This paper has been accepted at 58th Asilomar Conference on Signals, Systems, and Computers, 2024.

DL-SCADS: Deep Learning-Based Post-Silicon Side-Channel Analysis Using Decomposed Signal

Dipayan Saha and Farimah Farahmandi
Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL, USA
dsaha@ufl.edu, farimah@ece.ufl.edu

***Abstract*—Side-channel analysis (SCA) does not aim at the algorithm's weaknesses but rather its implementations. The rise of machine learning (ML) and deep learning (DL) is giving adversaries advanced capabilities to perform stealthy attacks. In this paper, we propose DL-SCADS, a DL-based approach along with signal decomposition techniques to leverage the power of secret key extraction from post-silicon EM/power side-channel traces. We integrate previously proven effective ideas of model ensembling and automated hyperparameter tuning with signal decomposition to develop an efficient and robust side-channel attack. Extensive experiments are performed on Advanced Encryption Standard (AES) and Post-Quantum Cryptography (PQC) to demonstrate the efficacy of our approach. The evaluation of the performance of the side-channel attack employing various decomposition techniques and comparison with the proposed approach in a range of datasets are also tabulated.**

***Index Terms*—Side-Channel Analysis, Deep Learning, Signal Decomposition, Empirical Mode Decomposition**

I. Introduction

The rapid growth of embedded devices, driven by the proliferation of IoT technologies, has been paralleled by an alarming increase in hardware security threats, exposing these systems to critical attacks such as side-channel analysis (SCA) [1]. Side-channel attacks make use of the physical environment (software or hardware) of a cryptosystem (timing, power, electromagnetic radiations, etc.) to retrieve the secret information of microelectronics. Since their discovery in the 1990s [2], side-channel attacks have been the focus of extensive research. However, deep learning (DL)-driven side-channel attacks [3] have gained popularity due to their stealthy and highly effective nature, making them a significant threat. The major benefit of such DL-based techniques is the ability of these approaches to break through even in strong masking and hiding countermeasures. Despite their success, DL-based approaches in side-channel analysis have room for improvement, particularly in enhancing the feature extraction step for more efficient attacks.

Signal decomposition is another powerful processing technique that unleashes the intrinsic features unseen in raw signals. Very few works in the domain of side-channel analysis have adopted signal decomposition for key recovery. Moreover, existing related works based on signal decomposition used it to denoise signals as a pre-processing step [4]. This leaves room for investigating the potentiality of the feature extraction of signal decomposition techniques for the SCA task. To the best of our knowledge, the proposed DL-SCADS is the first work that utilizes signal decomposition as a feature extraction step for training neural networks to execute a side-channel attack.

A. Specific Contributions

The specific contributions of this work are multifold:

- Implementation of the signal decomposition technique along with the deep learning approach to take advantage of the power of secret key extraction from power/EM side-channel traces in the presence of different masking and hiding countermeasures.
- Integration of model ensembling and automated hyperparameter tuning with the idea of signal decomposition to develop an efficient and robust side-channel attack on Advanced Encryption Standard (AES) and Post-Quantum Cryptography (PQC).
- Evaluation of side-channel attack performance employing various decomposition techniques and comparison with the proposed approach in various datasets.

B. Related Works

In the literature, research on profiling side-channel attacks is increasing due to the advantages these techniques have. However, effective hyperparameter tuning remains a challenging aspect. Wu *et al.* [5] demonstrated the effectiveness of automated hyperparameter tuning through Bayesian optimization, enabling more successful side-channel attacks. Similarly, Zaid *et al.* [6] explored efficient, shallow convolutional neural network (CNN) architectures using techniques such as weight and gradient visualization and heatmaps. Van *et al.* [7] focused on model compression through mimicking strategies, while Perin *et al.* [8] advocated for ensemble models to enhance attack generalization. The effectiveness of transferring 1-D side-channel traces to 2-D images was also investigated in [9].

The paper is organized as follows: Section II outlines the proposed methodology. Section III details the experimental setup and results. Finally, Section IV provides the conclusion.

II. Proposed Methodology

Empirical Mode Decomposition (EMD) [10] is an observation-driven method that decomposes a signal into intrinsic mode functions (IMFs) by analyzing local oscillatory behavior. In this study, we employ EMD as a feature extraction technique for deep learning-based side-channel attacks (SCA). Figure 1 provides an overview of the proposed approach.

A. Algorithms

The proposed methodology, like other profiling side-channel attacks, comprises two stages; profiling and attack. The Algorithm 1 represents the profiling stage and Algorithm 2 outlines

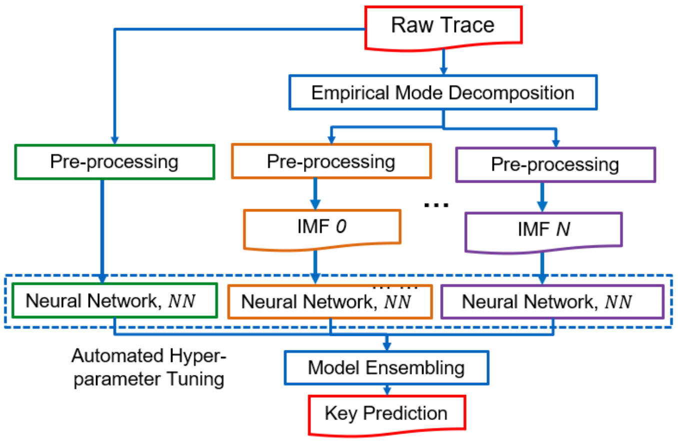


Fig. 1: Overview of proposed methodology DL-SCADS.

Algorithm 1 Profiling Stage of DL-SCADS

```
1: Input: Raw ($t_r^i \in \mathcal{T}_r$, $i = 1, 2, 3, ..., n$), plaintext ($p^i \in \mathcal{P}$, $i = 1, 2, 3, ..., n$), key ($k^i \in \mathcal{K}$, $i = 1, 2, 3, ..., n$)
2: Output: Trained $\mathcal{NN}$
3: Apply signal decomposition on raw trace T$_r$ and derive $\mathcal{T}_r^j$, $j = 0, 1, 2, ..., N$
4: Calculate intermediate value $v^i$ from ($p^i$,$k^i$) and set as label ($l^i \in \mathcal{L}$, $i = 1, 2, 3, ..., n$)
5: Take raw trace and Construct the dataset, $\mathcal{D}$ = ($\mathcal{T}_r$,$\mathcal{L}$)
6: Split the dataset $\mathcal{D}$ into $\mathcal{D}_{train}$, $\mathcal{D}_{val}$, $\mathcal{D}_{test}$
7: Automatic Hyperparameter Tuning:
8: Choose network architecture type based on nature of dataset
9: Set objective function for Bayesian Optimization
10: Set hyperparameter search space
11: Select best combination of optimization and model hyperparameter
12: Neural Network Training:
13: Initialize the weights of neural network
14: Train $\mathcal{NN}$ on ($\mathcal{T}_{r,train}$,$\mathcal{L}_{train}$)
15: return Trained $\mathcal{NN}$
16: for $j$=0:$N$ do
17:     construct dataset, $\mathcal{D}^j$ = ($\mathcal{T}_r^j$,$\mathcal{L}$)
18:     repeat Lines 6-15 on dataset $\mathcal{D}^j$
19: end for
```

the attack stage. In the profiling stage, the first step is to apply a signal decomposition technique and split the raw trace into multiple decomposed signals / intrinsic mode functions (IMFs) (**Line 3**). Before applying the decomposition algorithm, preprocessing steps such as normalization and down-sampling may be required. The necessity of these pre-processing steps depends on the nature of the raw trace. In the next step, intermediate values v^i must be calculated from available plaintext p^i and the correct key value k^i. These values are labeled to train a neural network (**Line 4**). Since model performance varies considerably depending on the hyperparameters, it is critical to select appropriate hyperparameter values. As a result, after the training, validation, and test datasets have been constructed, a Bayesian optimization (BO)-based automated hyperparameter technique [11] is employed to produce the best possible hyperparameters. To discover a suitable neural network architecture and train the network efficiently, this technique checks both the optimizer and the model hyperparameters (**Lines 8-11**). Later, with the selected hyperparameters, a neural network model is trained for the dataset of raw traces. The same process described in **Lines 6-15** is then followed for the decomposed signals.

Algorithm 2 Attack Stage of DL-SCADS

```
1: Input: Raw/decomposed trace $\mathcal{D}_{test}$, plaintext $\mathcal{P}_{test}$, key $\mathcal{K}_{test}$, Trained $\mathcal{NN}_e$, $e = 0, 1, 2, ..., N$
2: Output: key rank, guessing entropy (GE)
3: for $i = 1 : N_{exp}$ do
4:     for $e = 1 : N$ do
5:         Select $N_{te}$ traces from $\mathcal{D}_{test,e}$ and shuffle
6:         for $n = 1 : N_{te}$ do
7:             Predict probability $p_e^{t^n,c}$ of class $c$ for trace $t_e^n$
8:             using $\mathcal{NN}_e$
9:             Calculate probability $p_e^{t^n,k_g}$ for key $k_g \in [0, 255]$
10:        end for
11:        Calculate aggregated probability for $k_g$ using $\mathcal{NN}_e$
12:                $S_e(k_g) = \sum_{m=1}^{n} p_e^{t^m,k_g}$
13:    end for
14:    Calculate ensembled probability for key $k_g$
15:            $S(k_g) = \sum_{e=1}^{N} S_e(k_g)$
16:    Calculate rank for secret key $k_{test}$ based on $S(k_g)$
17: end for
18: Estimate guessing entropy
```

In this work, the attack stage, described in Algorithm 2, is performed in N_{exp} number of experiments. In each experiment for evaluation, the N_{tr} number of traces is chosen from the test set $\mathcal{T}_{test}$ and shuffled (**Line 3-5**). For each trace, the probability of class prediction is measured for each of the previously trained neural networks (**Line 6-8**). From each of the intermediate values predicted, the corresponding key value is calculated using the plaintext value (**Line 9**). Later, the prediction probabilities of all possible keys for current $\mathcal{NN}_e$ are calculated (**Line 11-12**). As written in **Line 14-15**, these probabilities are summed for all $\mathcal{NN}_e$ and then a key ranking algorithm is applied to these scores to determine the rank of the true secret key k_{test}, identifying the key with the highest probability as the most likely candidate. (**Line 16**).

B. *Neural Network Architecture Design*

In **Line 8** of Algorithm 1, the presence of a hiding countermeasure in the AES implementation determines the type of network architecture. The hiding countermeasure introduces random delay and creates misalignment in the trace, making the side-channel attack difficult to execute. CNNs are expected to perform better in these situations because they have a unique capacity to detect features of translational invariance, eliminating the need to align misaligned traces [12]. As a result, in this work, CNN is chosen as the network architecture if the implementation includes a hiding countermeasure. MLP is another key candidate, especially without a hiding countermeasure.

After selecting the architecture, Bayesian Optimization (BO) is applied for automated hyperparameter tuning [11]. BO uses a probabilistic model, represented by a Gaussian process in this work, to iteratively explore and exploit the search space, maximizing the target function with minimal iterations. It determines the optimal combination of both optimization and model-specific hyperparameters, which influence the architecture's structure. In the case of MLP, the number of fully connected layers (FC), the percentage of neurons in the initial layer, and the percentage of neuron shrinkage are considered model hyperparameters. However, the number of convolutional layers, the number of filters, the stride value in the average pooling, the percentage of neurons in the early FC layers, and the percentage of neuron shrinkage in the later FC layers are considered model hyperparameters for CNN.

After each layer, batch normalization is utilized to speed up the learning process and add regularization. When necessary, the dropout layer is also used to prevent overfitting. The task of attacking AES is formulated as a 256-class classification problem, since the proposed method is based on an 8-bit intermediate value. Each architecture includes a Softmax function for 256 classes to predict the probability of the class. In another task, an attack is performed on the hardware implementation of Saber targeting the vector multiplication. Since we target polynomial 4-bit sub-key through the attack, this is a 16-class classification problem. After the declaration of the function to be maximized, a search space for these hyperparameters mentioned above needs to be declared by specifying the corresponding bounds. With 10 steps of random exploration, 100 iterations are used to maximize the class accuracy.

III. Experiments and Results

In this work, exhaustive experiments are carried out on both AES and PQC (Saber [13]) algorithms to evaluate the efficacy of the proposed methodology DL-SCADS.

A. Datasets and Evaluation Metrics

This work utilizes four publicly available post-silicon trace datasets for the attack on AES, summarized in Table I. For the attack on Saber, power traces collected by [14] are used. For the attack on Saber, training is carried out on 16k traces, and performance is evaluated on a separate set of 16k traces.

In this work, the performance of the proposed DL-SCADS technique and other comparison methods is estimated with respect to the guessing entropy (GE), the success rate (SR), and the computational complexity. GE indicates the average rank of the correct key in all possible combinations of key values. The number of traces required for GE to reach less than 1, denoted by $n_{GE<1}$, is considered the key performance metric in this work. For GE calculation, we perform the attack 100 times with randomly selected sub-samples of the test to find the average number of traces to achieve. However, the success rate is the ratio of successful attempts to total attempts to recover the key. Computational complexity is measured through the total number of parameters of the neural network architectures used in the experiments.

TABLE I: Description of the datasets used for experiments.

Dataset	Side-Channel	Platform	Counter-measure	Feature size	# of traces
DPA.V4 [15]	Power	ATmega163 (Software)	Masking	4000	5,000
DPA. V2 [16]	Power	SASEBO GII(FPGA)	NA	3253	100,000
AES_RD [17]	Power	AVR MCU (Software)	Hiding	3500	50,000
ASCAD [18]	EM	ATmega 8515 (Software)	Masking	700	60,000

TABLE II: Brief overview of network architectures used in DPA.V2, DPA.V4 and ASCAD datasets

Dataset	**Input Features**	**# layers**	**Initial %**	**Neural shrink %**
DPA.V4	Raw,IMF0-3	{4,4,4, 5, 5}	{0.77,.063,0.188, 0.226,0.189}	{0.74, 0.36, 0.90, 0.44, 0.40}
DPA.V2	Raw,IMF0-2	{4,6, 5, 5}	{0.40,.25, 0.16, 0.10}	{0.63,0.77, 0.78,0.80}
ASCAD	Raw,IMF2	{6,4}	{0.52,0.445}	{0.877,0.708}

B. Comparison Methods

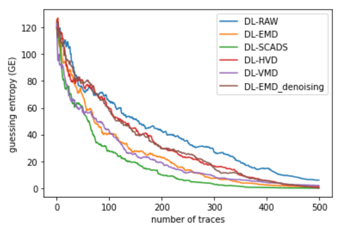
The performance of the proposed methodology is compared with that of some different strategies. A brief discussion of the comparison methods goes as follows:

- **DL-Raw**: Only raw traces are used in this approach to train a neural network requiring no model ensembling.
- **DL-EMD_Denoising**: In this approach, EMD is used to denoise the raw trace, and then the denoised trace is fed into the neural network architecture.
- **DL-HVD**: In DL-HVD, the IMFs generated by Hilbert Vibration Decomposition (HVD) [19] are used to train a deep neural network to perform a profiling attack.
- **DL-VMD**: DL-VMD uses the modes generated by Variation Mode Decomposition (VMD) [20] to execute the profiling attack through deep learning.

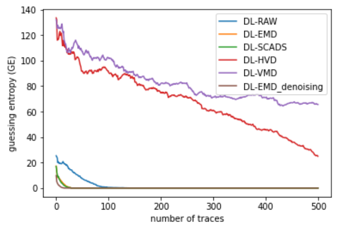
C. Model Setup

As described in Section II-B, the neural network architectures are tailored to the datasets. For DPA.V4, DPA.V2, and ASCAD datasets, MLP networks are used, while CNNs are implemented for the AES RD dataset. Hyperparameters were optimized using a defined search space: number of FC layers (2 to 6), initial percentage of neurons (0.1 to 1), neuron shrink percentage (0.1 to 1), learning rates (10^{-3} to 10^{-5}), batch sizes (32, 64, 128), dropout rates (0.1 to 0.5) and activation functions (ReLU and SeLU). Table II provides a brief description of the neural networks used in terms of model hyperparameters tuned by the algorithm described before. Training employs minibatch optimization using the RMSprop algorithm to minimize categorical cross-entropy loss

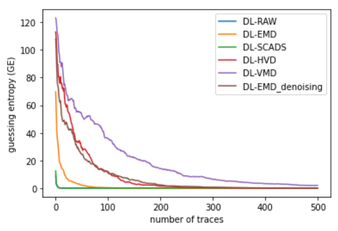
For DPA.V4 and AES_RD, raw traces are down-sampled 4 and 3 times, respectively. ReLU is used as the activation function across all datasets, except ASCAD, which employs SeLU. For the AES_RD dataset, CNNs have 3 convolutional layers for raw traces and 4 for IMF0 traces, each followed



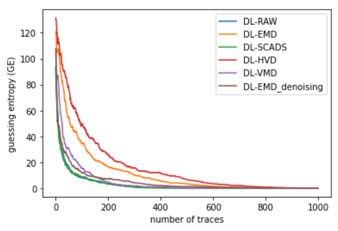
(a) $DPA.V4$



(b) $DPA.V2$



(c) AES_RD



(d) $ASCAD$

Fig. 2: Performance analysis on DPA.V4, DPA.V2, AES_RD, and ASCAD datasets.

TABLE III: Performance comparison of different methods in terms of $n_{GE<1}$and computational complexity (in million)

Dataset	Metrics	DPA.V4	DPA.V2	AES_RD	ASCAD
DL-Raw	$n_{GE<1}$	>500	88	4	301
	Comp.(M)	1.4	6.01	1.02	0.69
DL-EMD	$n_{GE<1}$	470	23	66	639
	Comp.(M)	1.45	4.39	0.93	0.4
Denoising	$n_{GE<1}$	478	9	272	557
	Comp.(M)	0.16	3.49	1.39	0.39
DL-HVD	$n_{GE<1}$	490	>500	228	766
	Comp.(M)	4.15	2.67	6.96	2.21
DL-VMD	n_GE<1	>500	>500	>500	343
	Comp.(M)	4.15	2.17	6.25	1.78
DL-SCADS	$n_{GE<1}$	358	21	5	285
	Comp.(M)	2.85	10.4	2.95	1.08

by batch normalization, dropout, and average pooling. Filters used are [32, 32, 64] for raw traces and [16, 32, 64, 128] for IMF0, with pooling steps of 3 and 5, respectively. Fully connected layers include 2 for raw traces and 3 for IMF0, each followed by batch normalization and dropout. On the other hand, for the attack on Saber, only MLPs are considered and the hyperparameters are searched in the same way as described before.

D. Results

1) DL-SCADS on AES: Figure 2 compares the GE performance of the proposed DL-SCADS method on various datasets with other approaches. Table III summarizes $n_{GE<1}$ and computational complexity for all methods. For the DPA.V4 dataset (Figure 2a), DL-SCADS achieves the best performance, with $n_{GE<1} = 358$, outperforming other methods. DL-Raw shows the highest GE values, failing to reach $GE < 1$ within 500 traces, while DL-EMD improves performance with $n_{GE<1} = 470$. Combining raw traces with EMD in DL-SCADS enhances attack performance by adding multidimensional features. DL-EMD_Denoising performs comparable to $n_{GE<1} = 478$, though with a lower computational cost. For the DPA.V2 dataset (Figure 2b), DL-SCADS demonstrates superior performance with $n_{GE<1} = 21$, while DL-EMD achieves $n_{GE<1} = 23$. DL-EMD_Denoising achieves the best result at $n_{GE<1} = 9$. In contrast, DL-HVD and DL-VMD show limited performance, with DL-VMD failing to extract the correct key effectively.

In the AES_RD dataset (Figure 2c), despite the misalignment of the trace, DL-SCADS and DL-Raw recover the key with few traces, although DL-Raw slightly outperforms DL-SCADS

TABLE IV: Performance comparison of existing approaches in terms of $n_{GE<1}$.

Approach	DPA.V2	AES_RD	ASCAD
Hettwer *et al.* [9]	>1500	-	$n_{GE<2}$ = 275
Van *et al.* [7]	-	>100	$\approx$ 8000
Prouff *et al.* [18]	-	-	1146
Picek *et al.* [21]	$\approx$ 3000	-	-
Zaid *et al.* [6]	-	5	191
DL-SCADS	21	5	$n_{GE<1}$ = 285 $n_{GE<2}$ = 260

in $n_{GE<1}$. DL-EMD performance is degraded due to downsampling during IMF construction, while DL-EMD_Denoising and DL-VMD underperform. For the ASCAD dataset (Figure 2d), all methods converge within 1000 traces, with DL-SCADS achieving the lowest $n_{GE<1}$, reducing it by 16 compared to DL-Raw. DL-VMD performs better on ASCAD than on other datasets. In general, DL-SCADS demonstrates the most consistent performance across datasets.

Table IV compares DL-SCADS with recent work in which experimental setups are comparable. DL-SCADS excels for DPA.V2 and matches the best-performing method for AES_RD. For ASCAD, it ranks as the second best method. Due to differences in experimental setups, DPA.V4 results are excluded. In particular, this study assumes an unknown mask for DPA.V4, unlike many works that use known mask values.

2) DL-SCADS on PQC: Figure 3 illustrates the performance comparison of the proposed approach in Saber with varying input features. The lowest success rates are observed when using only raw traces. However, incorporating IMF0 with the raw traces significantly improves performance, increasing single-trace attack success rates by 15.66% and multiple-trace attack success rates by 76.59%. IMF0, the first decomposed signal generated by EMD, introduces additional observations, enhancing the effectiveness of profiled attacks. The highest attack performance is achieved when raw traces are combined with three decomposed signals (IMF0 to IMF2), resulting in a success rate of approximately 77.03% for multiple-trace attacks. Despite the low signal-to-noise ratio (SNR) in this setup, the computational overhead remains manageable and is well justified by the substantial performance gains.

Figure 4 summarizes the performance of the side-channel attack in the Saber algorithm using different signal decomposition techniques for single- and multiple-trace attacks. From the figure, it can be seen that the DL-SCADS approach out-

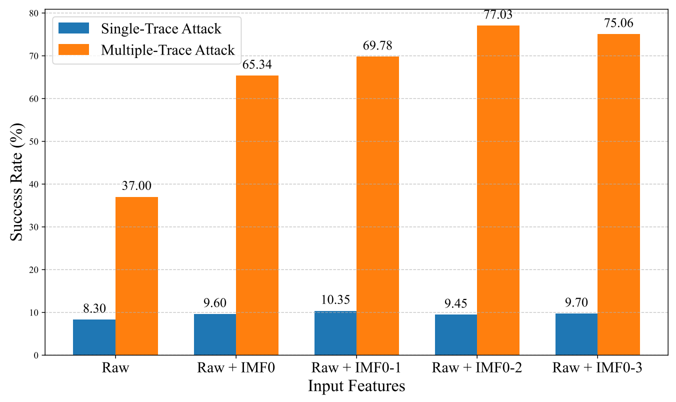


Fig. 3: Performance of DL-SCADS on Saber using different combinations of features for single- and multiple-trace attack.

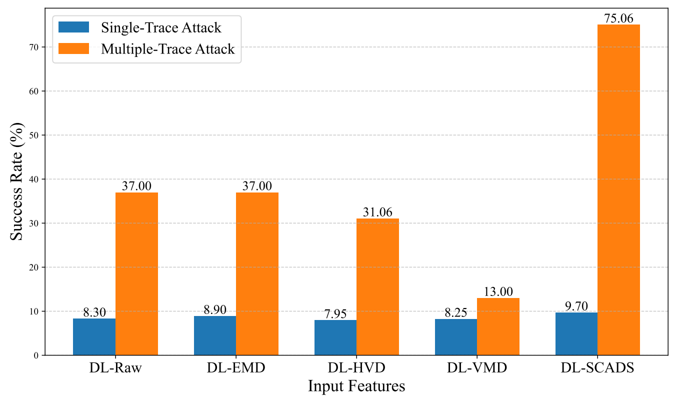


Fig. 4: Performance of different signal decomposition techniques on Saber for single-trace and multiple-trace attack.

performs the others. The SR attained by DL-EMD is equal to the case where only raw traces are used. Interestingly, when both empirical modes generated by EMD and raw traces are ensembled simultaneously, the highest success rates are attained in single-trace and multiple-trace attacks. The EMD features and raw traces concentrate on completely different target subkey classes. Due to these unseen observations, when these discriminative features are merged, the number of misclassifications decreases.

IV. Conclusion

To the best of our knowledge, this is the first work to combine the feature extraction capability of signal decomposition and DL in the SCA domain. The work proposes DL-SCADS, which uses EMD to decompose signals and automated hyperparameter tuning to find the best possible network architecture with minimum effort. This work performs exhaustive experiments on both AES and PQC algorithms and compares the experimental results with existing recent works and other different decomposition-based approaches. The experimental results prove the argument that signal decomposition has the potential to help DL approaches in performing a superior side-channel attack.

V. Acknowledgment

This work was supported by the U.S. National Science Foundation (NSF) through the CAREER Award under Grant 2339971.

References

[1] S. Picek, G. Perin, L. Mariot, L. Wu, and L. Batina, "Sok: Deep learning-based physical side-channel analysis," *ACM Computing Surveys*, vol. 55, no. 11, pp. 1–35, 2023.

[2] P. C. Kocher, "Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems," in *Annual Int. Cryptology Conference*, pp. 104–113, Springer, 1996.

[3] H. Maghrebi, T. Portigliatti, and E. Prouff, "Breaking cryptographic implementations using deep learning techniques," in *Security, Privacy, and Applied Cryptography Engineering: 6th International Conference, SPACE 2016, Hyderabad, India, December 14-18, 2016, Proceedings 6*, pp. 3–26, Springer, 2016.

[4] S. Zhang *et al.*, "A highly effective data preprocessing in side-channel attack using empirical mode decomposition," *Security and Communication Networks*, vol. 2019, no. 1, p. 6124165, 2019.

[5] L. Wu, G. Perin, and S. Picek, "I choose you: Automated hyperparameter tuning for deep learning-based side-channel analysis," *IEEE Transactions on Emerging Topics in Computing*, 2022.

[6] G. Zaid, L. Bossuet, A. Habrard, and A. Venelli, "Methodology for efficient cnn architectures in profiling attacks," *IACR Trans. Cryptographic Hardware and Embedded Systems*, vol. 2020, no. 1, pp. 1–36, 2020.

[7] van der Valk *et al.*, "Learning from a big brother-mimicking neural networks in profiled side-channel analysis," in *57th ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6, IEEE, 2020.

[8] G. Perin, Ł. Chmielewski, and S. Picek, "Strength in numbers: Improving generalization with ensembles in machine learning-based profiled side-channel analysis," *IACR Trans. Cryptographic Hardware and Embedded Systems*, pp. 337–364, 2020.

[9] B. Hettwer, T. Horn, S. Gehrer, and T. Güneysu, "Encoding power traces as images for efficient side-channel analysis," in *IEEE Int. Symposium on Hardware Oriented Security and Trust (HOST)*, pp. 46–56, IEEE, 2020.

[10] N. E. Huang *et al.*, "The empirical mode decomposition and the hilbert spectrum for nonlinear and non-stationary time series analysis," *in Proc. the Royal Society of London. Series A: mathematical, physical and engineering sciences*, vol. 454, no. 1971, pp. 903–995, 1998.

[11] F. Nogueira, "Bayesian Optimization: Open source constrained global optimization tool for Python," 2014.

[12] R. Gens and P. M. Domingos, "Deep symmetry networks," *Advances in neural information processing systems*, vol. 27, pp. 2537–2545, 2014.

[13] D'Anvers *et al.*, "Saber: Module-lwr based key exchange, cpa-secure encryption and cca-secure kem," in *Progress in Cryptology–AFRICACRYPT 2018: 10th International Conference on Cryptology in Africa, Marrakesh, Morocco, May 7–9, 2018, Proceedings 10*, pp. 282–305, Springer, 2018.

[14] J. Park, N. N. Anandakumar, D. Saha, D. Mehta, N. Pundir, F. Rahman, F. Farahmandi, and M. M. Tehranipoor, "Pqc-sep: Power side-channel evaluation platform for post-quantum cryptography algorithms.," *IACR Cryptol. ePrint Arch.*, vol. 2022, p. 527, 2022.

[15] S. Bhasin *et al.*, "Analysis and improvements of the dpa contest v4 implementation," in *Int. Conf. Security, Privacy, and Applied Cryptography Engineering*, pp. 201–218, Springer, 2014.

[16] F.-X. Standaert, P. Bulens, G. de Meulenaer, and N. Veyrat-Charvillon, "Improving the rules of the dpa contest.," *IACR Cryptol. ePrint Arch.*, vol. 2008, p. 517, 2008.

[17] J.-S. Coron and I. Kizhvatov, "An efficient method for random delay generation in embedded software," in *Int. Workshop on Cryptographic Hardware and Embedded Systems*, pp. 156–170, Springer, 2009.

[18] E. Prouff, R. Strullu, R. Benadjila, E. Cagli, and C. Dumas, "Study of deep learning techniques for side-channel analysis and introduction to ascad database," *Cryptology ePrint Archive*, 2018.

[19] M. Feldman, "Time-varying vibration decomposition and analysis based on the hilbert transform," *Journal of Sound and Vibration*, vol. 295, no. 3-5, pp. 518–530, 2006.

[20] K. Dragomiretskiy and D. Zosso, "Variational mode decomposition," *IEEE Trans. signal processing*, vol. 62, no. 3, pp. 531–544, 2013.

[21] S. Picek, I. P. Samiotis, J. Kim, A. Heuser, S. Bhasin, and A. Legay, "On the performance of convolutional neural networks for side-channel analysis," in *Int. Conf. Security, Privacy, and Applied Cryptography Engineering*, pp. 157–176, Springer, 2018.