Federated Learning in Side-Channel Analysis

Huanyu Wang [1] **Elena Dubrova** [1]

Abstract

Recently introduced federated learning is an attractive framework for the distributed training of deep learning models with thousands of participants. However, it can potentially be used with malicious intent. For example, adversaries can use their smartphones to jointly train a classifier for extracting secret keys from the smartphones' SIM cards without sharing their side-channel measurements with each other. With federated learning, each participant might be able to create a strong model in the absence of sufficient training data. Furthermore, they preserve their anonymity. In this paper, we investigate this new attack vector in the context of side-channel attacks. We compare the federated learning, which aggregates model updates submitted by N participants, with two other aggregating approaches: (1) training on combined side-channel data from N devices, and (2) using an ensemble of N individually trained models. Our first experiments on 8-bit Atmel ATxmega128D4 microcontroller implementation of AES show that federated learning is capable of outperforming the other approaches.

1. Introduction

Federated Learning (FL) is a new paradigm in machine learning that can help meet regulatory requirements (GDPR (Voigt & Von dem Bussche, 2017), HIPAA (Atchinson & Fox, 1997)) and mitigate privacy concerns while taking advantage of massive distributed data (Konečnỳ et al., 2016b;a; McMahan et al., 2016). FL allows its participants to collaboratively train a global model without sharing participant's local training data. At every communication round, each participant trains a local model based on his/her training data and submits the model updates to the server. The server employs a secure aggregation (Bonawitz et al., 2017) to build a global model by averaging the local models' weights. Motivating applications for FL include image classifiers for self-driving cars, keyboard next-word predictors, and personalized product recommendation services (Li et al., 2019).

However, as any great scientific discovery, FL can potentially be used with malicious intent. Since FL preserves not only training data confidentiality, but also participant's anonymity, its setting is very appealing to adversaries. Furthermore, an adversary who does not have enough training data might still be able to create a strong deep-learning model by training in a FL framework. For example, adversaries can use their smartphones to jointly train a classifier for extracting secret keys from the smartphones' SIM cards without sharing their local side-channel measurements with each other. At each round, every participant independently trains a local model update based on traces captured from his/her profiling device and uploads it to the aggregator, where the submitted updates are combined to construct a global model. The aggregator can be either a participant, or a third party.

In this paper, we investigate this new attack vector in the context of Deep-Learning Side Channel Attacks (DL-SCAs). DL-SCA are one of the most powerful attacks against implementations of cryptographic algorithms at present (Perin et al., 2018). During the execution of a cryptographic algorithm, physical implementations tend to leak side-channel information which is related to the secret key. An adversary first trains a deep-learning model on power traces captured from profiling devices which he/she controls, and then applies the trained model to recover the key of a victim device. Using more than one device for profiling (called *multi-source training*), is known to reduce the negative effect of inter-chip variation, which is prominent in advanced technologies, and help generalization (Wang et al., 2019; Das et al., 2019; Wang et al., 2020).

Another known technique for reducing generalization error in machine learning is *bootstrap aggregating*, or *bagging* (Breiman, 1996). In bagging, several different, separately trained models are used in an ensemble to vote on the output results. Since different models usually do not make the same errors on the test set, on average, an ensemble of N models is expected to perform better than its

[1]School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, Stockholm, Sweden. Correspondence to: <huanyu@kth.se>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

members (Goodfellow et al., 2016). Bagging has been successfully applied to power analysis of hardware implementations of Advanced Encryption Standard (AES) (Wang & Dubrova, 2020). The attack presented in (Wang & Dubrova, 2020) uses an ensemble of three CNN models trained on different attack points.

While it is obvious that a DL-SCA in FL framework will outperform a DL-SCA based a single classifier trained on a single profiling device, the outcome of a competition between FL (model-level aggregation), bagging (output-level aggregation), and multi-source training (data-level aggregation) methods is not evident. We present such an evaluation in this paper. We apply FL, bagging, and multi-source training aggregation methods to power analysis of a microcontroller implementation of AES. Our first experiments show that FL is capable of outperforming the other two approaches.

The rest of the paper is organized as follows. Section 2 gives a background on deep-learning side-channel attacks. Section 3 describes model-level, output-data and data-level aggregation methods in the side-channel analysis context. Section 4 presents the experimental setup. Section 5 shows how local models are trained. Section 6 summarizes the evaluation results. Section 7 concludes this paper and discusses open problems.

2. Deep Learning Side-Channel Attacks

Side-channel attacks were pioneered by Paul Kocher in his seminal paper on *timing analysis* (Kocher, 1996) where he has shown that non-constant running time of a cipher can leak information about its key. Kocher has also introduced *power analysis* (Kocher et al., 1999) which exploits the fact that circuits typically consume differing amounts of power based on their input data. The power consumption remains one of the most successfully exploited side-channels today. We focus on power analysis in this paper.

The target of a side-channel attack is to recover an n-bit key $k \in \mathcal{K}$, where $\mathcal{K}$ is the set of all possible keys. To recover the key, the attacker uses of a set of known input data (e.g. the plaintext) and a set of the physical measurements (e.g. power consumption). Usually a divide-and-conquer strategy is used in which the key k is divided into m-bit parts k_i, called *subkeys*, and the subkeys k_i are recovered independently, for $i \in \{1, 2, \ldots, \frac{n}{m}\}$. Typically $m = 8$.

Deep learning can be used in side-channel analysis in two settings: profiling and non-profiling. *Profiling* attacks (Martinasek et al., 2015) first learn a leakage profile of the cryptographic algorithm under attack, and then attack. *Non-profiling* attacks (Timon, 2018) attack directly, as the traditional Differential Power Analysis (Kocher et al., 1999) or Correlation Power Analysis (CPA) (Brier et al., 2004). In this paper, we focus on profiling attacks.

2.1. Assumptions

Profiling side-channel attacks assume that:

1. The attacker has at least one device, called the *profiling* device, which is similar to the device under attack and runs the same implementation of the same cryptographic algorithm.
2. The attacker has a full control over the profiling device (can apply chosen plaintext, program chosen keys, and do physical measurements).
3. The attacker has a physical access to the victim device to measure some side-channel signals during the execution of the cryptographic algorithm.

In addition, in this paper we assume that only a single power trace from a victim device is available to the attacker. Single-trace attacks are particularly threatening because they can recover the key even if the key if changed for every session.

2.2. Attack Stages

A profiling deep-learning side-channel attack is done in two stages.

At the *profiling* stage, the selected type of deep-learning model is trained to learn a leakage profile of the cryptographic algorithm under attack for all possible values of the sensitive variable. The sensitive variable is typically a subkey. The training is done using a large set of side-channel data captured from profiling device(s) which are labeled according to the selected leakage model.

At the *attack* stage, the trained model is used to classify the side-channel data from a victim device.

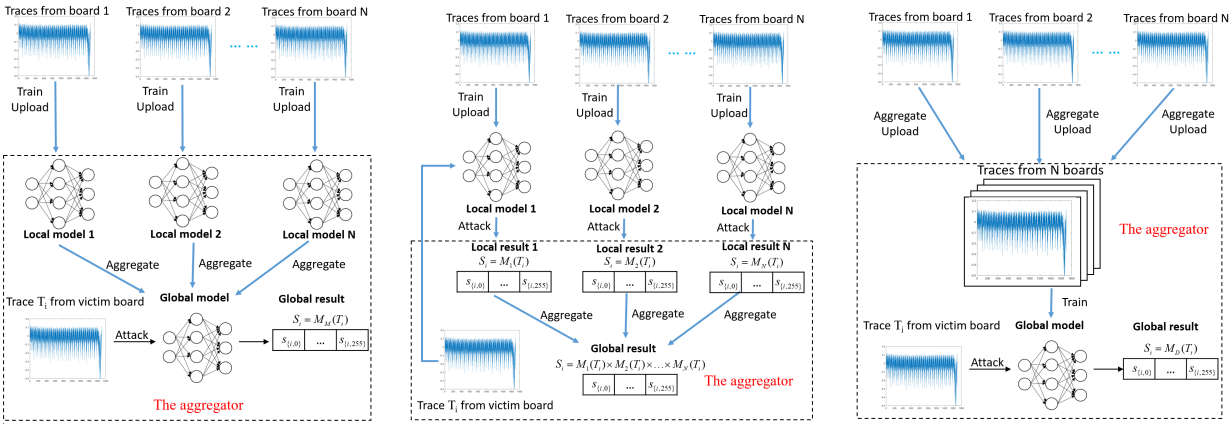
3. Aggregation methods

This section describes model-, output- and data-level aggregation methods in the side-channel analysis context.

3.1. Model-level aggregation

Figure 1(a) illustrates the model-level aggregation. There are N participants (clients) jointly constructing a federated deep-learning model. Each client i has n_i private data samples from his/her profiling device, for $i \in \{1, \ldots, N\}$. The total number of training data samples of N clients is denoted by n, with $n = \sum_{i=1}^{N} n_i$.

At the beginning of the training process, a typical model structure is initialized by an aggregator (server) and sent to each client. At each communication round t, a random fraction $\eta_t \in [0, 1]$ of N clients is selected by the aggregator to independently update local models based on their private



(a) Model-level aggregation (FL) (b) Output-level aggregation (c) Data-level aggregation

Figure 1. Model-, output- and data-level aggregation in the side-channel analysis context.

data and upload the updates to the aggregator. In our experiment, we set $\eta_t = 1$, which means that all clients contribute to the global model in each communication round.

For each client i, local updates are typically done using *Stochastic Gradient Descent* (SGD) taken on the private data the client i based on the weights ω_0^t of the shared global model:

$$\omega_i^{t+1} = \omega_0^t - \alpha \nabla \phi(\omega_i^t) \tag{1}$$

where α is the learning rate, $\nabla\phi$ is gradient of the classification loss ϕ, and ω_i^t are weights of current local model of the client i.

A typical aggregation approach of federated learning is *averaging*. The aggregator computes the weights ω_0^{t+1} of the global model by averaging the weights of submitted local models:

$$\omega_0^{t+1} = \sum_{i=1}^{N} \frac{n_i}{n} \omega_i^{t+1} \tag{2}$$

At the end of communication round t, the aggregator sends the global model with the weights ω_0^{t+1} back to each client.

All clients can use the global model to classify the data samples from a victim device.

3.2. Output-level aggregation

Figure 1(a) shows the output-level aggregation approach inspired by bagging (Breiman, 1996) meta-algorithm which is a type of ensemble learning. N different classifiers are trained and their score vectors are combined. In this way, a stronger classifier can potentially be created from several weaker ones. Bagging may help avoid overfitting and reduce variance (Polikar, 2012). In output-level aggregation, N participants train their local models independently from each other. Each participant i uses n_i private data samples from his/her profiling device to train the model M_i, for $i \in \{1, \ldots, N\}$. After training, all participants submit their trained models to the aggregator. The aggregator creates the ensemble model M_O from M_1, M_2, M_3 by aggregating the outputs of M_1, M_2, M_3 and returns M_O back to each participant.

All clients can use M_O to classify the data samples from a victim device.

3.3. Data-level aggregation

Figure 1(c) illustrates the data-level aggregation. There are N participants. Each participant i uploads n_i private data samples from from his/her profiling device to the aggregator, for $i \in \{1, \ldots, N\}$. The aggregator combines the local data into the global data set with $n = \sum_{i=1}^{N} n_i$ samples and trains the model M_D. After training, the aggregator sends the model M_D to each participant.

All clients can use M_D to classify the data samples from a victim device.

4. Experimental Setup

The section describes our experimental setup.

4.1. Equipment for Power Analysis

The equipment we use for power analysis is shown in Figure 2. It consists of the ChipWhisperer-Lite board, the CW308 UFO mother board and nine CW308T-XMEGA target boards. In the sequel, we refer to these boards as $D_1, D_2, \ldots, D_9$.

The ChipWhisperer is a hardware security evaluation

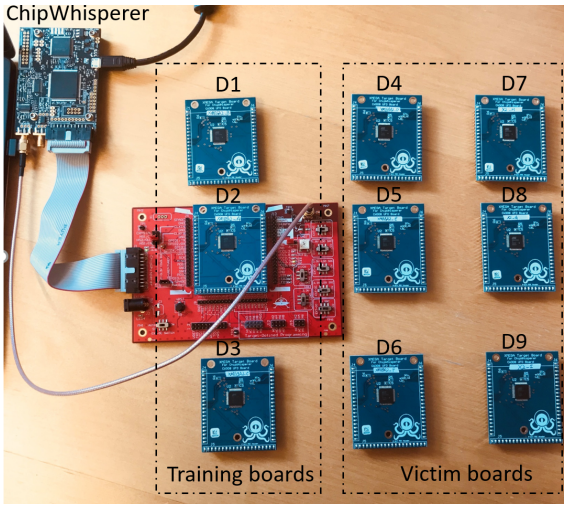


Figure 2. Equipment for power analysis.

toolkit based on a low-cost open hardware platform and an open source software (NewAE Technology Inc.). The ChipWhisperer-Lite can be used to measure power consumption with the maximum sampling rate of 105 MS/sec.

The CW308 UFO board is a generic platform for evaluating multiple targets (CW308 UFO Target). The target board is plugged in a dedicated U connector.

The CW308T-XMEGA target board contains an 8-bit ATxmega128D4 microcontroller. We programmed the microcontrollers to the same implementation of AES-128 encryption algorithm in Electronic codebook (ECB) mode of operation.

4.2. Power Trace Acquisition

We used D_1, D_2, D_3 as profiling devices and $D_4 - D_9$ as victim devices. To collect training data, 300K power traces were captured from each profiling device during the execution of AES for randomly selected plaintexts and keys. To collect testing data, 1K power traces were captured from each target device during the execution of AES for randomly selected plaintexts and fixed keys.

5. Training of Local Models

In this section we describe how local models are trained.

5.1. Choice of Neural Network Type

Previous work investigated which type of deep neural networks is suitable for various side-channel analysis scenarios. For example, Convolutional Neural Networks (CNNs) can overcome trace misalignment and jitter-based countermeasure (Cagli et al., 2017; Perin et al., 2018; Gilmore et al., 2019). If traces are synchronized and there is no need to handle noise, Multiple Layer Perception (MLP) seems to be a more suitable choice. MLPs are shown successful in extracting keys from software (Das et al., 2019; Wang et al., 2019; Benadjila et al., 2018; Martinasek et al., 2016; Maghrebi, 2019) and hardware (Kubota et al., 2019) implementations of AES.

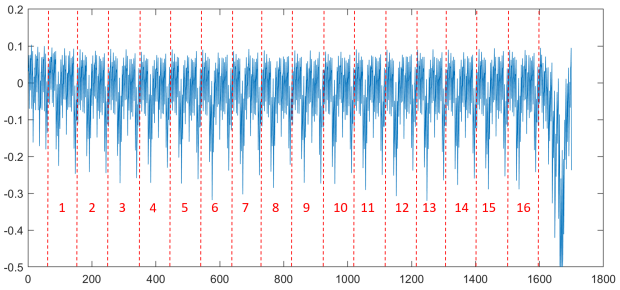


Figure 3. Segment of a power trace from an 8-bit ATxmega128D4 microcontroller representing 16 executions of S-box.

In our experiments, we use an unprotected software implementation of AES-128 on an 8-bit microcontroller. In a software implementation instructions are executed sequentially, thus signal-to-noise ratio is much higher compared to a hardware implementation. Furthermore, we capture traces using ChipWhisperer which assures perfect trace alignment. For this reason, we use MLPs as a neural network type.

5.2. Training Process

Given a set of power traces $\{T_1, \ldots, T_n\}$, $T_i \in \mathbb{R}^m$, where m is the number of data points in a trace, and a set of classification classes C, the objective is classify traces according their labels $l(T_i) \in C$.

Fig. 3 shows the segment of a power trace from an 8-bit ATxmega128D4 microcontroller representing 16 executions of S-box in the 1st encryption round. The S-box is a 8×8 invertible mapping. AES-128 executes S-box 16 times in each round. One can see the distinct shape of each S-box execution. For all models, we use 8-bit values of the S-box output in the 1st round as labels (*identity* power model[1]), i.e. $C = \{0, 1, \ldots, 255\}$.

A neural network can be viewed as a function $M : \mathbb{R}^m \rightarrow \mathbb{I}^{|C|}$, where $\mathbb{I} := \{x \in \mathbb{R} \mid 0 \leq x \leq 1\}$, which maps a trace T_i into a *score* vector $S_i = M(T_i) \in \mathbb{I}^{|C|}$ whose elements $s_{i,j}$ represent the probability of the label with value $j \in C$.

We use *categorical cross-entropy loss* to quantify the classification error of the network. To minimize the loss, the gradient of the loss with respect the score S_i is computed

[1]Identity power model assumes that the power consumption is proportional to the value of the data processed at the attack point.

Layer Type	Output Shape	Parameter #
Input (Dense)	(None, 200)	19400
Dense 1	(None, 200)	40200
Dense 2	(None, 200)	40200
Dense 3	(None, 200)	40200
Dense 4	(None, 200)	40200
Output (Dense)	(None, 256)	51456
Total Parameters: 231,656		

Table 1. Local model's architecture summary.

and back-propagated through the network to tune its internal parameters according to the *RMSprop* optimizer, which is one of the advanced adaptations SGD algorithm (Robbins & Monro, 1951). This is repeated for a chosen number of iterations called *epochs*.

Once the network is trained, to classify a trace T_i whose label $l(T_i)$ is unknown, we determine the most likely label $\tilde{l}$ among all $|C|$ candidate labels as

$$\tilde{l} = \arg\max_{i \in |C|} S_i.$$

If $\tilde{l} = l(T_i)$, the classification is successful.

5.3. Choice of Neural Network Architecture

The architecture of MLP networks used in our experiments is shown in Table 1. The network contains an input layer, four hidden layers and an output layer. The input size 96 corresponds to the number data samples in one S-box execution. The output size is $|C| = 256$.

6. Evaluation Results

In this section, we apply model-, output- and data-level aggregation methods to power analysis of AES-128 and compare their results.

In all experiments, we simulate a scenario with $N = 3$ participants having the same number of training traces.

6.1. Results of model-level aggregation

In this experiment, three participants jointly create a global MLP model, M_M, each using $n_i = 300$K traces from D_i for training his/her local model, for $i \in \{1, 2, 3\}$. For all local models, we used RMSprop with a learning rate $\alpha = 0.0001$ and trained for 40 epochs with local minimum batch size 128. The training is carried out for 20 communication rounds. After each round, the aggregator sends the global model back to each participant and the global model is further trained on local training sets.

Since each of the participants uses the same number of training traces for updating the local model, the weights ω_0^{t+1} of the global model are updated based on the weights

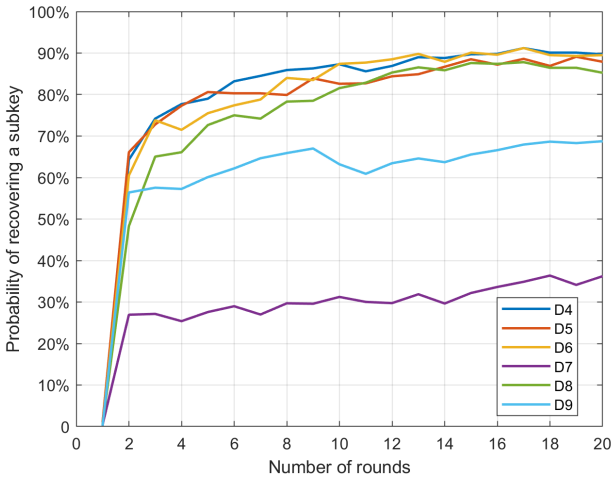


Figure 4. Probability of recovering a subkey from a single trace from devices $D_4 - D_6$ using the global model M_M (average for 1,000 tests).

ω_i^{t+1}, $i \in \{1, 2, 3\}$, of local models as:

$$\omega_0^{t+1} = \frac{1}{3}(\omega_1^{t+1} + \omega_2^{t+1} + \omega_3^{t+1}).$$

After each round, we test the resulting global model on a randomly selected single trace T_i from each victim device D_j, for $j \in \{4, 5, \ldots, 9\}$. If the correct subkey value has the highest probability in the score vector $S_i = M_M(T_i)$, the attack is successful. Otherwise, the attack fails. Figure 4 shows the average probability of recovering a subkey from a single trace from devices $D_4 - D_9$ for 1,000 tests for different numbers of rounds.

From Figure 4, we can see that the federated model built in the $17th$ communication round has the highest average success probability over all rounds. The 2nd column in Table 2 shows the probability of recovering a subkey from a single trace using this model. We can see that the average is 79.7%.

6.2. Results of output-level aggregation

In this experiment, three participants train their local MLP models independently from each other. Each participant i trains the model M_i on $n_i = 300$K traces from D_i with 60K traces set aside for validation, for $i \in \{1, 2, 3\}$. For all models, we used RMSprop with a learning rate $\alpha = 0.0001$ and trained for 70 epochs with batch size 128. The score vector of the ensemble model M_O is computed by multiplying score vectors $S_i = M_j(T_i)$ for all $j \in \{1, 2, 3\}$. Such an approach is known to work well for power analysis of hardware implementations of AES (Wang & Dubrova, 2020). Note, however, that we used the same attack point (S-box output in the 1st round) to create labels for models M_1, M_2

Table 2. Probability of recovering a subkey from a single trace using aggregated models (average for 1,000 tests).

Device	Aggregation method		
	Model-level M_M	Output-level M_O	Data-level M_D
D_4	89.8%	70.0%	71.3%
D_5	91.2%	60.6%	79.4%
D_6	91.4%	62.6%	65.3%
D_7	35.5%	28.7%	41.5%
D_8	88.5%	56.3%	49.6%
D_9	69.6%	62.7%	70.0%
average	79.7%	66.4%	70.7%

Table 3. Probability of recovering a subkey from a single trace using local models (average for 1,000 tests).

Device	M_1	M_2	M_3
D_4	40.3%	31.9%	32.7%
D_5	44.6%	40.7%	15.6%
D_6	34.3%	28.6%	46.7%
D_7	12.8%	10.2%	65.2%
D_8	22.5%	28.8%	45.8%
D_9	55.7%	66.2%	12.1%
average	35.0%	34.4%	36.4%

and M_3. In (Wang & Dubrova, 2020), a different attack point is used for each model. This might have negatively affected the ensemble's results.

The 3rd column of Table 2 shows the probability of recovering a subkey from a single trace using M_O. For a comparison, Table 3 also shows the results for local models M_1, M_2 and M_3. One see that that success probabilities of the models M_1, M_2 and M_3 vary a lot for different devices. This is because different pairs of devices have different amounts of variability. Some devices are less different, some are more different. For example, on one hand, models M_1 and M_2 can recover a subkey from a single power trace from D_5 in 44.6% and 40.7% of cases, respectively. Contrary, for model M_3, the subkey recovery rate from D_5 is only 15.6%. Probably D_1 and D_2 are less different from D_5 than D_3. On the other hand, M_3 significantly outperforms M_1 and M_2 on D_7 (65.3% vs 12.8% and 10.2%, respectively). Probably D_3 is similar to D_7, while D_1 and D_2 are very different from D_7. High dissimilarly of D_1 and D_2 from D_7 might be the reason why the global model M_M in performs so poorly on D_7 in the federated learning case. From Table 3 we can also conclude that D_3 is very different from D_9, which explains the worse result of M_M for D_9 as compared to the results of M_M for $D4 - D6$ and D_8.

Using an ensemble improves generalization ability of individual models (Sh9, 1999), as we can see from the 3rd column of Table 2. The average probability of recovering a subkey from a single trace using M_O is 66.4%.

6.3. Results of data-level aggregation

In this experiment, the participants 1, 2 and 3 upload their sets of 300K traces from D_1, D_2 and D_3, respectively, to the aggregator. The aggregator combines theses sets into a training set of size 900K and sets aside 270K traces for validation. Then, the aggregator trains the MLP model M_D. We used RMSprop with a learning rate $\alpha = 0.0001$ and trained for 40 epochs with batch size 128. The 4th column of Table 2 shows the probability of recovering a subkey from a single trace using M_D. The average is 70.7%.

7. Conclusion

We compared the federated learning approach to two other aggregating approaches - on data and on output levels. Our first results show that federated learning is capable of outperforming the other approaches. This is quite surprising. Intuitively, it should be more difficult to train a global model in a federated learning framework due to challenges related to training on distributed data while keeping these data private. Moves due to averaging of weights of local models are more random compared to the moves due to the SGD. This randomness seems beneficial for the optimization of the objective function in the case of power analysis, when generalization is particularly important.

We plan to further investigates this phenomena by training new models using other combinations of profiling devices.

8. Acknowledgements

This work was supported in part by the research grant 2018-04482 from the Swedish Research Council and by the Vinnova Competence Center for Trustworthy Edge Computing Systems and Applications at KTH Royal Institute of Technology.

References

In Sharkey, A. (ed.), *Combining Artificial Neural Nets: Ensemble and Modular Multi-Net Systems*, New York, 1999. Springer-Verlag.

Atchinson, B. K. and Fox, D. M. From the field: The politics of the health insurance portability and accountability act. *Health affairs*, 16(3):146–150, 1997.

Benadjila, R., Prouff, E., Strullu, R., Cagli, E., and Dumas, C. Study of deep learning techniques for side-channel analysis and introduction to ascad database. *ANSSI, France & CEA, LETI, MINATEC Campus, France*, 22: 2018, 2018. URL https://eprint.iacr.org/2018/053.pdf.

Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A.,

McMahan, H. B., Patel, S., Ramage, D., Segal, A., and Seth, K. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1175–1191, 2017.

Breiman, L. Bagging predictors. *Machine learning*, 24(2): 123–140, 1996.

Brier, E., Clavier, C., and Olivier, F. Correlation power analysis with a leakage model. In Joye, M. and Quisquater, J.-J. (eds.), *Cryptographic Hardware and Embedded Systems - CHES 2004*, pp. 16–29, 2004. ISBN 978-3-540-28632-5.

Cagli, E., Dumas, C., and Prouff, E. Convolutional neural networks with data augmentation against jitter-based countermeasures. In *International Conference on Cryptographic Hardware and Embedded Systems*, pp. 45–68. Springer, 2017.

CW308 UFO Target. https://wiki.newae.com/CW308_UFO_Target.

Das, D., Golder, A., Danial, J., Ghosh, S., Raychowdhury, A., and Sen, S. X-deepsca: Cross-device deep learning side channel attack. In *Proceedings of the 56th Annual Design Automation Conference 2019*, pp. 1–6, 2019.

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

Goodfellow, I., Bengio, Y., and Courville, A. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

Kocher, P., Jaffe, J., and Jun, B. Differential power analysis. pp. 388–397. Springer-Verlag, 1999.

Kocher, P. C. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In *Proc. of the 16th Annual Int. Cryptology Conf. on Advances in Cryptology*, pp. 104–113, 1996. ISBN 3-540-61512-1.

Konečnỳ, J., McMahan, H. B., Ramage, D., and Richtárik, P. Federated optimization: Distributed machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527*, 2016a.

Konečnỳ, J., McMahan, H. B., Yu, F. X., Richtárik, P., Suresh, A. T., and Bacon, D. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016b.

Kubota, T., Yoshida, K., Shiozaki, M., and Fujino, T. Deep learning side-channel attack against hardware implementations of AES. In *2019 22nd Euromicro Conference on Digital System Design (DSD)*, pp. 261–268. IEEE, 2019.

Li, Q., Wen, Z., Wu, Z., Hu, S., Wang, N., and He, B. A survey on federated learning systems: Vision, hype and reality for data privacy and protection, 2019.

Maghrebi, H. Deep learning based side channel attacks in practice. Technical report, IACR Cryptology ePrint Archive 2019, 578, 2019.

Martinasek, Z., Malina, L., and Trasy, K. Profiling power analysis attack based on multi-layer perceptron network. In *Computational Problems in Science and Engineering*, pp. 317–339. Springer, 2015.

Martinasek, Z., Dzurenda, P., and Malina, L. Profiling power analysis attack based on mlp in dpa contest v4. 2. In *2016 39th International Conference on Telecommunications and Signal Processing (TSP)*, pp. 223–226. IEEE, 2016.

McMahan, H. B., Moore, E., Ramage, D., Hampson, S., et al. Communication-efficient learning of deep networks from decentralized data. *arXiv preprint arXiv:1602.05629*, 2016.

NewAE Technology Inc. Chipwhisperer. https://newae.com/tools/chipwhisperer.

Perin, G., Ege, B., and van Woudenberg, J. Lowering the bar: Deep learning for side-channel analysis (white-paper). In *Proc. BlackHat*, pp. 1–15, 2018.

Polikar, R. Ensemble learning. In *Ensemble machine learning*, pp. 1–34. Springer, 2012.

Robbins, H. and Monro, S. A stochastic approximation method. *Ann. Math. Statist.*, 22:400–407, 1951.

Timon, B. Non-profiled deep learning-based side-channel attacks. IACR Cryptology ePrint Archive, 2018:196, 2018.

Voigt, P. and Von dem Bussche, A. The eu general data protection regulation (gdpr). *A Practical Guide, 1st Ed., Cham: Springer International Publishing*, 2017.

Wang, H. and Dubrova, E. Tandem deep learning side-channel attack against FPGA implementation of AES. *arXiv preprint arXiv*, 2020.

Wang, H., Brisfors, M., Forsmark, S., and Dubrova, E. How diversity affects deep-learning side-channel attacks. In *2019 IEEE Nordic Circuits and Systems Conference (NORCAS): NORCHIP and International Symposium of System-on-Chip (SoC)*, pp. 1–7. IEEE, 2019.

Wang, H., Forsmark, S., Brisfors, M., and Dubrova, E. Multi-source training deep learning side-channel attacks. *IEEE 50th International Symposium on Multiple-Valued Logic*, 2020.