently, training the SECNNP model with two *convolutional* filter kernel size requires double the efforts compared with that of the single *convolutional* filter kernel size SECNNP model.

6 CONCLUSION

This paper has proposed a modified approach for building CNN-based models for profiling SCAs. A MLS layer which combines the Maximum likelihood scores of multiple models and multiple traces into the training of the model is proposed. A new network structure, which includes the input of the MLS and the probability of the hypothesis keys predicted from different traces by different CNNP models into the re-training process is introduced for improved results over the state-of-the-art. The proposed SECNNP models require half of the traces in comparison with the state-of-the-art CNNP models in attacking a masked AES implementation from the open-source ASCAD database and key recovery from just 20 traces is possible when targeting the variable key database.

While considerable work has been done in the area of ML for SCA, there is still significant scope for improvement in current approaches. The experimentation in this work is conducted on traces acquired from an embedded device with software based countermeasures. Future work could involve an analysis of how the plaintext embedding approach transfers to attack against hardware based countermeasures such as threshold or dual-rail logic approaches.

ACKNOWLEDGEMENTS

We would like to thank the people and organizations who support the research and helped improve the paper: The UK Research Institute in Secure Hardware and Embedded Systems (RISE) and The EPSRC Quantum Communications Hub (EP/T001011/1).

REFERENCES

Brier, E., Clavier, C., and Olivier, F. (2004). Correlation Power Analysis with a Leakage Model. volume 3156, pages 16–29.

Chen, Z. and Zhou, Y. (2006). Dual-Rail Random Switching Logic: A Countermeasure to Reduce Side Channel Leakage. In Goubin, L. and Matsui, M., editors, *Cryptographic Hardware and Embedded Systems - CHES 2006*, pages 242–254, Berlin, Heidelberg. Springer Berlin Heidelberg.

D, J. and Tymen, C. (2002). Multiplicative Masking and Power Analysis of AES. *IACR Cryptology ePrint Archive*, 2002:91.

Gebotys, C. H., Ho, S., and Tiu, C. C. (2005). EM Analysis of Rijndael and ECC on a Wireless Java-Based PDA. In Rao, J. R. and Sunar, B., editors, *Cryptographic Hardware and Embedded Systems – CHES 2005*, pages 250–264, Berlin, Heidelberg. Springer Berlin Heidelberg.

Gilmore, R., Hanley, N., and O'Neill, M. (2015). Neural network based attack on a masked implementation of AES. In *2015 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 106–111.

Goubin L., M. A. . (2011). *Protecting AES with Shamir's Secret Sharing Scheme.*, volume 6917 of *Lecture Notes in Computer Science*. Springer, Berlin, Heidelberg.

Hettwer, B., Gehrer, S., and Güneysu, T. (2018). Profiled Power Analysis Attacks Using Convolutional Neural Networks with Domain Knowledge. In Cid, C. and Jr., M. J. J., editors, *Selected Areas in Cryptography - SAC 2018 - 25th International Conference, Calgary, AB, Canada, August 15-17, 2018, Revised Selected Papers*, volume 11349 of *Lecture Notes in Computer Science*, pages 479–498. Springer.

Hoang, A.-T. and Fujino, T. (2014). Intra-Masking Dual-Rail Memory on LUT Implementation for SCA-Resistant AES on FPGA. *ACM Trans. Reconfigurable Technol. Syst.*, 7(2):10:1–10:19.

Hoang, A.-T., Hanley, N., and O'Neill, M. (2020). Plaintext: A Missing Feature for Enhancing the Power of Deep Learning in Side-Channel Analysis? Breaking multiple layers of side-channel countermeasures. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(4):49–85.

Kim, J., Picek, S., Heuser, A., Bhasin, S., and Hanjalic, A. (2019). Make Some Noise. Unleashing the Power of Convolutional Neural Networks for Profiled Side-channel Analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2019(3):148–179.

Kocher, P. C. (1996). Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In *Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '96, pages 104–113, London, UK, UK. Springer-Verlag.

Kocher, P. C., Jaffe, J., and Jun, B. (1999). Differential Power Analysis. In *Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '99, pages 388–397, Berlin, Heidelberg. Springer-Verlag.

Lu, X., Zhang, C., Cao, P., Gu, D., and Lu, H. (2021). Pay Attention to Raw Traces: A Deep Learning Architecture for End-to-End Profiling Attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021(3):235–274.

Maghrebi, H. (2019). Deep Learning based Side Channel Attacks in Practice. *IACR Cryptology ePrint Archive*, 2019:578.

Maghrebi, H., Portigliatti, T., and Prouff, E. (2016). Breaking Cryptographic Implementations Using Deep Learning Techniques. *IACR Cryptology ePrint Archive*, 2016:921.

Markowitch, O., Medeiros, S., Bontempi, G., and Lerman, L. (2013). A Machine Learning Approach Against a Masked AES. volume 5.

Martinasek, Z., Dzurenda, P., and Malina, L. (2016). Profiling power analysis attack based on MLP in DPA contest V4.2. In *2016 39th International Conference on Telecommunications and Signal Processing (TSP)*, pages 223–226.

Nassar, M., Souissi, Y., Guilley, S., and Danger, J.-L. (2012). RSM: A Small and Fast Countermeasure for AES, Secure Against 1st and 2Nd-order Zero-offset SCAs. In *Proceedings of the Conference on Design, Automation and Test in Europe*, DATE '12, pages 1173–1178, San Jose, CA, USA. EDA Consortium.

Perin, G., Chmielewski, Å., and Picek, S. (2020). Strength in Numbers: Improving Generalization with Ensembles in Machine Learning-based Profiled Side-channel Analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(4):337–364.

Picek, S., Samiotis, I. P., Kim, J., Heuser, A., Bhasin, S., and Legay, A. (2018). On the Performance of Convolutional Neural Networks for Side-Channel Analysis. In *SPACE*.

Popp, T. and Mangard, S. (2005). Masked Dual-Rail Precharge Logic: DPA-Resistance Without Routing Constraints. In Rao, J. R. and Sunar, B., editors, *Cryptographic Hardware and Embedded Systems – CHES 2005*, pages 172–186, Berlin, Heidelberg. Springer Berlin Heidelberg.

Prouff, E., Strullu, R., Benadjila, R., Cagli, E., and Dumas, C. (2018). Study of Deep Learning Techniques for Side-Channel Analysis and Introduction to ASCAD Database. Cryptology ePrint Archive, Report 2018/053. https://eprint.iacr.org/2018/053.

Reparaz, O., Bilgin, B., Nikova, S., Gierlichs, B., and Verbauwhede, I. (2015). Consolidating Masking Schemes. In *CRYPTO*.

Standaert, F.-X., Malkin, T. G., and Yung, M. (2009). A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks. In Joux, A., editor, *Advances in Cryptology - EUROCRYPT 2009*, pages 443–461, Berlin, Heidelberg. Springer Berlin Heidelberg.

Timon, B. (2019). Non-Profiled Deep Learning-based Side-Channel attacks with Sensitivity Analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2019(2):107–131.

Tiri, K., Akmal, M., and Verbauwhede, I. (2002). A dynamic and differential CMOS logic with signal independent power consumption to withstand differential power analysis on SmartCards. pages 403 – 406.

Wang, W. (2008). Some fundamental issues in ensemble methods. In *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pages 2243–2250.

Weissbart, L., Picek, S., and Batina, L. (2019). One trace is all it takes: Machine Learning-based Side-channel Attack on EdDSA. Cryptology ePrint Archive, Report 2019/358. https://eprint.iacr.org/2019/358.