Deep Learning-based Side-channel Analysis: How and Why

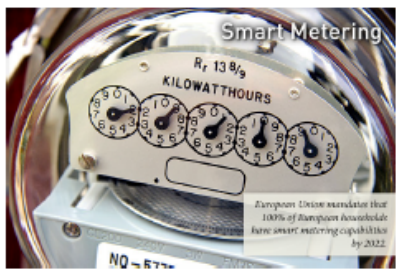
Stjepan Picek

Trieste, June 29, 2023

Outline

Outline

Challenges



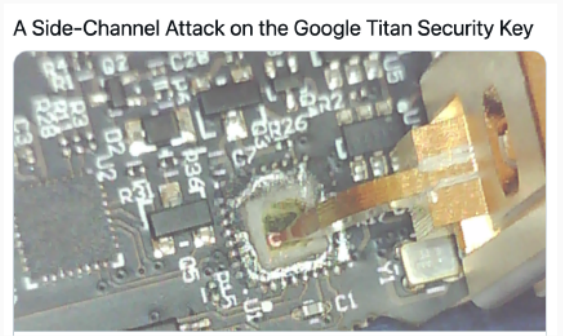
Relevance

November 13, 2019

October 3, 2019

May 28, 2020

January 7, 2021



Cryptographic Theory vs Physical Reality

- Cryptographic algorithms are (supposed to be) theoretically secure.
- Implementations leak in physical world.

Blackbox Scenario

- Cryptographic function is a black box, parameterized with key, that maps plaintext into ciphertext.
- Analyzing the security in the blackbox scenario relates to classical cryptanalysis.
- Adversary's goal: secret key or plaintext recovery by observing plaintext/ciphertext pairs.

Graybox Scenario

- Cryptographic algorithm is implemented on a real device such as a microcontroller, FPGA, ASIC etc.
- We can measure and process certain physical quantities in the device's vicinity.
- Adversary's goal: secret key or plaintext recovery by observing plaintext/ciphertext pairs and a side channel.
- Side-channel information relates to all sorts of leakage on the algorithm's internal computations.
- Examples: execution/reaction time, power consumption, electromagnetic emission, sound, temperature.
- Assuming limited access to the internal computations through this side channel window brings us to the graybox scenario.
- Security in the blackbox scenario does not imply security under the graybox scenario.

Implementation Attack Categories

- Side-channel attacks.
- Faults.
- Microprobing.

Implementation Attacks

Implementation attacks

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

- **Side-channel attacks** (SCAs) are passive, non-invasive attacks.
- SCAs represent one of the most powerful category of attacks on crypto devices.

Taxonomy of Implementation Attacks

- Active vs passive.
- Active:
  1. Active: the key is recovered by exploiting some abnormal behavior.
  2. Insertion of signals.
- Passive:
  1. The device operates within its specifications.
  2. Reading hidden signals.

The Goals of Attackers

- Secret data.
- Location.
- Reverse engineering.
- Theoretical cryptanalysis.
- ...

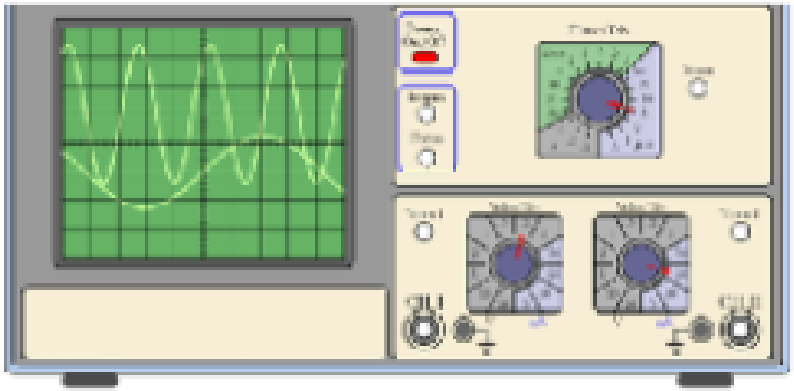
Physical Security in the Beginning

- Tempest – already known in 1960s that computers generate EM radiation that leaks information about the processed data.
- 1965: MI5 used a microphone positioned near the rotor machine used by Egyptian embassy to deduce the positions of rotors.
- 1996: first academic publication on SCA – timing.
- 1997: Bellcore attack.
- 1999: first publication of SCA – power.

Outline

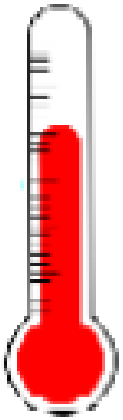
Side Channels

Power

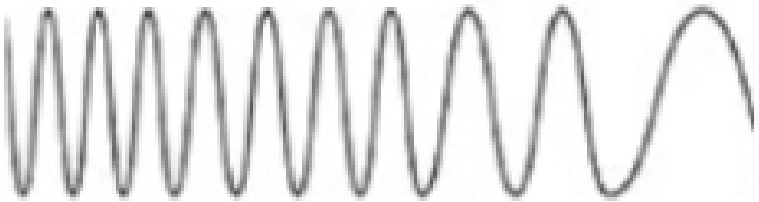


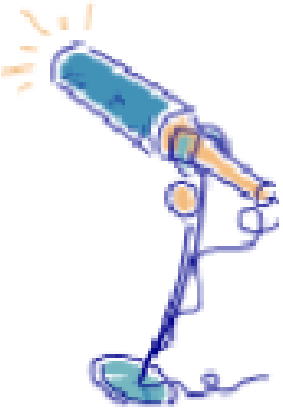
Timing

Temperature



Side channels





Sound

Timing

- One of the earliest side-channel attacks due to easy measurements collection.
- Can also be exploited remotely.
- Exploit some not foreseen effects of caches to crypto implementations.
- Applied to symmetric and asymmetric cryptography.

Timing

- Software for PIN code verification

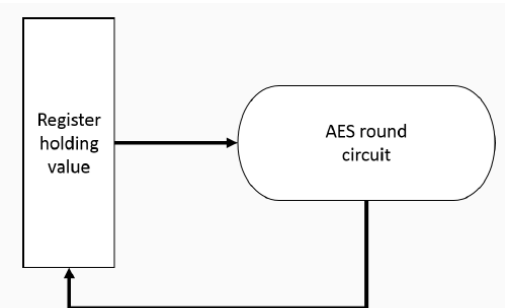
```
Input: 4-digit PIN code
Output: PIN verified or rejected
Process CheckPIN (pin[4])
int pin_ok=0;
if (pin[0]==5)
   if (pin[1]==9)
      if (pin[2]==0)
         if (pin[3]==2)
            pin_ok=1;
         end
      end
   end
end
return pin_ok;
EndProcess
```

Power Consumption

- CMOS is one of the most popular technologies for chip design.
- CMOS circuits exhibit several types of leakage.
- Charge and discharge of the CMOS load capacitance leads to side-channel leakage (dynamic power consumption).
- Power analysis attack exploits the fact that the dynamic power consumption depends on the data and instructions being processed.
- Dynamic power consumption is produced by CMOS transitions from state 0 to 1 and from state 1 to 0.

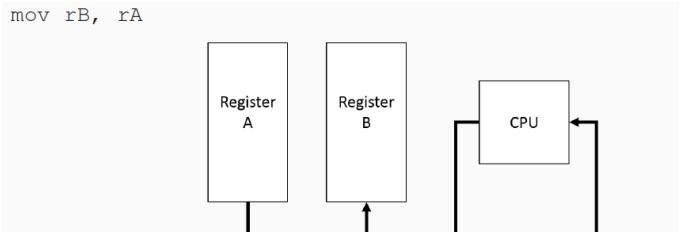
How to Model the Leakage

- We use the number of transitions to model the leakage.
- The Hamming distance model counts the number of $0 \rightarrow 1$ and $1 \rightarrow 0$ transitions.
- Example 1: A register R is storing the result of an AES round and initial value v_0 gets overwritten with v_1.
- The power consumption because of the register transition $v_0 \rightarrow v_1$ is related to the number of bit flips that occurred.
- Modeled as HammingDistance(v0, v1) = $HammingWeight(v_0 \oplus v_1)$.
- Common leakage model for hardware implementations (FPGA, ASIC).

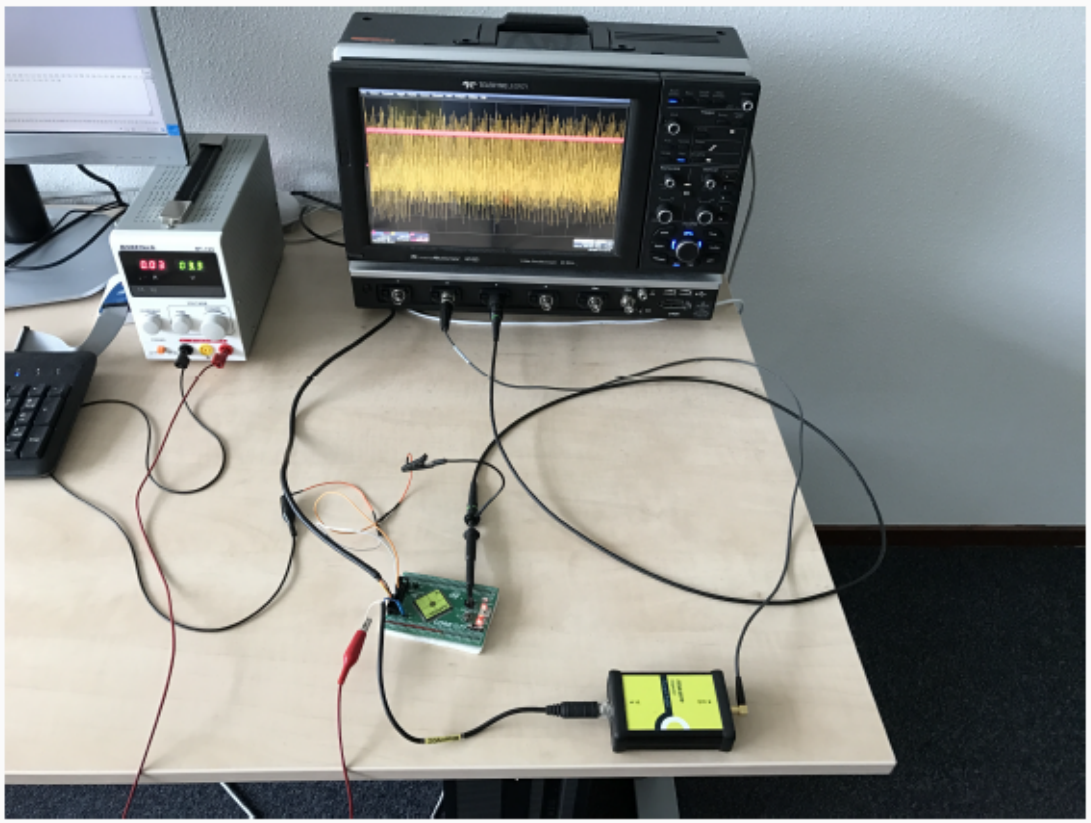


How to Model the Leakage

- Example 2: In a microcontroller, a register A contains value v_0 and an assembly instruction moves the content of register A to B.
- This instruction transfers v_0 from A to B via the CPU, using the bus.
- Typically the bus is precharged at all bits being zeros or one (busInitialValue).
- The power consumption of the instruction can be modeled as $\mathit{HammingDistance}(\mathit{busInitialValue}, v_0) = \mathit{HammingWeight}(v_0 \oplus 0) = \mathit{HW}(v_0)$.
- Common leakage model for software implementations (AVR/ARM).



Measurement Setup



Outline

Analysis Capabilities

- Direct attacks:
  1. Simple side-channel analysis.
  2. Differential side-channel analysis.
  3. Higher order attacks.
  4. ...
- Two-stage (profiling) attacks:
  1. Template attack.
  2. Stochastic models.
  3. Machine learning-based attacks.
  4. ...

Scientific Metrics

- The most common evaluation metrics in the side-channel analysis are success rate (SR) and guessing entropy (GE).
- Success rate defines the estimated averaged probability of success.
- The average key rank is given by the guessing entropy.
- More precisely, GE states the average number of key candidates an adversary needs to test in order to reveal the secret key after conducting a side-channel analysis.

Scientific Metrics

- In particular, given Q samples in the attacking phase, an attack outputs a key guessing vector $g = [g_1, g_2, \ldots, g_{|\mathcal{K}|}]$ in decreasing order of probability with $|\mathcal{K}|$ being the size of the keyspace.
- So, g_1 is the most likely and $g_{|\mathcal{K}|}$ the least likely key candidate.
- The guessing entropy is the average position of k_a^* in g over multiple experiments.
- The success rate is defined as the average empirical probability that g_1 is equal to the secret key k_a^*.

Scientific Metrics

- In practice one may consider leakage models $Y(\cdot)$ that are bijective functions, thus each output probability calculated from the classifiers for $Y(k)$ directly relates to one single key candidate k.
- In case $Y(\cdot)$ is not bijective, several key candidates k may get assigned with the same output probabilities, which is why on average a single trace attack ($Q = 1$) may not be possible in case of non-bijective leakage models.
- Further, to calculate the key guessing vector g over Q amount of samples, the (log-)likelihood principle is used.

Practical Evaluation Testing

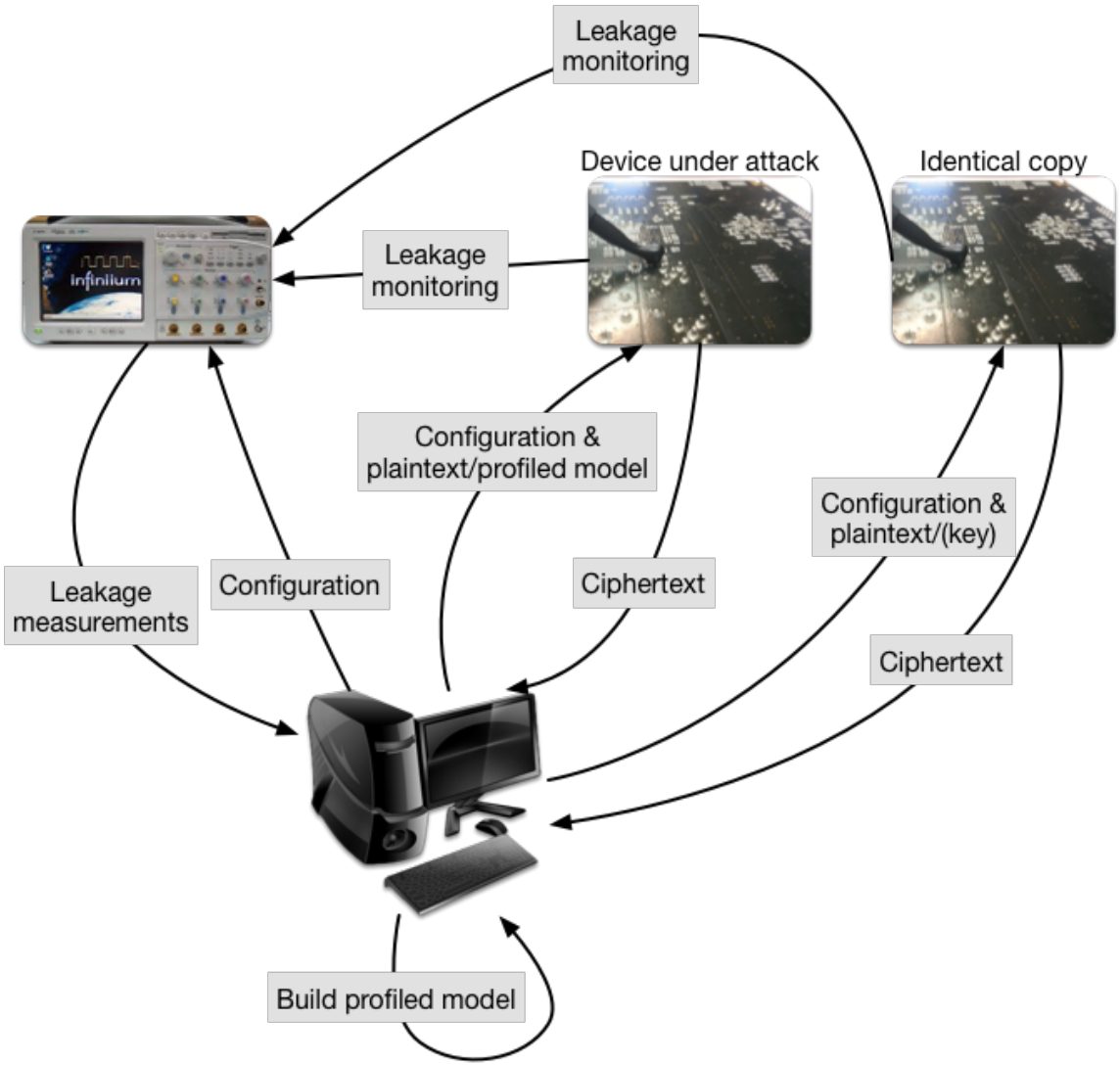
- In practice, there are two main practical schemes:
  1. Test-based schemes, such as NIST FIPS 140 and ISO/IEC 17825.
  2. Evaluation-based schemes, such as Common Criteria (CC, ISO/IEC 15408).

Outline

Profiling Attacks

- Profiling attacks have a prominent place as the most powerful among side channel attacks.
- Within profiling phase the adversary estimates leakage models for targeted intermediate computations, which are then exploited to extract secret information in the actual attack phase.
- Template Attack (TA) is the most powerful attack from the information theoretic point of view.
- Some machine learning (ML) techniques also belong to the profiling attacks.

Profiling Attacks



Profiling Attacks

- Two stage (profiling) attacks are more complicated than the direct attacks.
- The attacker must have access to a copy of the device to be attacked.

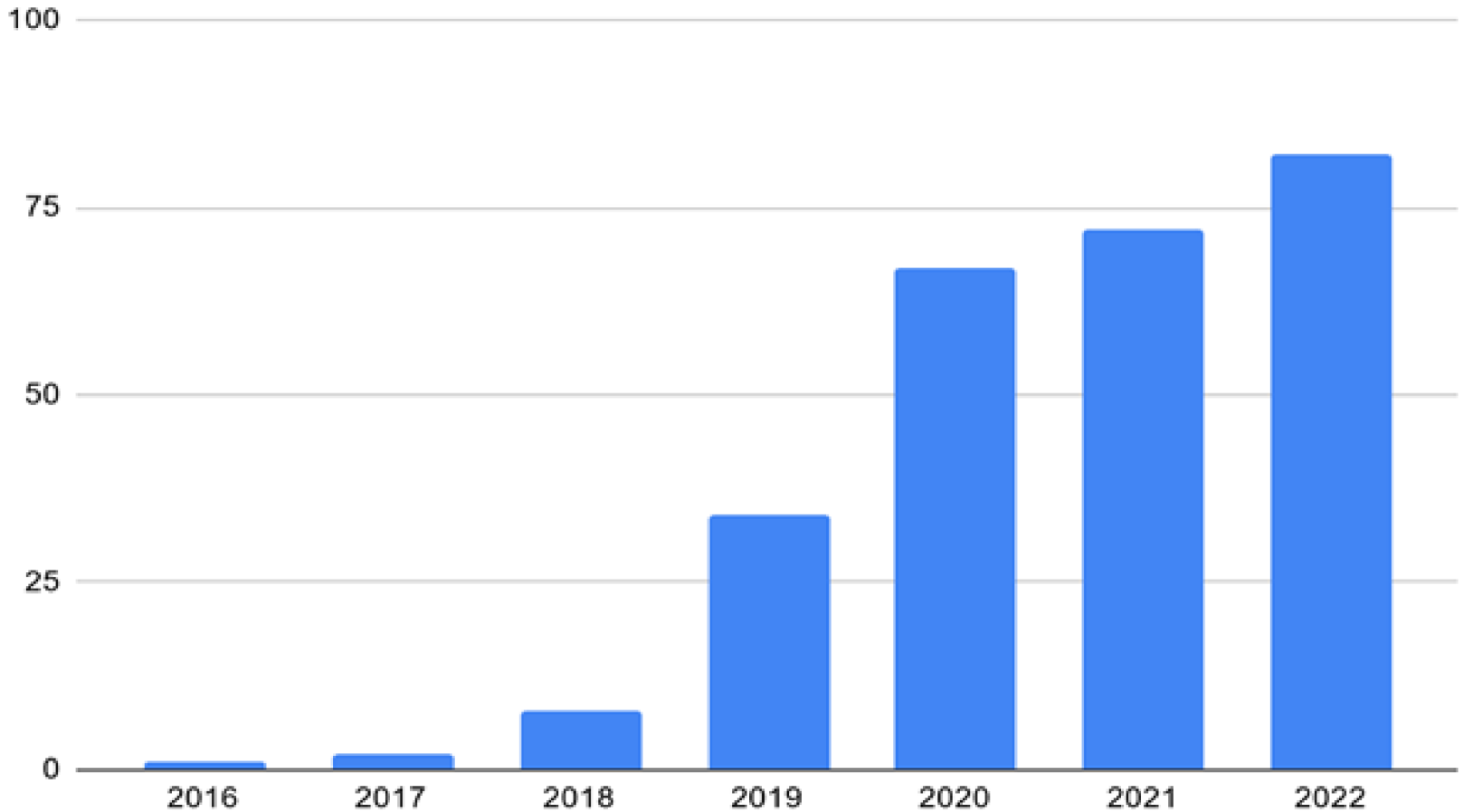
Template Attack

- Using the copy of device, record a large number of measurements using different plaintexts and keys. We require information about every possible subkey value.
- Create a template of device's operation. A template is a set of probability distributions that describe what the power traces look like for many different keys.
- On device that is to be attacked, record a (small) number of measurements (called attack traces) using different plaintexts.
- Apply the template to the attack traces. For each subkey, record what value is the most likely to be the correct subkey.
- When using high-quality templates made from many traces, it is possible to attack a system with a single trace.
- Template attack can become unstable if there are more points of interest than measurements per value.

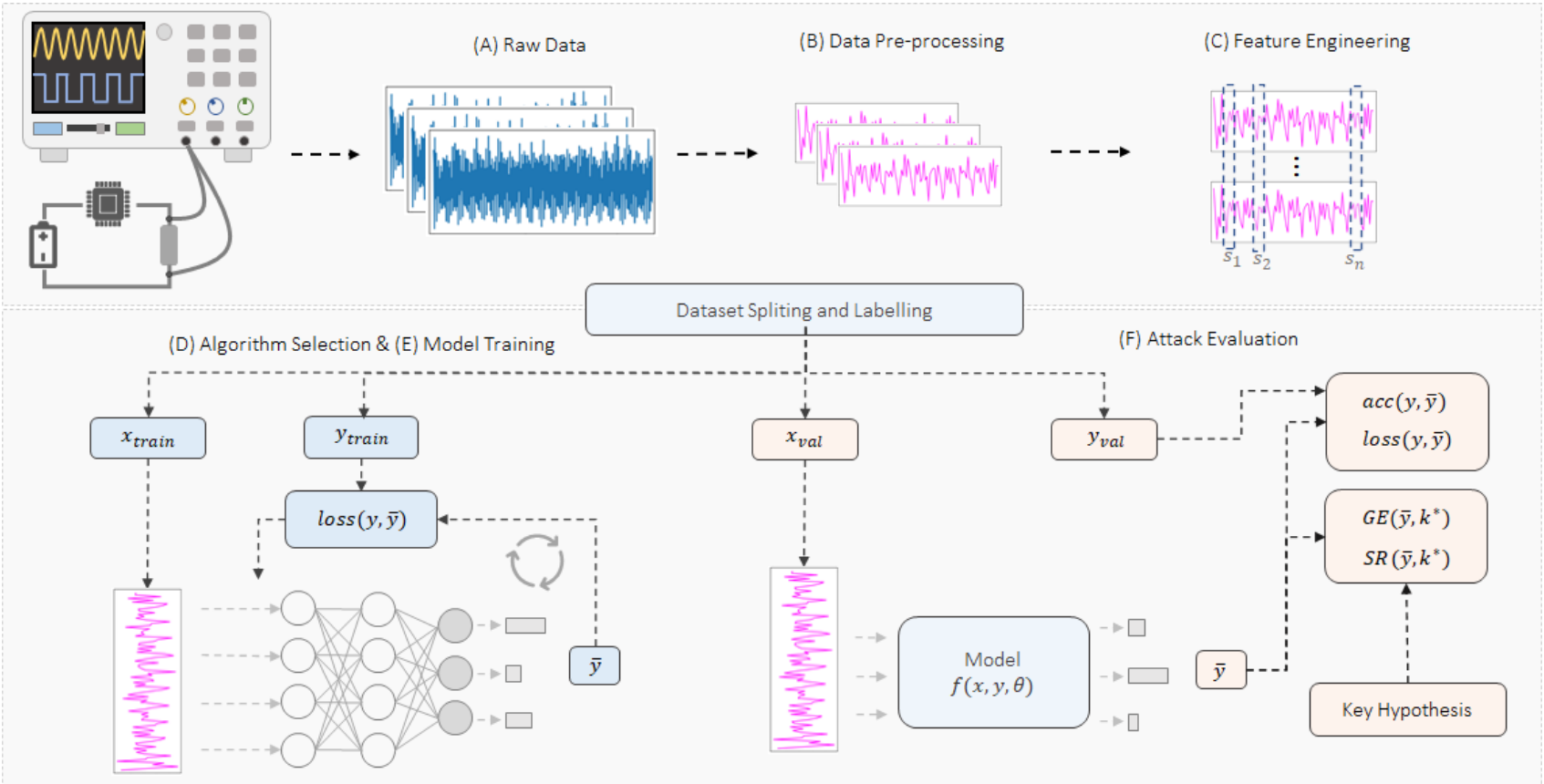
Side-channel Analysis and Machine Learning

- Machine learning techniques also represent an extremely powerful paradigm in side-channel analysis.
- We can observe how profiling scenario in SCA has clear connections with supervised machine learning.
- We can use machine learning in SCA for classification, clustering, feature engineering, preprocessing.
- More recently, deep learning positioned itself as the most powerful choice for profiling SCA.

Deep Learning SCA Publications



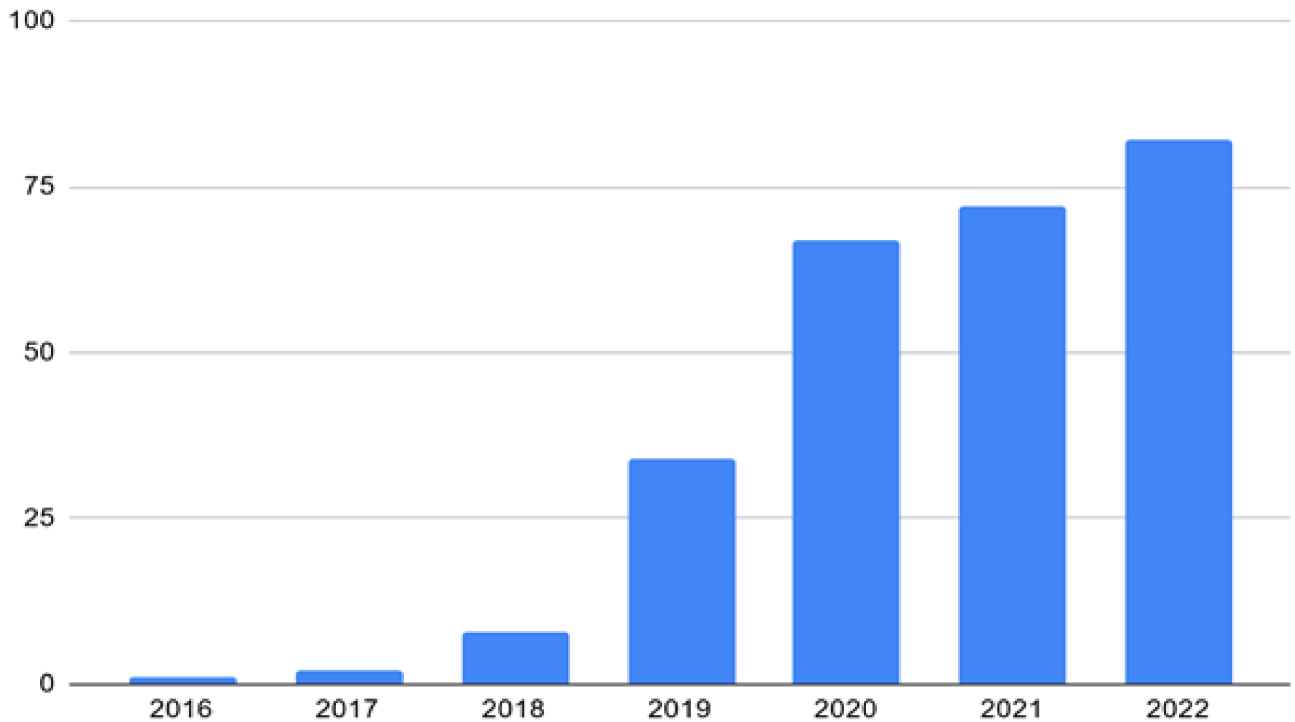
Machine Learning Process Flow



Neural Network Types in SCA

- Multilayer perceptron.
- Convolutional neural network.
- Autoencoder.
- Recurrent neural network.
- Residual neural network.
- Generative Adversarial Network.

Deep Learning Publications



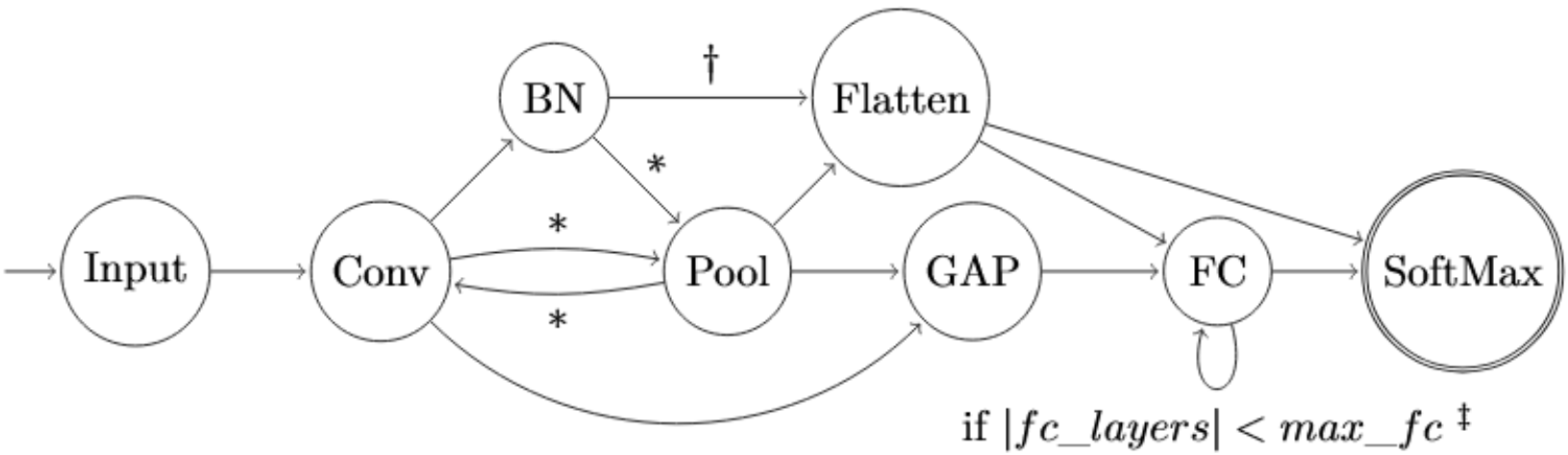
Outline

Methodologies

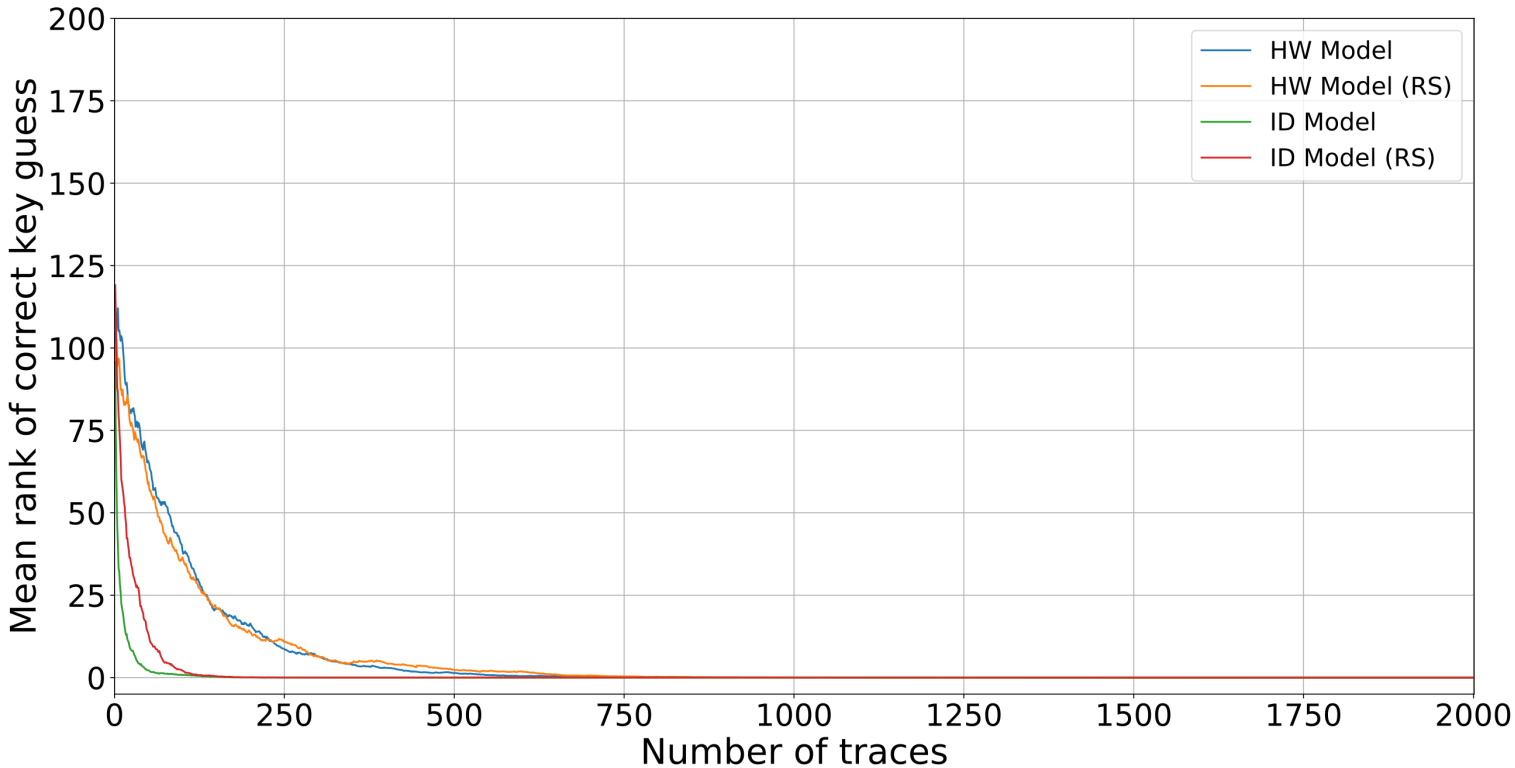
- Zaid, G., Bossuet, L., Habrard, A., & Venelli, A. (2019). Methodology for Efficient CNN Architectures in Profiling Attacks. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2020(1), 1-36. https://doi.org/10.13154/tches.v2020.i1.1-36.
- Wouters, L., Arribas, V., Gierlichs, B., & Preneel, B. (2020). Revisiting a Methodology for Efficient CNN Architectures in Profiling Attacks. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2020(3), 147-168. https://doi.org/10.13154/tches.v2020.i3.147-168.

Reinforcement Learning

- Reinforcement learning attempts to teach an agent how to perform a task by letting the agent experiment and experience the environment, maximizing some reward signal.
- `https://github.com/AISyLab/Reinforcement-Learning-for-SCA`



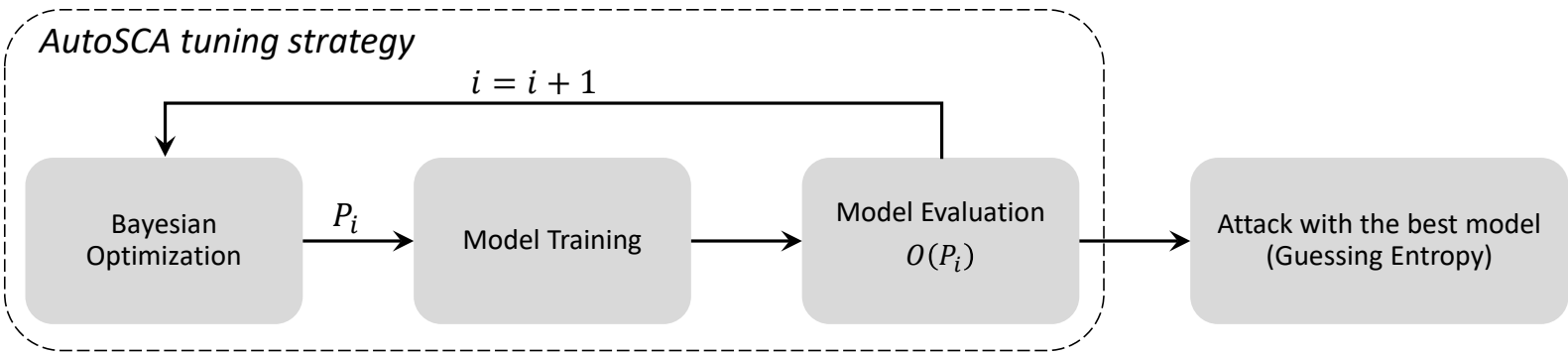
Reinforcement Learning



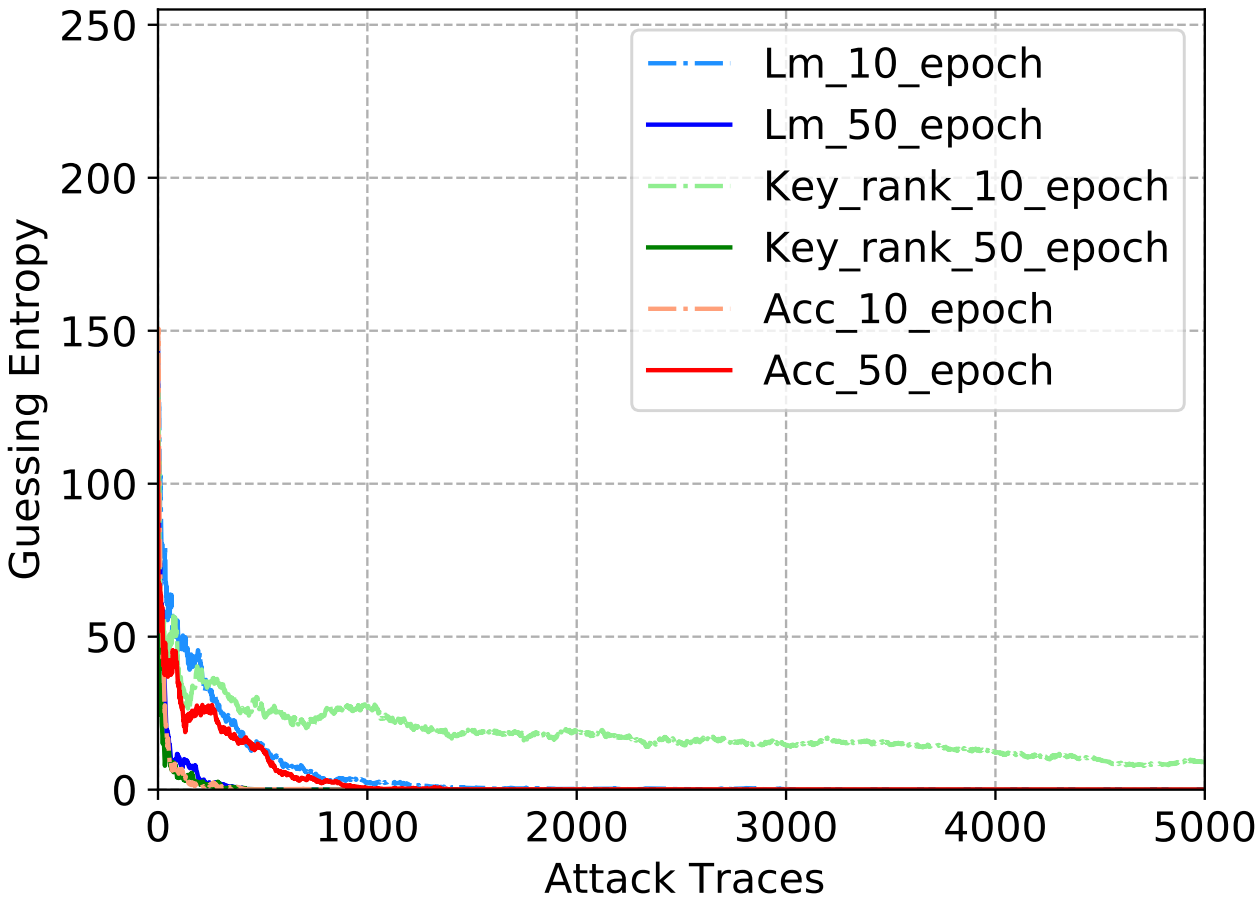
Bayesian Optimization

- In Bayesian optimization, the aim is to build a probabilistic model of the underlying function.
- We first require a probabilistic model of a function (often referred to as the surrogate model), where there are several ways to model it.
- Second, we require an acquisition function for Bayesian optimization to generate the next neural network architecture to observe, i.e., to select what point to sample next.

Bayesian Optimization



Bayesian Optimization



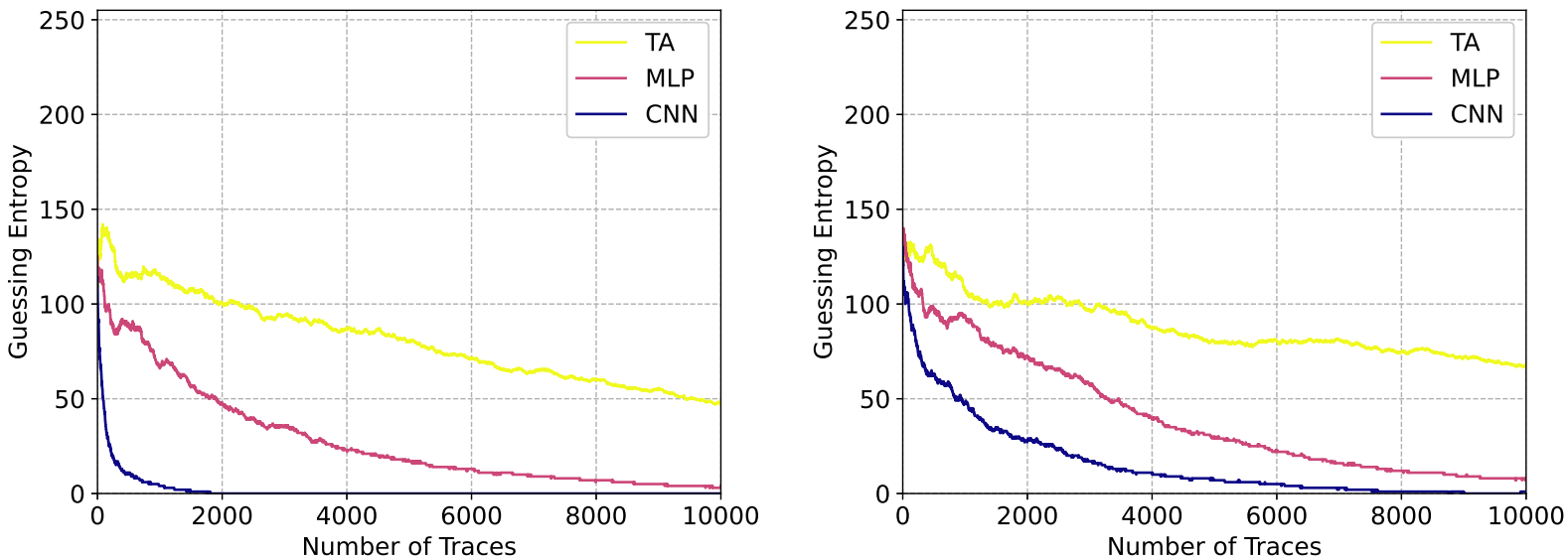
Countermeasures as Noise

- Various countermeasures make SCAs significantly more complex, and such countermeasures can be further combined to make the attacks even more challenging.
- Could we consider those as noise and use some techniques to remove the noise?
- We propose a new approach to remove several common hiding countermeasures with a denoising autoencoder.
- Gaussian noise, random delay interrupts, desynchronization, jitter, shuffling.

Denoising Autoencoder

- An autoencoder consists of two parts: encoder (ϕ) and decoder (ψ).
- The goal of the encoder is to transfer the input to its latent space $\mathcal{F}$, i.e., $\phi : \mathcal{X} \rightarrow \mathcal{F}$.
- The decoder, on the other hand, reconstructs the input from the latent space, which is equivalent to $\psi : \mathcal{F} \rightarrow \mathcal{X}$.
- When applying the autoencoder for the denoising purpose, the input and output are not identical but represented by noisy-clean data pairs.

Denoising Autoencoder

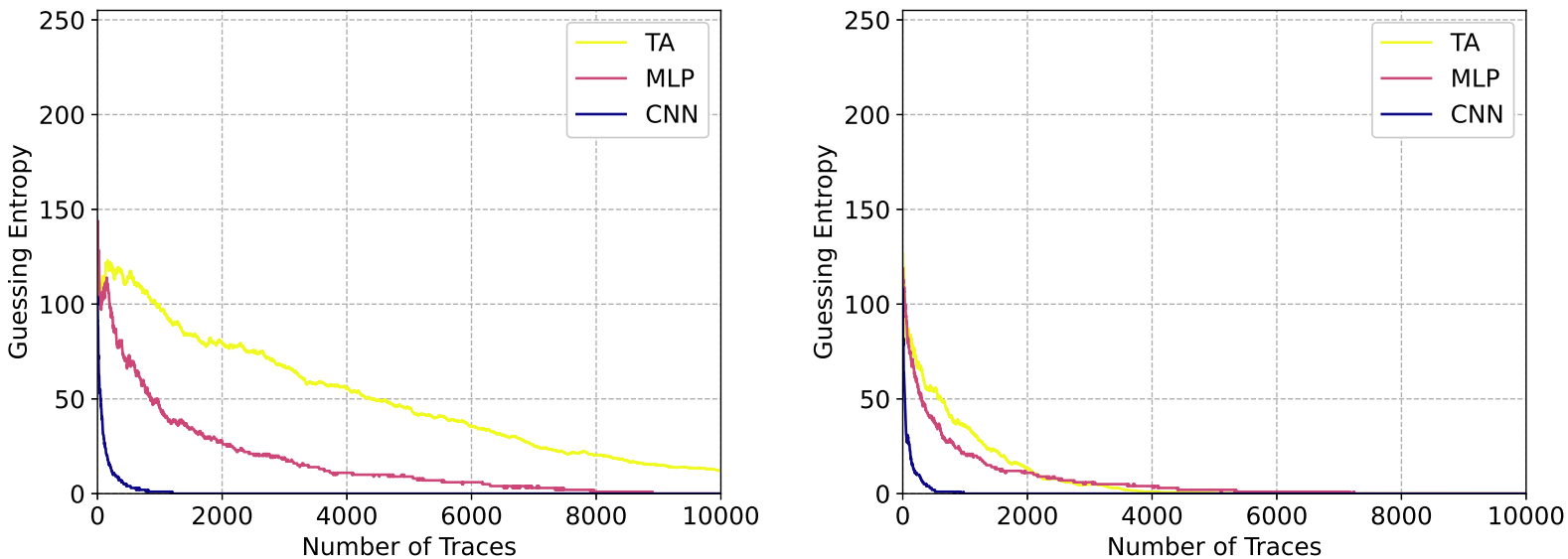


(a) GE: denoise with averaging.

(b) GE: denoise with CAE.

Figure: Guessing entropy: denoising Gaussian noise with averaging (a) and CAE (b).

Denoising Autoencoder



(a) GE: denoise with static alignment.

(b) GE: denoise with CAE.

Figure: Guessing entropy: denoising desynchronization with static alignment (a) and CAE (b).

Generalization of Function Approximation

- While we use machine learning metrics to drive the training, we are interested in results as observed through SCA metrics.
- Ideally, we should always train a neural network until it achieves the maximum quality in generalization to the validation set.
- Underfitting, generalization, and overfitting phases.

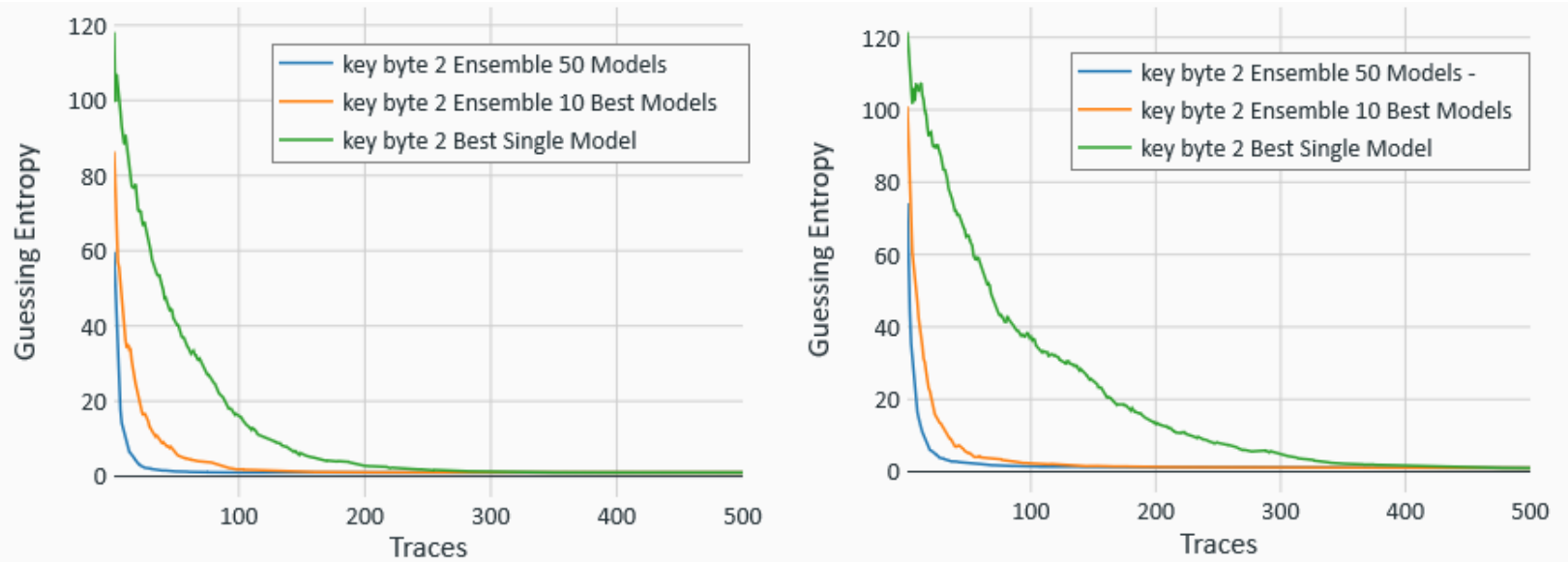
Generalization of Function Approximation

- In SCA, the generalization phase is directly related to the key recovery, and it may start very soon after the training starts because a low accuracy can already represent the turning point from underfitting to generalization.
- Can a low accuracy (sometimes close to random guessing) still be associated with this *good enough* generalization phase?

How to Improve Generalization?

- There are many ways to improve generalization (more powerful classification methods, better hyperparameter tuning, regularization, etc.).
- We can also do something simpler!
- Commonly, in the experimental phase, one runs a number of evaluations to find the best hyperparameters.
- Can we somehow use multiple results?
- It sounds reasonable to take the most out of the hyperparameter tuning phase and explore whether one can use more than a single machine learning model obtained during the tuning phase.

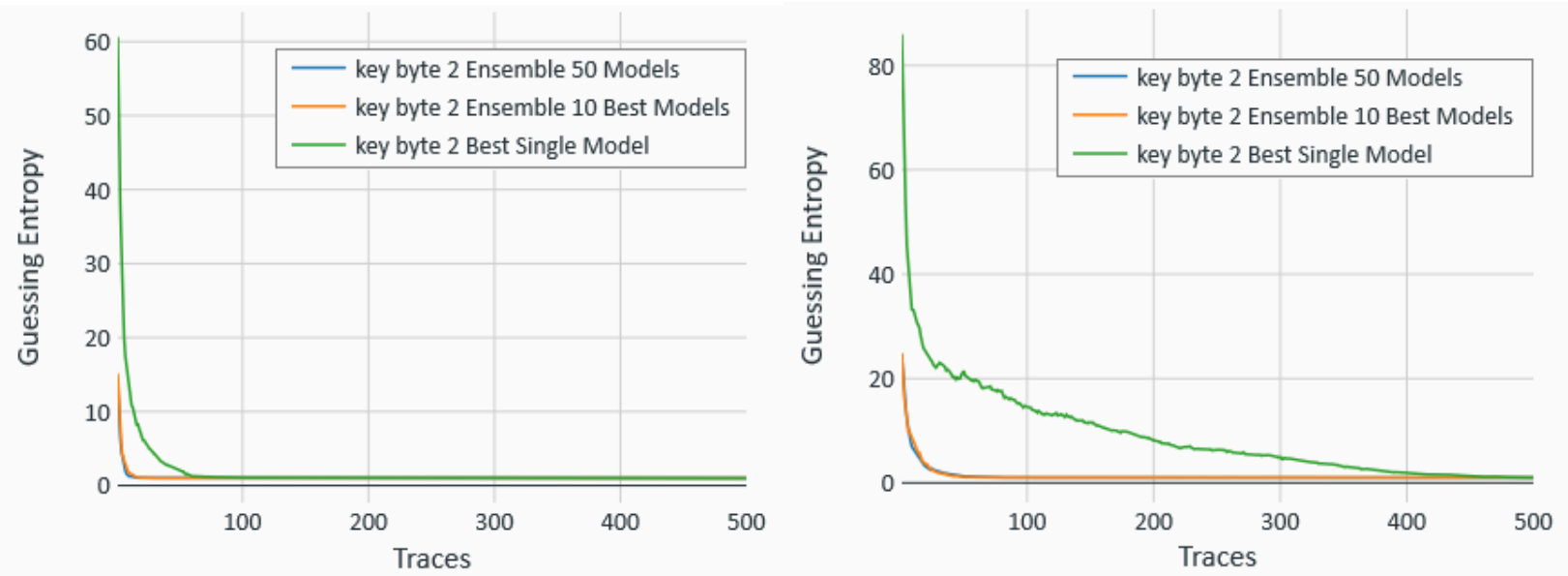
Deep Learning Ensembles



(a) MLP results. (b) CNN results.

Figure: Guessing entropy for ASCAD for the Hamming weight leakage model.

Deep Learning Ensembles



(a) MLP results. (b) CNN results.

Figure: Guessing entropy for ASCAD for the identity leakage model.

Outline

Important Hyperparameters

- MLP: activation functions, number of dense layers, number of neurons in dense layers, regularization, learning rate, optimizer, loss function, number of epochs, batch size, initialization.
- CNN: Number of convolutional layers, number and size of kernels, activation functions, pooling type and size, stride, number of dense layers, number of neurons in dense layers, regularization, learning rate, optimizer, loss function, number of epochs, batch size, initialization.

Many Things Actually Work

- Simple hyperparameter search.
- Data Augmentation.
- Various types of architectures.
- Small architectures.
- Custom metrics and neural network elements.
- ...

Feature Selection for Deep Learning SCA

Scenario	**Knowledge of r mask share**	**POI selection and pre-processing**	**Noisy/non-leaking samples**
RPOI	Yes	Main SNR peaks of r and s_r. No pre-processing required.	No
OPOI	Yes	Minimum trace interval including SNR peaks of r and s_r. No pre-processing required.	Reduced
NOPOI	No	No POI selection and pre-processing is required.	All available

Table: Possible feature selection scenarios for deep learning-based SCA with the synchronized measurements.

Feature Selection for Deep Learning SCA

Dataset	RPOI	OPOI	NOPOI	Total
ASCADf	up to 1 000 SNR peaks from NOPOI interval	[45 400, 46 100]	[0, 100 000]	100 000
ASCADr	up to 1 000 SNR peaks from NOPOI interval	[80 945, 82 345]	[0, 250 000]	250 000
DPAv4.2	up to 1 000 SNR peaks from NOPOI interval	[170 000, 174 000] + [206 000, 210 000]	[250 000, 400 000]	1 700 000
CHES CTF	-	[0, 10 000] + [120 000, 150 000]	[0, 150 000]	650 000

Table: Selected intervals for each feature selection scenario. '-' denotes that we did not explore that specific setting.

Feature Selection for Deep Learning SCA

Table: Points of interest, minimum number of attack traces to get guessing entropy equal to 1, model search success (when GE=1), and number of trainable parameters for all datasets and feature selection scenarios.

Dataset	Neural Network Model	Feature Selection Scenario	Amount of POIs (HW/ID)	Attack Traces (HW/ID)	Search Success (%) (HW/ID)	Trainable Parameters (HW/ID)
ASCADf	MLP	RPOI	200/100	5/**1**	99.22%/96.86%	82 209/429 256
ASCADf	CNN	RPOI	400/200	5/**1**	99.23%/99.08%	499 533/158 108
ASCADf	MLP	OPOI	700/700	480/104	82.80%/68.80%	16 309/10 266
ASCADf	CNN	OPOI	700/700	744/87	55.53%/35.33%	594 305/62 396
ASCADf	MLP	NOPOI	2 500/2 500	7/**1**	74.50%/39.00%	2 203 009/5 379 256
ASCADf	CNN	NOPOI	10 000/10 000	7/ **1**	15.40%/2.45%	545 693/439 348
ASCADf	CNN	NOPOI desync	10 000/10 000	532/36	2.44%/2.64%	268 433/64 002
ASCADr	MLP	RPOI	200/20	3/**1**	99.23%/100%	565 209/639 756
ASCADr	CNN	RPOI	400/30	5/**1**	100%/100%	575 369/636 224
ASCADr	MLP	OPOI	1 400/1 400	328/129	71.40%/37.25%	31 149/34 236
ASCADr	CNN	OPOI	1 400/1 400	538/78	47.92%/23.95%	270 953/87 632
ASCADr	MLP	NOPOI	25 000/25 000	6/ **1**	44.39%/7.02%	5 243 209/12 628 756
ASCADr	CNN	NOPOI	25 000/25 000	7/ **1**	19.17%/4.35%	369 109/721 012
ASCADr	CNN	NOPOI desync	25 000/25 000	305/73	0.71%/1.04%	22 889/90 368

Introduction

- There are already few settings that consider unsupervised deep learning-based SCA.
- First approach (DDLA) is by B. Timon.
- While it works, the attack performance is not great and the computational complexity is high.

Plaintext/Ciphertext-based Non-profiling SCA

- Supervised deep learning-based SCA learns a mapping based on known plaintexts and keys.
- Then, the adversary estimates the conditional probability given a leakage trace with the unknown key.
- In unsupervised setting, we do not know the key.
- But, the key is commonly fixed for all traces.

Plaintext/Ciphertext-based Non-profiling SCA

- Supervised deep learning-based SCA learns a mapping based on known plaintexts and keys.
- Then, the adversary estimates the conditional probability given a leakage trace with the unknown key.
- In unsupervised setting, we do not know the key.
- But, the key is commonly fixed for all traces.
- The label $\mathsf{I}(k, d_i)$ and d_i would satisfy:

$$d_i \longmapsto \mathsf{I}(k, d_i). \tag{1}$$

Plaintext/Ciphertext-based Non-profiling SCA

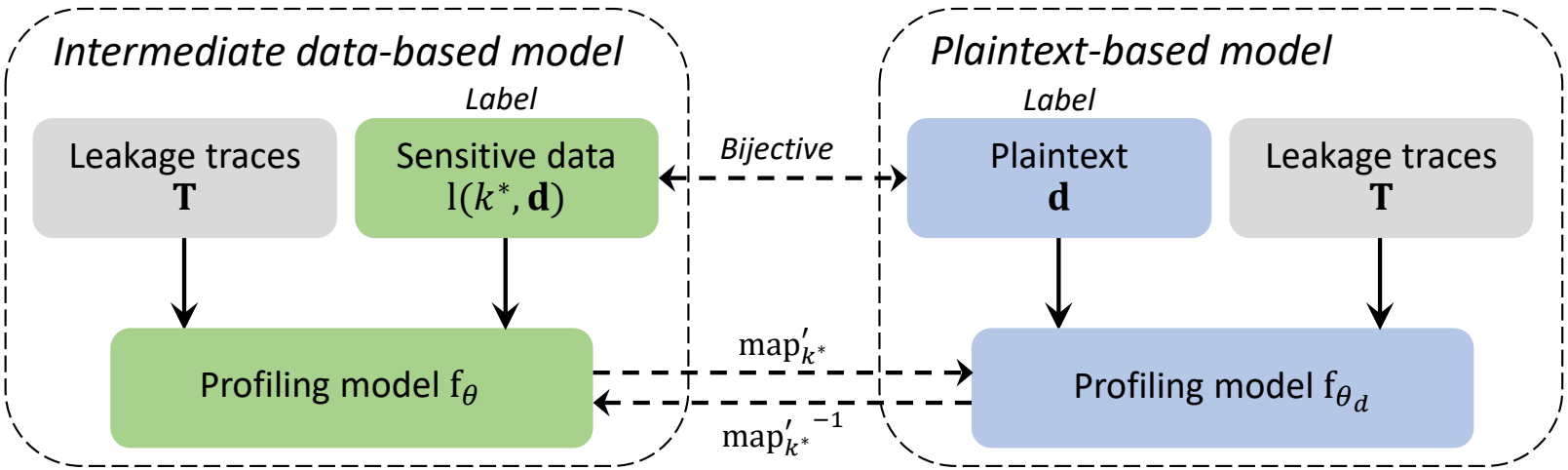


Figure: The relationship between intermediate data-based model and plaintext-based model.

Plaintext/Ciphertext-based Non-profiling SCA

- For supervised DLSCA, if a profiling model is generalized well on the leakage traces, the probability of the incorrect value is closely correlated with the correct label.
- In unsupervised setting, we can still estimate the label distance, providing us with plaintext/ciphertext distribution.

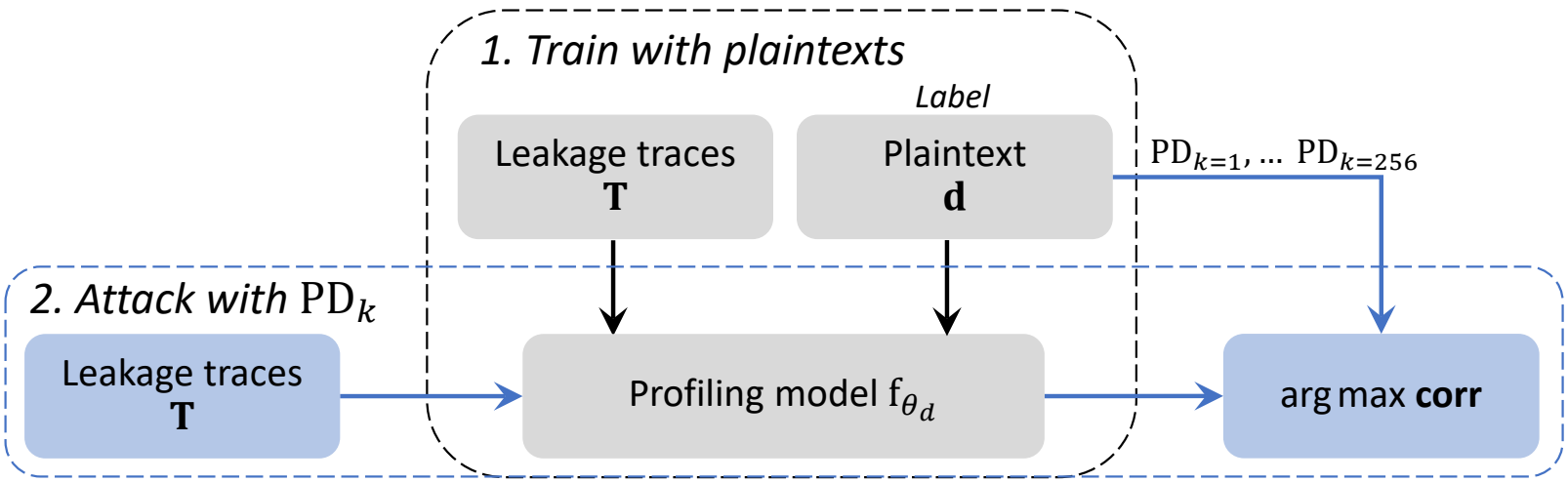


Figure: Attack scheme of the Plaintext Labeling Deep Learning (PLDL).

Plaintext/Ciphertext-based Non-profiling SCA

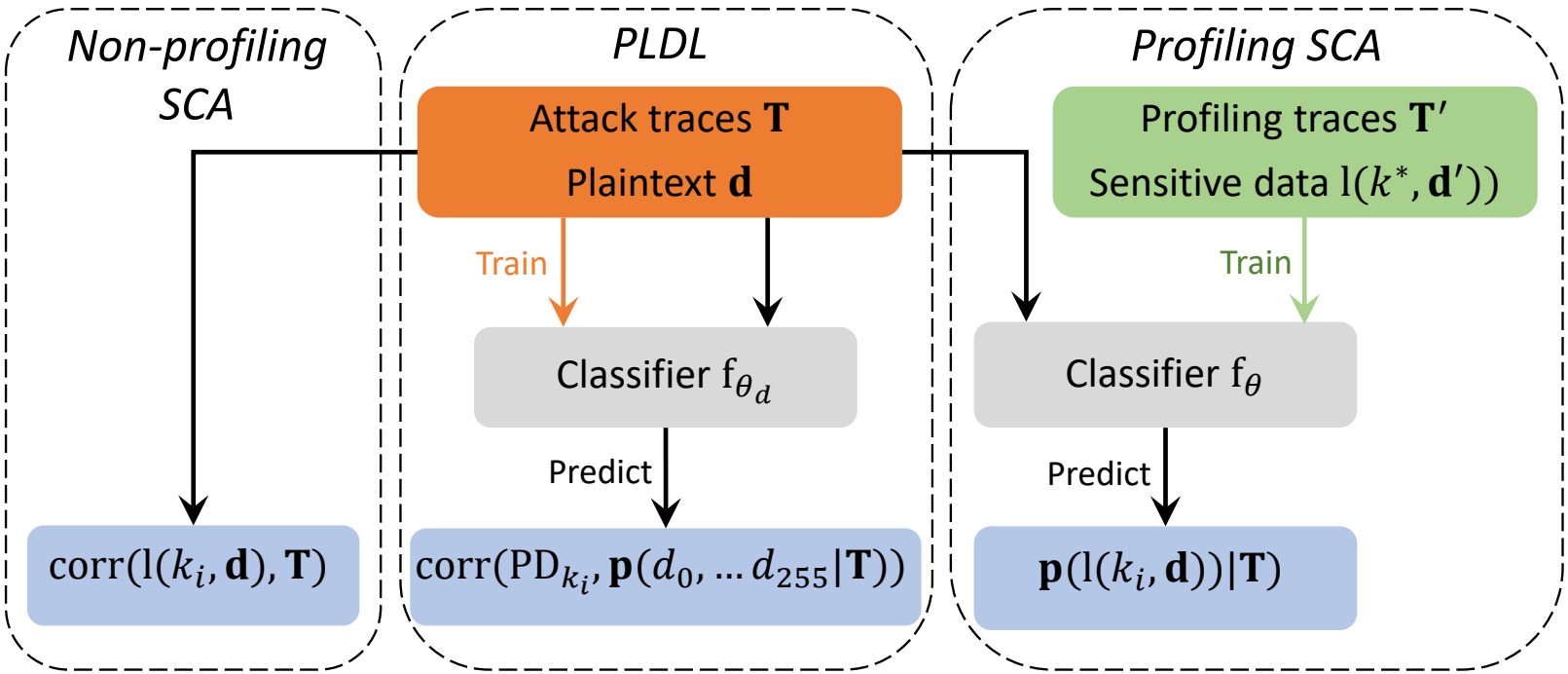


Figure: A demonstration of non-profiling SCA, PLDL, and profiling SCA.

Plaintext/Ciphertext-based Non-profiling SCA

Table: Performance benchmark with non-profiling attacks.

Dataset	CPA	MOR	DDLA	PLDL
ASCAD_F	KR161/KR47	1 957/638	KR7/309	**8**/**111**
ASCAD_R	KR64/KR8	KR28/KR9	27 266/KR48	**20**/**19**
CHES_CTF	KR139/KR220	KR6/KR31	KR54/KR85	**6 121**/**KR2**
AES_RD	KR2/KR31	KR33/3 112	2 541/KR2	**1**/**57**
AES_HD	KR19/KR145	5 593/KR10	KR26/KR20	**60**/**KR6**

Some More Things

- Explanaibility.
- Custom loss functions.
- Portability.

Outline

Raw Data

- Most of the targets are software implementations offering limited countermeasures (e.g., first-order masking and simulation of misalignment).
- Break higher-order masking dataset without knowing the shares.

Data Pre-processing

- We are missing a clear set of guidelines on what pre-processing techniques to use and in what settings.
- Considering data augmentation techniques, we can recognize two directions: either use standard machine learning techniques or customize data augmentation for SCA. There is a need for a systematic comparison of those approaches.
- If we know something about the traces, why not to use this knowledge?

Feature Engineering

- Feature engineering is not needed for deep learning-based SCA, or we need only very basic techniques. As using side-channel traces with thousands (or tens of thousands) features is common, we must investigate the possible drawbacks of using extremely lengthy traces.
- If feature engineering is required, we must understand what techniques to use.

Algorithm Selection

- As research papers consider relatively small datasets, there are no efficient guidelines to determine the hyperparameters on more realistic settings containing millions of noisy and protected side-channel traces.

Model Training

- Recognize the most important hyperparameters for deep learning-based SCA.
- Evaluate how custom neural networks can enhance the attack performance and generalize for different settings.
- As one of the dominant problems in DL-SCA is overfitting, it is necessary to investigate how to prevent or, at least reduce it.

Attack Evaluation

- Little is understood about the relationship between commonly used SCA metrics (GE, SR) and model-learned parameters (i.e., the neural network weights).
- It is unlikely to find a universal profiling model that could defeat all types of available countermeasures and that could be used in a wide variety of targets. We should measure and understand how the selected hyperparameters make the model succeed (or fail) in fitting existing leakages.

AI Explainability and SCA

- Understand how neural networks process masking countermeasures.
- Propose efficient countermeasures based on the AI explainability that are tuned to fight against deep learning-based SCA.

Outline

Conclusions

- Deep learning is efficient and powerful option for SCA.
- Current results are very promising as we can break protected targets even with very small architectures.
- What is less clear is how would the approach scale for more realistic targets.
- It is not easy to select the most promising approaches from all that is available.
- The big challenges are unsupervised deep learning-based SCA and explainable AI for SCA.