Machine Learning and Implementation Attacks

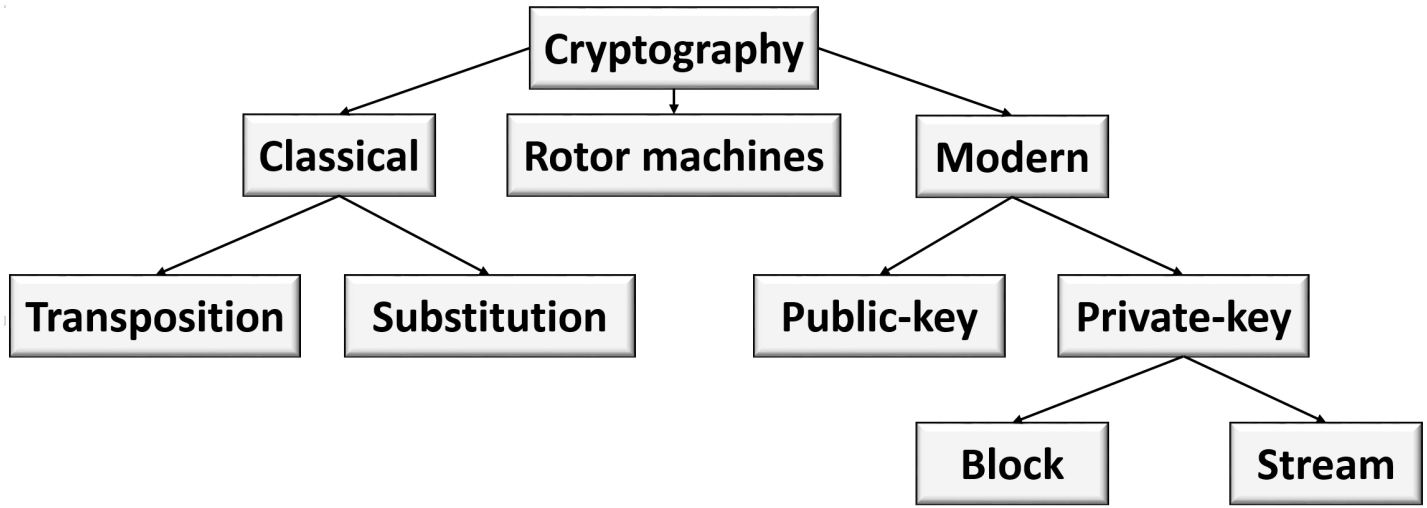
Stjepan Picek

Machine Learning and Security School, Padua, September 11, 2019

Outline

Outline

Fast Introduction to Cryptography



Where to use Machine Learning in Cryptology

- Machine learning is data driven approach.
- It seems more difficult to use such techniques for design.
- Additional benefit from using them in attacks: it is easy to validate the solution.

Where to use Machine Learning - Classical Applications

- **Side-channel attacks**.
- **Fault injection**.
- Modeling attacks on PUFs.
- Detecting Hardware Trojans.
- Machine learning over encrypted data.

Where to use Machine Learning - Exotic Applications

- Factoring numbers.
- Design of ciphers.

Outline

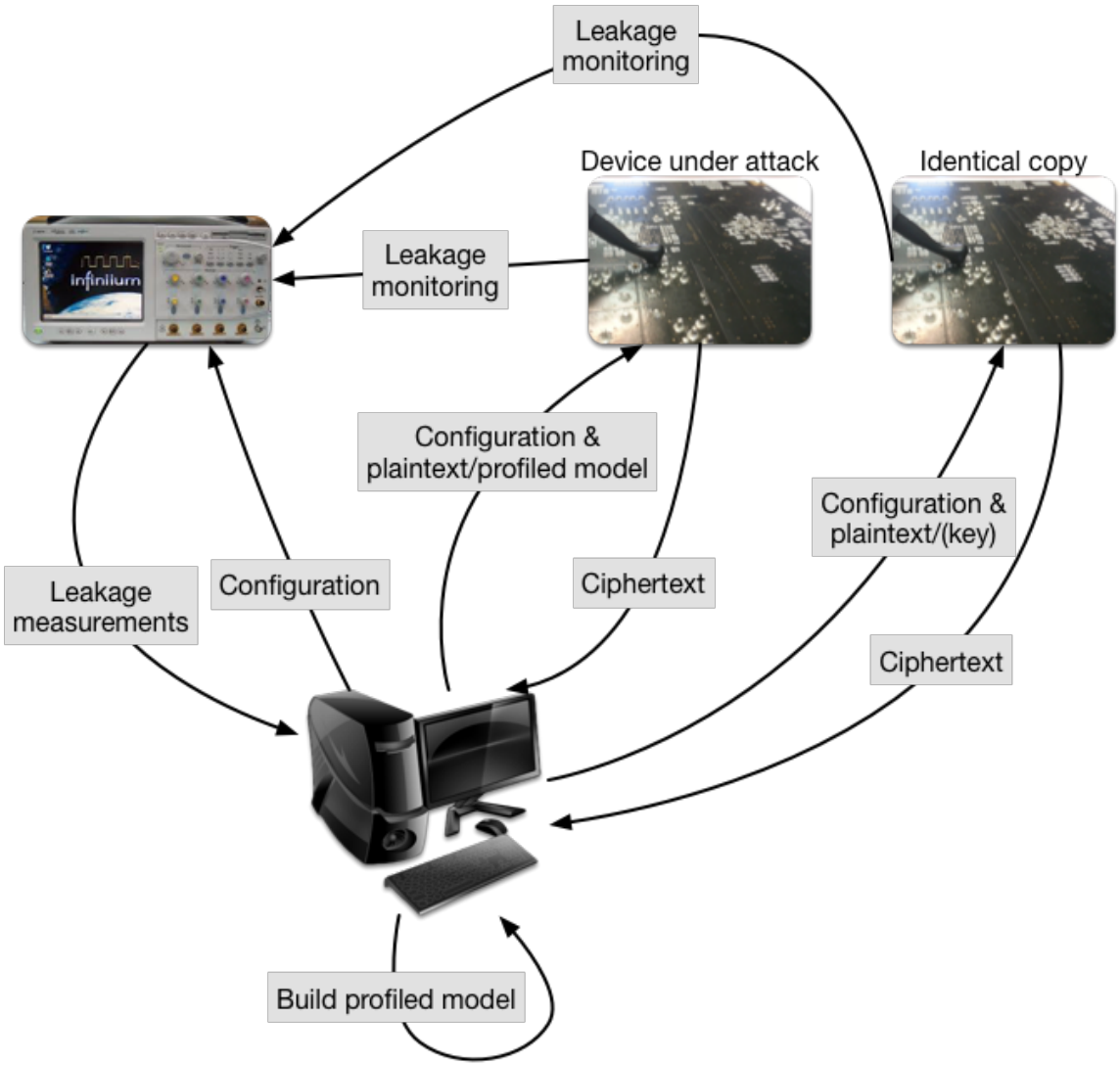
Implementation Attacks and SCA

Implementation attacks

Implementation attacks do not aim at the weaknesses of the algorithm, but on its implementation.

- **Side-channel attacks** (SCAs) – passive, non-invasive attacks.
- SCAs – one of the most powerful category of attacks on crypto devices.
- Profiled attacks – the most powerful among SCAs.
- Within profiling phase the adversary estimates leakage models for targeted intermediate computations, which are exploited to extract secret information in the actual attack phase.

Profiled Attacks



Profiled Attacks

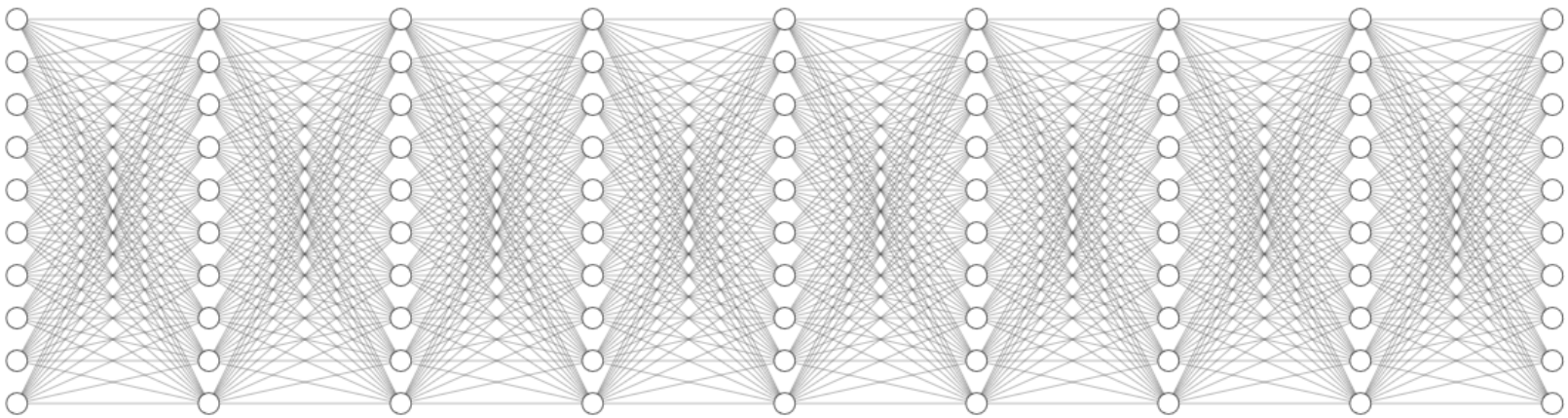
- Template Attack (TA) is the most powerful attack from the information theoretic point of view.
- Some machine learning (ML) techniques also belong to the profiled attacks.
- Deep learning has been shown to be able to reach top performance even if the device is protected with countermeasures.

SCA and Machine Learning

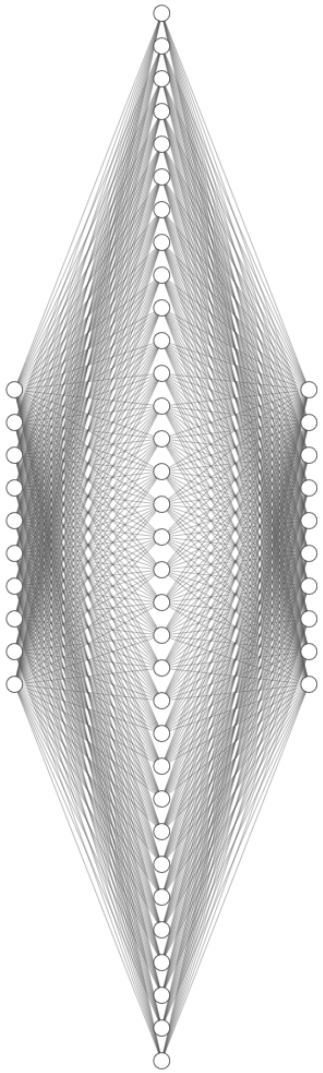
Table: Overview of profiling side-channel attacks used in literature (up to March 2019 and limited to symmetric key crypto).

Algorithm	Reference
Naive Bayes and its variants	[1, 2, 3, 4, 5, 6]
Random Forest	[2, 3, 4, 7, 8, 6, 9, 10, 11, 12, 13, 14]
Rotation Forest	[15, 4, 5, 16]
XGB	[5]
MultiBoost	[15]
Self-organizing maps	[9]
Support Vector Machines	[15, 4, 7, 8, 6, 17, 18, 9, 10, 11, 12, 19, 13, 20, 16]
Multivariate regression analysis	[21, 11, 12]
Multilayer Perceptron	[2, 3, 5, 7, 8, 6, 22, 23, 24, 25, 26, 27, 28]
Convolutional Neural Networks	[8, 5, 7, 29, 30, 22, 28]
Autoencoders	[8]
Recurrent Neural Networks	[8]
Template Attack and its variants	[1, 15, 4, 7, 8, 29, 30, 6, 17, 9, 10, 11, 12, 19, 13, 28, 16]
Stochastic attack	[11, 12, 7]

Multilayer Perceptron - “Many” Hidden Layers



Multilayer Perceptron - One Hidden Layer



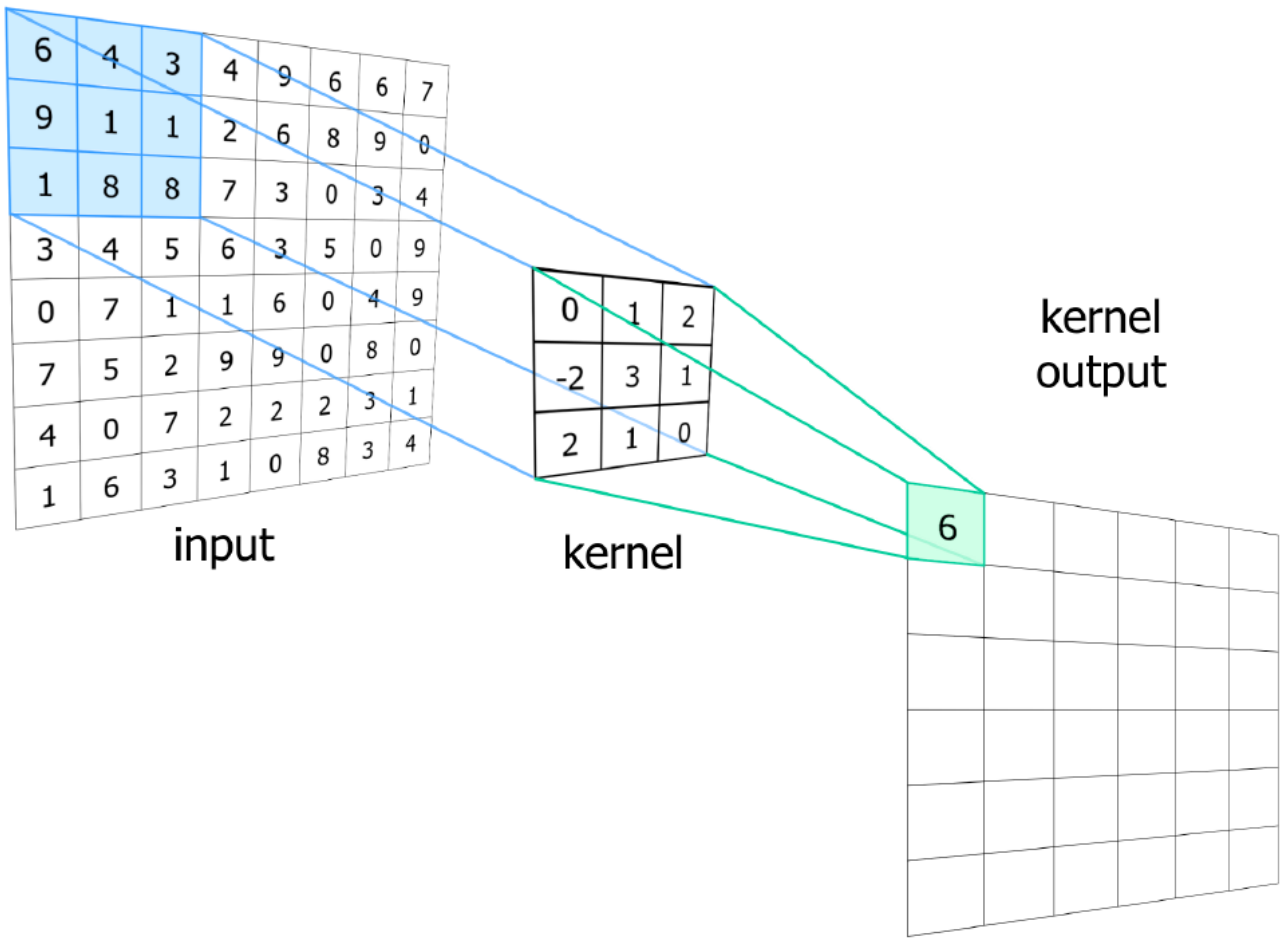
Universal Approximation Theorem

- A feed-forward network with a single hidden layer containing a finite number of neurons can approximate continuous functions on compact subsets of $\mathbb{R}^n$.
- Given enough hidden units and enough data, multilayer perceptrons can approximate virtually any function to any desired accuracy.
- Valid results if and only if there is a sufficiently large number of training data in the series.

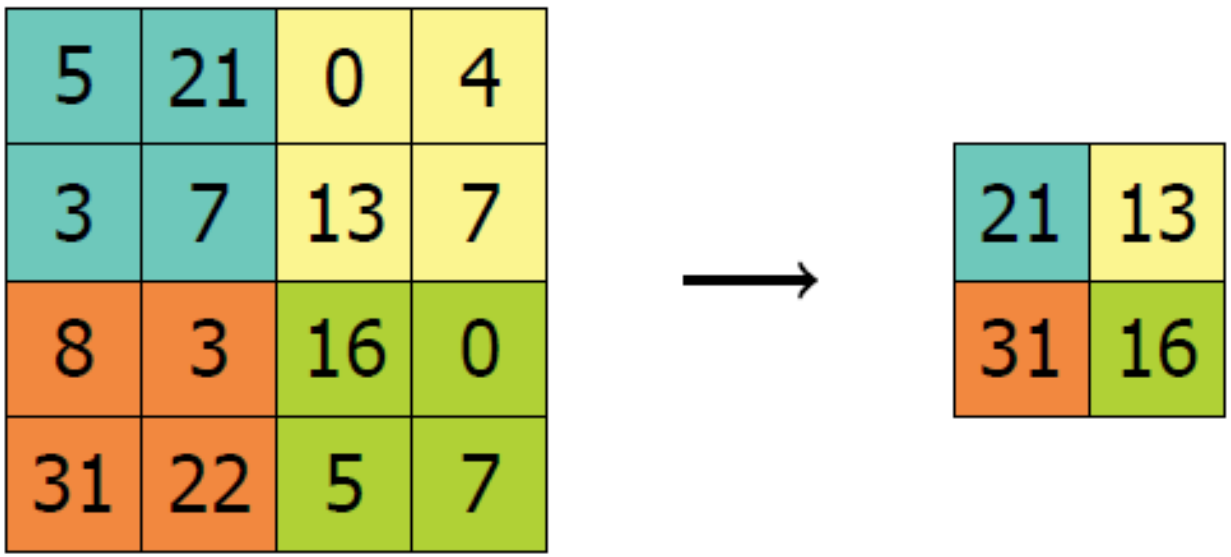
Convolutional Neural Networks

- CNNs represent a type of neural networks which were first designed for 2-dimensional convolutions.
- They are primarily used for image classification but lately, they have proven to be powerful classifiers in other domains.
- From the operational perspective, CNNs are similar to ordinary neural networks: they consist of a number of layers where each layer is made up of neurons.
- CNNs use three main types of layers: convolutional layers, pooling layers, and fully-connected layers.

Convolutional Neural Networks - Convolution Layer



Convolutional Neural Networks - Pooling



Design Principle - VGG Like CNN

- Small kernel size: 3×3 for every layer.
- Max pooling with 2 x 2 windows, with stride 2.
- Increasing number of filters per layer: doubled after every max pooling layer.
- Convolutional blocks are added until the spatial dimension is reduced to 1.
- After the fully connected layers is the output layey.
- The convolutional and fully connected layers use ReLu activations, the output layer uses Softmax to normalize the predictions.

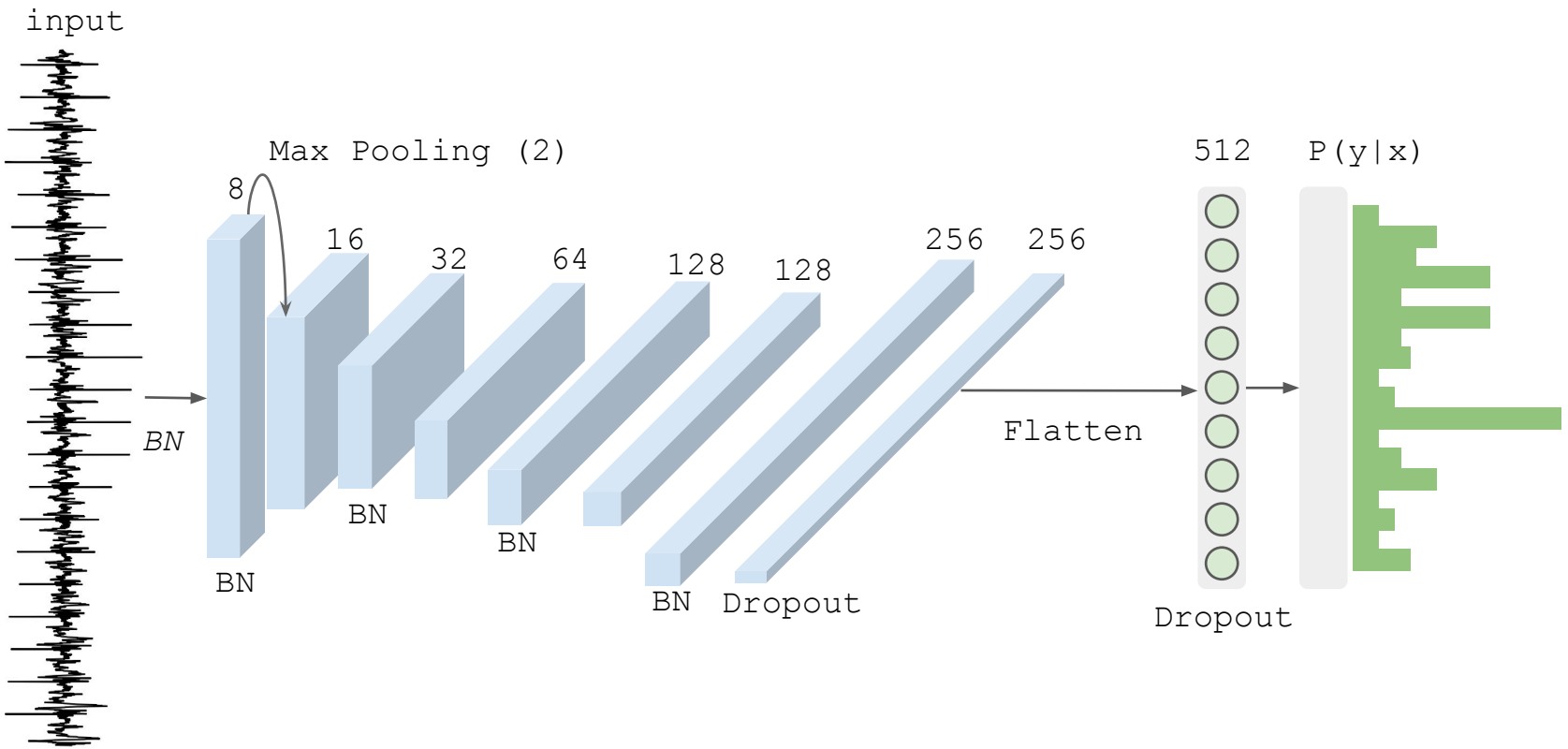
Design Principle - VGG Like CNN

$$\texttt{net} = \texttt{fc}_{\theta,\texttt{softmax}} \circ \prod_{p=1}^{P} \texttt{fc}_{\theta^p,\texttt{ReLU}} \circ \prod_{q=1}^{Q} \left(\texttt{pool}_{\texttt{Max}} \circ \prod_{r=1}^{R_q} \texttt{conv}_{\phi^r,\texttt{ReLU}} \right), \tag{1}$$

$$\texttt{conv}_{\phi,\sigma}(X) = \sigma(\phi * X), \tag{2}$$

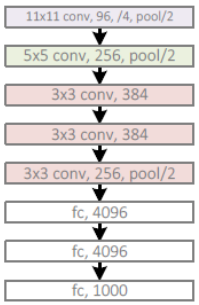
$$\texttt{fc}_{\theta,\sigma}(x) = \sigma(\theta^{\mathsf{T}} x). \tag{3}$$

Convolutional Neural Networks - Final

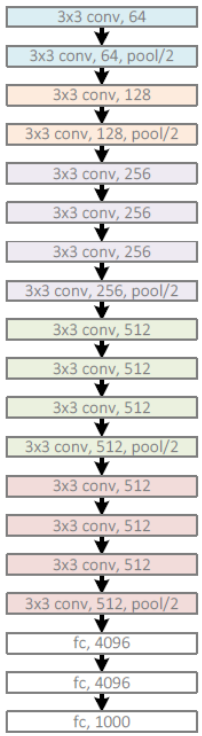


More Complex Architectures

AlexNet, 8 layers
(ILSVRC 2012)



VGG, 19 layers
(ILSVRC 2014)



GoogleNet, 22 layers
(ILSVRC 2014)

More Complex Architectures

AlexNet, 8 layers
(ILSVRC 2012)

VGG, 19 layers
(ILSVRC 2014)

ResNet, 152 layers
(ILSVRC 2015)

Adding Noise

- To reduce the overfitting of the model, we introduce noise to the training phase.
- Since in our case, the input normalization is also learned during the training process via the BN layer, we added the noise tensor after the first BN.

$$X^* = BN_0(X) + \Psi, \quad \Psi \sim \mathcal{N}(0, \alpha). \tag{4}$$

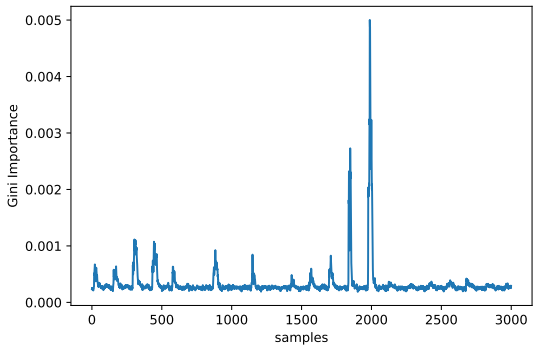
- The noise tensor follows the normal distribution.

Datasets

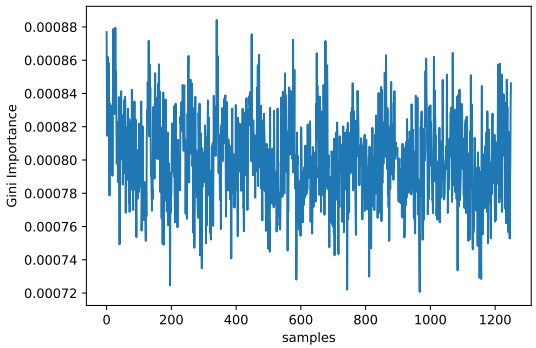
Table: Statistical information for publicly available datasets.

Dataset	Nr measurements	Nr features	SNR	Countermeasure
DPAcontest v4	100 000	4 000	5.8577	–
AES_HD	100 000	1 250	0.0096	–
AES_RD	50 000	3 500	0.0556	Random delay
ASCAD	60 000	700	$\approx 0.8/0$	Masking

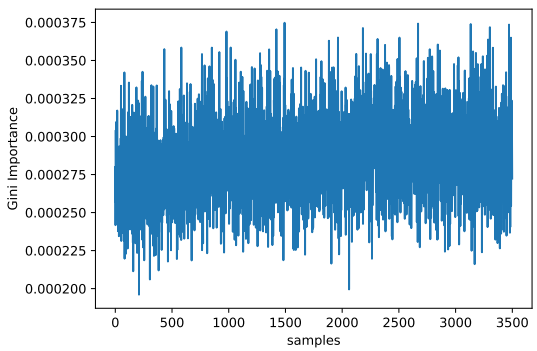
Datasets



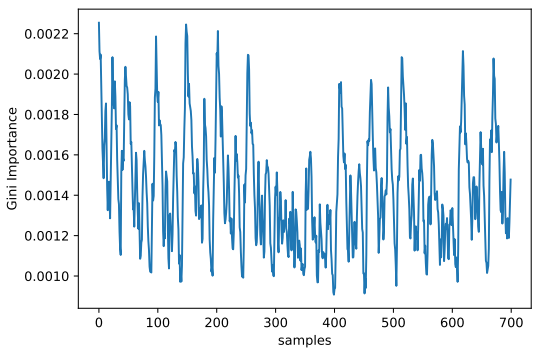
(a) DPAcontest v4 dataset



(b) AES_HD dataset

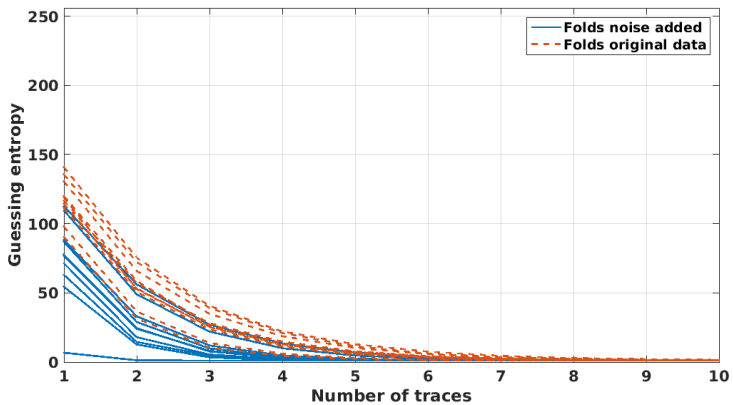


(c) Random Delay dataset

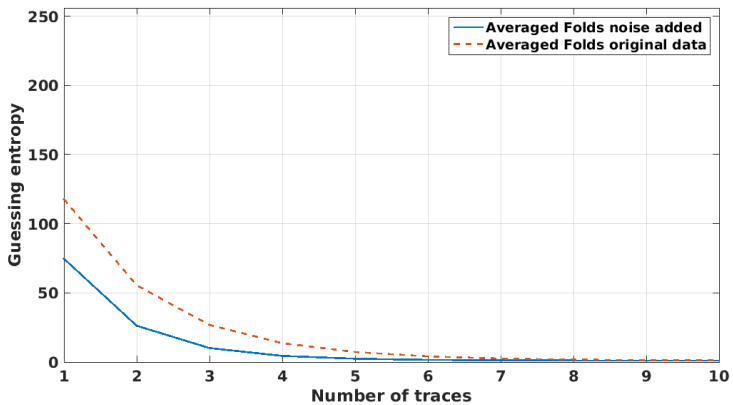


(d) ASCAD dataset

Results DPAv4

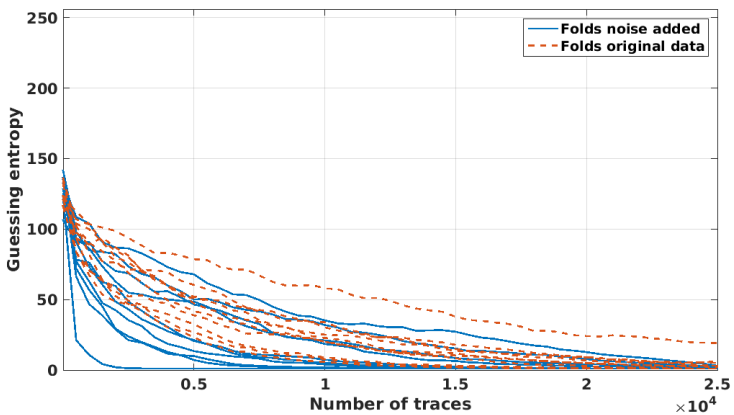


(e) RD network per fold

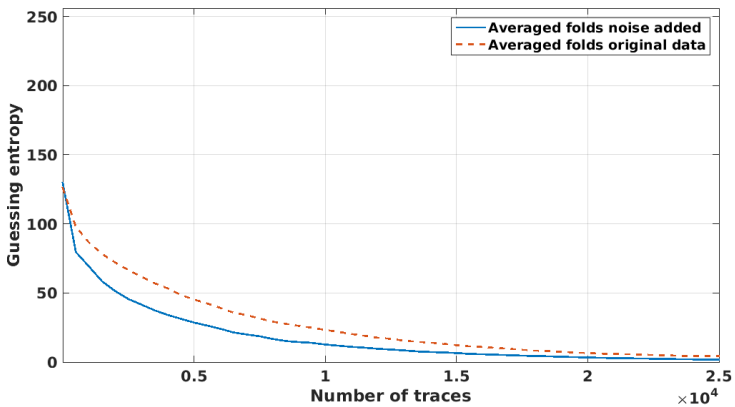


(f) RD network averaged

Results AES_HD

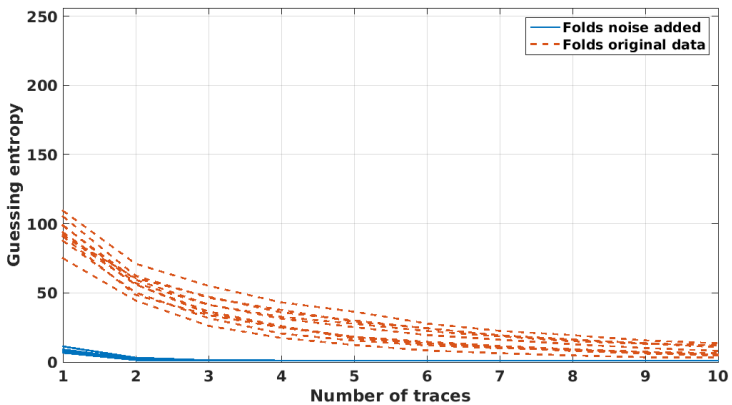


(g) ASCAD network per fold

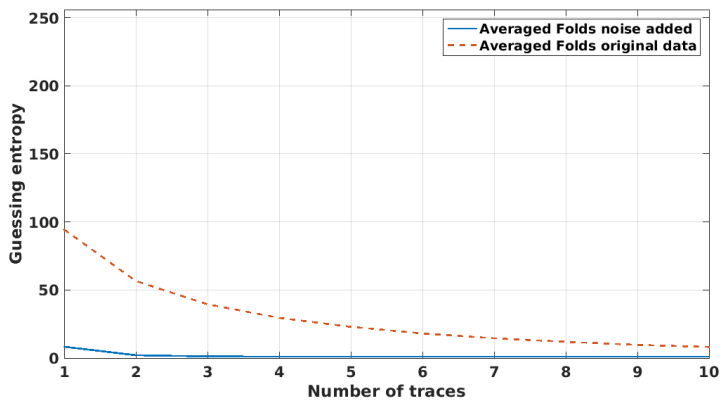


(h) ASCAD network averaged

Results AES_RD

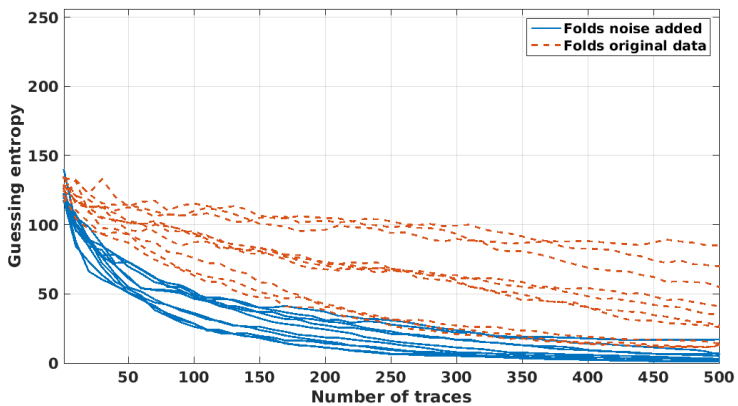


(i) RD network per fold

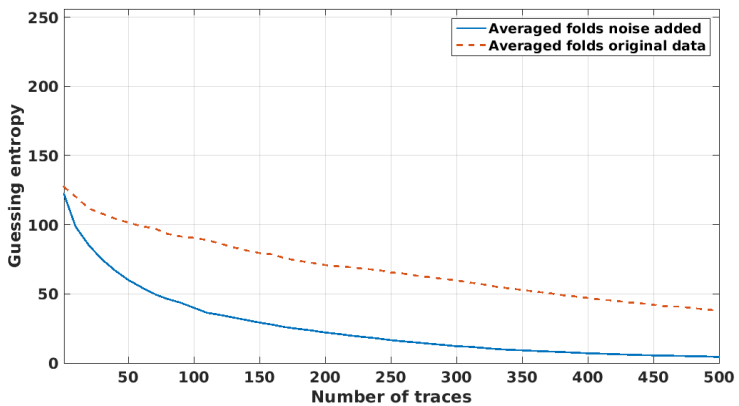


(j) RD network averaged

Results ASCAD



(k) ASCAD network per fold

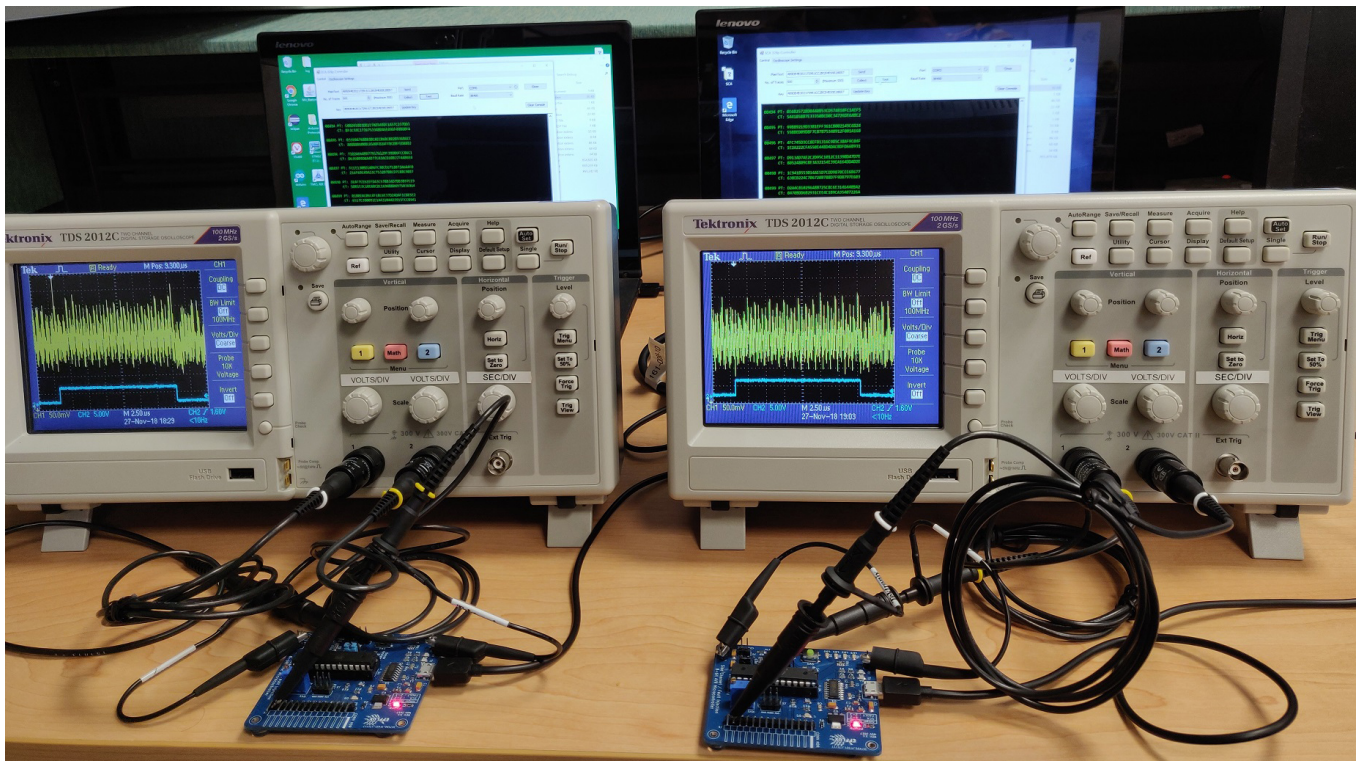


(l) ASCAD network averaged

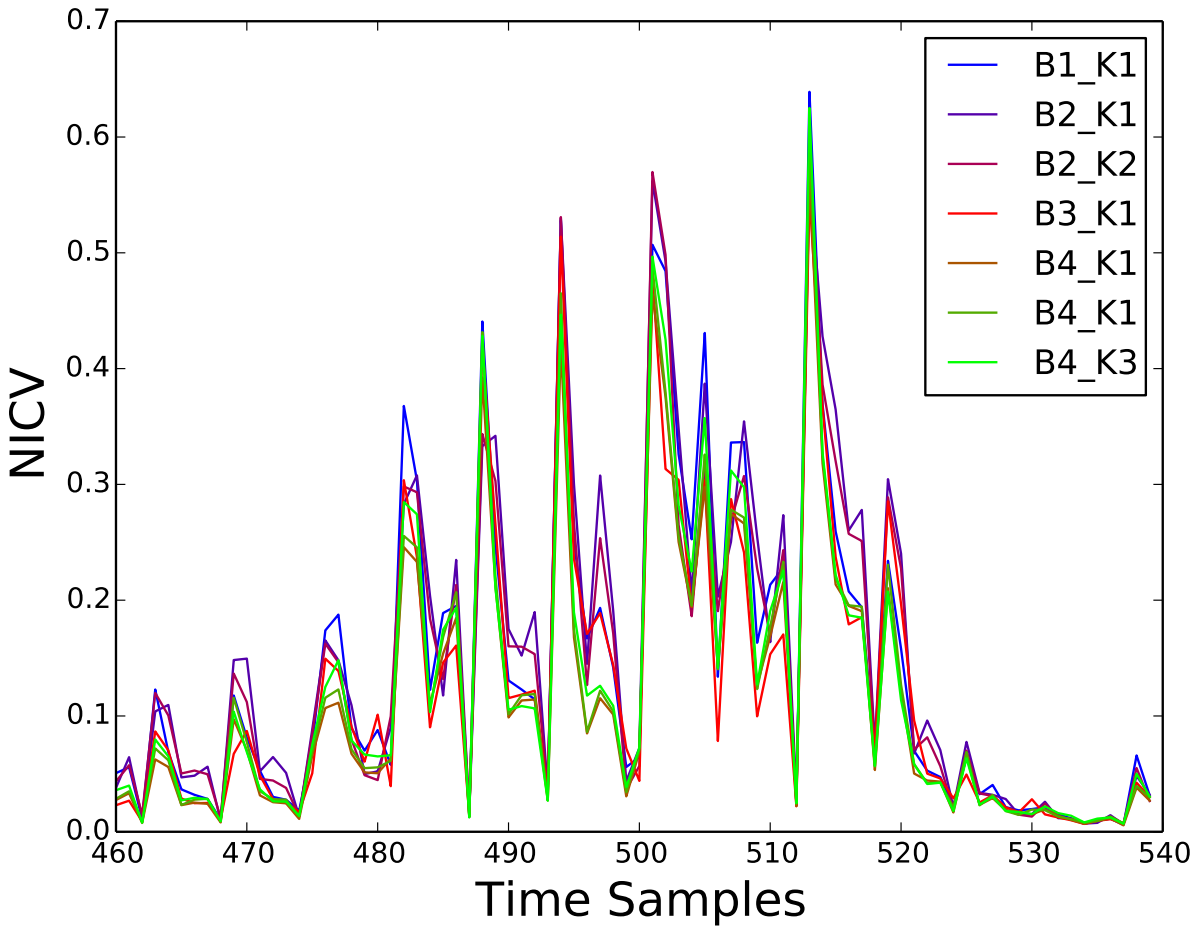
Profiling Attacks

- There are two devices: one for training and the second one for attack.
- Two devices, different keys.
- Usually, we make our lives simpler and assume only one device and the same key.
- It is the same?
- Or not?

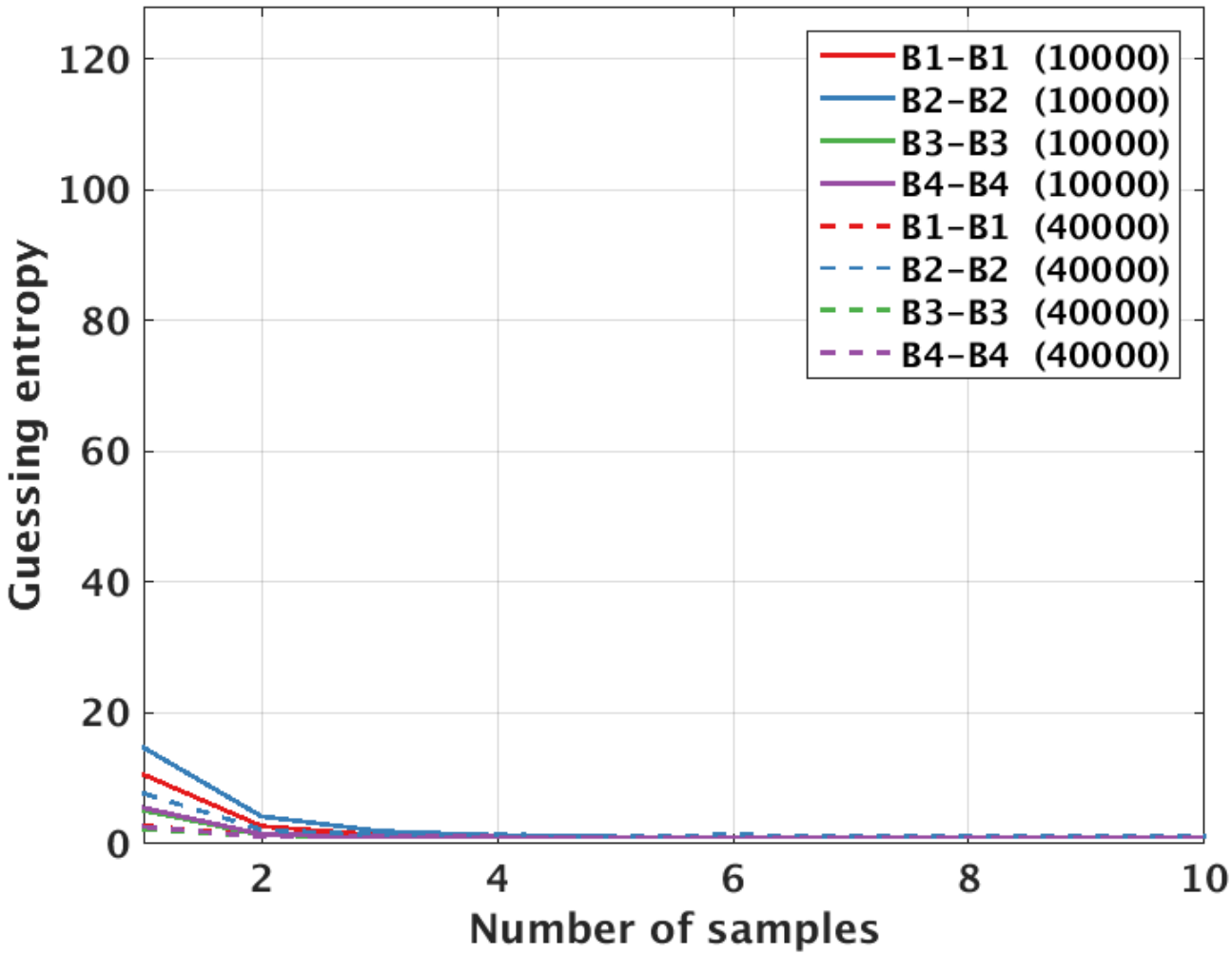
Setup



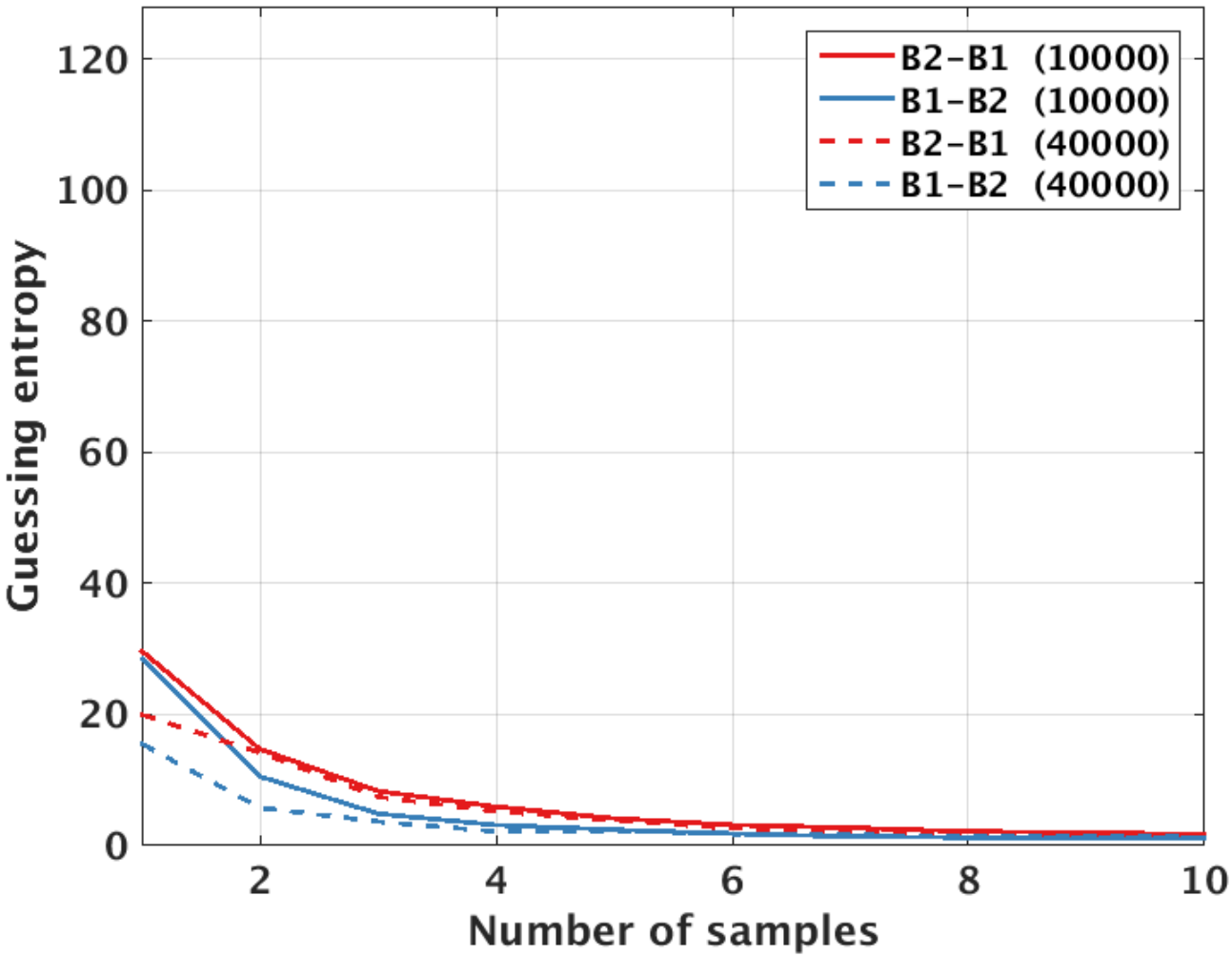
NICV


NICV
Time Samples
B1_K1
B2_K1
B2_K2
B3_K1
B4_K1
B4_K1
B4_K3
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
460
470
480
490
500
510
520
530
540

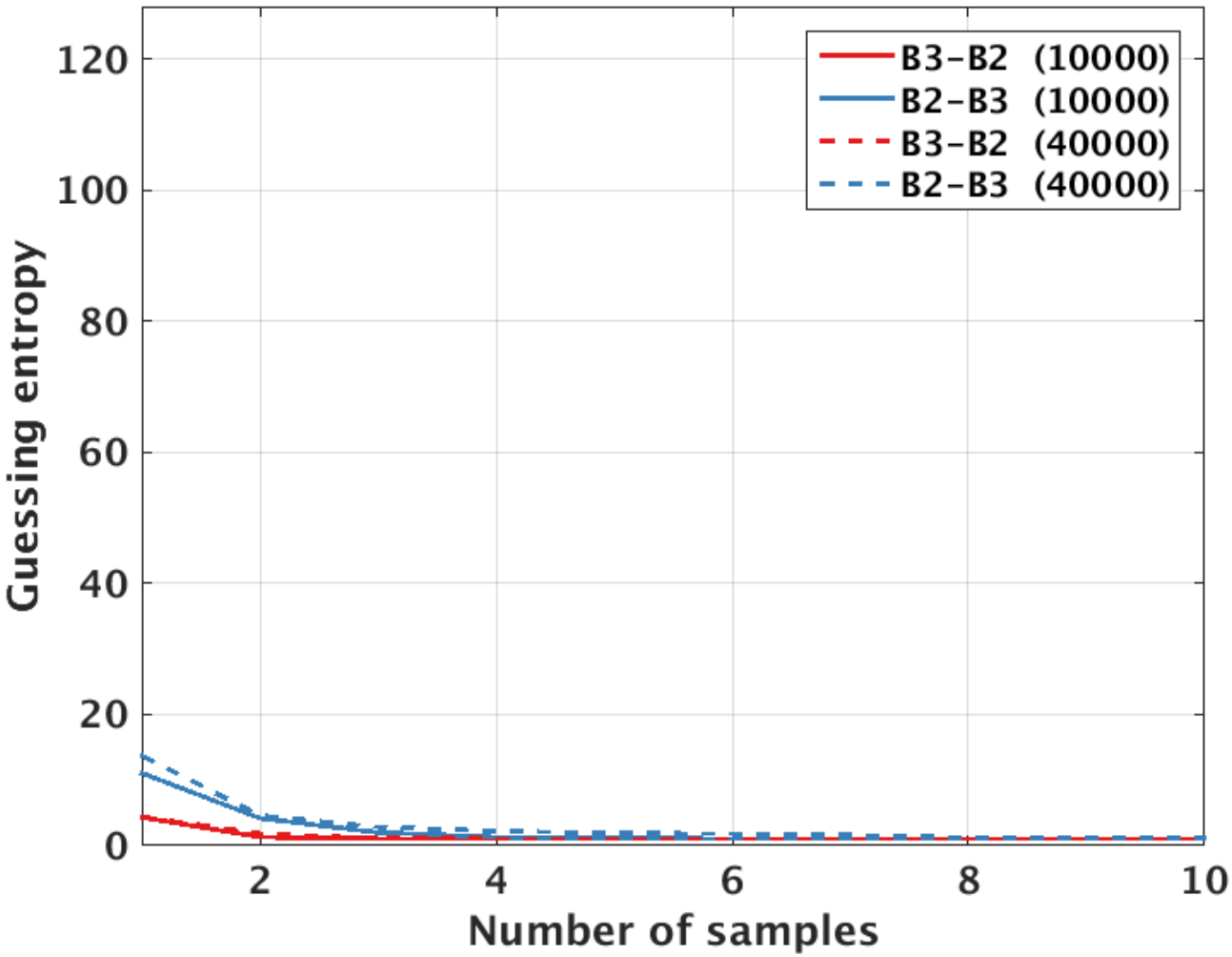
Same Key and Device



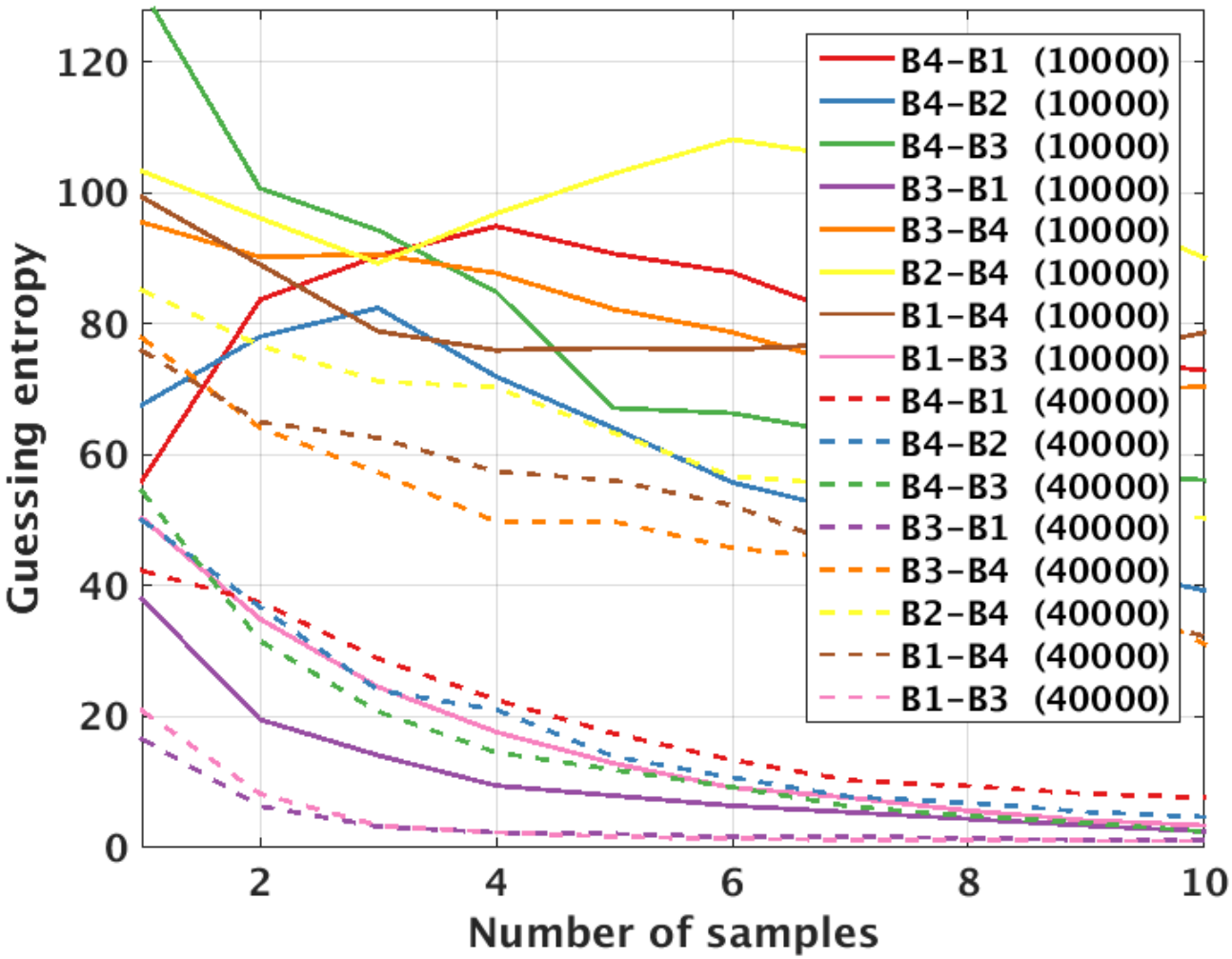
Same Key and Different Device



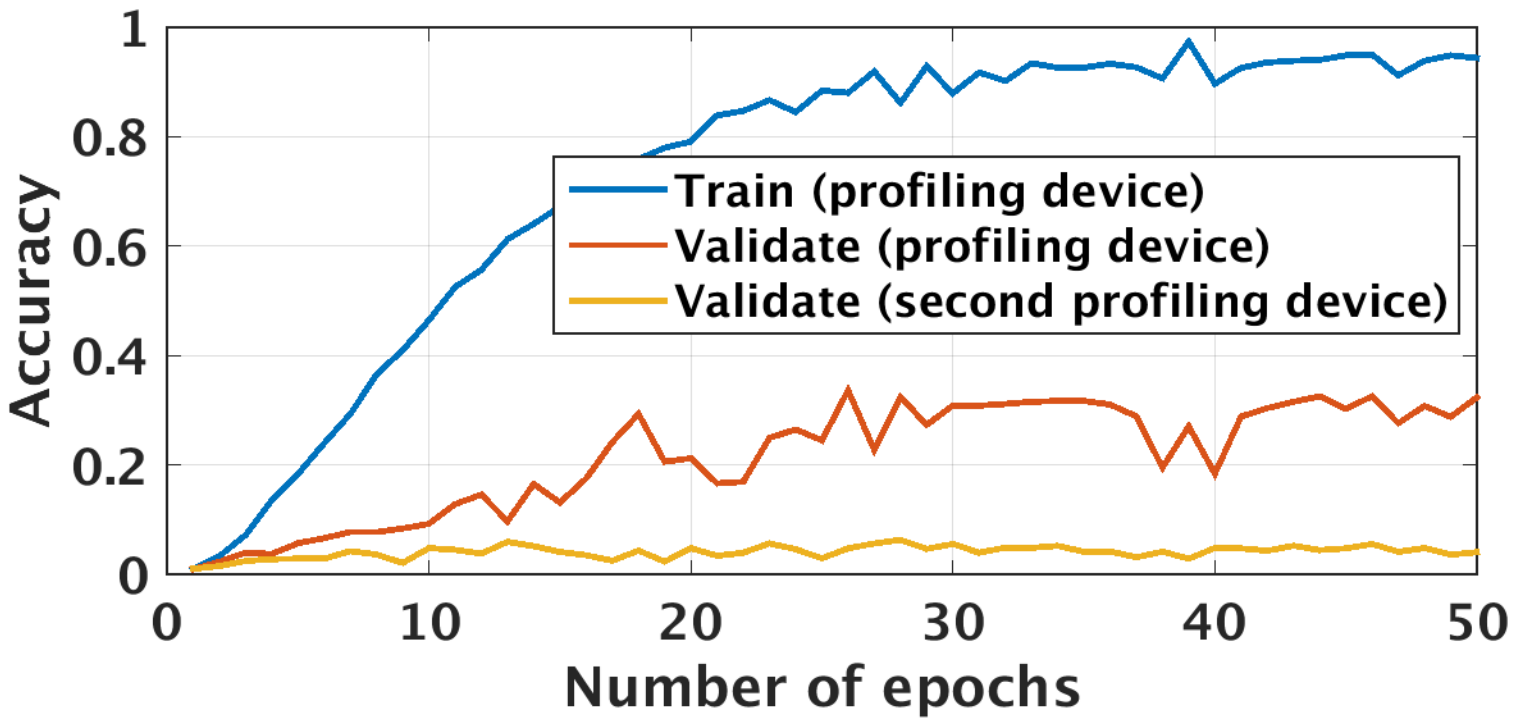
Different key and Same Device



Different key and Device



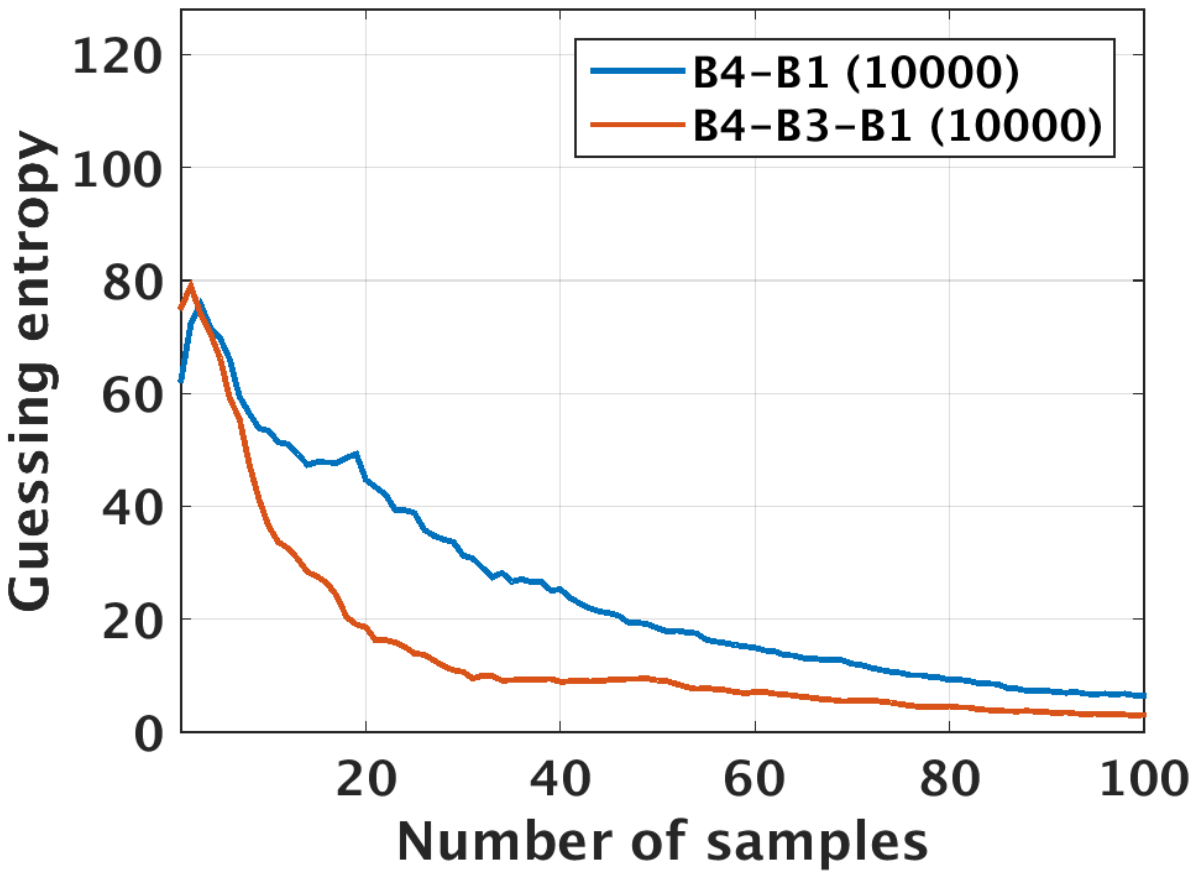
Validation


Accuracy
1
0.8
0.6
0.4
0.2
0
0
10
20
30
40
50
Number of epochs
Train (profiling device)
Validate (profiling device)
Validate (second profiling device)

Multiple Device Model

- Instead of validating on the same device as training, we need one more device!
- Separate devices for train, validation, attack.
- If we do not have a third device, we can use artificial noise.

Multiple Device Model



Problems and "Problems"

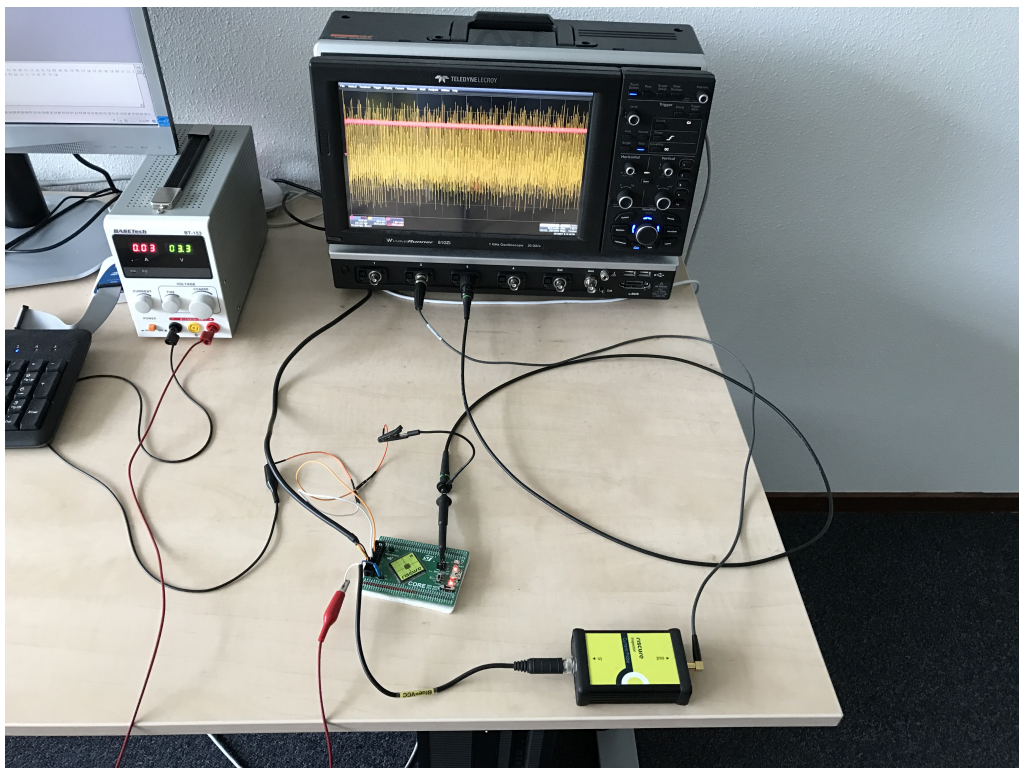
- Overfitting and underfitting.
- Selection of ML techniques and hyper-parameter tuning.
- Portability.
- Lack of frameworks/reproducibility.
- Lack of datasets.
- Explainability.
- Still no clear connection between machine learning and side-channel analysis metrics.
- Countermeasures.
- ...

Setup

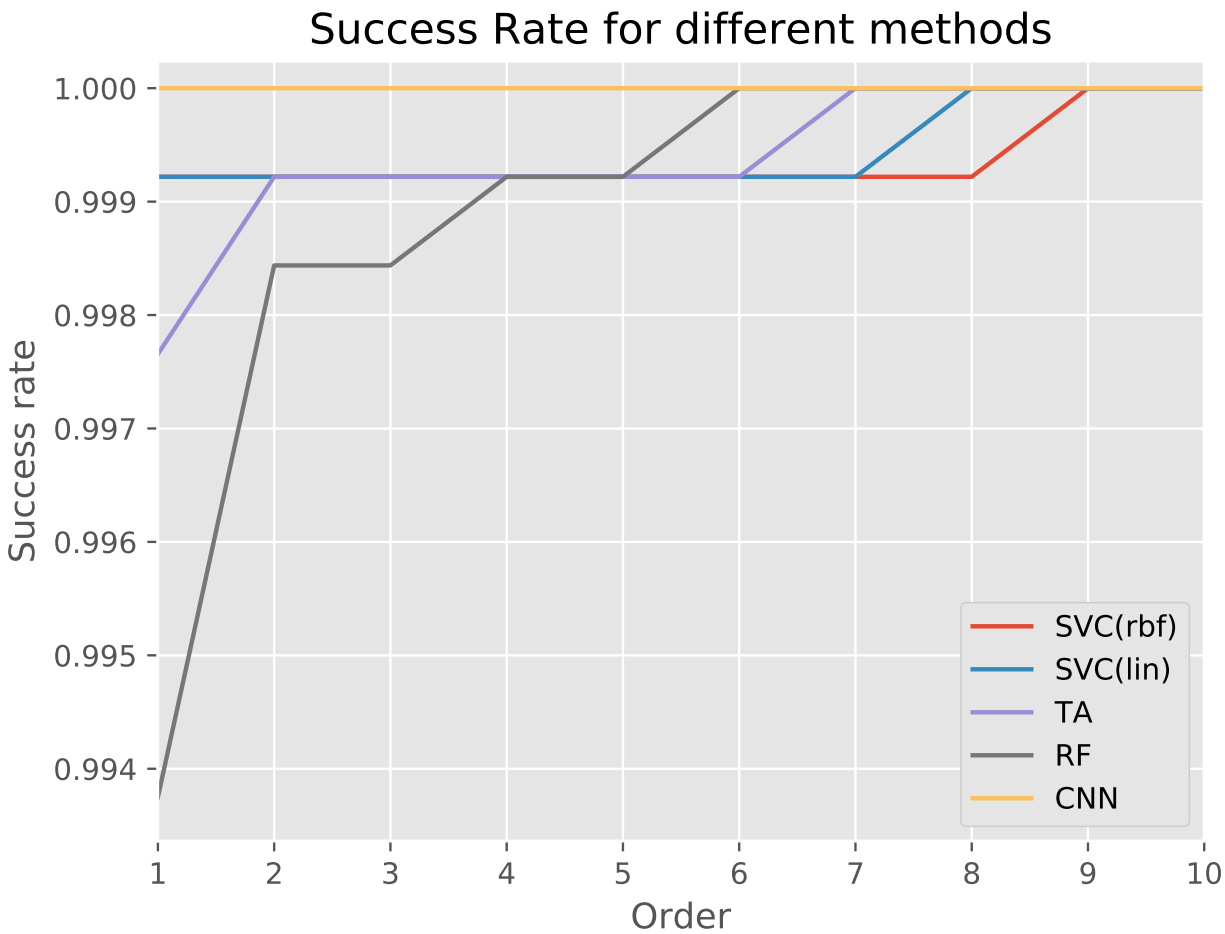
- We mount a power analysis attack on EdDSA using the curve Curve25519 as implemented in WolfSSL.
- EdDSA is a variant of the Schnorr digital signature scheme using Twisted Edward Curves, a subgroup of elliptic curves that uses unified formulas, enabling speed-ups for specific curve parameters.
- The aim of the attacker is the same as for every ECDSA attack: recover the secret scalar a.

Setup

Figure: The measurement setup



State-of-the-art for ECC



Introduction

- A fault injection (FI) attack is successful if after exposing the device to a specially crafted external interference, it shows an unexpected behavior exploitable by the attacker.
- Insertion of signals has to be precisely tuned for the fault injection to succeed.
- Finding the correct parameters for a successful FI can be considered as a search problem.
- The search space is typically too large to perform an exhaustive search.

Verdict classes

- FI testing equipment can output only verdict classes that correspond to successful measurements.
- Several possible classes for classifying a single measurement:
  1. NORMAL: smart card behaves as expected and the glitch is ignored
  2. RESET: smart card resets as a result of the glitch
  3. MUTE: smart card stops all communication as a result of the glitch
  4. INCONCLUSIVE: smart card responds in a way that cannot be classified in any other class
  5. SUCCESS: smart card response is a specific, predetermined value that does not happen under normal operation

Approaches

- For voltage glitching and EMFI, we can use various heuristics, like genetic algorithms.
- For laser FI, the situation is more complex as laser can easily break the target so we use deep learning.

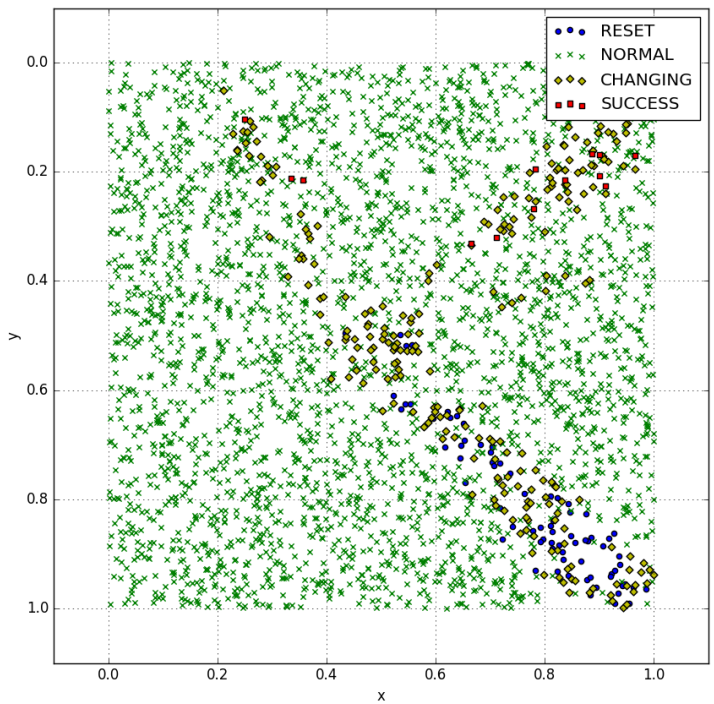
EMFI and Keccak

- We combine GA and local search.
- Approaches like exhaustive search difficult to work: the EMFI example would last 29 000 years.

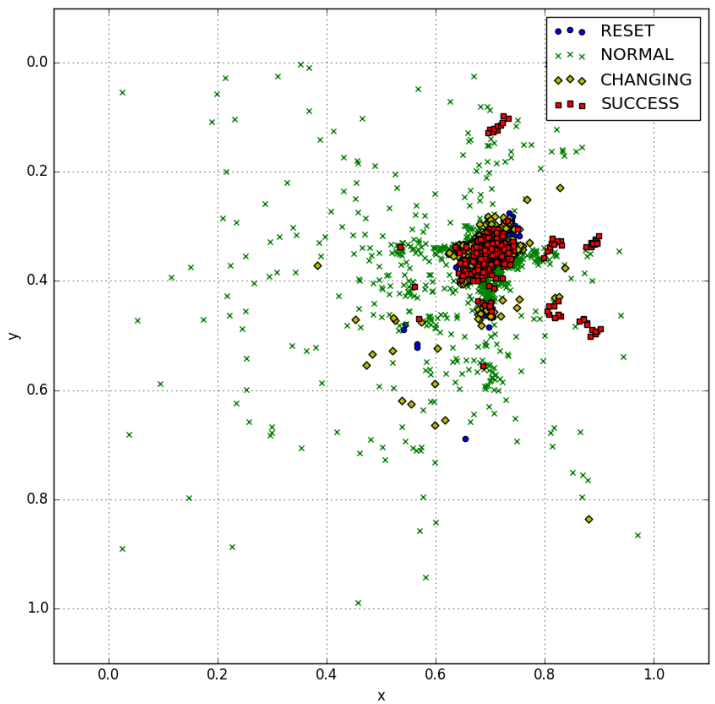
Table: Statistical results for GA and random search.

	GA	Random
NORMAL	662.8 (18.9%)	2 995.8 (90.7%)
RESET	496.4 (15.0%)	65.0 (2.0%)
CHANGING	375.2 (11.4%)	232.4 (7.0%)
SUCCESS	1 807.2 (54.7%)	8.8 (0.3%)

EMFI and Keccak

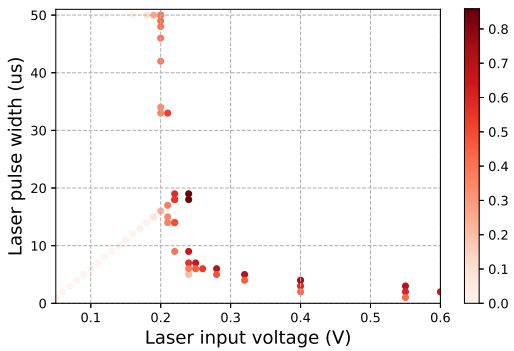


(a) Random search

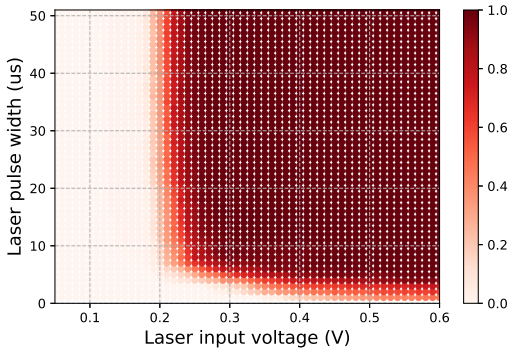


(b) GA and local search

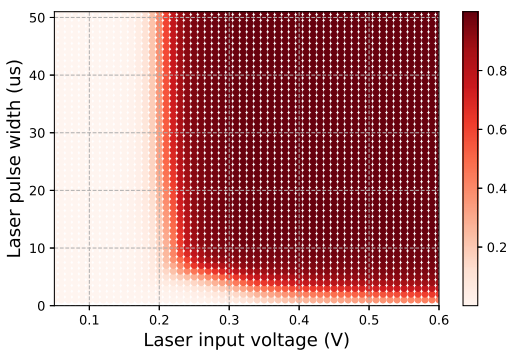
LFI and DES



(c) Characterization

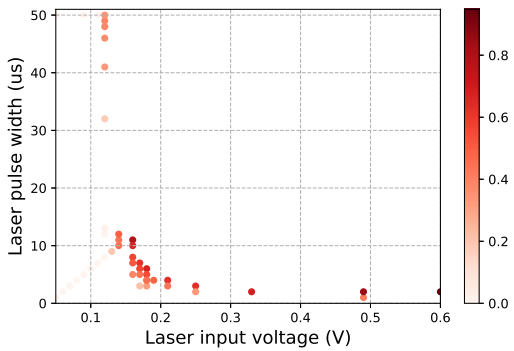


(d) Exhaustive search

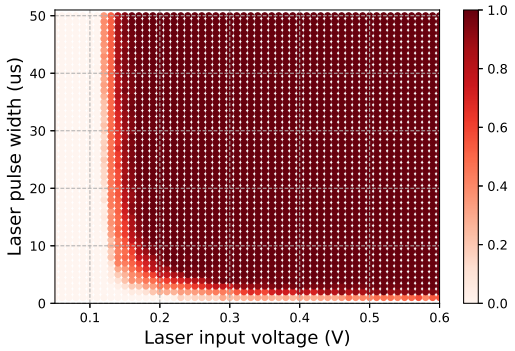


(e) Prediction with neural network

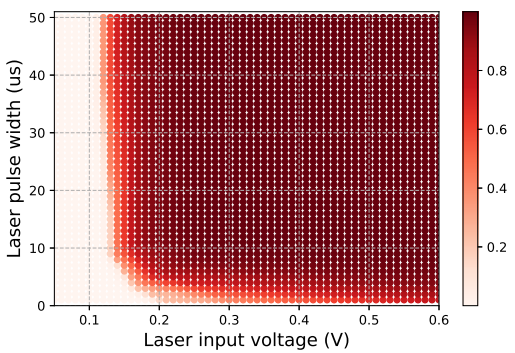
LFI and AES



(f) Characterization



(g) Exhaustive search



(h) Prediction with neural network

Outline

Machine Learning and Security

- Machine learning has become mainstream across industries.
- It is also widely used in security applications.
- Having strong ML models is an asset, on which many companies invest a significant amount of time and money to develop.
- How secure are such ML models against reverse engineering attacks?

Machine Learning and Security

- People investigate the leakage of sensitive information from machine learning models about individual data records.
- ML model provided by malicious attacker can give information about the training set.
- Reverse engineering of CNNs via timing and memory leakage.
- Exploits of the line buffer in a convolution layer of a CNN.

Neural Networks

- In this work, we consider neural networks.
- Multilayer perceptron and convolutional neural networks.
- We consider MLP and CNNs since: 1) they are commonly used machine learning algorithms in modern applications; 2) they consist of different types of layers that are also occurring in other architectures like recurrent neural networks; and 3) in the case of MLP, the layers are all identical, which makes it more difficult for SCA and could be consequently considered as the worst-case scenario.

Activation Functions

- An activation function of a node is a function f defining the output of a node given an input or set of inputs.
- Sigmoid, tanh, softmax, ReLU.

$$y = Activation(\sum(weight \cdot input) + bias). \tag{5}$$

Activation Functions

$$f(x) = \frac{1}{1 + e^{-x}}. \tag{6}$$

$$f(x) = tanh(x) = \frac{2}{1 + e^{-2x}} - 1. \tag{7}$$

$$f(\mathbf{x})_j = \frac{e^{x_j}}{\sum_{k=1}^{K} e^{x_k}}, \ for\ j = 1, \ldots, K. \tag{8}$$

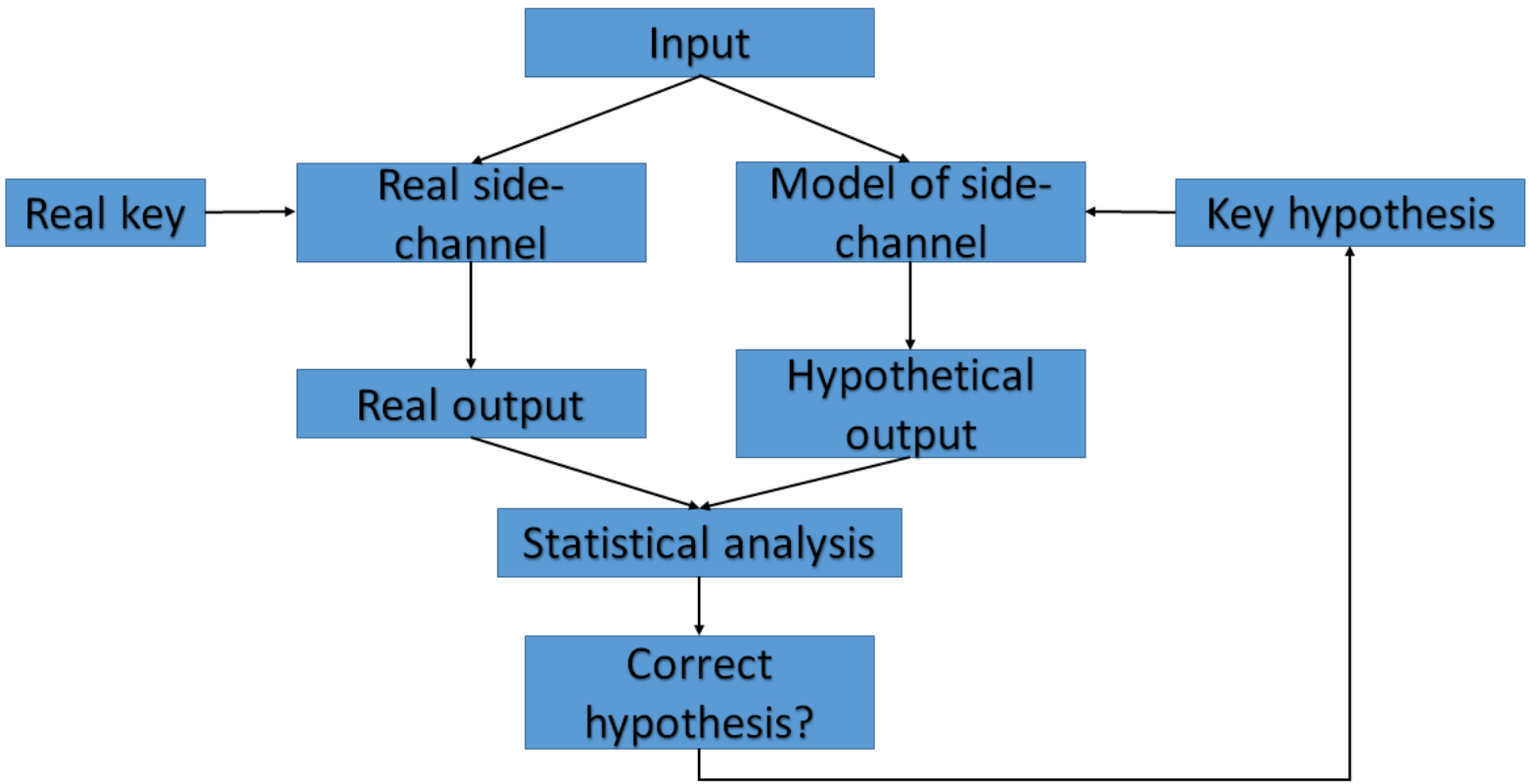
$$f(x) = max(0, x). \tag{9}$$

Side-channel Analysis

- DPA or DEMA is an advanced form of SCA, which applies statistical techniques to recover secret information from physical signatures.
- The attack normally tests for dependencies between actual physical signature (or measurements) and hypothetical physical signature, i.e., predictions on intermediate data. The hypothetical signature is based on a leakage model and key hypothesis.

Differential Power Analysis

- Statistical analysis of measurements.



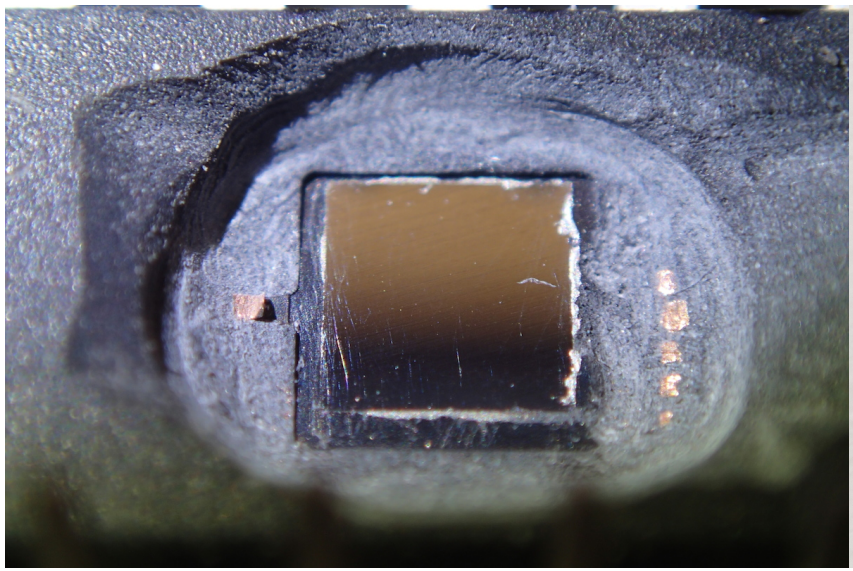
Threat Model

- Recover the neural network architecture using only side-channel information.
- No assumption on the type of inputs or its source, as we work with real numbers.
- We assume that the implementation of the machine learning algorithm does not include any side-channel countermeasures.

Attacker's Capability

- The attacker in consideration is a passive one.
- Acquiring measurements of the device while operating "normally" and not interfering with its internal operations by evoking faulty computations.
- Attacker does not know the architecture of the used network but can feed random (and hence known) inputs to the architecture.
- Attacker is capable of measuring side-channel information leaked from the implementation of the targeted architecture.
- Targets are Atmel ATmega328P and ARM Cortex-M3.

Setup

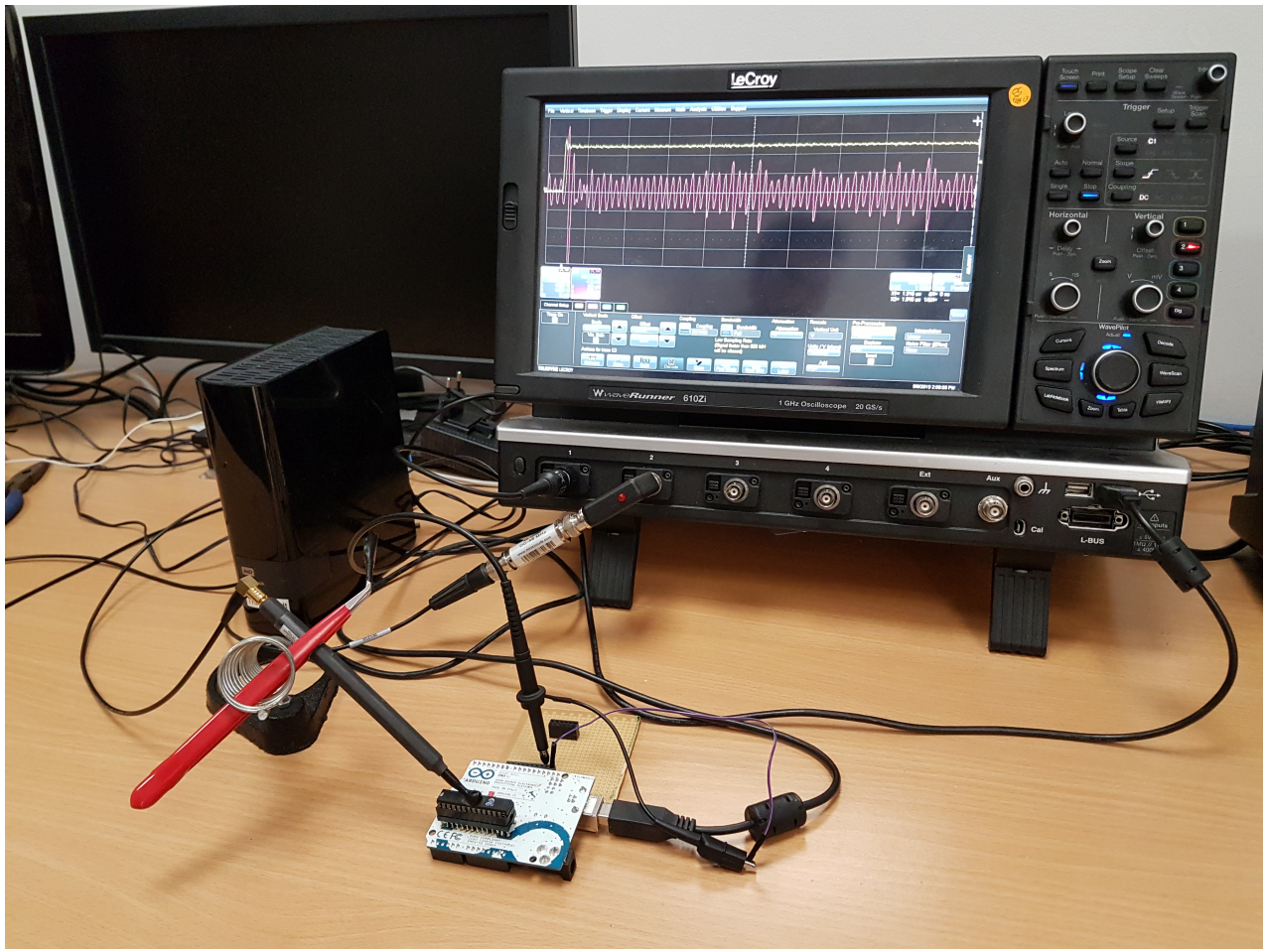


(i) Target 8-bit microcontroller mechanically decapsulated

(j) Langer RF-U 5-2 Near Field Electromagnetic passive Probe

Figure: Experimental Setup 1

Setup



(a) The complete measurement setup

Setup

- The exploited leakage model of the target device is the Hamming weight (HW) model.
- A microcontroller loads sensitive data to a data bus to perform indicated instructions.
- The training phase is conducted offline, and the trained network is then implemented in C language and compiled on the microcontroller.

$$HW(x) = \sum_{i=1}^{n} x_i \ , \tag{10}$$

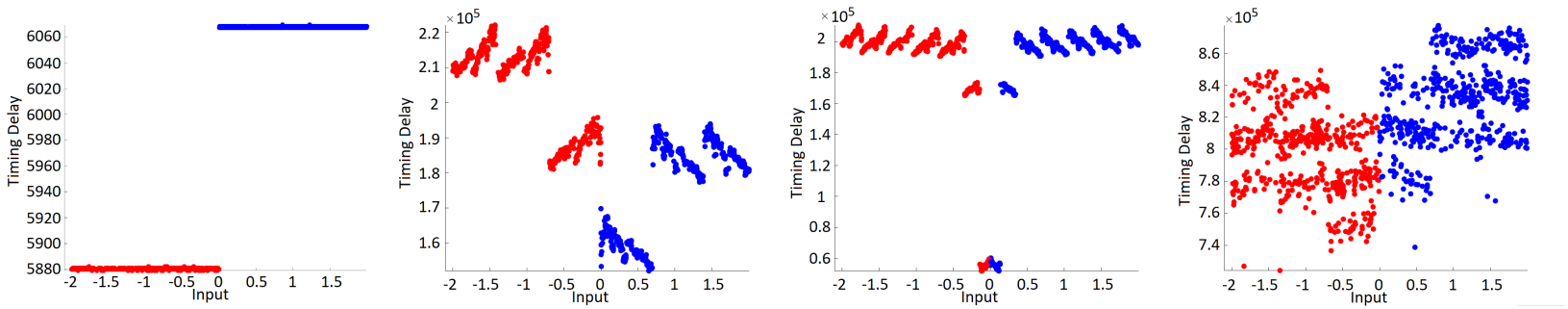
Reverse Engineering the Activation Functions

- The timing behavior can be observed directly on the EM trace.
- We collect EM traces and measure the timing of the activation function computation from the measurements.

Table: Minimum, Maximum, and Mean computation time (in *ns*) for different activation functions

Activation Function	Minimum	Maximum	Mean
ReLU	5 879	6 069	5 975
Sigmoid	152 155	222 102	189 144
Tanh	51 909	210 663	184 864
Softmax	724 366	877 194	813 712

Reverse Engineering the Activation Functions



(b) ReLU (c) Sigmoid (d) Tanh (e) Softmax

Figure: Timing behavior for different activation functions

Reverse Engineering the Multiplication Operation

- For the recovery of the weights, we use the Correlation Power Analysis (CPA) i.e., a variant of DPA using the Pearson's correlation as a statistical test.
- CPA targets the multiplication $m = x \cdot w$ of a known input x with a secret weight w.
- Using the HW model, the adversary correlates the activity of the predicted output m for all hypothesis of the weight.
- Thus, the attack computes $\rho(t, w)$, for all hypothesis of the weight w, where ρ is the Pearson correlation coefficient and t is the side-channel measurement.
- The correct value of the weight w will result in a higher correlation standing out from all other wrong hypotheses w^*, given enough measurements.

Reverse Engineering the Multiplication Operation

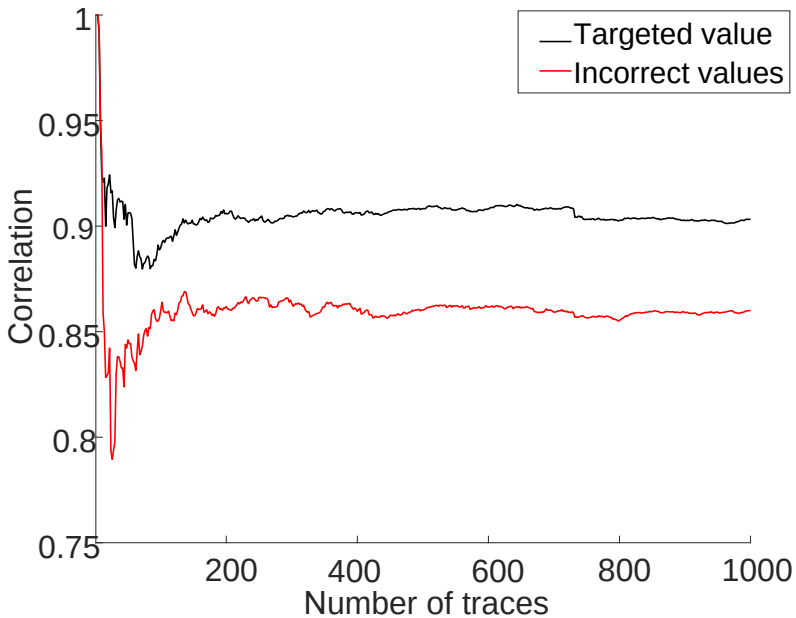
- We start by analyzing the way the compiler is handling floating-point operations for our target.
- The generated assembly confirms the usage of IEEE 754 compatible representation.
- Since the target device is an 8-bit microcontroller, the representation follows a 32-bit pattern $(b_{31}...b_0)$, being stored in 4 registers.
- The 32-bit consist of: 1 sign bit (b_{31}), 8 biased exponent bits $(b_{30}...b_{23})$ and 23 mantissa (fractional) bits $(b_{22}...b_0)$.

$$(-1)^{b_{31}} \times 2^{(b_{30}...b_{23})_2 - 127} \times (1.b_{22}...b_0)_2.$$

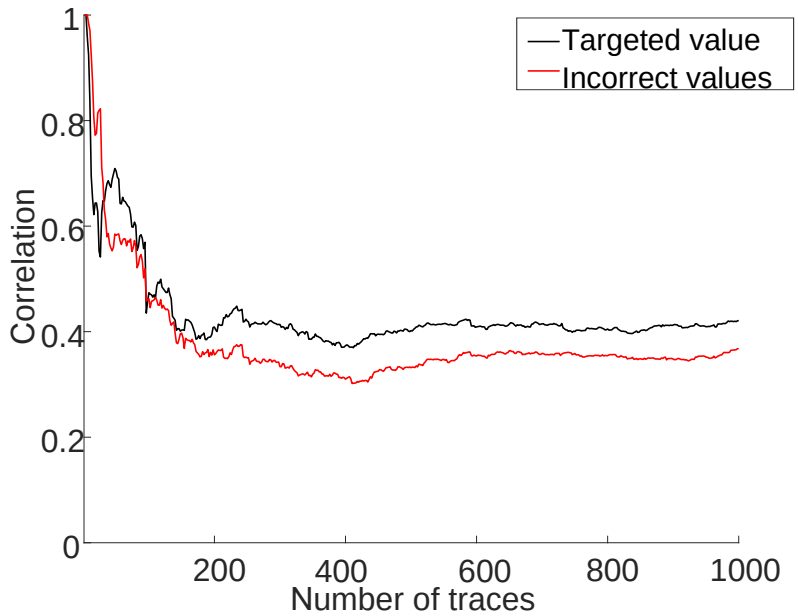
Reverse Engineering the Multiplication Operation

- We target the result of the multiplication m of known input values x and unknown weight w.
- For every input, we assume different possibilities for weight values.
- We then perform the multiplication and estimate the IEEE 754 binary representation of the output.
- Then, we perform the recovery of the 23-bit mantissa of the weight.
- The sign and exponent could be recovered separately.

Reverse Engineering the Multiplication Operation



(a) First byte recovery (sign and 7-bit exponent)

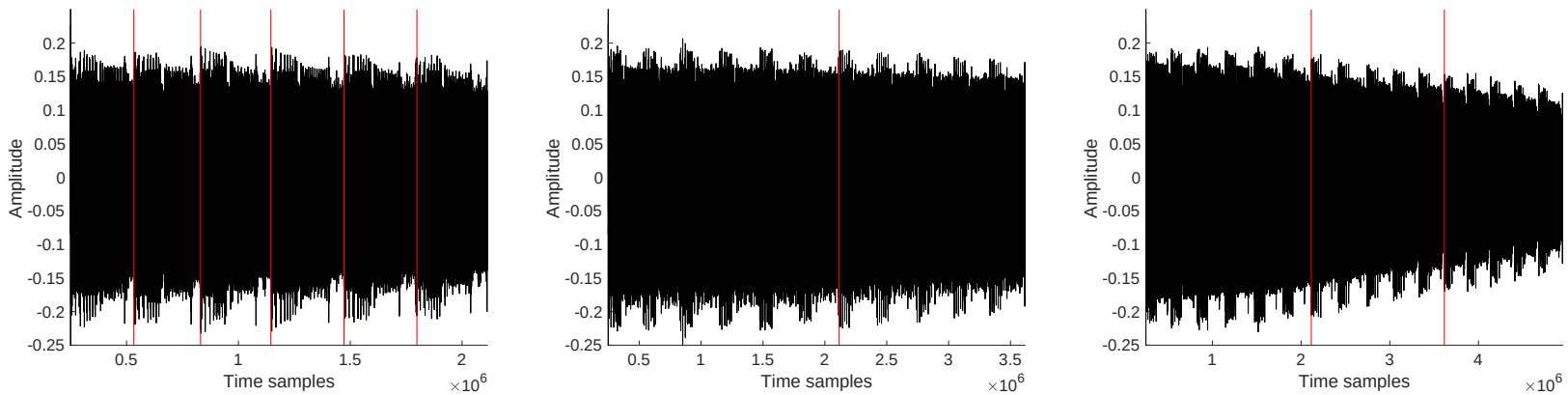


(b) Second byte recovery (lsb exponent and mantissa)

Figure: Recovery of the weight

Reverse Engineering the Number of Neurons and Layers

- To perform the reverse engineering of the network structure, we first use SPA (SEMA).



(a) One hidden layer with 6 neurons (b) 2 hidden layers (6 and 5 neurons each) (c) 3 hidden layers (6,5,5 neurons each)

Figure: SEMA on hidden layers

Reverse Engineering the Number of Neurons and Layers

- To determine if the targeted neuron is in the same layer as previously attacked neurons, or in the next layer, we perform a weight recovery using two sets of data.
- Let us assume that we are targeting the first hidden layer (the same approach can be done on different layers as well).
- Assume that the input is a vector of length N_0, so the input x can be represented $x = \{x_1, ..., x_{N_0}\}$.
- For the targeted neuron y_n in the hidden layer, perform the weight recovery on 2 different hypotheses.

Reverse Engineering the Number of Neurons and Layers

- For the first hypothesis, assume that the y_n is in the first hidden layer. Perform weight recovery individually using x_i, for $1 \leq i \leq N_0$.
- For the second hypothesis, assume that y_n is in the next hidden layer (the second hidden layer).
- Perform weight recovery individually using y_i, for $1 \leq i \leq (n - i)$.
- For each hypothesis, record the maximum (absolute) correlation value, and compare both.
- Since the correlation depends on both inputs to the multiplication operation, the incorrect hypothesis will result in a lower correlation value.

Recovery of the Full Network Layout

- The combination of previously developed individual techniques can thereafter result in full reverse engineering of the network.
- The full network recovery is performed layer by layer, and for each layer, the weights for each neuron have to be recovered one at a time.
- The first step is to recover the weight w_{L_0} of each connection from the input layer (L_0) and the first hidden layer (L_1).
- In order to determine the output of the sum of the multiplications, the number of neurons in the layer must be known.
- Using the same set of traces, timing patterns for different inputs to the activation function can be built.
- The same steps are repeated in the subsequent layers $L_1, ..., L_{N-1}$.

Reverse Engineering the Number of Neurons and Layers

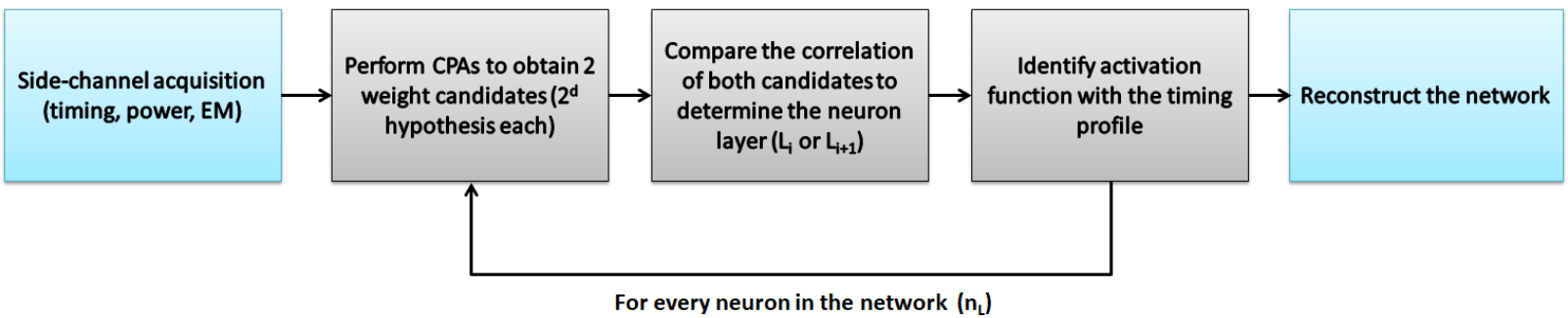
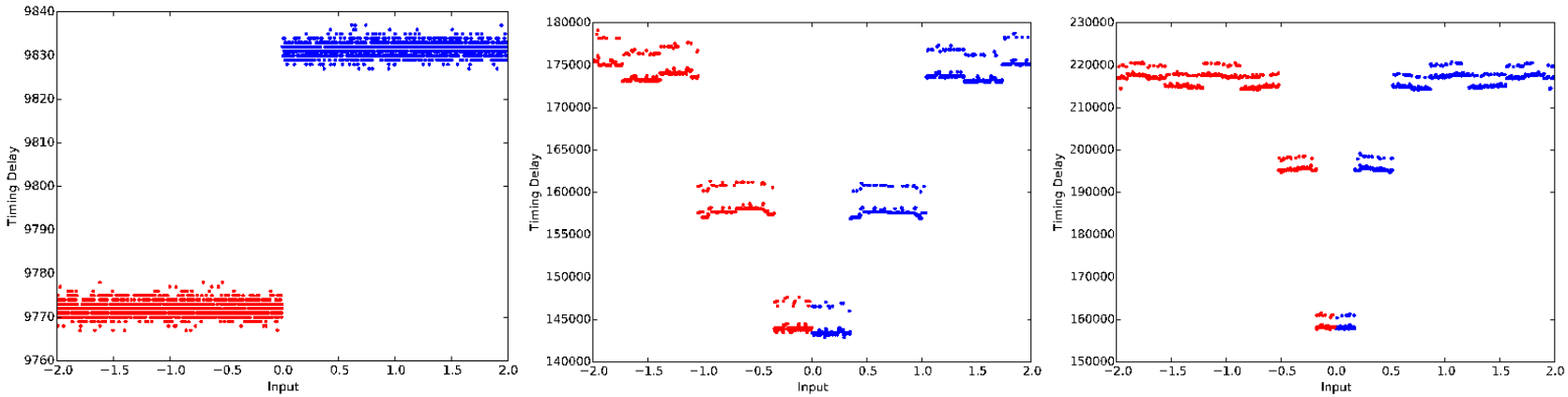


Figure: Methodology to reverse engineer the target neural network

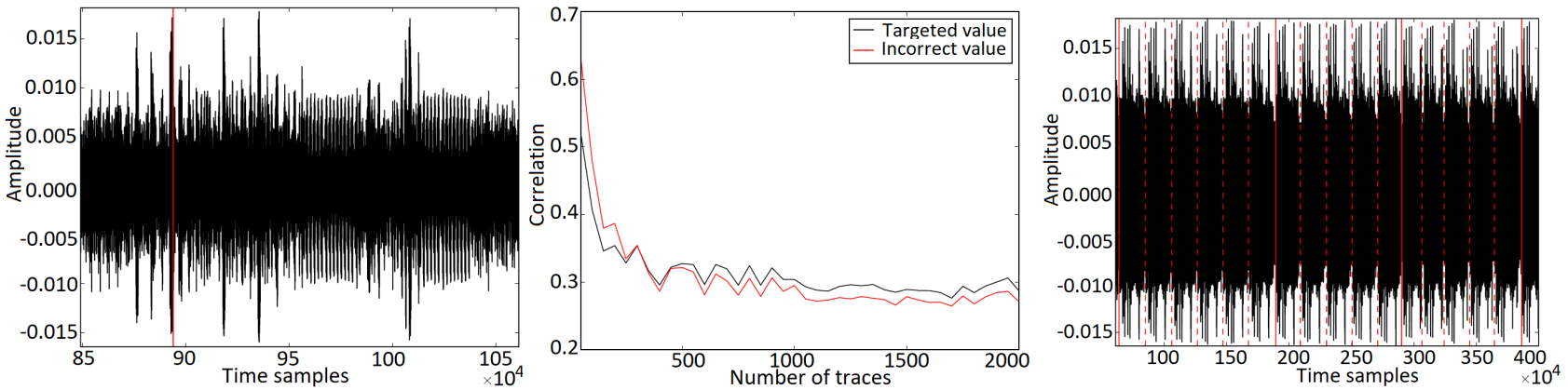
ARM Cortex M-3 and MLP



(a) ReLU (b) Sigmoid (c) Tanh

Figure: Timing behavior for different activation functions

ARM Cortex M-3 and MLP



(a) Observing pattern and timing of multiplication and activation function

(b) Correlation comparison between correct and incorrect mantissa for weight=2.453

(c) SEMA on hidden layers with 3 hidden layers (6,5,5 neurons each)

Figure: Analysis of an (6,5,5,) neural network

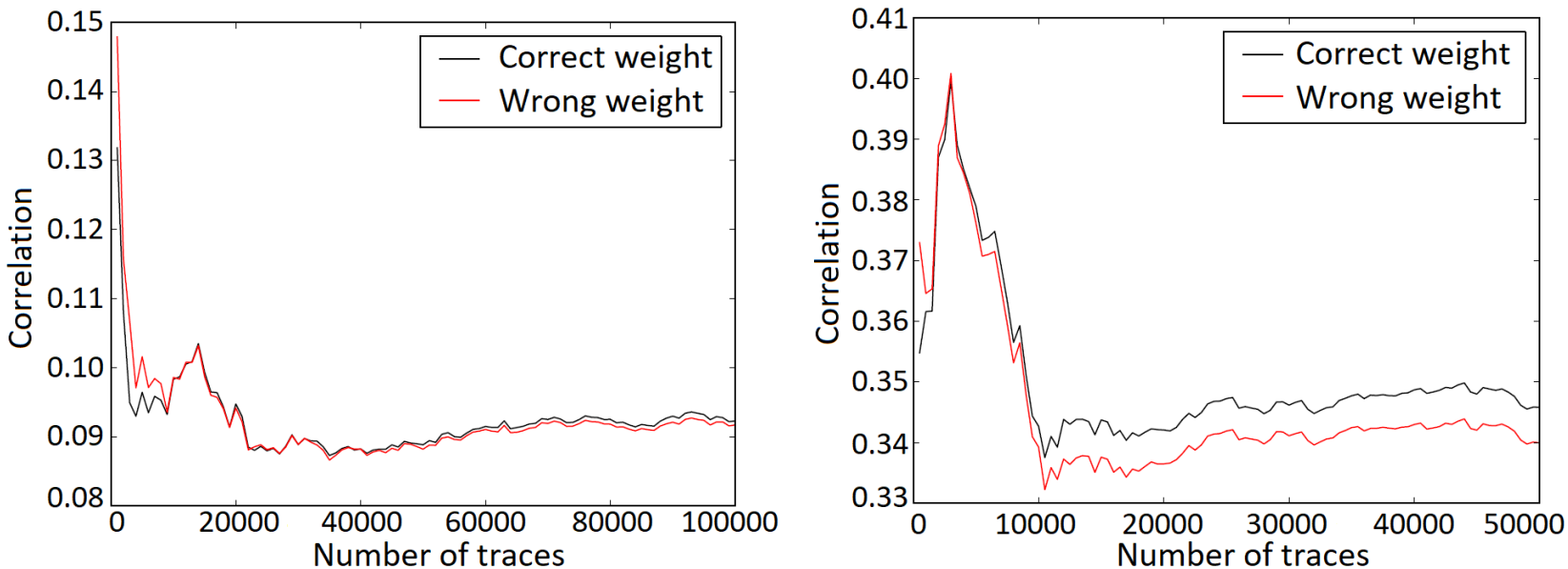
ARM Cortex M-3 and MLP

- Tests with MNIST and DPAv4 datasets.
- DPAv4: the original accuracy equals 60.9% and the accuracy of the reverse engineered network is 60.87%.
- MNIST: the accuracy of the original network is equal to 98.16% and the accuracy of the reverse engineered network equals 98.15%, with an average weight error converging to 0.0025.

ARM Cortex M-3 and CNN

- We target CIFAR-10 dataset.
- We choose as target the multiplication operation from the input with the weight, similar as in previous experiments.
- Differing from previous experiments, the operations on real values are here performed using fixed-point arithmetic.
- The numbers are stored using 8-bit data type – `int8` (`q7`).
- The resulting multiplication is stored in temporary `int` variable.
- The original accuracy of the CNN equals 78.47% and the accuracy of the recovered CNN is 78.11%.

ARM Cortex M-3 and CNN



(a) `int` scenario (b) `int8` scenario

Figure: The correlation of correct and wrong weight hypotheses on different number of traces targeting the result of multiplication operation stored as different variable type: (a) `int`, (b) `int8`

Threat Model

- The underlying neural network architecture of the used network is public and all the weights are known.
- Attacker is capable of measuring side-channel information leaked from the implementation of the targeted architecture.
- The crucial information for this work are the weights of the first layer.
- Indeed, when MLP reads the input, it propagates it to all the nodes, performing basic arithmetic operations.
- This arithmetic operation with different weights and common unknown input leads to input recovery attack via side-channel.

Experimental Setup

- A microcontroller loads sensitive data to/from a data bus to perform indicated instructions.
- This data bus is pre-charged to all '0's' before every instruction.
- Note that data bus being pre-charged is a natural behavior of microcontrollers and not a vulnerability introduced by the attacker.
- Thus, the power consumption (or EM radiation) assigned to the value of the data being loaded is modeled as the number of bits equal to '1'.

Experimental Setup

- In other words, the power consumption of loading data x is:

$$HW(x) = \sum_{i=1}^{n} x_i \ , \tag{11}$$

  where x_i represents the i^{th} bit of x.
- In our case, it is the product of secret input and known weight which is regularly stored to the memory after computation and results in the HW leakage.

Experimental Setup

- The training phase is conducted offline, and the trained network is then implemented in C language and compiled on the microcontroller.
- In our experiments, we consider MLP architectures consisting of a different number of layers and nodes in those layers.
- Note, we are only interested in the input layer where a higher number of neurons is beneficial for the attacker.

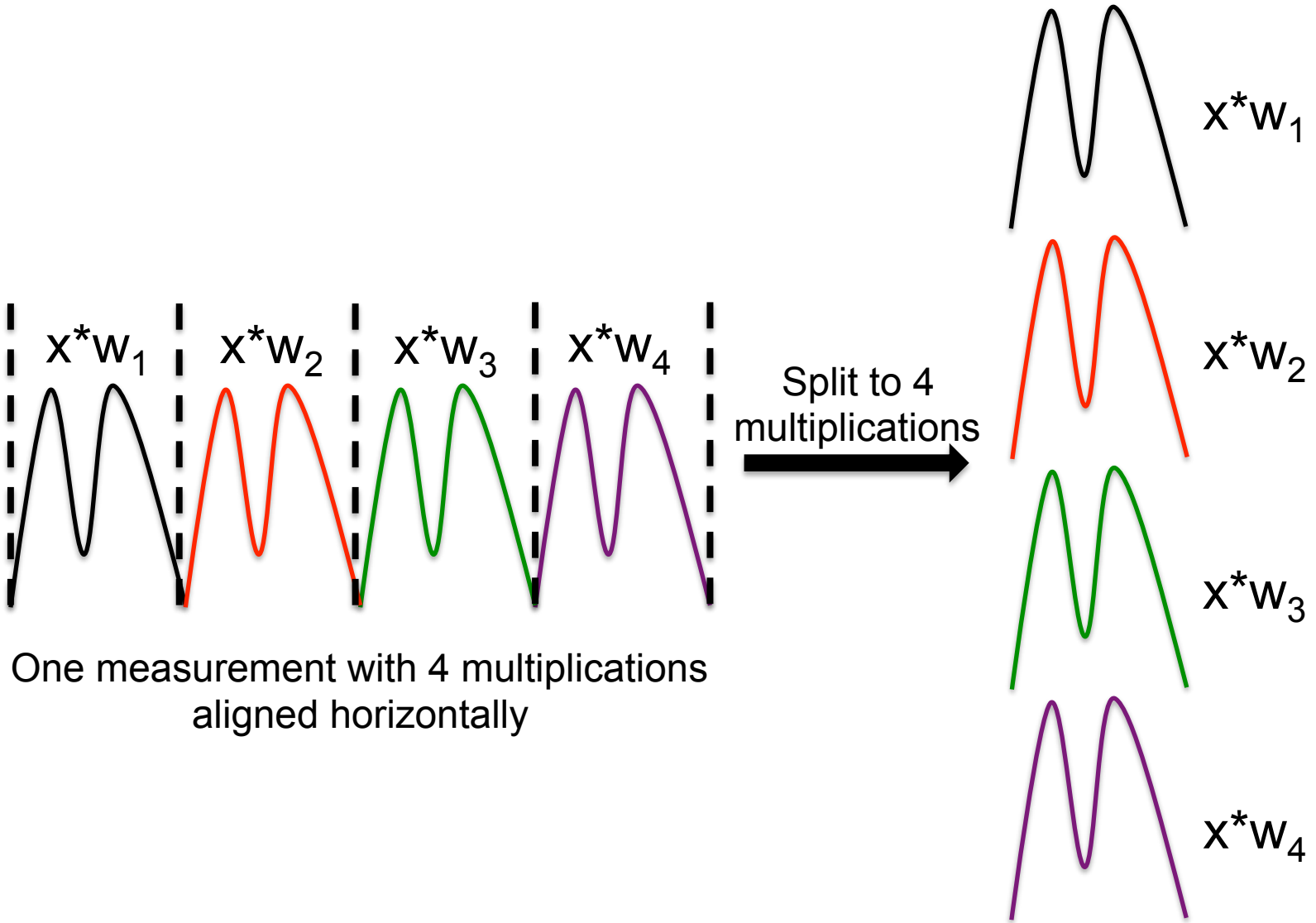
Results

- It can be extremely complex to recover the input by observing outputs from a known network.
- The proposed attack targets the multiplication operation in the first hidden layer.
- The main target for CPA is the multiplication $m = x \cdot w$ of a known weight w with a secret input x.
- As x changes from one measurement (input) to another, information learned from one measurement cannot be used with another measurement, preventing any statistical analysis over a set of different inputs.

Results

- To perform information exploitation over a single measurement, we perform a horizontal attack.
- The weights in the first hidden layer are all multiplied with the same input x, one after the other.
- M multiplications, corresponding to M different weights (or neurons) in the first hidden layer are isolated.
- A single trace is cut into M smaller traces, each one corresponding to one multiplication with a known associated weight.
- Next, the value of the input is statistically inferred by applying a standard CPA as explained before on the M smaller traces.

HPA


x*w1
x*w2
x*w3
x*w4
Split to 4 multiplications
x*w1
x*w2
x*w3
x*w4
One measurement with 4 multiplications aligned horizontally

Results on ATMega

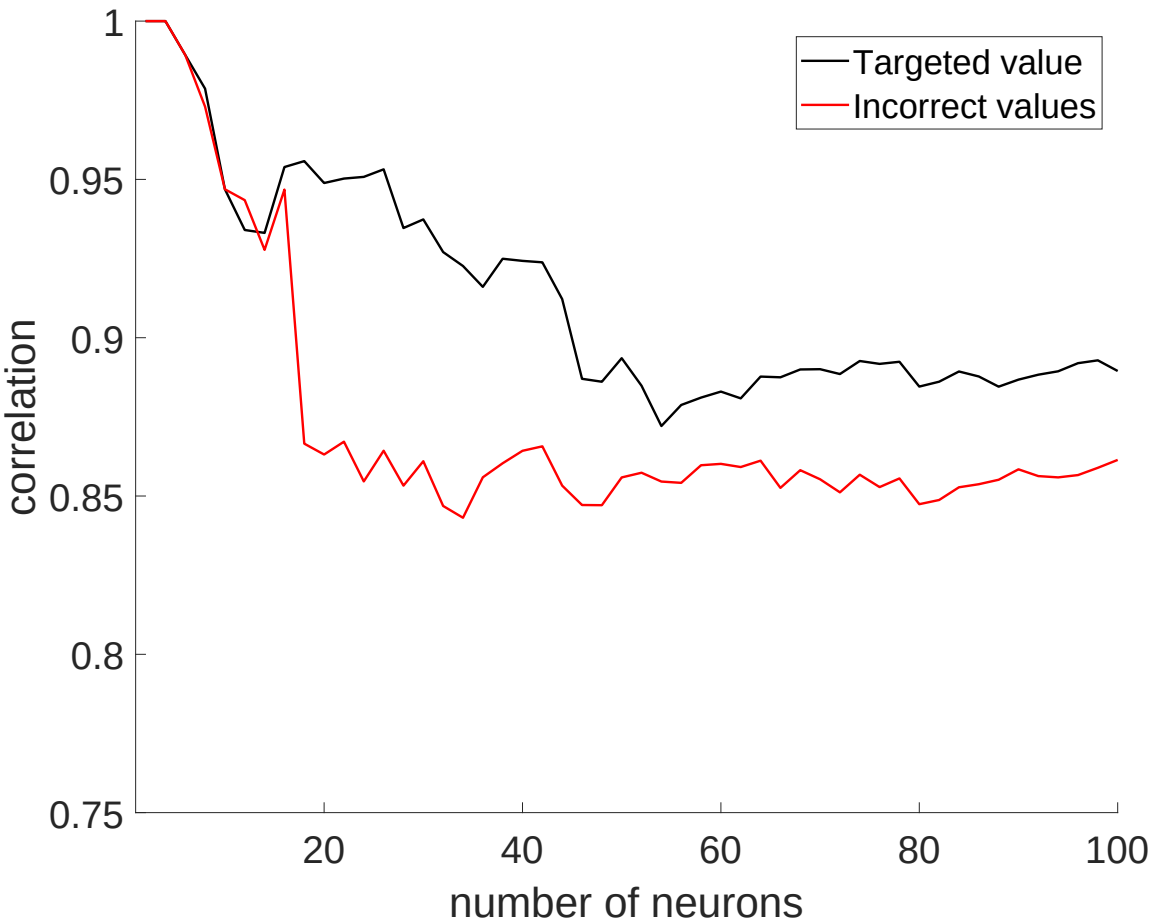


Figure: The first byte recovery (sign and 7-bit exponent).

Results

- The attack needs around 20 or more multiplications to reliably recover the input.
- In general, 70 multiplications are enough to recover all the bytes of the input, up to the desired precision of 2 decimal digits.
- This means that in the current setting, the proposed attack works very well on medium to large-sized networks, with at least 70 neurons in the first hidden layer, which is no issue in modern architectures used today.

Results on ARM Cortex M3

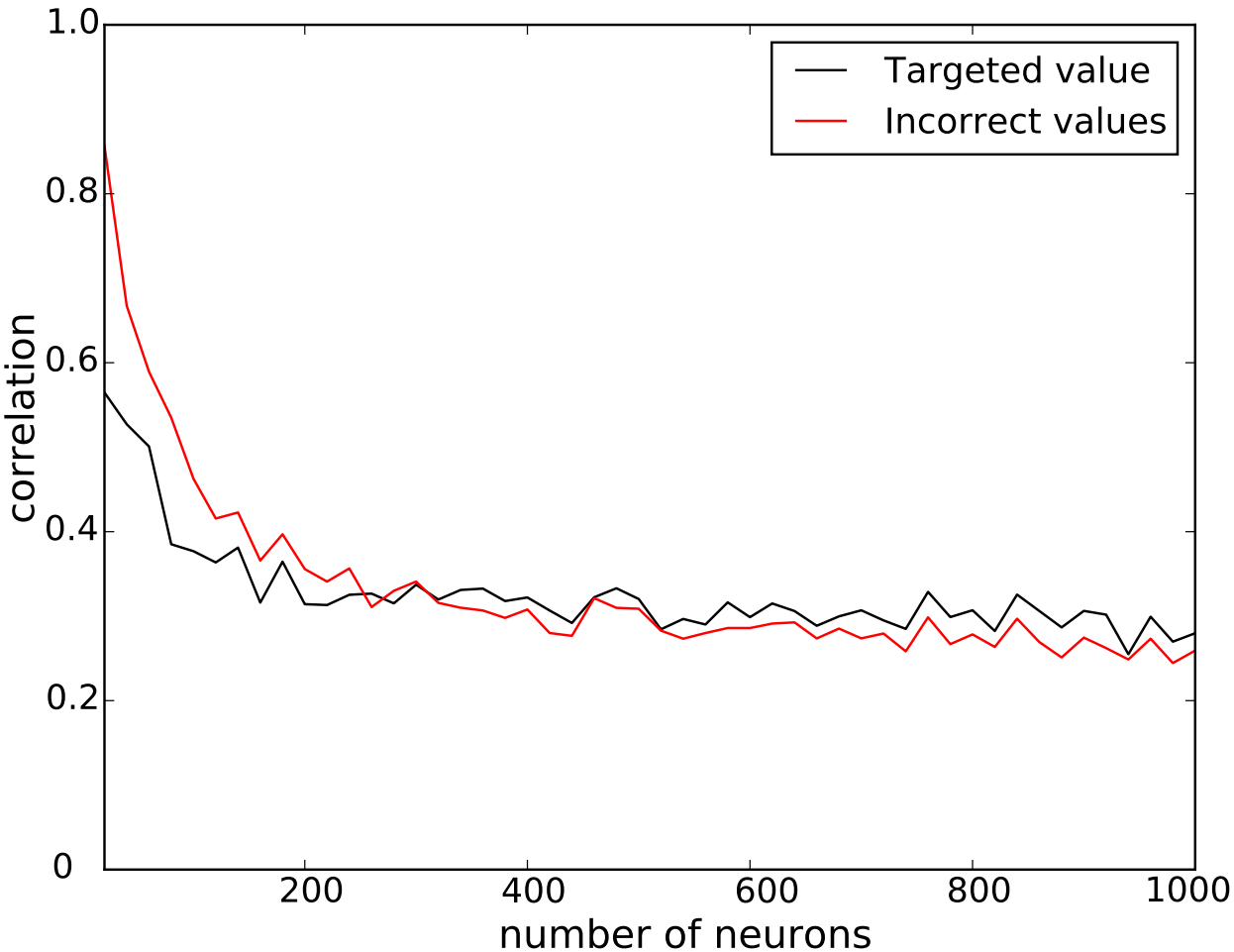


Figure: Correlation comparison between correct and incorrect inputs for target value 2.453.

Attack on MNIST Database

Figure: Original images (top) and recovered images with precision error (bottom).

Outline

Conclusions

- Machine learning (and even wider, artificial intelligence) play important role in cryptography.
- Currently, attacks perspective seem to be more developed.
- In implementation attacks, machine learning represents even the most powerful option.
- Still, our state-of-the-art techniques are usually much simpler than in other domains.
- There are some specific parts one does not encounter in other domains, but much of the knowledge is transferable.

Questions?

Thanks for your attention! Q?


Picek, S., Heuser, A., Guilley, S.:
Template attack versus Bayes classifier.
Journal of Cryptographic Engineering **7**(4) (Nov 2017) 343–351


Heuser, A., Picek, S., Guilley, S., Mentens, N.:
Side-channel analysis of lightweight ciphers: Does lightweight equal easy?
In: Radio Frequency Identification and IoT Security - 12th International Workshop, RFIDSec 2016, Hong Kong, China, November 30 - December 2, 2016, Revised Selected Papers. (2016) 91–104


Heuser, A., Picek, S., Guilley, S., Mentens, N.:
Lightweight ciphers and their side-channel resilience.
IEEE Transactions on Computers **PP**(99) (2017) 1–1


Picek, S., Heuser, A., Jovic, A., Legay, A.:
Climbing down the hierarchy: Hierarchical classification for machine learning side-channel attacks.
In Joye, M., Nitaj, A., eds.: Progress in Cryptology - AFRICACRYPT 2017: 9th International Conference on Cryptology in Africa, Dakar, Senegal, May 24-26, 2017, Proceedings, Cham, Springer International Publishing (2017) 61–78


Picek, S., Samiotis, I.P., Kim, J., Heuser, A., Bhasin, S., Legay, A.:
On the performance of convolutional neural networks for side-channel analysis.
In Chattopadhyay, A., Rebeiro, C., Yarom, Y., eds.: Security, Privacy, and Applied Cryptography Engineering, Cham, Springer International Publishing (2018) 157–176


Picek, S., Heuser, A., Alippi, C., Regazzoni, F.:
When theory meets practice: A framework for robust profiled side-channel analysis.
Cryptology ePrint Archive, Report 2018/1123 (2018) https://eprint.iacr.org/2018/1123.


Picek, S., Heuser, A., Jovic, A., Bhasin, S., Regazzoni, F.:
The curse of class imbalance and conflicting metrics with machine learning for side-channel evaluations.
IACR Trans. Cryptogr. Hardw. Embed. Syst. **2019**(1) (2019) 209–237

Maghrebi, H., Portigliatti, T., Prouff, E.:

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. (2016) 3–26

Lerman, L., Bontempi, G., Markowitch, O.:

Power analysis attack: An approach based on machine learning.
Int. J. Appl. Cryptol. **3**(2) (June 2014) 97–115

Lerman, L., Poussier, R., Bontempi, G., Markowitch, O., Standaert, F.:

Template Attacks vs. Machine Learning Revisited (and the Curse of Dimensionality in Side-Channel Analysis).
In: COSADE 2015, Berlin, Germany, 2015. Revised Selected Papers. (2015) 20–33

Lerman, L., Medeiros, S.F., Bontempi, G., Markowitch, O.:

A Machine Learning Approach Against a Masked AES.
In: CARDIS. Lecture Notes in Computer Science, Springer (November 2013) Berlin, Germany.

Lerman, L., Bontempi, G., Markowitch, O.:

A machine learning approach against a masked AES - Reaching the limit of side-channel attacks with a learning model.
J. Cryptographic Engineering **5**(2) (2015) 123–139

Lerman, L., Bontempi, G., Ben Taieb, S., Markowitch, O.:

A time series approach for profiling attack.
In Gierlichs, B., Guilley, S., Mukhopadhyay, D., eds.: Security, Privacy, and Applied Cryptography Engineering, Berlin, Heidelberg, Springer Berlin Heidelberg (2013) 75–94

Najm, Z., Jap, D., Jungk, B., Picek, S., Bhasin, S.:

On comparing side-channel properties of AES and chacha20 on microcontrollers.

In: 2018 IEEE Asia Pacific Conference on Circuits and Systems, APCCAS 2018, Chengdu, China, October 26-30, 2018, IEEE (2018) 552–555

Picek, S., Heuser, A., Jovic, A., Ludwig, S.A., Guilley, S., Jakobovic, D., Mentens, N.:
Side-channel analysis and machine learning: A practical perspective.
In: 2017 International Joint Conference on Neural Networks, IJCNN 2017, Anchorage, AK, USA, May 14-19, 2017. (2017) 4095–4102

Lerman, L., Medeiros, S.F., Veshchikov, N., Meuter, C., Bontempi, G., Markowitch, O.:
Semi-supervised template attack.
In Prouff, E., ed.: Constructive Side-Channel Analysis and Secure Design, Berlin, Heidelberg, Springer Berlin Heidelberg (2013) 184–199

Heuser, A., Zohner, M.:
Intelligent Machine Homicide - Breaking Cryptographic Devices Using Support Vector Machines.
In Schindler, W., Huss, S.A., eds.: COSADE. Volume 7275 of LNCS., Springer (2012) 249–264

Hospodar, G., Gierlichs, B., De Mulder, E., Verbauwhede, I., Vandewalle, J.:
Machine learning in side-channel analysis: a first study.
Journal of Cryptographic Engineering **1** (2011) 293–302 10.1007/s13389-011-0023-x.

Bartkewitz, T., Lemke-Rust, K.:
Efficient template attacks based on probabilistic multi-class support vector machines.
In Mangard, S., ed.: Smart Card Research and Advanced Applications, Berlin, Heidelberg, Springer Berlin Heidelberg (2013) 263–276

Hospodar, G., De Mulder, E., Gierlichs, B.:
Least squares support vector machines for side-channel analysis.
Center for Advanced Security Research Darmstadt (01 2011) 99–104

Sugawara, T., Homma, N., Aoki, T., Satoh, A.:

Profiling attack using multivariate regression analysis.
IEICE Electron. Express **7**(15) (2010) 1139–1144

Timon, B.:

Non-profiled deep learning-based side-channel attacks.
Cryptology ePrint Archive, Report 2018/196 (2018) https://eprint.iacr.org/2018/196.

Pfeifer, C., Haddad, P.:

Spread: a new layer for profiled deep-learning side-channel attacks.
Cryptology ePrint Archive, Report 2018/880 (2018) https://eprint.iacr.org/2018/880.

Gilmore, R., Hanley, N., 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). (May 2015) 106–111

Martinasek, Z., Hajny, J., Malina, L.:

Optimization of power analysis using neural network.
In Francillon, A., Rohatgi, P., eds.: Smart Card Research and Advanced Applications, Cham, Springer International Publishing (2014) 94–107

Yang, S., Zhou, Y., Liu, J., Chen, D.:

Back propagation neural network based leakage characterization for practical security analysis of cryptographic implementations.
In Kim, H., ed.: Information Security and Cryptology - ICISC 2011, Berlin, Heidelberg, Springer Berlin Heidelberg (2012) 169–185

Martinasek, Z., Zeman, V.:

Innovative method of the power analysis.
Radioengineering **22**(2) (2013)

Hettwer, B., Gehrer, S., Güneysu, T.:

Profiled power analysis attacks using convolutional neural networks with domain knowledge.
In Cid, C., Jr., M.J.J., eds.: 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., Springer (2018) 479–498

Kim, J., Picek, S., Heuser, A., Bhasin, S., Hanjalic, A.:

Make some noise: Unleashing the power of convolutional neural networks for profiled side-channel analysis.
Cryptology ePrint Archive, Report 2018/1023 (2018) https://eprint.iacr.org/2018/1023.

Cagli, E., Dumas, C., Prouff, E.:

Convolutional Neural Networks with Data Augmentation Against Jitter-Based Countermeasures - Profiling Attacks Without Pre-processing.
In: Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings. (2017) 45–68