Classifying Side-Channel Desynchronized Signals with Convolutional Neural Networks

Eleonora Cagli

16/04/2019, WRAC’H 2019

LETI ITSEF - Information Technology Security Evaluation Facility - CEA Grenoble
Contents

1. Context and State of the Art

2. Deep Learning against Misalignment
   2.1 Neural Network Classifiers
   2.2 Data Augmentation
   2.3 Experimental Results

3. Gradient Visualization

4. Conclusions
Side-Channel Vulnerability of Embedded Cryptography

Attack $\Rightarrow$ a secret

<table>
<thead>
<tr>
<th>Classical Attacks</th>
<th>Side-Channel Attacks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mathematical vulnerability</td>
<td>Black Box</td>
</tr>
</tbody>
</table>
Classifying Side-Channel Desynchronized Signals with Convolutional Neural Networks

Side-Channel Vulnerability of Embedded Cryptography

**Diagram:**
- **Enrollment**
- **Plaintext → Key → Ciphertext**

**Table:**
<table>
<thead>
<tr>
<th>Classical Attacks</th>
<th>Side-Channel Attacks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mathematical vulnerability</td>
<td>Black Box</td>
</tr>
</tbody>
</table>

**Attack:**
- " ATTACK "

*16/04/2019, WRAC’H 2019 | Eleonora Cagli | 3/42*
Side-Channel Vulnerability of Embedded Cryptography

Classical Attacks | Side-Channel Attacks
---|---
Mathematical vulnerability | Physical vulnerability
Black Box | Grey Box / Divide-and-conquer
Side-Channel Vulnerability of Embedded Cryptography

Simple attack: a single input

Classical Attacks | Side-Channel Attacks
--- | ---
Mathematical vulnerability, Black Box | Physical vulnerability, Grey Box / Divide-and-conquer
Advanced Side-Channel Attacks
Advanced Side-Channel Attacks
Advanced Side-Channel Attacks
Advanced Side-Channel Attacks

Classifying Side-Channel Desynchronized Signals with Convolutional Neural Networks
Advanced Side-Channel Attacks

Classifying Side-Channel Desynchronized Signals with Convolutional Neural Networks

models

$S_1$

$S_2$

$S_3$

acquisitions

$x_1$

$x_2$

$x_3$

HELLO
WHAT’S
UP?
Advanced Side-Channel Attacks

Classifying Side-Channel Desynchronized Signals with Convolutional Neural Networks
Advanced Side-Channel Attacks

Classifying Side-Channel Desynchronized Signals with Convolutional Neural Networks

HELO
WHAT'S
UP?

S3

acquisitions

x1
x2
x3

Comparison

models

S1
S2
S3

Key Hypothesis

HELLO
WHAT'S
UP?

S1
S2
S3

S1
S2
S3

16/04/2019, WRAC'H 2019 | Eleonora Cagli | 4/42
Advanced Side-Channel Attacks

Classifying Side-Channel Desynchronized Signals with Convolutional Neural Networks
Classifying Side-Channel Desynchronized Signals with Convolutional Neural Networks

Advanced Side-Channel Attacks

Differential Power Analysis [KJJ99]
Correlation Power Analysis [BCO04]
Mutual Information Analysis [Gie+08; Bat+11]...

Non-profiling attacks
Profiling attacks

Key Hypothesis

models

acquisitions

Comparison
Profiling Attacks...Supervised Learning

Target device

Clone device
Profiling Attacks...Supervised Learning

Machine Learning

Supervised Learning
Profiling Attacks...Supervised Learning

Machine Learning

Learn from data via statistic models
Task - Performance - Experience [TM97]

Supervised Learning
Profiling Attacks...Supervised Learning

Machine Learning

*Learn* from data via statistic models
Task - Performance - Experience [TM97]

Supervised Learning

The *supervised* learning algorithms access to a dataset of examples, each associated in general to a *target* or *label*. 
Classroom Side-Channel Attacks

\[ S_1 \]

Clone device

models

\[ S_1 \]
Classroom Side-Channel Attacks
Classification

Classification problem

Assign to a datum \( \tilde{X} \) a label \( Z \) among a set of possible labels \( Z = \{s_1, s_2, s_3\} \), or probabilities.
Classification

Classification problem

Assign to a datum $\tilde{X}$ a label $Z$ among a set of possible labels $Z = \{s_1, s_2, s_3\}$, or probabilities.

Advanced Attack as Multiple Classification Problems

acquisitions

$\begin{align*}
x_1 \rightarrow & \text{Classifier} \rightarrow S_3 \\
x_2 \rightarrow & \text{Classifier} \rightarrow S_2 \\
x_3 \rightarrow & \text{Classifier} \rightarrow S_1
\end{align*}$

RECOMBINATION

16/04/2019, WRAC’H 2019 | Eleonora Cagli | 7/42
Classification

Machine Learning classifiers in Side-Channel literature: SVM ([Hos+11; HZ12]), RF ([LBM14; LBM15])

Classification problem

Assign to a datum $\tilde{X}$ a label $Z$ among a set of possible labels $Z = \{s_1, s_2, s_3\}$, or probabilities.

Advanced Attack as Multiple Classification Problems
Notations and generalities

- Side-channel traces: realizations of a random vector $\vec{X} \in \mathbb{R}^D$
- $D$ is the number of time samples (or features)
- Target: a sensitive variable $Z = f(e, k)$ in $Z = \{s_1, \ldots, s_{|Z|}\}$

Profiling attack scenario

- labelled traces $\mathcal{D}_{\text{train}} = (\vec{x}_i, e_i, k_i)_{i=1}^N$, acquired under known secrets
- attack traces $\mathcal{D}_{\text{attack}} = (\vec{x}_i, e_i)_{i=1}^{N_a}$ acquired under unknown secrets
Profiling Attack

Profiling phase

- estimate
  - $p_{\tilde{X}} \mid z=z$

Attack phase

- Likelihood score for each key hypothesis $k$

$$d_k = p_{\tilde{X}} \mid z \left( (\tilde{x}_i)_{i=1,\ldots,N_a}, (f(e_i, k))_{i=1,\ldots,N_a} \right)$$
Profiling Attack

Profiling phase

- estimate
  - \( p_{\tilde{X} \mid z=z} \) (generative model)
  - \( p_{Z \mid \tilde{X}=\tilde{x}} \) (discriminative model)

Attack phase

- Likelihood score for each key hypothesis \( k \)
  \[
  d_k = p_{\tilde{X} \mid z} \left( (\tilde{x}_i)_{i=1,\ldots,N_a}, (f(e_i, k))_{i=1,\ldots,N_a} \right)
  \]

- A-posteriori probability score for each key hypothesis \( k \)
  \[
  d_k = p_{Z \mid \tilde{X}} \left( f(e_i, k)_{i=1,\ldots,N_a}, (\tilde{x}_i)_{i=1,\ldots,N_a} \right)
  \]
Profiling Attack

Profiling phase

▷ estimate
  ▷ $p_{\tilde{X}} \mid Z=z \ p_{\tilde{X}} \ p_{Z}$ (generative model)
  ▷ $p_{Z} \mid \tilde{X}=\tilde{x}$ (discriminative model)

Attack phase

▷ Likelihood score for each key hypothesis $k$
  \[ d_k = p_{\tilde{X}} \mid Z \left( (\tilde{x}_i)_{i=1,...,N_a}, (f(e_i, k))_{i=1,...,N_a} \right) \]

▷ A-posteriori probability score for each key hypothesis $k$
  \[ d_k = p_{Z} \mid \tilde{X} \left( f(e_i, k)_{i=1,...,N_a}, (\tilde{x}_i)_{i=1,...,N_a} \right) \]
Profiling Attack

**Profiling phase**

- estimate
  - \( p_{\vec{X} \mid Z=z} p_{\vec{X}} p_{Z} \) (generative model)
  - \( p_{Z \mid \vec{X}=\vec{x}} \) (discriminative model)

**Attack phase**

- Likelihood score for each key hypothesis \( k \)
  \[
  d_k = p_{\vec{X} \mid Z} \left( (\vec{x}_i)_{i=1,...,N_a}, (f(e_i, k))_{i=1,...,N_a} \right)
  \]

- A-posteriori probability score for each key hypothesis \( k \)
  \[
  d_k = p_{Z \mid \vec{X}} \left(f(e_i, k)_{i=1,...,N_a}, (\vec{x}_i)_{i=1,...,N_a} \right),
  \]
Classifying Side-Channel Desynchronized Signals with Convolutional Neural Networks

Profiling Attack

**Profiling phase**

- estimate
  - $p_{\tilde{X} \mid Z=z} \ p_{\tilde{X}} \ p_Z$ (generative model)
  - Gaussian hypothesis (Template Attack) [CRR03]
  - $p_{Z \mid \tilde{X}=\tilde{x}}$ (discriminative model)

**Attack phase**

- Likelihood score for each key hypothesis $k$
  $$d_k = p_{\tilde{X} \mid Z} \left((\tilde{x}_i)_{i=1,\ldots,N_a}, (f(e_i, k))_{i=1,\ldots,N_a}\right)$$

- A-posteriori probability score for each key hypothesis $k$
  $$d_k = p_{Z \mid \tilde{X}} \left(f(e_i, k)_{i=1,\ldots,N_a}, (\tilde{x}_i)_{i=1,\ldots,N_a}\right)$$

$\tilde{X} \in \mathbb{R}^D$

Curse of dimensionality!
Profiling Attack

Profiling phase

- mandatory dimensionality reduction $[\mathcal{D}_{\text{train}} \rightarrow \epsilon: \mathbb{R}^D \rightarrow \mathbb{R}^C]$
- estimate
  - $p_{\epsilon(\tilde{X})} | Z=z \ p_{\epsilon(\tilde{X})} \ p_Z$ (generative model)
  - Gaussian hypothesis (Template Attack) [CRR03]
  - $p_Z | \epsilon(\tilde{X})=\epsilon(\tilde{x})$ (discriminative model)

Attack phase

- Likelihood score for each key hypothesis $k$
  
  $d_k = p_{\epsilon(\tilde{X})} | Z \left( (\epsilon(\tilde{x}_i))_{i=1,\ldots,N_a}, (f(e_i, k))_{i=1,\ldots,N_a} \right)$

- A-posteriori probability score for each key hypothesis $k$
  
  $d_k = p_{Z} | \epsilon(\tilde{X}) \left( f(e_i, k)_{i=1,\ldots,N_a}, (\epsilon(\tilde{x}_i))_{i=1,\ldots,N_a} \right)$. 

Curse of dimensionality!
Profiling Attack

Profiling phase

- manage desynchronization problem \([\mathcal{D}_{\text{train}} \rightarrow \rho : \mathbb{R}^D \rightarrow \mathbb{R}^D]\)
- mandatory dimensionality reduction \([\mathcal{D}_{\text{train}} \rightarrow \epsilon : \mathbb{R}^D \rightarrow \mathbb{R}^C]\)
- estimate
  - \(p_{\epsilon(\rho(\vec{X}))} | Z=z \ p_{\epsilon(\rho(\vec{X}))} \) (generative model)
    - Gaussian hypothesis (Template Attack) [CRR03]
  - \(p_Z | \epsilon(\rho(\vec{X}))=\epsilon(\rho(\vec{X}))\) (discriminative model)

Attack phase

- Likelihood score for each key hypothesis \(k\)
  \[d_k = p_{\epsilon(\rho(\vec{X}))} | Z \left( (\epsilon(\rho(\vec{x}_i)))_{i=1,\ldots,N_a} , (f(e_i, k))_{i=1,\ldots,N_a} \right)\]

- A-posteriori probability score for each key hypothesis \(k\)
  \[d_k = p_Z | \epsilon(\rho(\vec{x})) \left( f(e_i, k)_{i=1,\ldots,N_a} , (\epsilon(\rho(\vec{x}_i)))_{i=1,\ldots,N_a} \right) ,\]
Classifying Side-Channel Desynchronized Signals with Convolutional Neural Networks

Mandatory Dimensionality Reduction

A vast domain

Features (Points of Interests - PoI) selection

- SOD [CRR03]
- SOST [BDP10]
- SNR [MOP08]/ NICV [Bha+14]
- \(t\)-test, \(F\)-test,... [GLRP06; CK14]

Feature extraction

- Principal Component Analysis (PCA) [Arc+06; BHW12]
- Linear Discriminant Analysis (LDA) [SA08; Bru+15]
- Projection Pursuits (PP) [Dur+15]

Figure: SNR computed on synchronized traces.
Manage desynchronization problem

### Misaligning Countermeasures
- Random Delays, Clock Jittering, ...
- In theory: insufficient to provide security, since information still leak (somewhere)
- In practice: one of the main issues for evaluators

**Figure**: SNR computed on desynchronized traces.
Manage desynchronization problem

Misaligning Countermeasures

- Random Delays, Clock Jittering, ...
- In theory: insufficient to provide security, since information still leak (somewhere)
- In practice: one of the main issues for evaluators

Realignment

Mandatory realignment preprocessing
- not a wide literature
- in practice: evaluation labs home-made realignment techniques
- signal deformations or pattern extraction based on prior unverified assumptions
- Risks:
  - deformations $\rightarrow$ information degradation
  - pattern extraction $\rightarrow$ information loss
Classifying Side-Channel Desynchronized Signals with Convolutional Neural Networks

This talk

Profiling phase

- manage de-synchronization problem \([D_{\text{train}} \rightarrow \rho: \mathbb{R}^D \rightarrow \mathbb{R}^D]\)
- mandatory dimensionality reduction \([D_{\text{train}} \rightarrow \epsilon: \mathbb{R}^D \rightarrow \mathbb{R}^C]\)
- estimate
  - \(p_{\epsilon(\rho(\vec{x}))) \mid Z=z, \epsilon(\rho(\vec{x}))}, p_{\epsilon}(\vec{Z})(\text{generative model})\)
    - Gaussian hypothesis (Template Attack)[CRR03]
  - \(p_{\epsilon}(\rho(\vec{x}))(\text{discriminative model})\)

This talk

Convolutional Neural Network: integrated approach (deal desynchronization + extraction feature + approximate a discriminative model)
This talk

**Profiling phase**

- manage de-synchronization problem \( D_{\text{train}} \to \rho : \mathbb{R}^D \to \mathbb{R}^D \)
- mandatory dimensionality reduction \( D_{\text{train}} \to \epsilon : \mathbb{R}^D \to \mathbb{R}^C \)
- estimate
  - \( p_\epsilon(\rho(\vec{X})) \mid Z = z \), \( p_\epsilon(\rho(\vec{X})) \), \( p_Z \) (generative model)
  - Gaussian hypothesis (Template Attack)\([CRR03]\)

- \( p_Z \mid \vec{x} \) (discriminative model)
  by means of a neural network \( \hat{\rho}(\vec{x}, W) \approx p_Z \mid \vec{x} = \vec{x} \)

This talk

Convolutional Neural Network: integrated approach (deal desynchronization + extraction feature + approximate a discriminative model)
Classifying Side-Channel Desynchronized Signals with Convolutional Neural Networks

Contents

1. Context and State of the Art

2. Deep Learning against Misalignment
   2.1 Neural Network Classifiers
   2.2 Data Augmentation
   2.3 Experimental Results

3. Gradient Visualization

4. Conclusions
Multi-Layer Perceptron

In SCA literature [MHM13; MZ13; MMT15; MDM16]

Multi-Layer Perceptron (MLP)

\[ \hat{p}(\vec{x}, W) = s \circ \lambda_n \circ \sigma_{n-1} \circ \lambda_{n-1} \circ \cdots \circ \lambda_1(\vec{x}) = \vec{y} \approx p_{Z \mid \vec{x} = \vec{x}} \]
Multi-Layer Perceptron

In SCA literature [MHM13; MZ13; MMT15; MDM16]

Multi-Layer Perceptron (MLP)

\[ \hat{p}(\tilde{x}, W) = s \circ \lambda_n \circ \sigma_{n-1} \circ \lambda_{n-1} \circ \cdots \circ \lambda_1(\tilde{x}) = \tilde{y} \approx p_Z | \tilde{x} = \tilde{x} \]

\( \lambda_i \) linear functions (linear combinations of time samples) depending on some trainable weights \( W \)

---

**Figure:** Linear layer in an MLP (Fully Connected Layer)
Classifying Side-Channel Desynchronized Signals with Convolutional Neural Networks

Multi-Layer Perceptron

In SCA literature [MHM13; MZ13; MMT15; MDM16]

Multi-Layer Perceptron (MLP)

\[
\hat{p}(\vec{x}, W) = s \circ \lambda_n \circ \sigma_{n-1} \circ \lambda_{n-1} \circ \ldots \circ \lambda_1(\vec{x}) = \vec{y} \approx p_{Z|\vec{x}=\vec{x}}
\]

- \(\lambda_i\) linear functions (linear combinations of time samples) depending on some \textbf{trainable weights} \(W\)
- \(\sigma_i\) non-linear \textit{activation} functions

![Matrix of weights 9x11 parameters](image)
Multi-Layer Perceptron

In SCA literature [MHM13; MZ13; MMT15; MDM16]

Multi-Layer Perceptron (MLP)

\[ \hat{p}(\tilde{x}, W) = s \circ \lambda_n \circ \sigma_{n-1} \circ \lambda_{n-1} \circ \cdots \circ \lambda_1(\tilde{x}) = \tilde{y} \approx p_{Z | \tilde{x} = \tilde{x}} \]

\( \lambda_i \) linear functions (linear combinations of time samples) depending on some **trainable weights** \( W \)

\( \sigma_i \) non-linear **activation** functions

\( s \) normalizing **softmax** function
Classifying Side-Channel Desynchronized Signals with Convolutional Neural Networks

Multi-Layer Perceptron

In SCA literature [MHM13; MZ13; MMT15; MDM16]

Multi-Layer Perceptron (MLP)

\[ \hat{\rho}(\vec{x}, W) = s \circ \lambda_n \circ \sigma_{n-1} \circ \lambda_{n-1} \circ \cdots \circ \lambda_1(\vec{x}) = \vec{y} \approx p_Z | \vec{x} = \vec{x} \]

- \( \lambda_i \) linear functions (linear combinations of time samples) depending on some **trainable weights** \( W \)
- \( \sigma_i \) non-linear **activation** functions
- \( s \) normalizing **softmax** function

Architecture hyper-parameters

![Matrix of weights](image_url)
Multi-Layer Perceptron

In SCA literature [MHM13; MZ13; MMT15; MDM16]

**Multi-Layer Perceptron (MLP)**

\[ \hat{p}(\vec{x}, W) = s \circ \lambda_n \circ \sigma_{n-1} \circ \lambda_{n-1} \circ \cdots \circ \lambda_1(\vec{x}) = \vec{y} \approx p_z | \vec{x} = \vec{z} \]

- \( \lambda_i \) linear functions (linear combinations of time samples) depending on some **trainable weights** \( W \)
- \( \sigma_i \) non-linear **activation** functions
- \( s \) normalizing **softmax** function

Architecture hyper-parameters

Universal approximation theorem
Convolutional Neural Networks

Translation-Invariance

Classification

Horse
Dog
Cat

Classifier

0% 20% 40% 60%

Horse  Dog  Cat
Convolutional Neural Networks

Translation-Invariance

Classifier

Classification

0% 20% 40% 60%

Horse | Dog | Cat

16/04/2019, WRAC’H 2019 | Eleonora Cagli | 15/42
Classifying Side-Channel Desynchronized Signals with Convolutional Neural Networks

Convolutional Neural Networks

Translation-Invariance

Classifier

Classification

0% 20% 40% 60%
Horse Dog Cat

0% 20% 40% 60%
Horse Dog Cat
Convolutional Neural Networks

Translation-Invariance

\[ P(Z|X=x) \]

\[ Z=1 \quad Z=0 \]

Classification of side-channel desynchronized signals with Convolutional Neural Networks.
Convolutional Neural Networks

Translation-Invariance

Classifier

P(Z|X=x)

Z=1
Z=0

x
Convolutional Neural Networks

Translation-Invariance

![Signal with Classifier](image)

\[ P(Z|X=x) \]

\[ Z=1 \quad Z=0 \]
Classifying Side-Channel Desynchronized Signals with Convolutional Neural Networks

Convolutional Layers

Figure: Linear layer in an MLP.

Figure: Convolutional layer in a CNN.
Pooling Layers

**Figure:** Convolutional layer in a CNN.

**Figure:** Pooling layer in a CNN.
A kind of CNN architecture

Architecture inspired by AlexNet \cite{KSH12}, VGG \cite{SZ14}, ResNet \cite{He+16} design rules:

- Reduce temporal features to only one
- Maintain time complexity of each layer (one-half pooling when number of feature maps is doubled)

CHES 2017 - Convolutional Neural Networks with Data Augmentation Against Jitter-Based Countermeasures - Profiling Attacks Without Pre-processing. E. Cagli - C. Dumas - E. Prouff

- 4 Conv + Pool layers
- tanh activations
- batch normalisation \cite{IS15}
- 1 fully connected layer + softmax
Training and Validation (1)

- Profiling Traces
- Validation Traces
- Training Traces
- Architecture Hyper-parameters
- Cost function
- Gradient Descent Parameters update
- Training Hyper-parameters
  - Learning Rate
Cost function - Cross-entropy

- Batch of training data \((\vec{x}_i, z_i)_{i \in I}\), outputs of the current model \((\vec{y}_i)_{i \in I}\)
- Labels \(z_i = s_j\) are one-hot encoded: \(\vec{z}_i = \vec{s}_j = (0, \ldots, 0, 1_j, 0, \ldots, 0)\)

Loss function

\[
\mathcal{L} = -\frac{1}{|I|} \sum_{i \in I} \sum_{t=1}^{|Z|} \vec{z}_i[t] \log \vec{y}_i[t] \tag{1}
\]

Maximum-a-posteriori or Cross-entropy

- \(\vec{y}_i \approx \Pr[Z \mid \vec{X} = \vec{x}_i]\)
Cost function - Cross-entropy

- batch of training data \((\vec{x}_i, z_i)_{i \in I}\), outputs of the current model \((\vec{y}_i)_{i \in I}\)
- labels \(z_i = s_j\) are one-hot encoded: \(\vec{z}_i = \vec{s}_j = (0, \ldots, 0, 1_j, 0, \ldots, 0)\)

Loss function

\[
\mathcal{L} = -\frac{1}{|I|} \sum_{i \in I} \sum_{t=1}^{|Z|} \vec{z}_i[t] \log \vec{y}_i[t]
\]

Maximum-a-posteriori or Cross-entropy

- \(\vec{y}_i \approx \Pr[Z | \vec{X} = \vec{x}_i]\)
- \(\vec{z}_i \approx \Pr[Z | Z = \vec{s}_j]\)
- \(\mathcal{H}(\vec{z}_i, \vec{y}_i) = \mathcal{H}(\vec{z}_i) + D_{KL}(\vec{z}_i || \vec{y}_i) = \mathbb{E}_{\vec{z}_i}[- \log \vec{y}_i] = - \sum_{t=1}^{|Z|} \vec{z}_i[t] \log \vec{y}_i[t]\)
Training and Validation (2)
Classifying Side-Channel Desynchronized Signals with Convolutional Neural Networks

Training and Validation (2)

Training Traces
- Batches

Validation Traces

Profiling Traces

Architecture Hyper-parameters

Cost function

Parameters

Stochastic Gradient Descent
- Parameters update
- Epochs
- Learning Rate

Training
- Hyper-parameters

16/04/2019, WRAC’H 2019 | Eleonora Cagli | 21/42
Training and Validation (2)

Profiling Traces

Validation Traces

Training Traces

Architecture
Hyper-parameters

\( \hat{p} \)

Parameters

Performance measure

Training set
~?

Validation set

VALIDATION

Training
Hyper-parameters
Training and Validation (2)

- Attack Traces
- Performance measure
- Attack

Parameters

TEST
Overfitting

Accuracy

\[
\frac{\text{Correct predictions}}{\text{Total predictions}}
\]

Evaluate and compare training and validation accuracy

Learn by heart (OVERFITTING)

Accuracy

Training

Validation

Epoch
Overfitting

**Accuracy**

\[
\text{Correct predictions} \over \text{Total predictions}
\]

**Evaluate and compare training and validation accuracy**

Understand significant features

Learn by heart (OVERFITTING)

Accuracy

Training

Validation

Epoch
Overfitting

Accuracy

\[
\text{Correct predictions} \quad \frac{}{} \quad \text{Total predictions}
\]

Evaluate and compare training and validation accuracy

Why?
- Too complex model
- Not enough training data

Solution?
- Reduce model capacity
- Regularization
- Dropout
- Early-Stopping
- Data augmentation

Learn by heart (OVERFITTING)

Accuracy

Training

Validation

Epoch
Overfitting

**Accuracy**

\[
\frac{\text{Correct predictions}}{\text{Total predictions}}
\]

**Evaluate and compare training and validation accuracy**

Why?
- Too complex model
- Not enough training data

Solution?
- Reduce model capacity
- Regularization
- Dropout
- Early-Stopping
- Data augmentation

Learn by heart (OVERFITTING)
Data Augmentation

Artificially generate new training data by deforming those previously acquired, applying transformations that preserve the label \( Z \).
Classifying Side-Channel Desynchronized Signals with Convolutional Neural Networks

Side-Channel Data Augmentation

Countermeasure Emulation Idea

Emulate the effects of misaligning countermeasures to generate new traces

**SHIFTING**

![Diagram showing shifting window and time samples](image)

**ADD-REMOVE**

![Diagram showing original and augmented traces](image)

**Figure: \( SH_T \)**

Parameter \( T \): \# of possible positions

Parameter \( R \): \# of added and removed points

Data Augmentation techniques are applied online during training phase.

**Figure: \( AR_R \)**
Training with Data Augmentation

1. **Profiling Traces**
2. **Validation Traces**
3. **Training Traces**
   - **Augmented Traces**
   - **Batches**

**Parameters**
- Architecture
- Hyper-parameters

**Cost function**

**Stochastic Gradient Descent**
- Parameters update
- Epochs
- Learning Rate

**TRAINING**

**16/04/2019, WRAC'H 2019 | Eleonora Cagli | 25/42**
Training with Data Augmentation

- **Profiling Traces**
- **Training Traces**
- **Validation Traces**

**TRAINING**

- Augmented Batch
- Batches
- Architecture
  - Hyper-parameters
- Parameters
- Cost function
- Stochastic Gradient Descent
  - Parameters update
  - Epochs
  - Learning Rate

**Training Hyper-parameters**
Experimental Results

- Random delays (software countermeasure)
- Artificial Jitter (simulated hardware countermeasure)
- Real Jitter (hardware countermeasure)

Keras 1.2.1 library with Tensorflow backend [Cho+15] (open source, today 2.2.4)
Experimental Results

- Random delays (software countermeasure)
- Artificial Jitter (simulated hardware countermeasure)
- Real Jitter (hardware countermeasure)

Keras 1.2.1 library with Tensorflow backend [Cho+15] (open source, today 2.2.4)
Random delays

(a) One leaking operation

Setup

- Target Chip: Atmega328P
- Target Variable: $Z = \text{HW}(\text{Sbox}(P \oplus K))$
- Acquisition: through ChipWhisperer[OC14] platform, $\approx 4,000$ time samples
- Countermeasure: Random Delays - insertion of $r$ \textit{nop} operations, $r \in [0, 127]$ uniform random
- 1,000 training traces
Classifying Side-Channel Desynchronized Signals with Convolutional Neural Networks

Random delays
Data augmentation vs overfitting

Training

<table>
<thead>
<tr>
<th>Epoch</th>
<th>Training Accuracy</th>
<th>Validation Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.2</td>
<td>0.1</td>
</tr>
<tr>
<td>20</td>
<td>0.4</td>
<td>0.3</td>
</tr>
<tr>
<td>40</td>
<td>0.6</td>
<td>0.5</td>
</tr>
<tr>
<td>60</td>
<td>0.8</td>
<td>0.7</td>
</tr>
<tr>
<td>80</td>
<td>1.0</td>
<td>0.9</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Epoch</th>
<th>Training Accuracy</th>
<th>Validation Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.2</td>
<td>0.1</td>
</tr>
<tr>
<td>20</td>
<td>0.4</td>
<td>0.3</td>
</tr>
<tr>
<td>40</td>
<td>0.6</td>
<td>0.5</td>
</tr>
<tr>
<td>60</td>
<td>0.8</td>
<td>0.7</td>
</tr>
<tr>
<td>80</td>
<td>1.0</td>
<td>0.9</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Epoch</th>
<th>Training Accuracy</th>
<th>Validation Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.2</td>
<td>0.1</td>
</tr>
<tr>
<td>20</td>
<td>0.4</td>
<td>0.3</td>
</tr>
<tr>
<td>40</td>
<td>0.6</td>
<td>0.5</td>
</tr>
<tr>
<td>60</td>
<td>0.8</td>
<td>0.7</td>
</tr>
<tr>
<td>80</td>
<td>1.0</td>
<td>0.9</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Epoch</th>
<th>Training Accuracy</th>
<th>Validation Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.2</td>
<td>0.1</td>
</tr>
<tr>
<td>20</td>
<td>0.4</td>
<td>0.3</td>
</tr>
<tr>
<td>40</td>
<td>0.6</td>
<td>0.5</td>
</tr>
<tr>
<td>60</td>
<td>0.8</td>
<td>0.7</td>
</tr>
<tr>
<td>80</td>
<td>1.0</td>
<td>0.9</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Predicted label</th>
<th>True label</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Predicted label</th>
<th>True label</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Predicted label</th>
<th>True label</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>

SH₀

SH₁₀₀

SH₅₀₀
Random delays
Data augmentation vs overfitting

Training

<table>
<thead>
<tr>
<th></th>
<th>SH&lt;sub&gt;0&lt;/sub&gt;</th>
<th>SH&lt;sub&gt;100&lt;/sub&gt;</th>
<th>SH&lt;sub&gt;500&lt;/sub&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Epoch</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>training accuracy</td>
<td>validation accuracy</td>
<td>training accuracy</td>
</tr>
<tr>
<td></td>
<td>0.2</td>
<td>0.3</td>
<td>0.4</td>
</tr>
<tr>
<td></td>
<td>0.9</td>
<td>0.8</td>
<td>0.7</td>
</tr>
<tr>
<td></td>
<td>0.8</td>
<td>0.7</td>
<td>0.6</td>
</tr>
<tr>
<td></td>
<td>0.7</td>
<td>0.6</td>
<td>0.5</td>
</tr>
<tr>
<td></td>
<td>0.6</td>
<td>0.5</td>
<td>0.4</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>0.4</td>
<td>0.3</td>
</tr>
</tbody>
</table>

Attack

Table: $N^*$ = number of attack traces to have $GE = 1.$
Random Delays - Two Leaking Operations

Two leaking operations

First operation - Test acc: 76.8%, $N^* = 7$
Second operation - Test acc: 82.5%, $N^* = 6$
Conclusions about CNN

- CNNs provide an integrated approach to construct a discriminative model from misaligned data
Conclusions about CNN

- CNNs provide an integrated approach to construct a discriminative model from misaligned data
- CNN models may have high capacity and require plenty of data to be trained
Conclusions about CNN

- CNNs provide an integrated approach to construct a discriminative model from misaligned data
- CNN models may have high capacity and require plenty of data to be trained
- Side-Channel-adapted Data Augmentation techniques
Conclusions about CNN

- CNNs provide an integrated approach to construct a discriminative model from misaligned data
- CNN models may have high capacity and require plenty of data to be trained
- Side-Channel-adapted Data Augmentation techniques
- Effectiveness/efficiency of the CNN+Data Augmentation approach experimentally verified
Conclusions about CNN

- CNNs provide an integrated approach to construct a discriminative model from misaligned data
- CNN models may have high capacity and require plenty of data to be trained
- Side-Channel-adapted Data Augmentation techniques
- Effectiveness/efficiency of the CNN+Data Augmentation approach experimentally verified
- Today Deep Learning attacks systematically performed in Side-Channel tests for embedded cryptography evaluation
Classifying Side-Channel Desynchronized Signals with Convolutional Neural Networks

Conclusions about CNN

- CNNs provide an integrated approach to construct a discriminative model from misaligned data
- CNN models may have high capacity and require plenty of data to be trained
- Side-Channel-adapted Data Augmentation techniques
- Effectiveness/efficiency of the CNN+Data Augmentation approach experimentally verified
- Today Deep Learning attacks systematically performed in Side-Channel tests for embedded cryptography evaluation

Among new problematics...

Deep Learning provides black-box models:

- Lack of posterior knowledge: how did the model learn?
- Lack of trust: where did the model get the information?
- No hints to correct vulnerability
Conclusions about CNN

- CNNs provide an integrated approach to construct a discriminative model from misaligned data
- CNN models may have high capacity and require plenty of data to be trained
- Side-Channel-adapted Data Augmentation techniques
- Effectiveness/efficiency of the CNN+Data Augmentation approach experimentally verified
- Today Deep Learning attacks systematically performed in Side-Channel tests for embedded cryptography evaluation

Among new problematics...

Deep Learning provides black-box models:

Lack of posterior knowledge: how did the model learn?
Conclusions about CNN

- CNNs provide an integrated approach to construct a discriminative model from misaligned data
- CNN models may have high capacity and require plenty of data to be trained
- Side-Channel-adapted Data Augmentation techniques
- Effectiveness/efficiency of the CNN+Data Augmentation approach experimentally verified
- Today Deep Learning attacks systematically performed in Side-Channel tests for embedded cryptography evaluation

Among new problematics...

Deep Learning provides black-box models:

Lack of posterior knowledge: how did the model learn?
Lack of trust: where did the model get the information?
Conclusions about CNN

- CNNs provide an integrated approach to construct a discriminative model from misaligned data
- CNN models may have high capacity and require plenty of data to be trained
- Side-Channel-adapted Data Augmentation techniques
- Effectiveness/efficiency of the CNN+Data Augmentation approach experimentally verified
- Today Deep Learning attacks systematically performed in Side-Channel tests for embedded cryptography evaluation

Among new problematics...

Deep Learning provides black-box models:

Lack of posterior knowledge: how did the model learn?
Lack of trust: where did the model get the information?
No hints to correct vulnerability
Contents

1. Context and State of the Art

2. Deep Learning against Misalignment
   2.1 Neural Network Classifiers
   2.2 Data Augmentation
   2.3 Experimental Results

3. Gradient Visualization

4. Conclusions
Gradient Visualization

L. Masure et al., *Gradient Visualization for General Characterization in Profiling Attacks*, COSADE 2019 (Darmstadt, 5th April 2019)

- proposes a characterization technique based on a trained CNN
Gradient Visualization

L. Masure et al., Gradient Visualization for General Characterization in Profiling Attacks, COSADE 2019 (Darmstadt, 5th April 2019)

- proposes a characterization technique based on a trained CNN
- able to detect Points of Interest (PoIs) as long as the model has learned something
L. Masure et al., *Gradient Visualization for General Characterization in Profiling Attacks*, COSADE 2019 (Darmstadt, 5th April 2019)

- proposes a characterization technique based on a trained CNN
- able to detect Points of Interest (PoIs) as long as the model has learned something
- already proposed in Image Recognition [SVZ13; Spr+14]
Gradient Visualization

L. Masure *et al.*, *Gradient Visualization for General Characterization in Profiling Attacks*, COSADE 2019 (Darmstadt, 5th April 2019)

▶ proposes a characterization technique based on a trained CNN
▶ able to detect Points of Interest (PoIs) as long as the model has learned something
▶ already proposed in Image Recognition [SVZ13; Spr+14]
▶ starts to be used in SCA [Tim19; HGG19]
An ideal case

Ideal case: we know $F^* = \Pr[Z|X]$ (i.e. $F^*: \mathbb{R}^D \rightarrow \mathcal{P}(\mathbb{Z}) \subset [0, 1]^{|\mathbb{Z}|}$)

An example

An explanation

- Assume the informative leakage is very localized (few PoIs)
An ideal case

Ideal case: we know $F^* = \Pr[Z|X]$ (i.e. $F^* : \mathbb{R}^D \rightarrow \mathcal{P}(\mathbb{Z}) \subset [0, 1]^{|\mathbb{Z}|}$)

An example

An explanation

- Assume the informative leakage is very localized (few PoIs)
- Consider a new trace and its label $\vec{x}, z$
An ideal case

Ideal case: we know $F^* = \Pr[Z|X]$ (i.e. $F^* : \mathbb{R}^D \rightarrow \mathcal{P}(\mathcal{Z}) \subset [0, 1]|\mathcal{Z}|$)

An example

Assume the informative leakage is very localized (few PIs)

$t_0$ non informative:

$\tilde{x}[t_0] \rightarrow \tilde{x}[t_0] + \epsilon$ not sensitive

In other words, $t_0$ non informative

$\rightarrow \frac{\partial}{\partial \tilde{x}[t_0]} F^*(\tilde{x})[z] \approx 0$
An ideal case

Ideal case: we know $F^* = \Pr[Z|X]$ (i.e. $F^* : \mathbb{R}^D \rightarrow \mathcal{P}(\mathcal{Z}) \subset [0, 1]^{\mid \mathcal{Z} \mid}$)

An example

▶ Assume the informative leakage is very localized (few PoIs)
▶ $t_0$ non informative: $\tilde{x}[t_0] \rightarrow \tilde{x}[t_0] + \epsilon$ not sensitive
▶ In other words, $t_0$ non informative $\rightarrow \frac{\partial}{\partial \tilde{x}[t_0]} F^*(\tilde{x})[z] \approx 0$
An ideal case

Ideal case: we know $F^* = \Pr[Z|X]$ (i.e. $F^* : \mathbb{R}^D \rightarrow \mathcal{P}(\mathbb{Z}) \subset [0, 1]^{\left|\mathbb{Z}\right|}$)

An example

An explanation

- Assume the informative leakage is very localized (few PoIs)
- $t_1$ informative: $\vec{x}[t_1] \mapsto \vec{x}[t_1] + \epsilon$ is likely to affect the optimal model’s decision
- $t_1$ informative
  \[ \left| \frac{\partial}{\partial \vec{x}[t_1]} F^*(\vec{x})[z] \right| > 0 \]
An ideal case

Ideal case: we know $F^* = \Pr[Z|X]$ (i.e. $F^*: \mathbb{R}^D \rightarrow \mathcal{P}(\mathcal{Z}) \subset [0,1]^{|\mathcal{Z}|}$)

An example

An explanation

- Assume the informative leakage is very localized (few PoIs)
- $t_1$ informative: $\vec{x}[t_1] \rightarrow \vec{x}[t_1] + \epsilon$ is likely to affect the optimal model’s decision
- $t_1$ informative
  $\rightarrow \left| \frac{\partial}{\partial \vec{x}[t_1]} F^*(\vec{x})[z] \right| > 0$

Consequences

If $t$ is a PoI, then it should be seen in the gradients $\nabla_{\vec{x}} F^*(\vec{x})[z]$. 
An ideal case

Ideal case: we know $F^* = \Pr[Z|X]$ (i.e. $F^*: \mathbb{R}^D \rightarrow \mathcal{P}(Z) \subset [0,1]|Z|$)

An example

![Graph showing SNR and trace with informative leakage localized to a few points and scores]

An explanation

- Assume the informative leakage is very localized (few P0Is)
- $t_1$ informative: $\overline{x}[t_1] \mapsto \overline{x}[t_1] + \epsilon$ is likely to affect the optimal model’s decision
- $t_1$ informative

Consequences

If $t$ is a P0I, then it should be seen in the gradients $\nabla_{\overline{x}} F^*(\overline{x})[z]$
Application on experimental data

Description
ASCAD dataset [Pro+18]: https://github.com/ANSSI-FR/ASCAD
50,000 traces, each of 700 points
Source codes of secure implementations of AES128 for public 8-bit architectures (https://github.com/ANSSI-FR/secAES-ATmega8515)
Corresponds to the first AES round
Three cases studied:
1. No countermeasure: synchronized traces, no masking
2. Artificial random shift
3. Synchronized traces, boolean masking (unknown masks)

Trained model
CNN with a VGG-like architecture
Grid search of hyperparameters
Best model: minimal trace number when the guessing entropy reaches 2
First experiment: no countermeasure

Average number of traces to recover the secret key: 3

SNR for $Z = SBox(p[3] \oplus k[3]) \oplus r_{out}$

Synchronized traces

Gradient averaged on a 5-fold cross validation
No masking, no desynchronization

Figure: SNR

Figure: Gradient Visualization
Second experiment: with desynchronization

Average number of traces to recover the secret key: 3.6

Figure: No Pol emphasized 😊

Figure: Band of peaks 😊
Second experiment: with desynchronization

Average number of traces to recover the secret key: 3.6

Figure: No Pol emphasized 😊

Figure: Characterization for each trace 😊
Classifying Side-Channel Desynchronized Signals with Convolutional Neural Networks

Third experiment: with masking

Average number of traces to recover the secret key: $\approx 100$

**Signal-to-Noise Ratios**

ASCAD database

$\text{SNR} = \frac{r_{\text{out}}}{Z \oplus r_{\text{out}}}$

**Loss function gradient (average)**

With masking, no shift

**Figure**: Requires knowledge of the masks 😊

**Figure**: No knowledge required 😊
Be careful not to overfit!

**Figure:** GV without overfitting 😊

**Figure:** GV with overfitting 😏

**Figure:** Solution: early-stopping

Accuracy

Training

Validation

Epoch
Conclusions on Gradient Visualization

- Reinforces trust into Deep Learning tools: in absence of overfitting information comes from well-identifiable regions of interest
- May be used to guide early-stopping and prevent overfitting
- Provides characterization of leakages, allows developers to correct the vulnerability
Contents

1. Context and State of the Art

2. Deep Learning against Misalignment
   2.1 Neural Network Classifiers
   2.2 Data Augmentation
   2.3 Experimental Results

3. Gradient Visualization

4. Conclusions
Conclusions

- Curse of dimensionality affects the potential optimality of profiling attacks
- Machine Learning: profiling attacks ≈ classification task
- Generative model approach: Template Attacks
- Discriminative model approach:
  - Neural Networks, big data scalability
  - CNN to integrate resynchronization in a unique model construction process
Conclusions

- Curse of dimensionality affects the potential optimality of profiling attacks
- Machine Learning: profiling attacks ≈ classification task
- Generative model approach: Template Attacks
- Discriminative model approach:
  - Neural Networks, big data scalability
  - CNN to integrate resynchronization in a unique model construction process

Today and in the future

- Going towards a community "ML for Embedded Security Analysis"
Conclusions

- Curse of dimensionality affects the potential optimality of profiling attacks
- Machine Learning: profiling attacks ≈ classification task
- Generative model approach: Template Attacks
- Discriminative model approach:
  - Neural Networks, big data scalability
  - CNN to integrate resynchronization in a unique model construction process

Today and in the future

- Going towards a community "ML for Embedded Security Analysis"
  (A whole session at WRAC’H today!)
Conclusions

- Curse of dimensionality affects the potential optimality of profiling attacks
- Machine Learning: profiling attacks \(\approx\) classification task
- Generative model approach: Template Attacks
- Discriminative model approach:
  - Neural Networks, big data scalability
  - CNN to integrate resynchronization in a unique model construction process

Today and in the future

- Going towards a community "ML for Embedded Security Analysis"
  (A whole session at WRAC’H today!)
  (since ASCAD publication \(\sim\) 15 published papers in which ASCAD is used as benchmark)
Conclusions

- Curse of dimensionality affects the potential optimality of profiling attacks
- Machine Learning: profiling attacks $\approx$ classification task
- Generative model approach: Template Attacks
- Discriminative model approach:
  - Neural Networks, big data scalability
  - CNN to integrate resynchronization in a unique model construction process

Today and in the future

- Going towards a community "ML for Embedded Security Analysis"
  (A whole session at WRAC’H today!)
  (since ASCAD publication $\sim$ 15 published papers in which ASCAD is used as benchmark)
- Beyond Classification
Conclusions

▶ Curse of dimensionality affects the potential optimality of profiling attacks
▶ Machine Learning: profiling attacks $\approx$ classification task
▶ Generative model approach: Template Attacks
▶ Discriminative model approach:
  ▶ Neural Networks, big data scalability
  ▶ CNN to integrate resynchronization in a unique model construction process

Today and in the future

▶ Going towards a community "ML for Embedded Security Analysis"
  (A whole session at WRAC’H today!)
  (since ASCAD publication $\sim$ 15 published papers in which ASCAD is used as benchmark)
▶ Beyond Classification
  ▶ Collision attacks $\approx$ verification task (siamese network)
Conclusions

- Curse of dimensionality affects the potential optimality of profiling attacks
- Machine Learning: profiling attacks ≈ classification task
- Generative model approach: Template Attacks
- Discriminative model approach:
  - Neural Networks, big data scalability
  - CNN to integrate resynchronization in a unique model construction process

Today and in the future

- Going towards a community "ML for Embedded Security Analysis"
  (A whole session at WRAC’H today!)
  (since ASCAD publication ∼ 15 published papers in which ASCAD is used as benchmark)
- Beyond Classification
  - Collision attacks ≈ verification task (siamese network)
  - Does "accuracy" matter? Need for specifying a proper "Advanced-attack-oriented machine learning task" (SCA-specific loss functions and metrics)
Thank You!

- Eleonora Cagli, Cécile Dumas, Emmanuel Prouff: *Convolutional Neural Networks with Data Augmentation against Jitter-Based Countermeasures - Profiling Attacks without Pre-Processing*. IACR Cryptology ePrint Archive 2017: 740 (2017) - CHES 2017:45-68


- Loïc Masure, Cécile Dumas, Emmanuel Prouff: *Gradient Visualization for General Characterization in Profiling Attacks* (COSADE 2019)
References


References


References


References IV


References V


References VI


References VII


References VIII


