



## Deep Learning for Embedded Security Evaluation

Emmanuel Prouff

*Joint work with* Ryad Benadjila, Eleonora Cagli (CEA LETI), Cécile Dumas (CEA LETI), Housseem Maghrebi (UL), Thibault Portigliatti (ex SAFRAN), Rémi Strullu and Adrian Thillard

Laboratoire de Sécurité des Composants, ANSSI, France  
Partially funded by **REASSURE** H2020 Project

June, Journées Nationales 2018 Pré-GDR Sécurité Informatique



## Contents

1. Context and Motivation
  - 1.1 Illustration
  - 1.2 Template Attacks
  - 1.3 Countermeasures
  - 1.4 State of the Art
2. A machine learning approach to classification
  - 2.1 Introduction
  - 2.2 The Underlying Classification Problem
  - 2.3 Convolutional Neural Networks
  - 2.4 Training of Models
3. Building a Community Around The Subject
  - 3.1 ASCAD Open Data-Base
  - 3.2 Leakage Characterization and Training
  - 3.3 Results
4. Conclusions



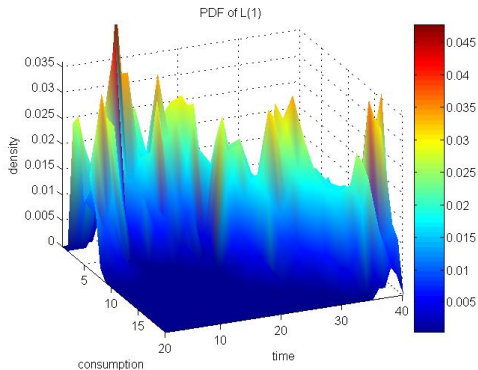
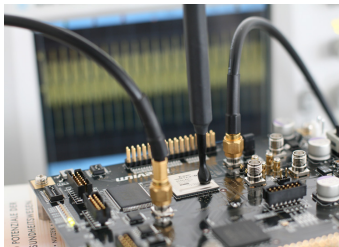
Probability distribution function (pdf) of Electromagnetic Emanations

Cryptographic Processing with a secret  $k = 1$ .



## Probability distribution function (pdf) of Electromagnetic Emanations

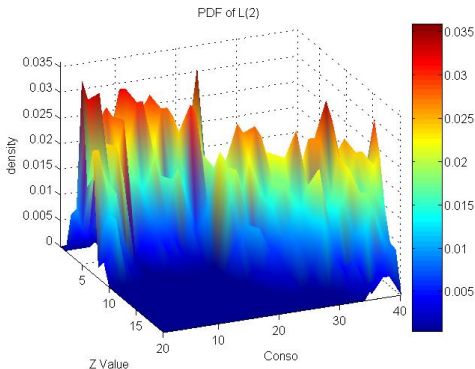
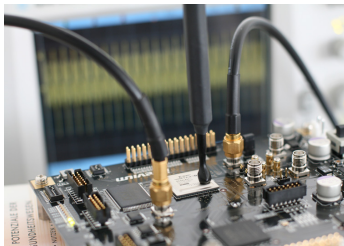
Cryptographic Processing with a secret  $k = 1$ .





## Probability distribution function (pdf) of Electromagnetic Emanations

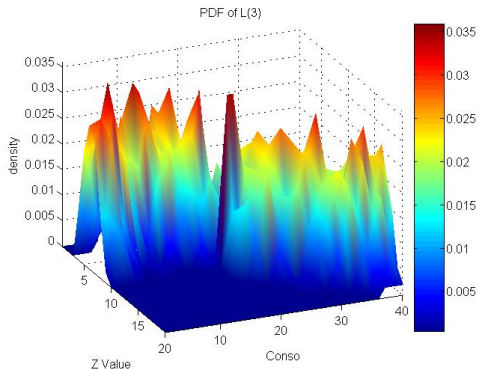
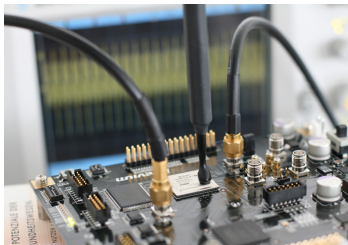
Cryptographic Processing with a secret  $k = 2$ .





## Probability distribution function (pdf) of Electromagnetic Emanations

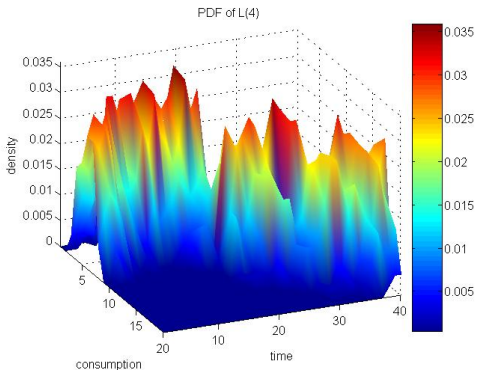
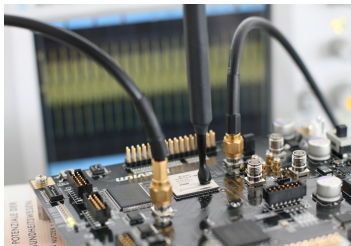
Cryptographic Processing with a secret  $k = 3$ .





## Probability distribution function (pdf) of Electromagnetic Emanations

Cryptographic Processing with a secret  $k = 4$ .



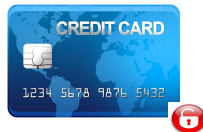


# Deep Learning for Embedded Security Evaluation

Context:



Target Device



Clone Device





Context:

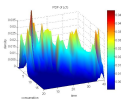


Target Device

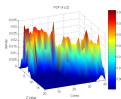


Clone Device

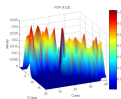
► [On Clone Device] For every  $k$  estimate the pdf of  $\vec{X} \mid K = k$ .



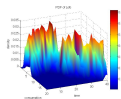
$k = 1$



$k = 2$



$k = 3$



$k = 4$

.....



Context:

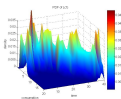


Target Device

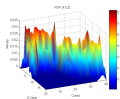


Clone Device

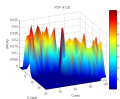
► [On Clone Device] For every  $k$  estimate the pdf of  $\vec{X} \mid K = k$ .



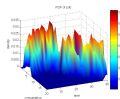
$k = 1$



$k = 2$



$k = 3$



$k = 4$

.....



Context:

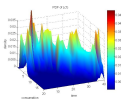


Target Device

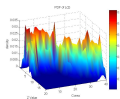


Clone Device

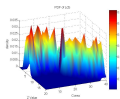
- ▶ [On Clone Device] For every  $k$  estimate the pdf of  $\vec{X} \mid K = k$ .



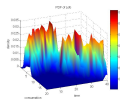
$k = 1$



$k = 2$



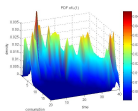
$k = 3$



$k = 4$

.....

- ▶ [On Target Device] Estimate the pdf of  $\vec{X}$ .



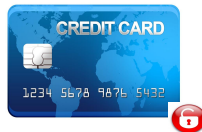
$k = ?$



Context:

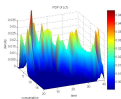


Target Device

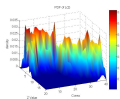


Clone Device

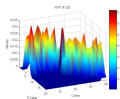
- ▶ [On Clone Device] For every  $k$  estimate the pdf of  $\vec{X} \mid K = k$ .



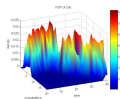
$k = 1$



$k = 2$



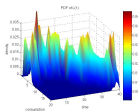
$k = 3$



$k = 4$

.....

- ▶ [On Target Device] Estimate the pdf of  $\vec{X}$ .



$k = ?$

- ▶ [Key-recovery] Compare the pdf estimations.



## Side Channel Attacks (Classical Approach)

### Notations

- ▶  $\vec{X}$  observation of the device behaviour
- ▶  $P$  public input of the processing
- ▶  $Z$  target (a cryptographic sensitive variable  $Z = f(P, K)$ )

Goal: make inference over  $Z$ , observing  $\vec{X}$



## Side Channel Attacks (Classical Approach)

### Notations

- ▶  $\vec{X}$  observation of the device behaviour
- ▶  $P$  public input of the processing
- ▶  $Z$  target (a cryptographic sensitive variable  $Z = f(P, K)$ )

Goal: make inference over  $Z$ , observing  $\vec{X}$

$\Pr[Z|\vec{X}]$



## Side Channel Attacks (Classical Approach)

### Notations

- ▶  $\vec{X}$  observation of the device behaviour
- ▶  $P$  public input of the processing
- ▶  $Z$  target (a cryptographic sensitive variable  $Z = f(P, K)$ )

Goal: make inference over  $Z$ , observing  $\vec{X}$

$\Pr[Z|\vec{X}]$

### Template Attacks

- ▶ Profiling phase (using profiling traces under known  $Z$ )
  
- ▶ Attack phase ( $N$  attack traces  $\vec{x}_i$ , e.g. with known plaintexts  $p_i$ )



## Side Channel Attacks (Classical Approach)

### Notations

- ▶  $\vec{X}$  observation of the device behaviour
- ▶  $P$  public input of the processing
- ▶  $Z$  target (a cryptographic sensitive variable  $Z = f(P, K)$ )

Goal: make inference over  $Z$ , observing  $\vec{X}$

$\Pr[Z|\vec{X}]$

### Template Attacks

- ▶ Profiling phase (using profiling traces under known  $Z$ )
  - ▶ estimate  $\Pr[\vec{X}|Z = z]$  by simple distributions for each value of  $z$
- ▶ Attack phase ( $N$  attack traces  $\vec{x}_i$ , e.g. with known plaintexts  $p_i$ )





## Side Channel Attacks (Classical Approach)

### Notations

- ▶  $\vec{X}$  observation of the device behaviour
- ▶  $P$  public input of the processing
- ▶  $Z$  target (a cryptographic sensitive variable  $Z = f(P, K)$ )

Goal: make inference over  $Z$ , observing  $\vec{X}$

$\Pr[Z|\vec{X}]$

### Template Attacks

- ▶ Profiling phase (using profiling traces under known  $Z$ )
  - ▶ estimate  $\Pr[\vec{X}|Z = z]$  for each value of  $z$
- ▶ Attack phase ( $N$  attack traces  $\vec{x}_i$ , e.g. with known plaintexts  $p_i$ )
  - ▶ Log-likelihood score for each key hypothesis  $k$

$$d_k = \sum_{i=1}^N \log \Pr[\vec{X} = \vec{x}_i | Z = f(p_i, k)]$$



## Side Channel Attacks (Classical Approach)

### Notations

- ▶  $\vec{X}$  observation of the device behaviour
- ▶  $P$  public input of the processing
- ▶  $Z$  target (a cryptographic sensitive variable  $Z = f(P, K)$ )

Goal: make inference over  $Z$ , observing  $\vec{X}$

$\Pr[Z|\vec{X}]$

### Template Attacks

- ▶ Profiling phase (using profiling traces under known  $Z$ )
  - ▶ mandatory dimensionality reduction
  - ▶ estimate  $\Pr[\vec{X}|Z = z]$  for each value of  $z$
- ▶ Attack phase ( $N$  attack traces  $\vec{x}_i$ , e.g. with known plaintexts  $p_i$ )
  - ▶ Log-likelihood score for each key hypothesis  $k$

$$d_k = \sum_{i=1}^N \log \Pr[\vec{X} = \vec{x}_i | Z = f(p_i, k)]$$



## Side Channel Attacks (Classical Approach)

### Notations

- ▶  $\vec{X}$  observation of the device behaviour
- ▶  $P$  public input of the processing
- ▶  $Z$  target (a cryptographic sensitive variable  $Z = f(P, K)$ )

Goal: make inference over  $Z$ , observing  $\vec{X}$

$\Pr[Z|\vec{X}]$

### Template Attacks

- ▶ Profiling phase (using profiling traces under known  $Z$ )
  - ▶ manage de-synchronization problem
  - ▶ mandatory dimensionality reduction
  - ▶ estimate  $\Pr[\varepsilon(\vec{X})|Z = z]$  for each value of  $z$
- ▶ Attack phase ( $N$  attack traces  $\vec{x}_i$ , e.g. with known plaintexts  $p_i$ )
  - ▶ Log-likelihood score for each key hypothesis  $k$

$$d_k = \sum_{i=1}^N \log \Pr[\varepsilon(\vec{X}) = \varepsilon(\vec{x}_i) | Z = f(p_i, k)]$$



## Defensive Mechanisms



### Misaligning Countermeasures

- ▶ Random Delays, Clock Jittering, ...
- ▶ In theory: assume to be insufficient to provide security
- ▶ In practice: one of the main issues for evaluators
- ▶  $\implies$  **Need for efficient resynchronization techniques**



## Defensive Mechanisms



### Misaligning Countermeasures

- ▶ Random Delays, Clock Jittering, ...
- ▶ In theory: assume to be insufficient to provide security
- ▶ In practice: one of the main issues for evaluators
- ▶  $\implies$  **Need for efficient resynchronization techniques**

### Masking Countermeasure

- ▶ Each key-dependent internal state element is randomly split into **2 shares**
- ▶ The crypto algorithm is adapted to always manipulate shares at  $\neq$  times
- ▶ The adversary needs to recover information on the two shares to recover  $K$
- ▶  $\implies$  **Need for efficient Methods to recover tuple of leakage samples that jointly depend on the target secret**

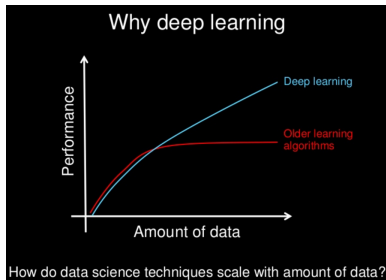


## Contents

1. Context and Motivation
  - 1.1 Illustration
  - 1.2 Template Attacks
  - 1.3 Countermeasures
  - 1.4 State of the Art
2. A machine learning approach to classification
  - 2.1 Introduction
  - 2.2 The Underlying Classification Problem
  - 2.3 Convolutional Neural Networks
  - 2.4 Training of Models
3. Building a Community Around The Subject
  - 3.1 ASCAD Open Data-Base
  - 3.2 Leakage Characterization and Training
  - 3.3 Results
4. Conclusions

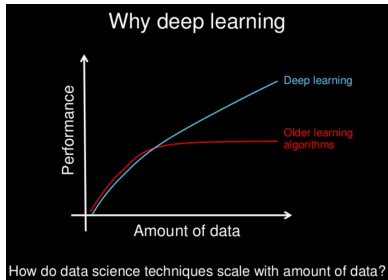


## Motivating Conclusions





## Motivating Conclusions



Now:

- ▶ preprocessing to prepare data
  - ▶ Traces resynchronisation
  - ▶ Selection of Pols
- ▶ make strong hypotheses on the statistical dependency
  - ▶ e.g. Gaussian approximation
- ▶ characterization to extract information
  - ▶ e.g. Maximum Likelihood

The proposed perspective:

- ▶ ~~preprocessing to prepare data~~
  - ▶ ~~Traces resynchronisation~~
  - ▶ ~~Selection of Pols~~
- ▶ ~~make strong hypotheses on the statistical dependency~~
  - ▶ ~~e.g. Gaussian approximation~~
- ▶ **Train** algorithms to directly extract information





## Side Channel Attacks

### Notations

- ▶  $\vec{X}$  side channel trace
- ▶  $Z$  target (a cryptographic sensitive variable  $Z = f(P, K)$ )

Goal: make inference over  $Z$ , observing  $\vec{X}$

$$\Pr[Z|\vec{X}]$$

### Template Attacks Machine Learning Side Channel Attacks

- ▶ Profiling phase (using profiling traces under known  $Z$ )
  - ▶ manage de-synchronization problem
  - ▶ mandatory dimensionality reduction
  - ▶ estimate  $\Pr[\vec{X}|Z = z]$  for each value of  $z$
- ▶ Attack phase ( $N$  attack traces, e.g. with known plaintexts  $p_i$ )
  - ▶ Log-likelihood score for each key hypothesis  $k$

$$d_k = \sum_{i=1}^N \log \Pr[\vec{X} = \vec{x}_i | Z = f(p_i, k)]$$



## Side Channel Attacks **with a Classifier**

### Notations

- ▶  $\vec{X}$  side channel trace
- ▶  $Z$  target (a cryptographic sensitive variable  $Z = f(P, K)$ )

Goal: make inference over  $Z$ , observing  $\vec{X}$

$\Pr[Z|\vec{X}]$

### ~~Template Attacks~~ Machine Learning Side Channel Attacks

- ▶ Profiling phase (using profiling traces under known  $Z$ )
  - ▶ manage de-synchronization problem
  - ▶ mandatory dimensionality reduction
  - ▶ estimate  $\Pr[\vec{X}|Z = z]$  for each value of  $z$
- ▶ Attack phase ( $N$  attack traces, e.g. with known plaintexts  $p_i$ )
  - ▶ Log-likelihood score for each key hypothesis  $k$

$$d_k = \sum_{i=1}^N \log \Pr[\vec{X} = \vec{x}_i | Z = f(p_i, k)]$$



## Side Channel Attacks **with a Classifier**

### Notations

- ▶  $\vec{X}$  side channel trace
- ▶  $Z$  target (a cryptographic sensitive variable  $Z = f(P, K)$ )

Goal: make inference over  $Z$ , observing  $\vec{X}$

$\Pr[Z|\vec{X}]$

### ~~Template Attacks~~ Machine Learning Side Channel Attacks

- ▶ Training phase (using training traces under known  $Z$ )
  - ▶ manage de-synchronization problem
  - ▶ mandatory dimensionality reduction
  - ▶ estimate  $\Pr[\vec{X}|Z = z]$  for each value of  $z$
- ▶ Attack phase ( $N$  attack traces, e.g. with known plaintexts  $p_i$ )
  - ▶ Log-likelihood score for each key hypothesis  $k$

$$d_k = \sum_{i=1}^N \log \Pr[\vec{X} = \vec{x}_i | Z = f(p_i, k)]$$



## Side Channel Attacks **with a Classifier**

### Notations

- ▶  $\vec{X}$  side channel trace
- ▶  $Z$  target (a cryptographic sensitive variable  $Z = f(P, K)$ )

Goal: make inference over  $Z$ , observing  $\vec{X}$

$$\Pr[Z|\vec{X}]$$

### ~~Template Attacks~~ Machine Learning Side Channel Attacks

- ▶ Training phase (using training traces under known  $Z$ )
  - ▶ manage de-synchronization problem
  - ▶ mandatory dimensionality reduction
  - ▶ construct a classifier  $F$  s.t.  $F(\vec{x})[z] = y \approx \Pr[Z = z|\vec{X} = \vec{x}]$
- ▶ Attack phase ( $N$  attack traces, e.g. with known plaintexts  $p_i$ )
  - ▶ Log-likelihood score for each key hypothesis  $k$

$$d_k = \sum_{i=1}^N \log \Pr[\vec{X} = \vec{x}_i | Z = f(p_i, k)]$$



## Side Channel Attacks **with a Classifier**

### Notations

- ▶  $\vec{X}$  side channel trace
- ▶  $Z$  target (a cryptographic sensitive variable  $Z = f(P, K)$ )

Goal: make inference over  $Z$ , observing  $\vec{X}$

$\Pr[Z|\vec{X}]$

### ~~Template Attacks~~ Machine Learning Side Channel Attacks

- ▶ Training phase (using training traces under known  $Z$ )
  - ▶ manage de-synchronization problem
  - ▶ mandatory dimensionality reduction
  - ▶ construct a classifier  $F$  s.t.  $F(\vec{x})[z] = y \approx \Pr[Z = z|\vec{X} = \vec{x}]$
- ▶ Attack phase ( $N$  attack traces, e.g. with known plaintexts  $p_i$ )
  - ▶ Log-likelihood score for each key hypothesis  $k$

$$d_k = \sum_{i=1}^N \log F(\vec{x}_i)[f(p_i, k)]$$



## Side Channel Attacks **with a Classifier**

### Notations

- ▶  $\vec{X}$  side channel trace
- ▶  $Z$  target (a cryptographic sensitive variable  $Z = f(P, K)$ )

Goal: make inference over  $Z$ , observing  $\vec{X}$

$$\Pr[Z|\vec{X}]$$

### ~~Template Attacks~~ Machine Learning Side Channel Attacks

- ▶ Training phase (using training traces under known  $Z$ )
    - ▶ manage de-synchronization problem
    - ▶ mandatory dimensionality reduction
    - ▶ construct a classifier  $F$  s.t.  
 $F(\vec{x})[z] = y \approx \Pr[Z = z | \vec{X} = \vec{x}]$
- } **Integrated approach**
- ▶ Attack phase ( $N$  attack traces, e.g. with known plaintexts  $p_i$ )
    - ▶ Log-likelihood score for each key hypothesis  $k$

$$d_k = \sum_{i=1}^N \log F(\vec{x}_i)[f(p_i, k)]$$



## Classification

### Classification problem

Assign to a datum  $\vec{X}$  (e.g. an image) a label  $Z$  among a set of possible labels  $Z = \{\text{Cat}, \text{Dog}, \text{Horse}\}$

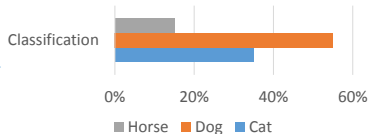
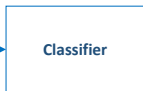




## Classification

### Classification problem

Assign to a datum  $\vec{X}$  (e.g. an image) a label  $Z$  among a set of possible labels  $Z = \{\text{Cat}, \text{Dog}, \text{Horse}\}$



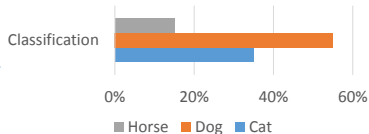




## Classification

### Classification problem

Assign to a datum  $\vec{X}$  (e.g. an image) a label  $Z$  among a set of possible labels  $\mathcal{Z} = \{\text{Cat}, \text{Dog}, \text{Horse}\}$



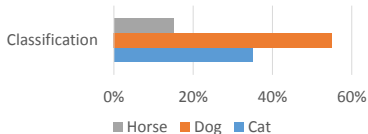
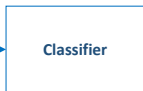
$$\Pr[Z|\vec{X}]$$



## Classification

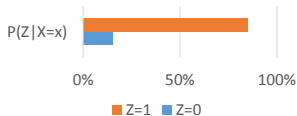
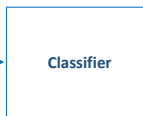
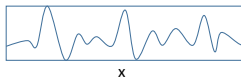
### Classification problem

Assign to a datum  $\vec{X}$  (e.g. an image) a label  $Z$  among a set of possible labels  $\mathcal{Z} = \{\text{Cat}, \text{Dog}, \text{Horse}\}$



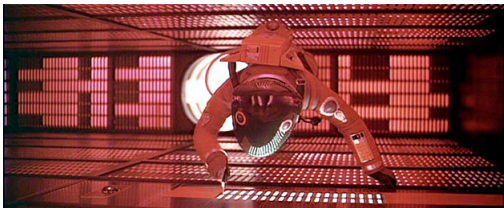
$$\Pr[Z|\vec{X}]$$

### SCA as a Classification Problem





## Machine Learning Approach



### Overview of Machine Learning Methodology

Human effort:

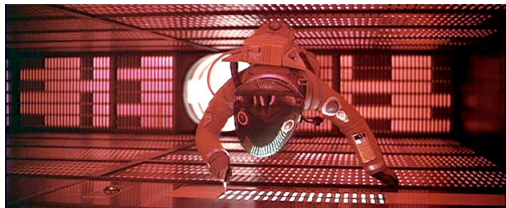
- ▶ choose a class of algorithms
- ▶ choose a model to fit + tune **hyper-parameters**

Automatic training:

- ▶ automatic tuning of **trainable parameters** to fit data



## Machine Learning Approach



### Overview of Machine Learning Methodology

Human effort:

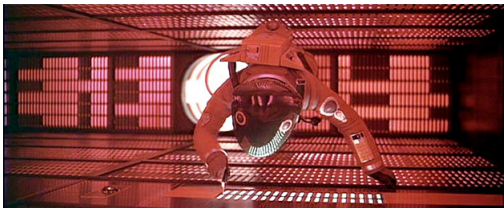
- ▶ choose a class of algorithms  
*Neural Networks*
- ▶ choose a model to fit +  
tune **hyper-parameters**

Automatic training:

- ▶ automatic tuning of  
**trainable parameters**  
to fit data  
*Stochastic Gradient Descent*



## Machine Learning Approach



### Overview of Machine Learning Methodology

Human effort:

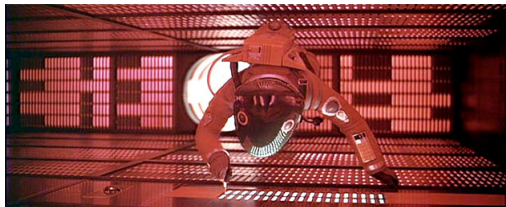
- ▶ choose a class of algorithms  
*Neural Networks*
- ▶ choose a model to fit +  
tune **hyper-parameters**  
*MLP, ConvNet*

Automatic training:

- ▶ automatic tuning of  
**trainable parameters**  
to fit data  
*Stochastic Gradient Descent*



## Machine Learning Approach



### Overview of Machine Learning Methodology

Human effort:

- ▶ choose a class of algorithms  
*Neural Networks*
- ▶ choose a model to fit +  
tune **hyper-parameters**  
*MLP, ConvNet*

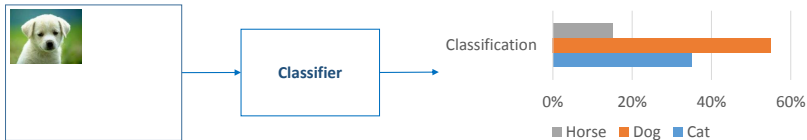
Automatic training:

- ▶ automatic tuning of  
**trainable parameters**  
to fit data  
*Stochastic Gradient Descent*



## Convolutional Neural Networks

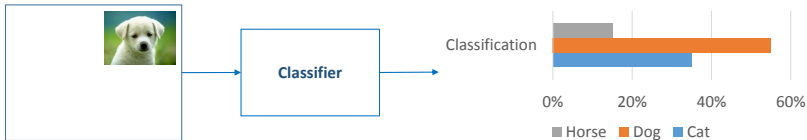
### An answer to translation-invariance





## Convolutional Neural Networks

### An answer to translation-invariance

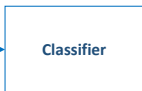
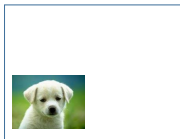




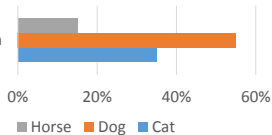


## Convolutional Neural Networks

### An answer to translation-invariance



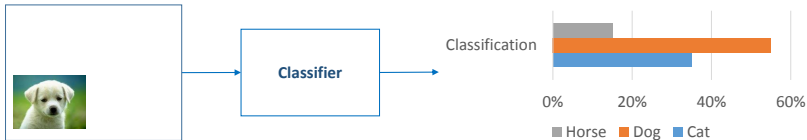
Classification





## Convolutional Neural Networks

### An answer to translation-invariance

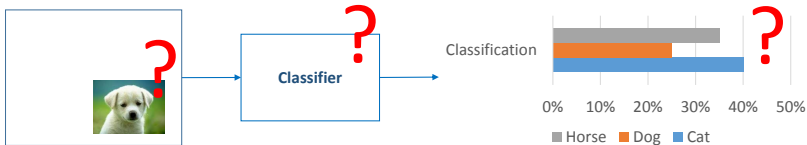


It is important to explicit the data translation-invariance



## Convolutional Neural Networks

### An answer to translation-invariance



It is important to explicit the data translation-invariance



## Convolutional Neural Networks

### An answer to translation-invariance



It is important to explicit the data translation-invariance



## Convolutional Neural Networks

### An answer to translation-invariance



It is important to explicit the data translation-invariance



## Convolutional Neural Networks

### An answer to translation-invariance



It is important to explicit the data translation-invariance



## Convolutional Neural Networks

An answer to translation-invariance



It is important to explicit the data translation-invariance  
Convolutional Neural Networks: share weights across space



## Convolutional Neural Networks

An answer to translation-invariance



It is important to explicit the data translation-invariance  
Convolutional Neural Networks: share weights across space





## Basic Example

Convolutional filtering:  $W = 2$ ,  $n_{\text{filter}} = 4$ , stride = 1, padding = same. Max pooling layer:  $W = \text{stride} = 3$ .



Example: masked manipulation of a sensitive datum  $Z$

Deep Learning Behaviour Against Masked Datum



Example: masked manipulation of a sensitive datum  $Z$

Deep Learning Behaviour Against Masked Datum



Example: masked manipulation of a sensitive datum  $Z$

Deep Learning Behaviour Against Masked Datum



## Training of Neural Networks

Trading Side-Channel Expertise for Deep Learning Expertise .... or huge computational power!

### Training



## Training of Neural Networks

Trading Side-Channel Expertise for Deep Learning Expertise .... or huge computational power!

### Training

Aims at finding the **parameters** of the function modelling for the dependency btw the target value and the leakage.



## Training of Neural Networks

Trading Side-Channel Expertise for Deep Learning Expertise .... or huge computational power!

### Training

Aims at finding the **parameters** of the function modelling for the dependency btw the target value and the leakage.

The search is done by solving a minimization problem with respect to some metric (aka loss function)



## Training of Neural Networks

Trading Side-Channel Expertise for Deep Learning Expertise .... or huge computational power!

### Training

Aims at finding the **parameters** of the function modelling for the dependency btw the target value and the leakage.

The search is done by solving a minimization problem with respect to some metric (aka loss function)

The training algorithm has itself some **training hyper-parameters**:  
the number of iterations (aka **epochs**) of the minimization procedure,  
the number of input traces (aka **batch**) treated during a single iteration.





## Training of Neural Networks

Trading Side-Channel Expertise for Deep Learning Expertise .... or huge computational power!

### Training

Aims at finding the **parameters** of the function modelling for the dependency btw the target value and the leakage.

The search is done by solving a minimization problem with respect to some metric (aka loss function)

The training algorithm has itself some **training hyper-parameters**:

- the number of iterations (aka **epochs**) of the minimization procedure,
- the number of input traces (aka **batch**) treated during a single iteration.

The trained model has **architecture hyper-parameters**:

- the size of the layers, the nature of the layers, the number of layers, etc.



## Training of Neural Networks

Trading Side-Channel Expertise for Deep Learning Expertise .... or huge computational power!

### Training

Aims at finding the **parameters** of the function modelling for the dependency btw the target value and the leakage.

The search is done by solving a minimization problem with respect to some metric (aka loss function)

The training algorithm has itself some **training hyper-parameters**:  
the number of iterations (aka **epochs**) of the minimization procedure,  
the number of input traces (aka **batch**) treated during a single iteration.

The trained model has **architecture hyper-parameters**:  
the size of the layers, the nature of the layers, the number of layers, etc.

### Tricky Points

Find sound hyper-parameters is the main issue in Deep Learning: **this can be done thanks to a good understanding of the underlying structure of the data and/or access to important computational power.**



## Contents

1. Context and Motivation
  - 1.1 Illustration
  - 1.2 Template Attacks
  - 1.3 Countermeasures
  - 1.4 State of the Art
2. A machine learning approach to classification
  - 2.1 Introduction
  - 2.2 The Underlying Classification Problem
  - 2.3 Convolutional Neural Networks
  - 2.4 Training of Models
3. Building a Community Around The Subject
  - 3.1 ASCAD Open Data-Base
  - 3.2 Leakage Characterization and Training
  - 3.3 Results
4. Conclusions



## Creation of an open database for Training and Testing

### ANSSI recently publishes

- ▶ source codes of secure implementations of AES128 for public 8-bit architectures (<https://github.com/ANSSI-FR/secAES-ATmega8515>)
  - ▶ **first version:** 10-masking + processing in random order
  - ▶ **second version:** affine masking + processing in random order (plus other minor tricks)
- ▶ data-bases of electromagnetic leakages (<https://github.com/ANSSI-FR/ASCAD>)
- ▶ example scripts for the training and testing of models in SCA contexts

### Goal

- ▶ Enable fair and easy benchmarking
- ▶ Initiate discussions and exchanges on the application of DL to SCA
- ▶ Create a community of contributors on this subject



### Nature of the Observations/Traces

Side-channel observations in ASCAD correspond to the masked processing of a simple cryptographic primitive

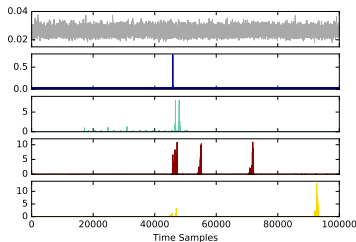
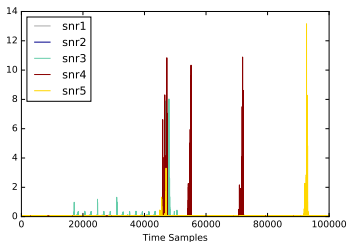
Information leakage validated thanks to SNR characterization



## Nature of the Observations/Traces

Side-channel observations in ASCAD correspond to the masked processing of a simple cryptographic primitive

Information leakage validated thanks to SNR characterization

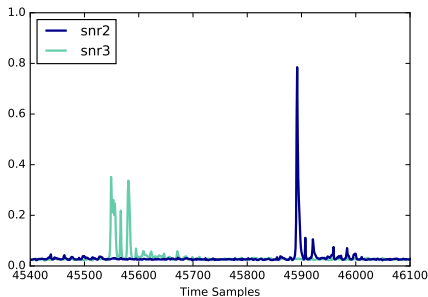


|             |                                    |                                   |
|-------------|------------------------------------|-----------------------------------|
| <i>snr1</i> | unmasked sbox output               | $SBox(p \oplus k)$                |
| <i>snr2</i> | masked sbox output                 | $SBox(p \oplus k) \oplus r_{out}$ |
| <i>snr3</i> | common sbox output mask            | $r_{out}$                         |
| <i>snr4</i> | masked sbox output in linear parts | $SBox(p \oplus k) \oplus r_{lin}$ |
| <i>snr5</i> | sbox output mask in linear parts   | $r_{lin}$                         |



## Nature of the Observations/Traces

Side-channel observations in ASCAD correspond to the masked processing of a simple cryptographic primitive  
Information leakage validated thanks to SNR characterization

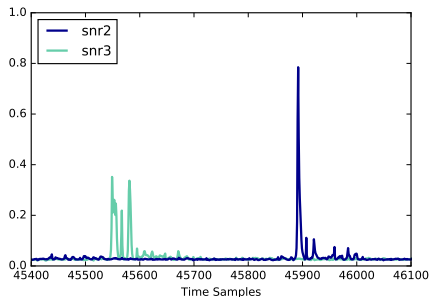


Validate that shares are **manipulated at different times**



## Nature of the Observations/Traces

Side-channel observations in ASCAD correspond to the masked processing of a simple cryptographic primitive  
Information leakage validated thanks to SNR characterization



Validate that shares are **manipulated at different times**  
Scripts are also proposed to add **artificial signal jittering**





## Our Training Strategy



### Our Training Strategy

Find a **base model architecture** and find training hyper-parameters for which a convergence towards the good key hypothesis is visible



### Our Training Strategy

Find a **base model architecture** and find training hyper-parameters for which a convergence towards the good key hypothesis is visible

Fine-tune all the hyper-parameters one after another to get the best efficiency/effectiveness trade-off



## Our Training Strategy

Find a **base model architecture** and find training hyper-parameters for which a convergence towards the good key hypothesis is visible

Fine-tune all the hyper-parameters one after another to get the best efficiency/effectiveness trade-off

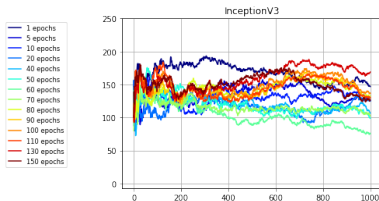
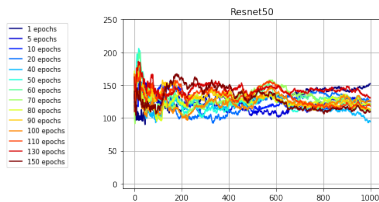
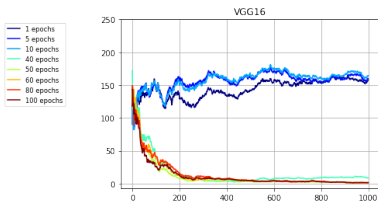
**Table:** Benchmarks Summary

| Parameter               | Reference              | Metric                 | Range                            | Choice    |
|-------------------------|------------------------|------------------------|----------------------------------|-----------|
| Training Parameters     |                        |                        |                                  |           |
| Epochs                  | -                      | rank vs time           | 10, 25, 50, 60, . . . , 100, 150 | up to 100 |
| Batch Size              | -                      | rank vs time           | 50, 100, 200                     | 200       |
| Architecture Parameters |                        |                        |                                  |           |
| Blocks                  | $n_{\text{blocks}}$    | rank, accuracy         | [2..5]                           | 5         |
| CONV layers             | $n_{\text{conv}}$      | rank, accuracy         | [0..3]                           | 1         |
| Filters                 | $n_{\text{filters},1}$ | rank vs time           | $\{2^i; i \in [4..7]\}$          | 64        |
| Kernel Size             | -                      | rank                   | {3, 6, 11}                       | 11        |
| FC Layers               | $n_{\text{dense}}$     | rank, accuracy vs time | [0..3]                           | 2         |
| ACT Function            | $\alpha$               | rank                   | ReLU, Sigmoid, Tanh              | ReLU      |
| Pooling Layer           | -                      | rank                   | Max, Average, Stride             | Average   |
| Padding                 | -                      | rank                   | Same, Valid                      | Same      |



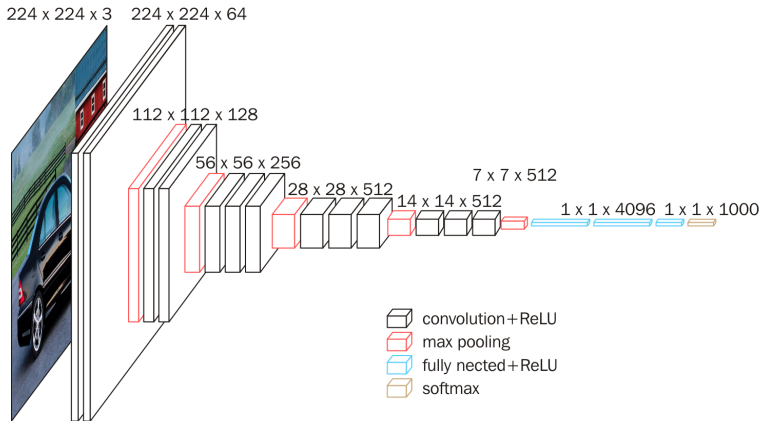
## The Base Architecture

h Mean rank of the good-key hypothesis obtained with **VGG-16**, **ResNet-50** and **Inception-v3** w.r.t. different epochs:



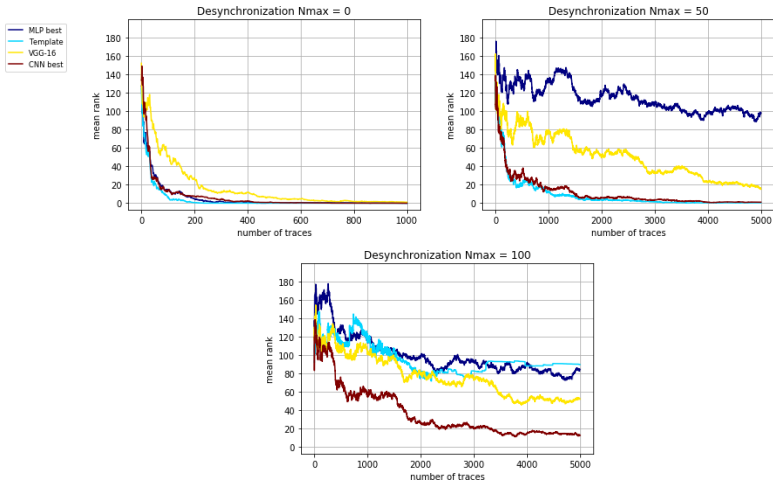


## VGG-16 Architecture





## Comparisons with State-Of-the-Art Methods





## Feedbacks & Open Issues

### Feedbacks





## Feedbacks & Open Issues

### Feedbacks

- ▶ The number of epochs for the training is between 100 and 1000



## Feedbacks & Open Issues

### Feedbacks

- ▶ The number of epochs for the training is between 100 and 1000
- ▶ Model architectures are relatively complex (more than 10 layers)



### Feedbacks & Open Issues

#### Feedbacks

- ▶ The number of epochs for the training is between 100 and 1000
- ▶ Model architectures are relatively complex (more than 10 layers)
- ▶ Data-bases for the training must be large



### Feedbacks & Open Issues

#### Feedbacks

- ▶ The number of epochs for the training is between 100 and 1000
- ▶ Model architectures are relatively complex (more than 10 layers)
- ▶ Data-bases for the training must be large
- ▶ Require important processing capacities (several GPUs, RAM memory, etc.)



## Feedbacks & Open Issues

### Feedbacks

- ▶ The number of epochs for the training is between 100 and 1000
- ▶ Model architectures are relatively complex (more than 10 layers)
- ▶ Data-bases for the training must be large
- ▶ Require important processing capacities (several GPUs, RAM memory, etc.)
- ▶ Importance of cross-validation



## Feedbacks & Open Issues

### Feedbacks

- ▶ The number of epochs for the training is between 100 and 1000
- ▶ Model architectures are relatively complex (more than 10 layers)
- ▶ Data-bases for the training must be large
- ▶ Require important processing capacities (several GPUs, RAM memory, etc.)
- ▶ Importance of cross-validation

### Open Issues



## Feedbacks & Open Issues

### Feedbacks

- ▶ The number of epochs for the training is between 100 and 1000
- ▶ Model architectures are relatively complex (more than 10 layers)
- ▶ Data-bases for the training must be large
- ▶ Require important processing capacities (several GPUs, RAM memory, etc.)
- ▶ Importance of cross-validation

### Open Issues

- ▶ Models are trained to recover manipulated values (e.g. sbx outputs) **but** not the key itself



## Feedbacks & Open Issues

### Feedbacks

- ▶ The number of epochs for the training is between 100 and 1000
- ▶ Model architectures are relatively complex (more than 10 layers)
- ▶ Data-bases for the training must be large
- ▶ Require important processing capacities (several GPUs, RAM memory, etc.)
- ▶ Importance of cross-validation

### Open Issues

- ▶ Models are trained to recover manipulated values (e.g. sbox outputs) **but** not the key itself
- ▶ Current loss functions measure the accuracy of pdf estimations **but** not the efficiency of the resulting attack





## Feedbacks & Open Issues

### Feedbacks

- ▶ The number of epochs for the training is between 100 and 1000
- ▶ Model architectures are relatively complex (more than 10 layers)
- ▶ Data-bases for the training must be large
- ▶ Require important processing capacities (several GPUs, RAM memory, etc.)
- ▶ Importance of cross-validation

### Open Issues

- ▶ Models are trained to recover manipulated values (e.g. sbox outputs) **but** not the key itself
- ▶ Current loss functions measure the accuracy of pdf estimations **but** not the efficiency of the resulting attack
- ▶ Adaptation to get (very) efficient key enumeration algorithms



## Contents

### 1. Context and Motivation

- 1.1 Illustration
- 1.2 Template Attacks
- 1.3 Countermeasures
- 1.4 State of the Art

### 2. A machine learning approach to classification

- 2.1 Introduction
- 2.2 The Underlying Classification Problem
- 2.3 Convolutional Neural Networks
- 2.4 Training of Models

### 3. Building a Community Around The Subject

- 3.1 ASCAD Open Data-Base
- 3.2 Leakage Characterization and Training
- 3.3 Results

### 4. Conclusions



## Conclusions

- ▶ State-of-the-Art Template Attack separates resynchronization/dimensionality reduction from characterization



### Conclusions

- ▶ State-of-the-Art Template Attack separates resynchronization/dimensionality reduction from characterization
- ▶ Deep Learning provides an integrated approach to directly extract information from rough data (no preprocessing)



### Conclusions

- ▶ State-of-the-Art Template Attack separates resynchronization/dimensionality reduction from characterization
- ▶ Deep Learning provides an integrated approach to directly extract information from rough data (no preprocessing)
- ▶ Many recent results validate the practical interest of the Machine Learning approach



### Conclusions

- ▶ State-of-the-Art Template Attack separates resynchronization/dimensionality reduction from characterization
- ▶ Deep Learning provides an integrated approach to directly extract information from rough data (no preprocessing)
- ▶ Many recent results validate the practical interest of the Machine Learning approach
- ▶ We are in the very beginning and we are still discovering how much Deep Learning is efficient



### Conclusions

- ▶ State-of-the-Art Template Attack separates resynchronization/dimensionality reduction from characterization
- ▶ Deep Learning provides an integrated approach to directly extract information from rough data (no preprocessing)
- ▶ Many recent results validate the practical interest of the Machine Learning approach
- ▶ We are in the very beginning and we are still discovering how much Deep Learning is efficient
- ▶ New needs:
  - ▶ big data-bases for the training,
  - ▶ platforms to enable comparisons and benchmarking,
  - ▶ create an open community "ML for Embedded Security Analysis",
  - ▶ encourage exchanges with the Machine Learning community,
  - ▶ understand the efficiency of the current countermeasures



### Conclusions

- ▶ State-of-the-Art Template Attack separates resynchronization/dimensionality reduction from characterization
- ▶ Deep Learning provides an integrated approach to directly extract information from rough data (no preprocessing)
- ▶ Many recent results validate the practical interest of the Machine Learning approach
- ▶ We are in the very beginning and we are still discovering how much Deep Learning is efficient
- ▶ New needs:
  - ▶ big data-bases for the training,
  - ▶ platforms to enable comparisons and benchmarking,
  - ▶ create an open community "ML for Embedded Security Analysis",
  - ▶ encourage exchanges with the Machine Learning community,
  - ▶ understand the efficiency of the current countermeasures

Thank You!

Questions?