Inter-Architecture Portability of Artificial Neural Networks and Side Channel Attacks

Manoj Gopale, Gregory Ditzler, Roman Lysecky, and Janet Roveda
University of Arizona
Tucson, AZ, USA
[manojgopale,ditzler,rlysecky,meilingw]@email.arizona.edu

ABSTRACT

Side-channel attacks (SCA) have been studied for several decades, which resulted in many techniques that use statistical models to extract system information from side channels. More recently, machine learning has shown significant promise to advance the ability for SCAs to expose vulnerabilities. Artificial neural networks (ANN) can effectively learn nonlinear relationships between features within a side channel. In this paper, we propose a multi-architecture data aggregation technique to profile power traces for a system with an embedded processor that is based on three types of deep NNs, namely, multi-layer perceptrons (MLP), convolutional neural networks (CNN), and recurrent neural networks (RNN). This is one of the first works to explore the inter-architecture portability of NNs and SCAs. We demonstrate the robustness of the ANNs performing power-based SCAs on multiple architecture configurations with different architectural features, such as L1/L2 caches' size and associativity, and system memory size. We provide a comprehensive set of benchmarks to demonstrate that architecturally identical devices are not essential for profile-based SCAs.

CCS CONCEPTS

• **Security and privacy → Security in hardware**; **Embedded systems security**.

KEYWORDS

Side channel attack, Portability embedded security.

ACM Reference Format:
Manoj Gopale, Gregory Ditzler, Roman Lysecky, and Janet Roveda. 2022. Inter-Architecture Portability of Artificial Neural Networks and Side Channel Attacks. In *Proceedings of the Great Lakes Symposium on VLSI 2022 (GLSVLSI '22), June 6–8, 2022, Irvine, CA, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3526241.3530382

1 INTRODUCTION

A critical security requirement for embedded systems is protection of sensitive information (e.g., cryptographic keys). Unfortunately, many embedded systems have constrained resources that limit the implementation of security measures. Side-channel attacks (SCAs) are a significant threat to embedded devices. Embedded devices can leak sensitive information when operations are performed on sensitive data. Many SCAs that can statistically infer cryptographic keys from embedded devices by using power side, cache access pattern, and electromagnetic side channels.

SCAs are categorized as profiled and non-profiled attacks. In a non-profiled attack, an adversary creates a side-channel leakage model based on knowledge of the target system. Once an adversary has the model, they can retrieve the secret key by using correlation between the side channel from the target and leakage models. A non-profiled leakage model's efficiency depends on multiple factors such as device architecture, software implementation, and manufacturing technology characteristics. Profiled SCAs, such as template attacks [3], use the complete side channel leakage for characterizing the target device. For every possible key, an adversary can profile and build a leakage template for the target device. Profiled SCAs are more robust to system noise and countermeasures [3].

In this work, we explore the portability of neural networks trained on one device and then evaluated on a different target device. *Inter-device portability* is the ability of an ANN trained on one device to attack a different instance of the device; however, the internal architecture of the two devices is identical. For inter-device portability, the differences between the training and attack devices primarily come from manufacturing variations. *Inter-architecture portability* is the ability for an ANN trained on a device with one internal architecture (i.e., processor, cache, etc.) to attack a device with a different internal architecture. Internal variations make portability analysis more challenging as they result in significant changes in the power traces compared to identical internal devices as seen in inter-device portability.

Recent work [5] explored the inter-device portability of ANNs for SCAs across physical instances of the same device. They showed that manufacturing variations and environmental conditions can significantly impact the inter-device performance. They improved the inter-device performance by merging the data from devices prior to training. Notably, this portability addresses the physical differences between devices with the same processor architecture, but not the inter-architecture portability across devices with different processor architecture configurations, which is explored herein.

Inter-architecture portability opens a new paradigm in profile-based SCAs. Research has shown that each device is vulnerable if an architecturally identical device is used for profiling SCAs. Specifically, for a power-based SCA, an adversary collects power traces from each architecturally different target devices then tunes an ANN for each target device. If the ANNs have inter-architecture portability, then an adversary only needs to collect and aggregate power traces from a few target devices to tune a ANN. Inter-architecture portability, thus, reduces the time complexity of attacks

This work is licensed under a Creative Commons Attribution International 4.0 License.

GLSVLSI '22, June 6–8, 2022, Irvine, CA, USA.
© 2022 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9322-5/22/06.
https://doi.org/10.1145/3526241.3530382

on architecturally different target devices. Subsequently, architecturally different devices are now vulnerable even if the adversary does not have an architecturally identical device for profiling.

This paper presents an approach to develop an inter-architecture portable ANN for a profiled power-based SCA by aggregating data from multiple processor architecture configurations. In this paper, we present: (1) the first work to comprehensively explore and provide empirical evidence for inter-architecture portability and ANN for power-based SCAs; (2) an ANN-based approach that demonstrates the need to aggregate data from multiple architecturally different configurations to improve the inter-architecture performance of ANNs; (3) MLPs, CNNs, and RNNs that generalize over 12 target processor configurations, considered herein, to give high inter-architecture performance. Our methods are thoroughly benchmarked on a large volume of data generated using the GEM5 and McPAT software. We show that a thoughtful, multi-architecture aggregate training dataset can significantly improve the performance across many configurations without the need to train on all possible configurations, which would be infeasible.

2 RELATED WORK

ANNs have shown considerable improvement over shallow ML techniques for SCAs [1]. ANNs can learn nonlinear and complex dependencies between input features and outputs in SCAs, which improves the attack efficiency [10]. However, these previous studies have only been applied to simple 8-bit microcontrollers. In contrast, this work targets a more complex 32-bit ARM processor, including multiple cache levels and main memory, which is more applicable to real-world systems.

Research has shown that ML approaches outperform template attacks for high dimensional data [7]. Principal component analysis (PCA) has been used to represent features from the power trace for single S-box operations to reduce the dimensionality of the input layer [9]. These dimensionality reduction approaches seek to ensure the reduced input features show maximum variability. In [9], the effectiveness of PCA and ANNs was evaluated using guessing entropy, which indicated a potential adversary would need on an average of four guesses to predict the correct key. In contrast, the approach presented herein operates on the entire power trace of a full AES encryption without reducing the dimensionality. Our ANNs need one guess (on average) to predict the correct key if we were to compare our results using guessing entropy.

Several recent works focused on analyzing SCAs and the inter-device portability of NNs across devices [5], specifically analyzing how a NN constructed from power traces of one device can be used to attack a different manufactured device with identical architectural parameters. These approaches show that device variability can degrade efficiency of NNs, which can be overcome by aggregating the power traces from multiple devices during training. However, these approaches focused on devices that both have the same internal processor architecture configuration. Thus, the portability of NNs studied in these previous efforts only addressed environmental and manufacturing variations. In contrast, this paper analyzes SCAs and inter-architectural portability of the NNs.Specifically, we seek to understand the portability of an adversary's NN to architecturally different devices from which it was trained.

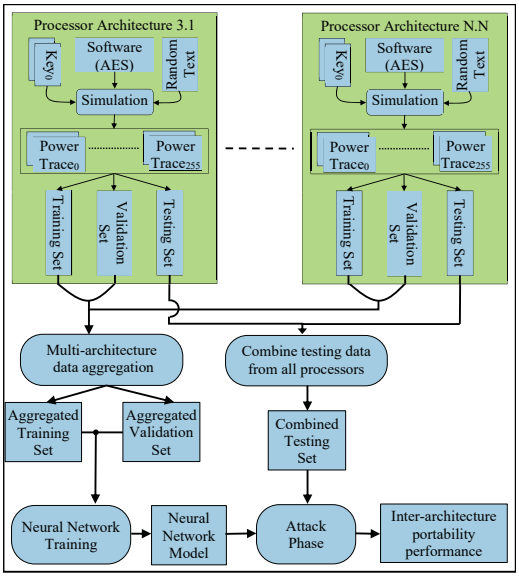
3 THREAT AND POWER MODELS

Our goal in this work is to demonstrate inter-architecture portability of ANNs, where an adversary can attack architecturally different devices. We assume an adversary has knowledge of the target system, including the processor core and architecture configuration for the caches and memory system. We assume the adversary has knowledge of the specific software implementation used for cryptographic operations. The adversary can train an ANN with simulation data of the target system. The final attack requires physical access to collect power traces from the target device. We focus on SCAs that target a 128-bit AES cryptography using electronic code book (ECB) mode executing on embedded devices incorporating a 32-bit ARM processor, with varying L1, L2 caches and memory configurations. An ANN is learned by the adversary to predict the first byte of the 128-bit cryptographic key. As all components of the cryptographic key are independent, we assume the same prediction method can be applied to each key byte.

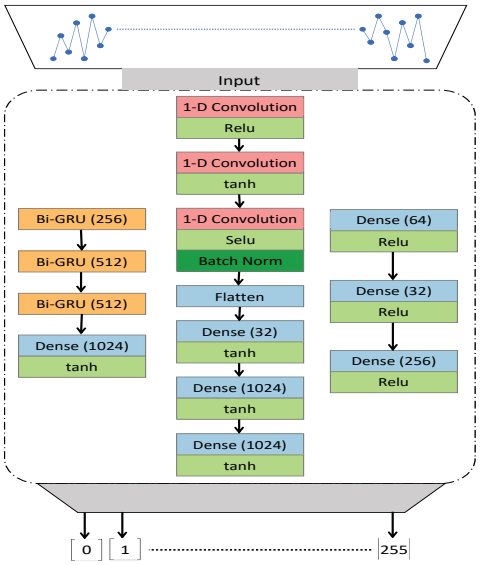
The power model in our approach includes the power consumed by the processor, L1 and L2 caches, and memory when executing an AES encryption. We use the ARM Mbed AES ECB for the AES encryption and GEM5 to simulate processor states, cache and memory access traces for each given system [2]. The processor states are used with an ARM processor power model to generate a cycle-by-cycle power consumption trace of the target processor. The cache and memory activity traces, along with typical physical characteristics for embedded system caches and memories, are fed into the memory organization power modeling tool McPAT to generate a corresponding cycle-by-cycle power trace of the caches and memory [8]. The final power traces account for power dissipation by the processor, L1 caches, L2 cache, and memory. As the simulation is used to generate the power traces, only one power sample per clock cycle is needed to reconstruct the power trace during simulation. The power trace collection is different when one is collecting data from a real device, where the sampling frequency should be at least twice that of the maximum operating frequency of the target device to accurately reconstruct the power trace.

4 METHODS

Figure 1a shows an overview of our ANN-based SCA approach that has four phases: dataset creation, multi-architecture data aggregation, training, and attack. Using the power model (Section 3), we calculate the power trace for the execution of a 128-bit secret key and a random plaintext encrypted using the AES algorithm. Power traces are collected for each processor by varying the secret key and random plaintext. The first byte of the secret key is controlled such that we generate 9,000 power traces for each byte ($9,000x256 = 2,304,000$ total power traces). Next, the dataset is split into a training set with 7,500 traces per key byte ($7500x256 = 1,920,000$ total power traces), a validation set with 1,000 power traces per byte and a testing set with 500 traces per byte. The dataset for 12 processor architecture configuration is generated in this manner. Datasets from two or more processor architecture configurations can be combined into aggregated datasets for training. Note that the training and validation set sizes in the aggregated datasets remain constant at 7,500 and 1,000 power traces per key byte, respectively. The ANNs are trained and tuned on the



(a) Inter-architecture portability



(b) ANN architectures

Figure 1: (a) Overview of the approach to explore inter-architecture portability for power-based SCAs. (b) Overview of the ANN architecture for inter-architecture portability of power-based SCAs.

aggregated training and validation set. A combined testing set for inter-architecture portability analysis is created by aggregating testing sets from all the processor architecture configurations. The trained ANNs then predict the first byte of the secret key from the power traces in combined testing set. The performances of ANNs are reported as the percentage of correct first key bytes predicted from the combined testing set.

Figure 1b shows the ANNs used for inter-architecture portability analysis. All ANNs for power-based SCAs have an input layer with 1,500 nodes, which correspond to the full cycle-accurate power trace for AES execution along with power traces from a random workload function. Power traces are standardized prior to training the ANNs. This standardization ensures that there is less variations in power traces at each time-step. We also ran experiments without standardization of the power traces; however, we found that the model was unable to successfully converge during training. Therefore, we have omitted the results for methods that do not use standardization because they were ineffective. We heuristically picked the hyper-parameters for ANN that performed well on the inter-architecture portability task. For training, we use the ADAM optimizer for calculating gradients in back-propagation, categorical cross-entropy as the loss function, and early stopping to reduce overfitting. The output layer of each ANN uses a softmax layer.

Multi-Layer Perceptrons (MLP). MLPs consist of an input layer, a number of hidden layers, and an output layer. The MLP's hidden layers learn relations between features to correctly predict the outcome of a specific task. The hidden layers are crucial and require tuning to determine the number of layers and node sizes that maximize the MLPs performance. The resulting MLP tuned for power-based SCAs has three hidden layers each having 64, 32, and 256 nodes, respectively. The Rectified Linear Unit (ReLu) is the activation function used for all hidden layers. A dropout layer is added after activation layer to limit overfitting the MLP to training data, thereby ensuring the MLP remains generalizable over a large dataset. Dropout rates of 10% and 40% on first two hidden layers worked well in practice. No dropout is added after the third hidden layer. Finally, the last layer in the MLP is the output layer with 256 nodes, which corresponds to the 256 possible values for the first key byte.

Convolutional Neural Networks (CNN). The proposed CNN consists of three convolutional blocks followed by three fully connected hidden layers. Each convolutional block consists of 1-D convolutions with filter sizes of 8, 16 and 8, respectively. Each of these filters are formed by convolving a kernel of size 3, 7, and 4, respectively. To reduce overfitting, we use a batch normalization layer after the third convolutional block. The output of last convolutional block is flattened, then passed through three fully connected layers of size 32, 1024, and 1024, respectively. Dropout rates of 10%, 10%, and 40% on the hidden layers worked well in practice.

Recurrent Neural Networks (RNN). RNNs perform well on time-series data by using information from previous time. The nature of the RNN can create exploding and vanishing gradients which impacts network performance. LSTMs [6] and GRUs [4] are modified versions of RNNs that solve the gradient problem by including memory blocks, which internally contain gating units that control flow of information. A bidirectional RNN architecture has an added benefit of training inputs in original and reverse sequence. We explored basic LSTM and GRU models and also bidirectional LSTM and GRU models. We found that bidirectional GRU performed best among the explored RNNs. The proposed RNN consists of three bidirectional GRU's each with 256, 512, and 512 memory units. This is followed by a hidden layer with size 1024 with "selu" activation and a dropout of 40%.

Table 1: Target processors used for inter-architecture portability analysis. The legend for the table is: PAC-processor architecture configuration, $-cache, A-associative

PAC	I-$		D-$		L2-$		Memory
	A	*Size*	*A*	*Size*	*A*	*Size*	*Size*
3.1	2	32k	2	64k	N/A		N/A
3.2	8	32k	8	64k			
3.3	2	64k	2	64k			
3.4	8	64k	8	64k			
4.1	2	32k	2	64k	8	2Mb	
4.2	8	32k	8	64k	8	2Mb	
4.3	2	32k	2	64k	16	4Mb	
4.4	8	64k	8	64k	16	4Mb	
5.1	2	32k	2	64k	8	2Mb	1024Mb
5.2	8	32k	8	64k	8	2Mb	1024Mb
5.3	2	64k	2	64k	16	4Mb	1024Mb
5.4	8	64k	8	64k	16	4Mb	1024Mb

5 EXPERIMENTAL RESULTS

For inter-architecture portability analysis, we simulated numerous embedded devices representative of real-world processors with different and unique combinations of L1 caches, L2 cache, and memory. Table 1 presents the processor architecture configurations for the 12 targeted embedded devices benchmarked in this work. The target processor architecture configurations are labeled using the format X.Y, where X indicates the number of components in the system, and Y identifies each unique configuration. For example, configuration 3.1 is the first target configuration with three components, and configuration 5.3 is the third target configuration with five components. Each target configuration consists of the same ARM Cortex-A32 processor core with different cache and memory configurations. We consider four L1 instruction cache configurations varying in size (32K or 64K) and associativity (2-way or 8-way), two L1 data cache configurations of size 64K with varying associativity (2-way or 8-way), two L2 cache configurations varying in size (2MB or 4MB) and associativity (8-way or 16-way), and an optional 1024 MB off-chip DRAM.

Results and Analysis. We conduct several experiments to analyze the performance of ANNs trained without multi-architecture data aggregation. Prior to training data aggregation, we have 12 different datasets for configurations 3.1 to 5.4 (see Table 1). Using the proposed MLP architecture, we train 12 models, one for each dataset. The inter-architecture performance of each model is measured on the combined testing set (i.e., all possible configurations). For comparison, the inter-device performance of the models are evaluated on the testing set of each individual processor architecture configuration. We repeat this evaluation procedure with the CNNs and RNNs.

Figure 2, shows the first group of results in this work. Each column in Figure 2 represents the inter-architecture performance whereas the diagonal represents the inter-device performance of the trained MLP. We observe a general trend that the inter-device performances are high (greater than 99.9%), but the inter-architecture performances are low (less than 12.1%). For example, the first column represents the performance of the MLP trained only on configuration 3.1 when attacking testing data of all configurations from

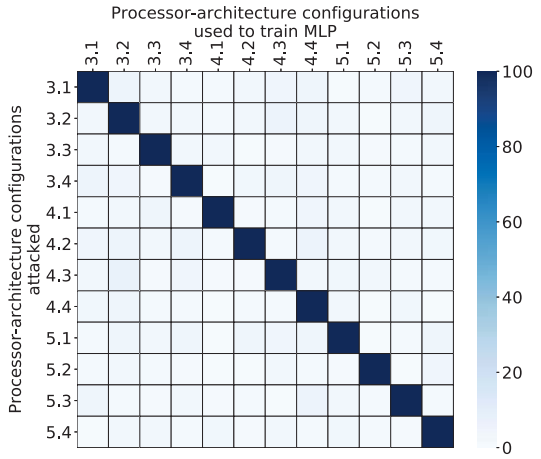


Figure 2: Inter-architecture portability for MLPs trained on a target configuration. The diagonal is the inter-device performance and each column shows the performance of the MLP on test data of other processor configurations. Similar heat-maps (not shown) for the CNN and RNN were observed.

3.1 to 5.4. The inter-device performance of the MLP trained on 3.1 is 100% when attacking testing dataset of 3.1. The inter-architecture performance of the MLP trained on 3.1 is 11.2% when attacking the combined testing set. This shows that the MLP trained on 3.1 is not able to generalize across other configurations. CNNs and RNNs also show a similar trend, having high inter-device performances (greater than 99.9% and 99.5%, respectively) but low inter-architecture performances (less than 15.9% and 15.3%, respectively).

We extend the previous experiments on aggregated datasets (i.e., the datasets with multiple processor architecture configurations). We aggregate power trace data from configurations 3.1 to 5.4 by combining 2, 3, 4, and 6 different configuration datasets, of which 12 unique sets (S1 to S12) are selected for analysis. Note, that the training and validation set size is fixed at 7,500 and 1,000 power traces per key byte in all 12 sets. If n is the number of selected combinations to aggregate, each configuration contributes $7500/n$ traces per key to the aggregated training set and $1000/n$ traces per key to the aggregated validation set. The MLP is trained on the 12 sets, giving us 12 MLPs. The training procedure is repeated for CNNs and RNNs. Inter-architecture performance is evaluated on the combined testing set as seen in Figure 3.

All ANNs trained on the 12 sets show better inter-architecture performance as compared to ANNs trained on datasets without aggregation. For example, consider set S1, which combines power traces from the 3.2 and 5.4 processor architecture configurations. The inter-architecture performance of the MLPs, CNNs, and RNNs trained on S1 are 44.8%, 37.0%, and 45.6%, respectively. We also see that there is a general increase in inter-architecture performance as we increase the number of configurations in the aggregated sets. Specifically, sets S11 and S12, which are aggregation of six different configurations, show a high inter-architecture performance irrespective of the ANNs used. The inter-architecture performance of MLP, CNN, and RNN for set S11 is 99.5%, 99.9%, and 98.1% and for S12 model is 99.7%, 99.6%, and 98.2% ,respectively. Note that S11 and S12 cover most of the possible combinations of L1 caches, L2 cache,

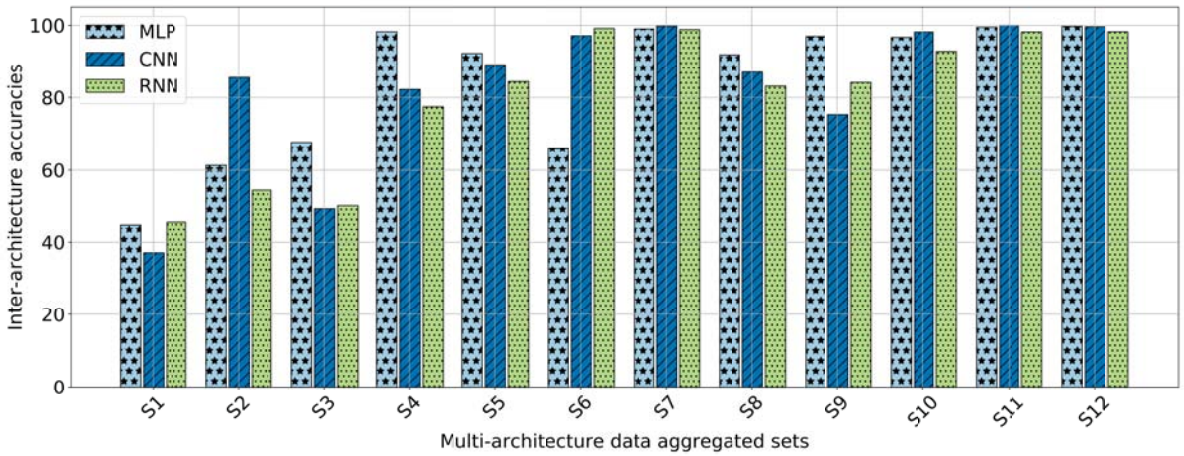


Figure 3: Inter-architecture performance of ANNs trained on aggregated sets evaluated on combined testing set. The details of each set are as follows (see Table 1): S1{3.2, 5.4}, S2{4.2, 4.4}, S3{3.2, 4.2}, S4{4.1, 5.1, 5.2}, S5{3.2, 5.1, 5.2}, S6{3.3, 4.1, 5.4}, S7{4.1, 5.1, 5.4}, S8{3.2, 4.2, 5.4}, S9{3.2, 3.3, 4.2}, S10{3.3, 4.4, 5.1, 5.3}, S11{3.2, 3.3, 4.2, 4.3, 5.2, 5.3}, S12{3.3, 3.4, 4.3, 4.4, 5.3, 5.4}.

and memory configuration from Table 1. The ANNs of S11 and S12 generalize better over others configurations as the coverage of their training data over the combined testing set is high.

With a decrease in the number of aggregated configurations in sets S1-S9, we observe variability in accuracy of trained ANNs for inter-architecture performance. The variability in inter-architecture performance can be attributed both to the coverage of training data of S1-S9 being low over the combined testing set and to how the MLP, CNN, and RNN capture different features while training using those sets. We note that ANNs trained using a lower number of aggregated datasets can achieve high accuracy, such as S7 that performs as well as the S11 and S12 ANNs. More analysis is needed to formulate a general strategy that can be used to find configurations that can be aggregated so that the ANNs generalize on the combined testing set.

Intuitively, ANNs trained with aggregated data from all 12 configurations will give high inter-architecture performance. But, it is not feasible, nor practical, to include all possible processor architecture configurations that are available in the market for creating a global dataset to train and tune ANNs. Instead, a subset of processor architecture configurations (e.g., S11 or S12) that share architectural features can be used for training ANNs. Thus, maintaining a reasonable balance between the classification performance and time / data complexity.

6 CONCLUSIONS

SCAs are of growing concern and the vulnerabilities of embedded systems has led to many different works. This is one of the first studies to explore the inter-architecture portability and ANNs for power-based SCAs. We present ANN based methodology to retrieve the first byte of the secret key by training ANNs (MLP, CNN, and RNN) on the power traces generated by the processor during the execution of 128-bit AES encryption. We demonstrate that the inter-architecture performance of ANNs increase with the addition of power traces from different processor architecture configurations. Specifically, we present a subset of configurations whose combination of power traces during training of ANNs shows high inter-architecture performance while maintaining a healthy trade-off between performance and time/data complexity. Additionally, we conclude that power traces from identical architecture processor is not required to profile ANNs for power-based SCAs.

ACKNOWLEDGMENTS

This work was supported by grants from the Department of Energy #DE-NA0003946, Army Research Lab W56KGU-20-C-0002, and the National Science Foundation CAREER #1943552.

REFERENCES

[1] Shivam Bhasin, Anupam Chattopadhyay, Annelie Heuser, Dirmanto Jap, Stjepan Picek, and Ritu Ranjan Shrivastwa. 2019. Mind the Portability: A Warriors Guide through Realistic Profiled Side-channel Analysis. Cryptology ePrint Archive, Report 2019/661.

[2] Nathan Binkert, Ronald Dreslinski, Lisa Hsu, Kevin T. Lim, Ali Saidi, and Steven K. Reinhardt. 2006. The M5 Simulator: Modeling Networked Systems. *IEEE Micro* 26 (2006), 52–60.

[3] Suresh Chari, Josyula R Rao, and Pankaj Rohatgi. 2003. Template Attacks. In *Cryptographic Hardware and Embedded Systems - CHES 2002*, Burton S Kaliski, çetin K Koç, and Christof Paar (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 13–28.

[4] Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. arXiv:1409.1259 [cs.CL]

[5] Debayan Das, Anupam Golder, Josef Danial, Santosh Ghosh, Arijit Raychowdhury, and Shreyas Sen. 2019. X-DeepSCA: Cross-Device Deep Learning Side Channel Attack. In *Design Automation Conference*.

[6] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[7] Liran Lerman, Romain Poussier, Olivier Markowitch, and François-Xavier Standaert. 2018. Template attacks versus machine learning revisited and the curse of dimensionality in side-channel analysis: extended version. *Journal of Cryptographic Engineering* 8, 4 (2018), 301–313.

[8] S Li, J H Ahn, R D Strong, J B Brockman, D M Tullsen, and N P Jouppi. 2009. McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures. In *IEEE/ACM International Symposium on Microarchitecture*. 469–480.

[9] Zdenek Martinasek, Lukas Malina, and Krisztina Trasy. 2015. *Profiling Power Analysis Attack Based on Multi-layer Perceptron Network*. 317–339.

[10] Emmanuel Prouff, Remi Strullu, Ryad Benadjila, Eleonora Cagli, and Cécile Canovas. 2018. Study of Deep Learning Techniques for Side-Channel Analysis and Introduction to ASCAD Database. *IACR Cryptology* (2018), 53.