Convolutional Neural Networks with Data Augmentation against Jitter-Based Countermeasures

Eleonora Cagli[1,3] Cécile Dumas[1]
Emmanuel Prouff[2,3]

[1]Univ. Grenoble Alpes, F-38000, Grenoble, France
CEA, LETI, MINATEC Campus, F-38054 Grenoble, France
{eleonora.cagli,cecile.dumas}@cea.fr

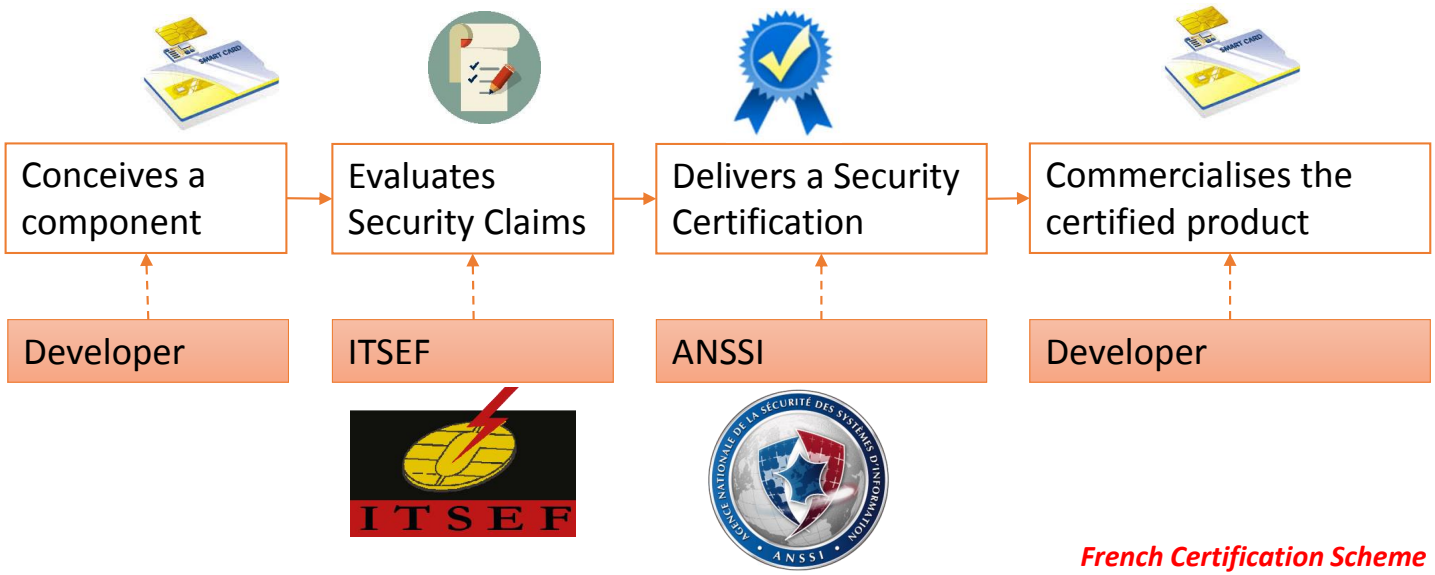
[2]Safran Identity and Security, France
emmanuel.prouff@safrangroup.com

[3]Sorbonne Universités, UPMC Univ. Paris 06, CNRS, INRIA, Laboratoire d'Informatique de Paris 6 (LIP6),
Équipe PolSys, 4 place Jussieu, 75252 Paris Cedex 05, France

26/09/2017, CHES 2017, Taipei, Taiwan

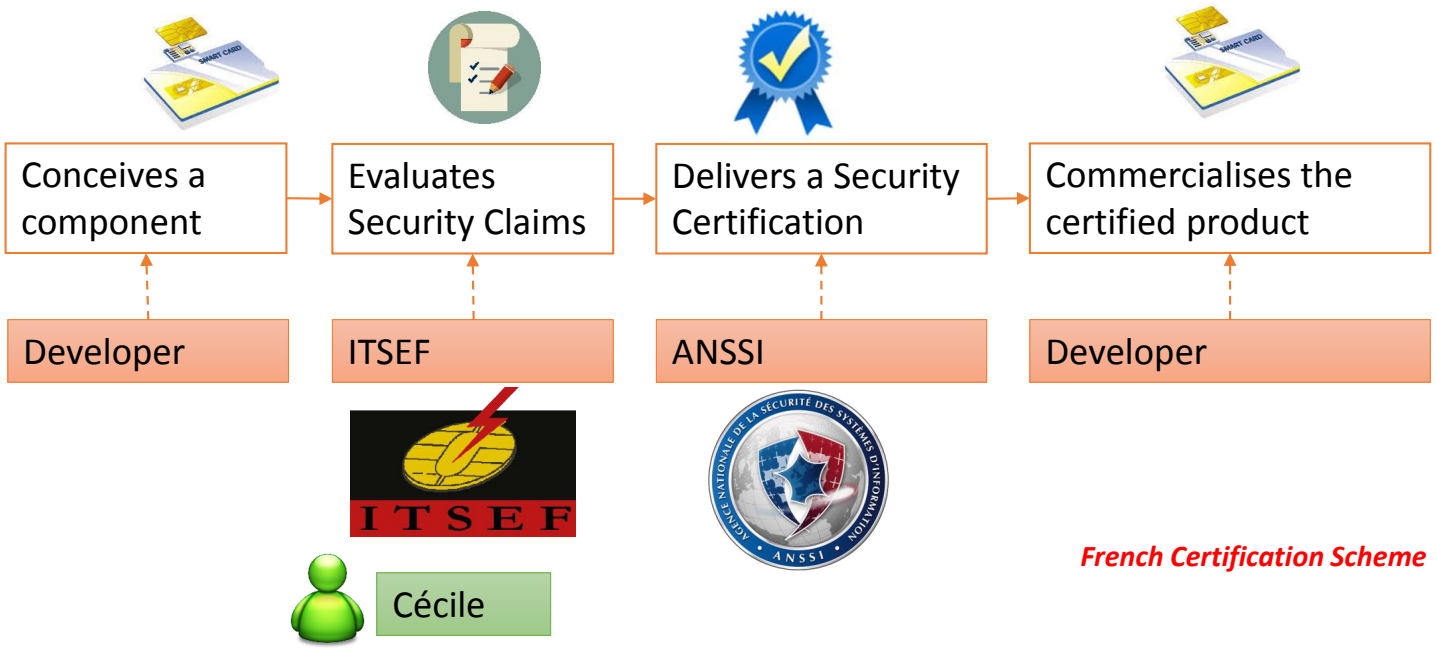
About the authors

Who we are



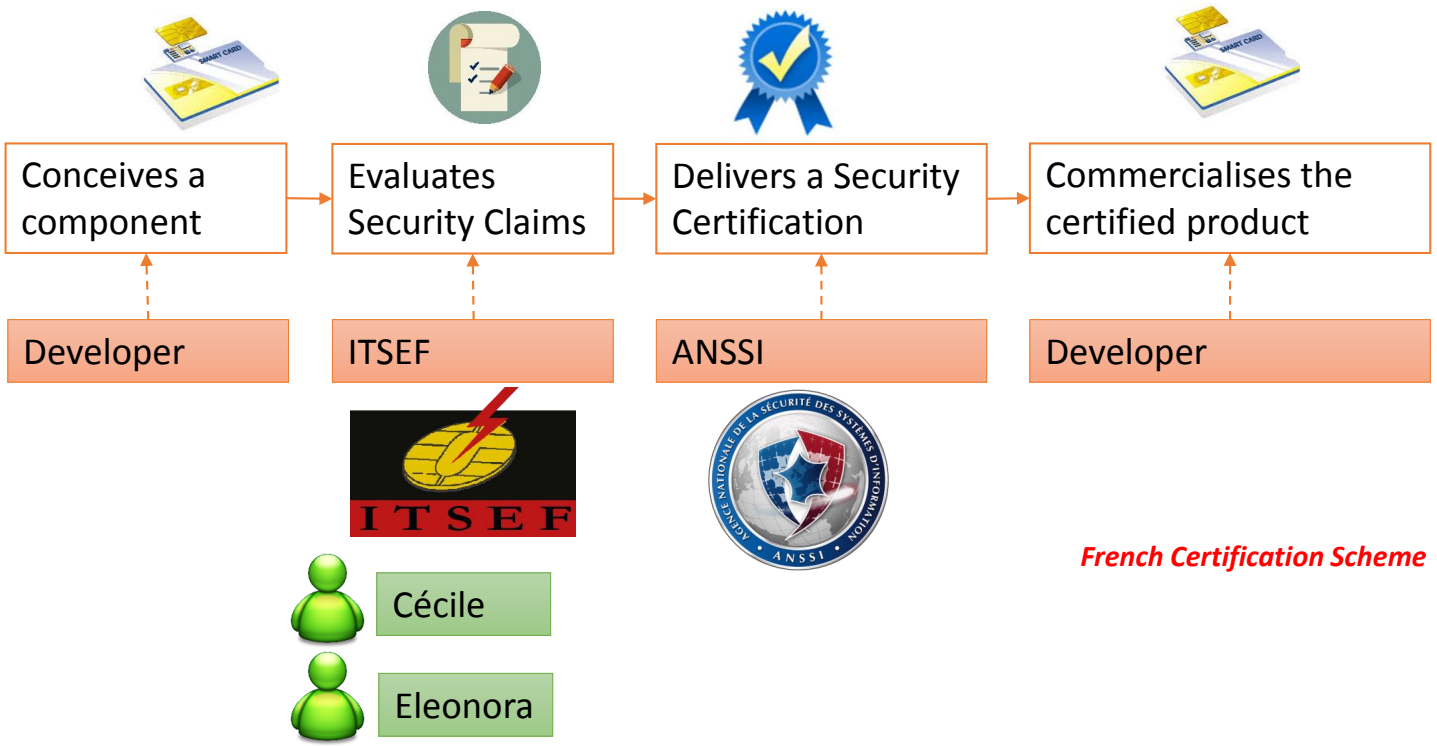
About the authors

Who we are



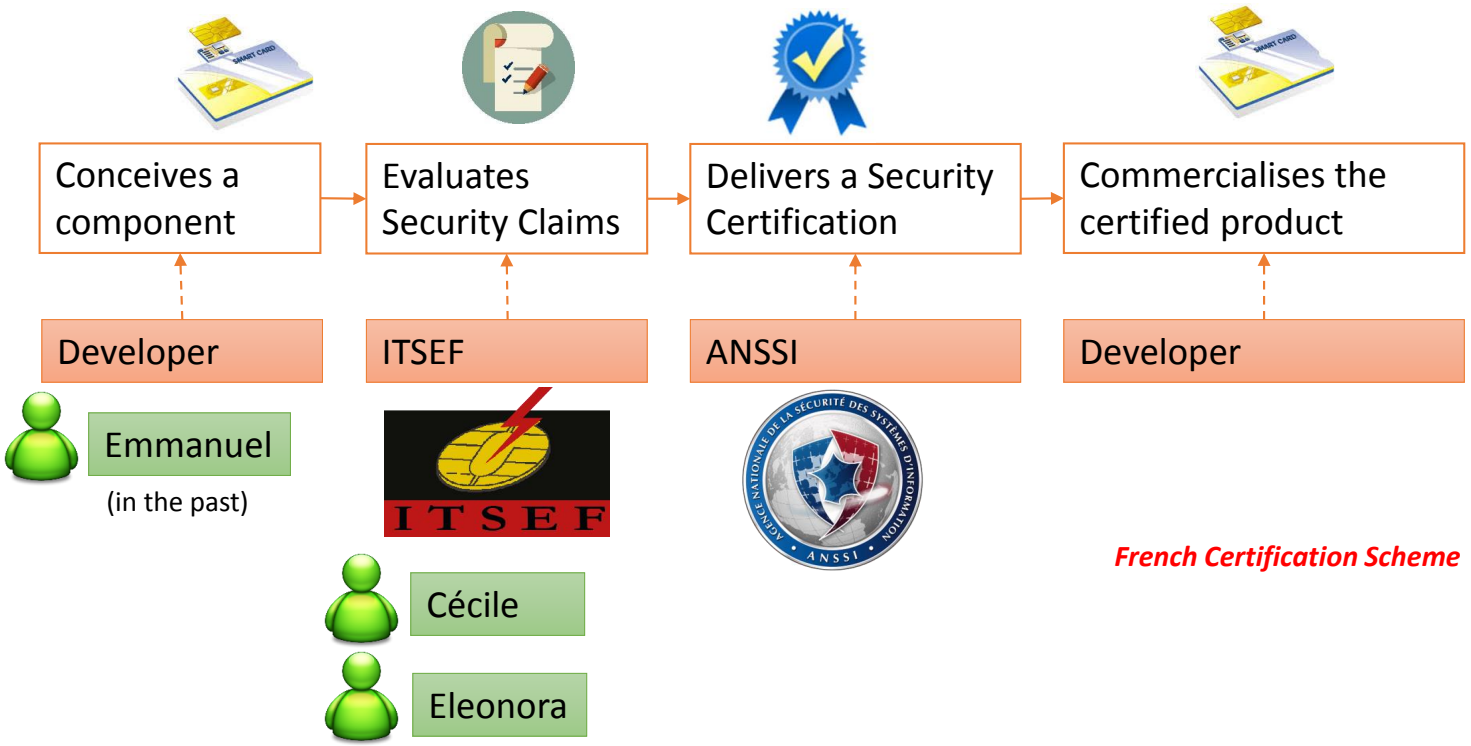
About the authors

Who we are



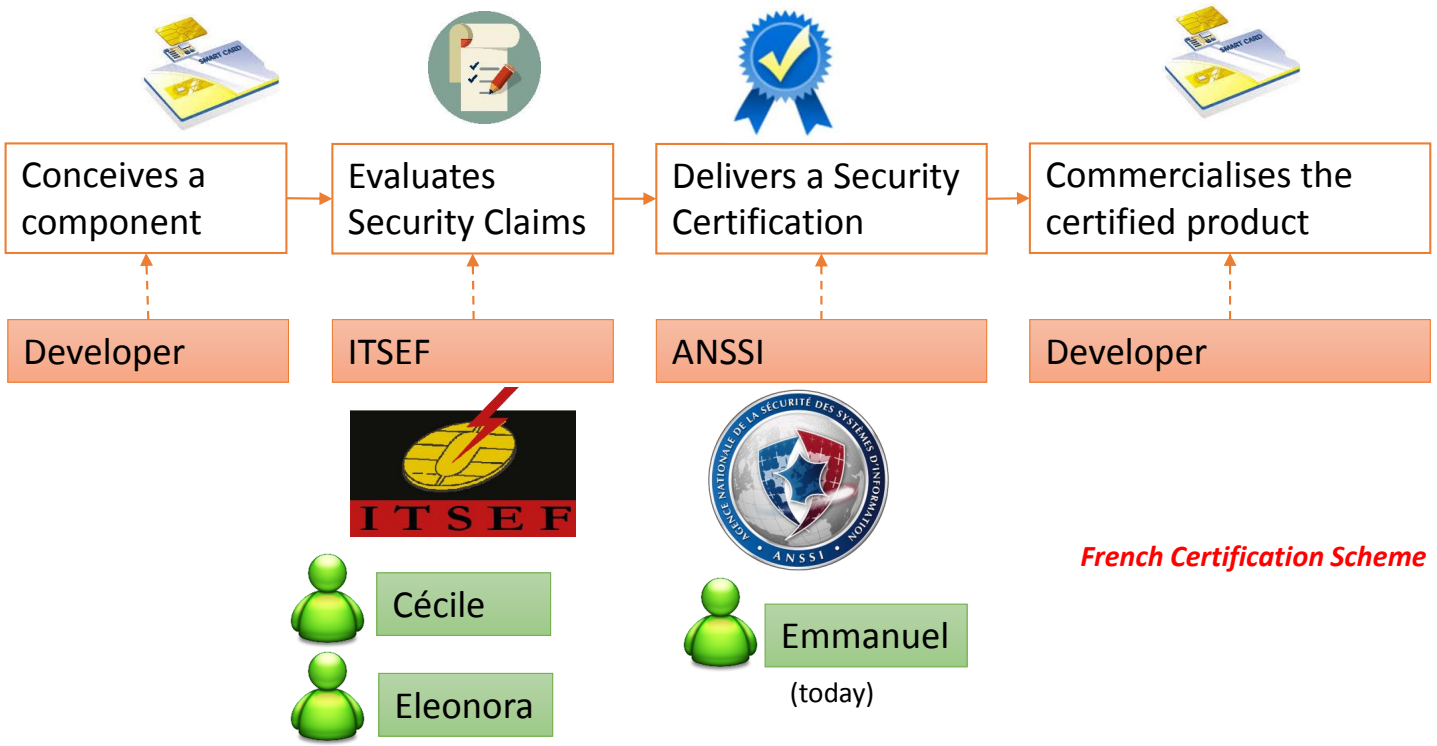
About the authors

Who we are



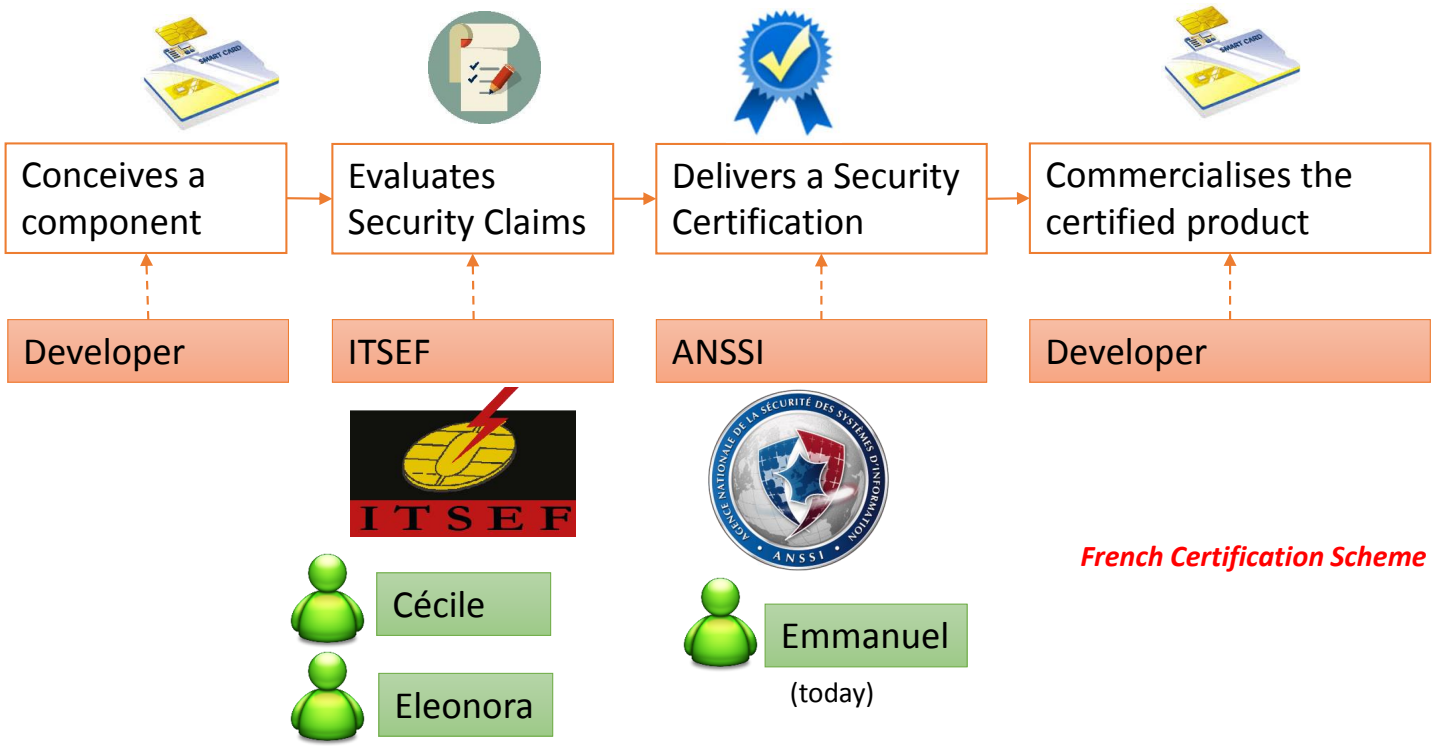
About the authors

Who we are



About the authors

Who we are



An evaluation point of view

- profiling attacks (worst-case security)
- practical aspects are concerned

Contents

Side Channel Attacks

Notations

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

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

Side Channel Attacks

Notations

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

Goal: make inference over **Z**, observing **X** $\Pr[\mathbf{Z}|\mathbf{X}]$

Side Channel Attacks

Notations

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

Goal: make inference over **Z**, observing **X** $\Pr[\mathbf{Z}|\mathbf{X}]$

Template Attacks

- Profiling phase (using profiling traces under known **Z**)
- Attack phase (N attack traces, e.g. with known plaintexts p_i)

Side Channel Attacks

Notations

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

Goal: make inference over **Z**, observing **X** $\Pr[\mathbf{Z}|\mathbf{X}]$

Template Attacks

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

Side Channel Attacks

Notations

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

Goal: make inference over $\mathbf{Z}$, observing $\mathbf{X}$ $\Pr[\mathbf{Z}|\mathbf{X}]$

Template Attacks

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

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

Side Channel Attacks

Notations

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

Goal: make inference over $\mathbf{Z}$, observing $\mathbf{X}$ $\Pr[\mathbf{Z}|\mathbf{X}]$

Template Attacks

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

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

Side Channel Attacks

Notations

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

Goal: make inference over $\mathbf{Z}$, observing $\mathbf{X}$ $\Pr[\mathbf{Z}|\mathbf{X}]$

Template Attacks

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

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

Misalignment

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

Misalignment

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

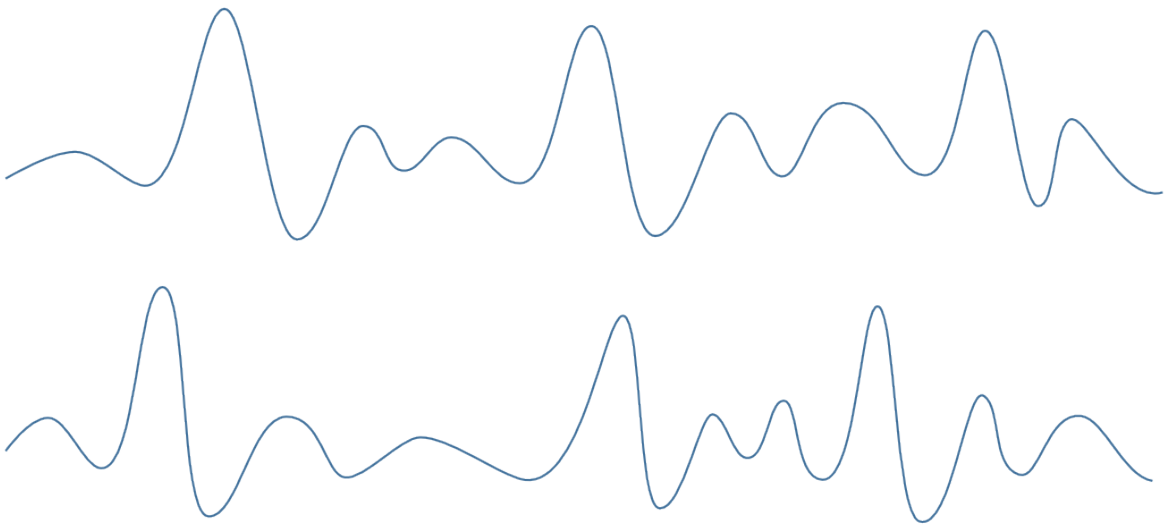
- not a wide literature
- in practice: evaluation labs home-made realignment techniques

Risks of realignment

An example

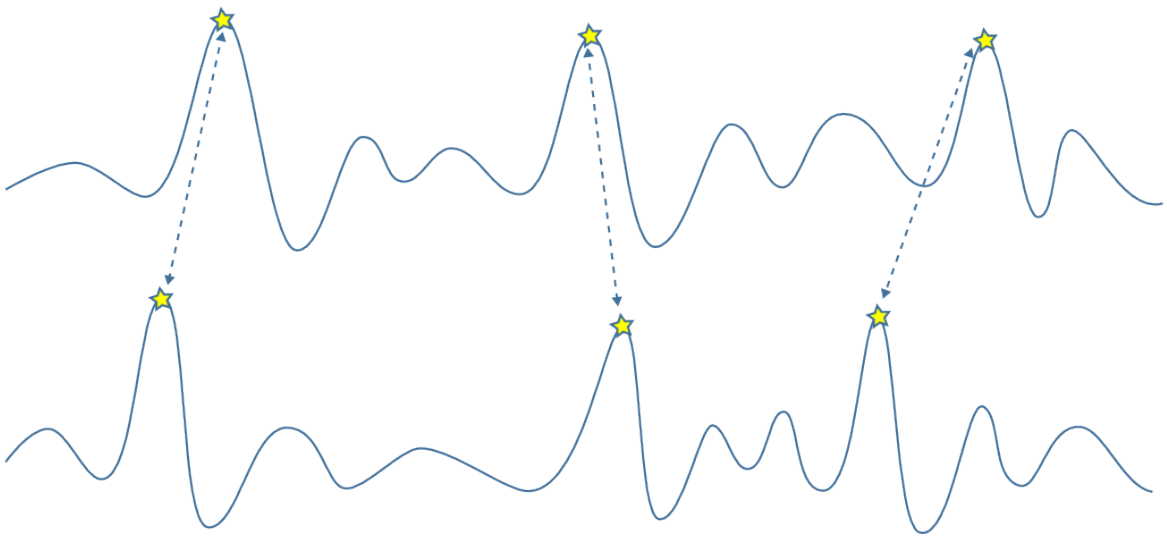
Risks of realignment

An example



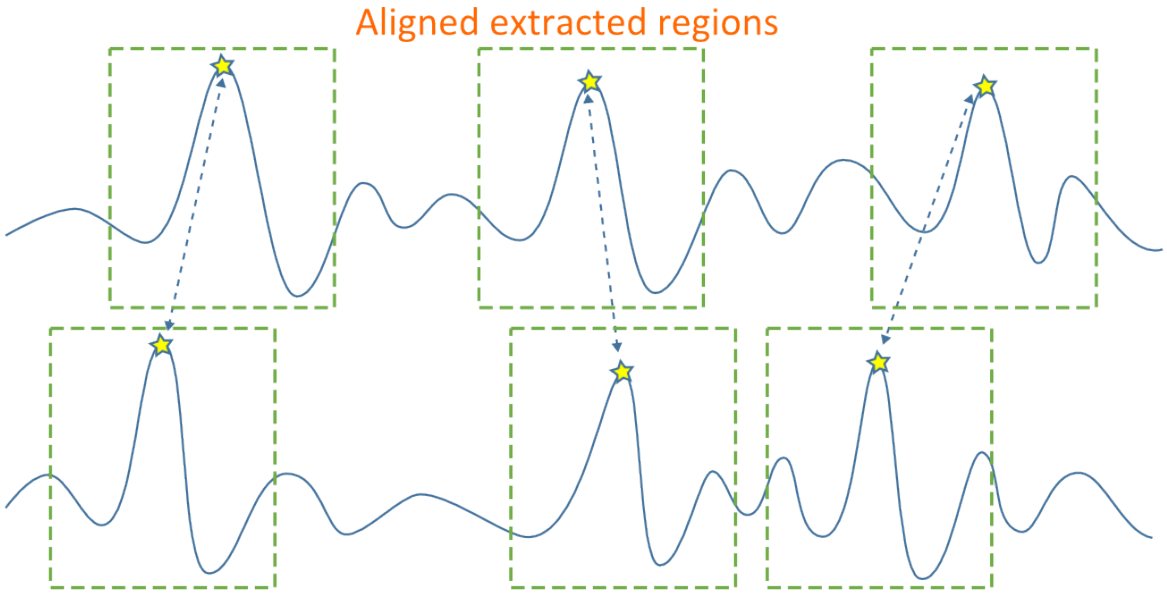
Risks of realignment

An example



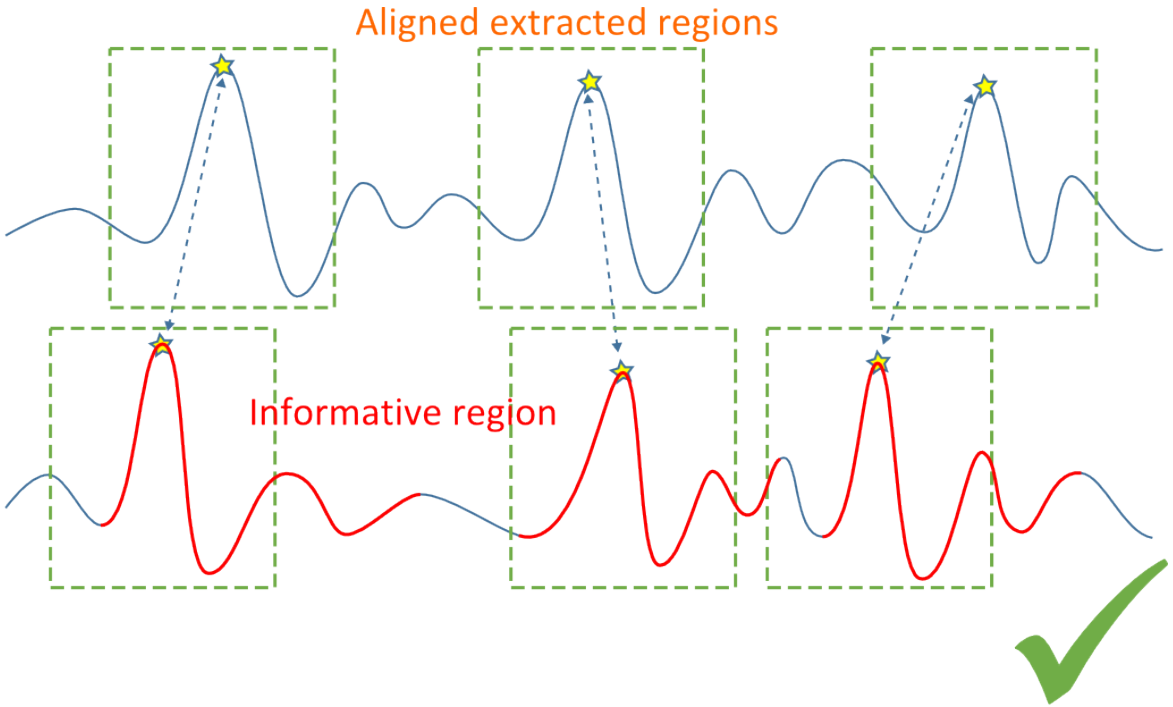
Risks of realignment

An example



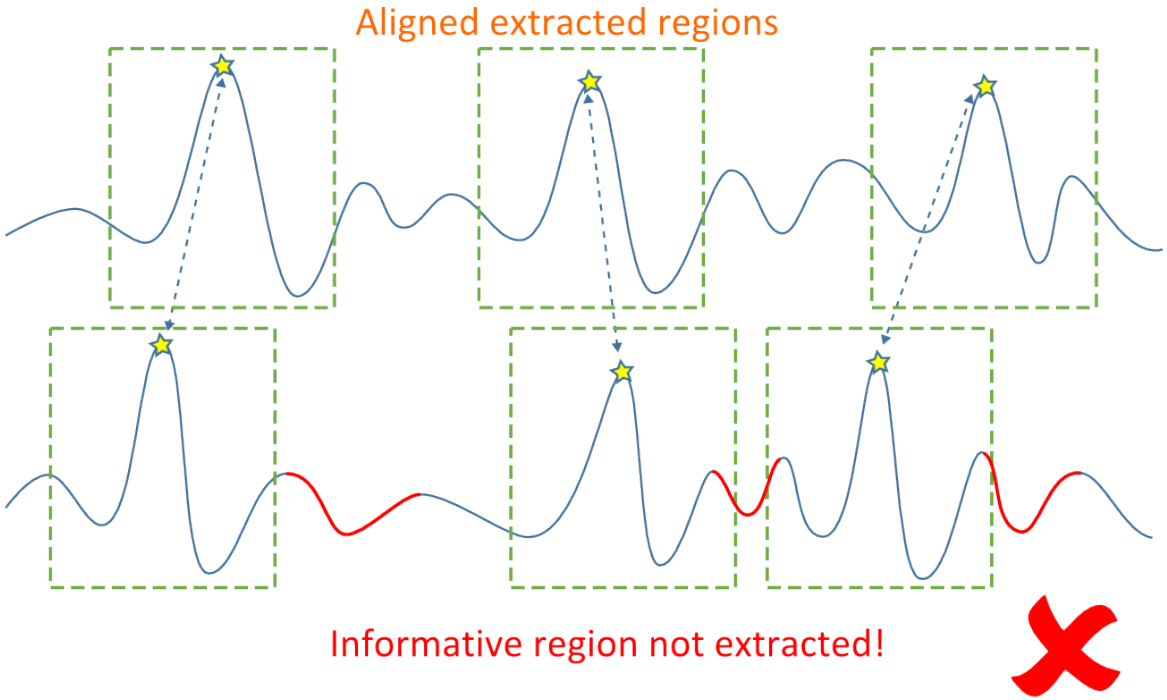
Risks of realignment

An example



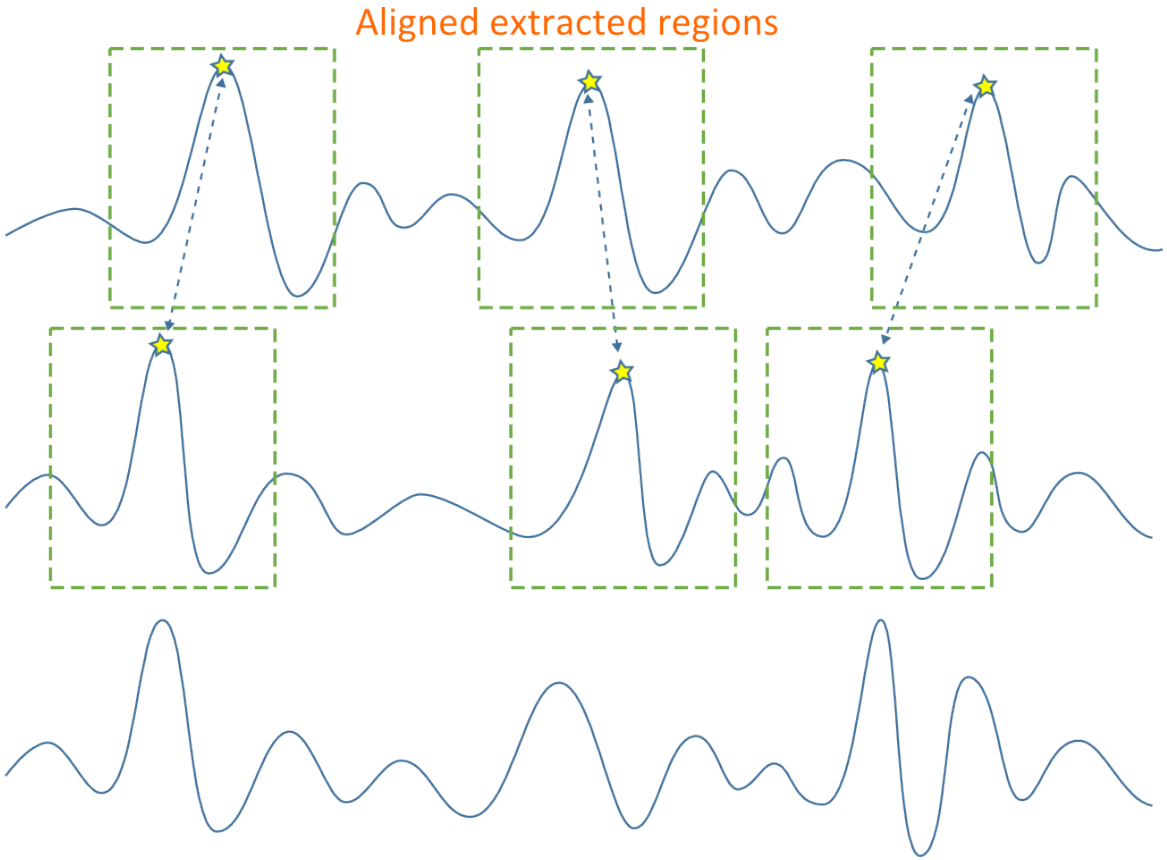
Risks of realignment

An example



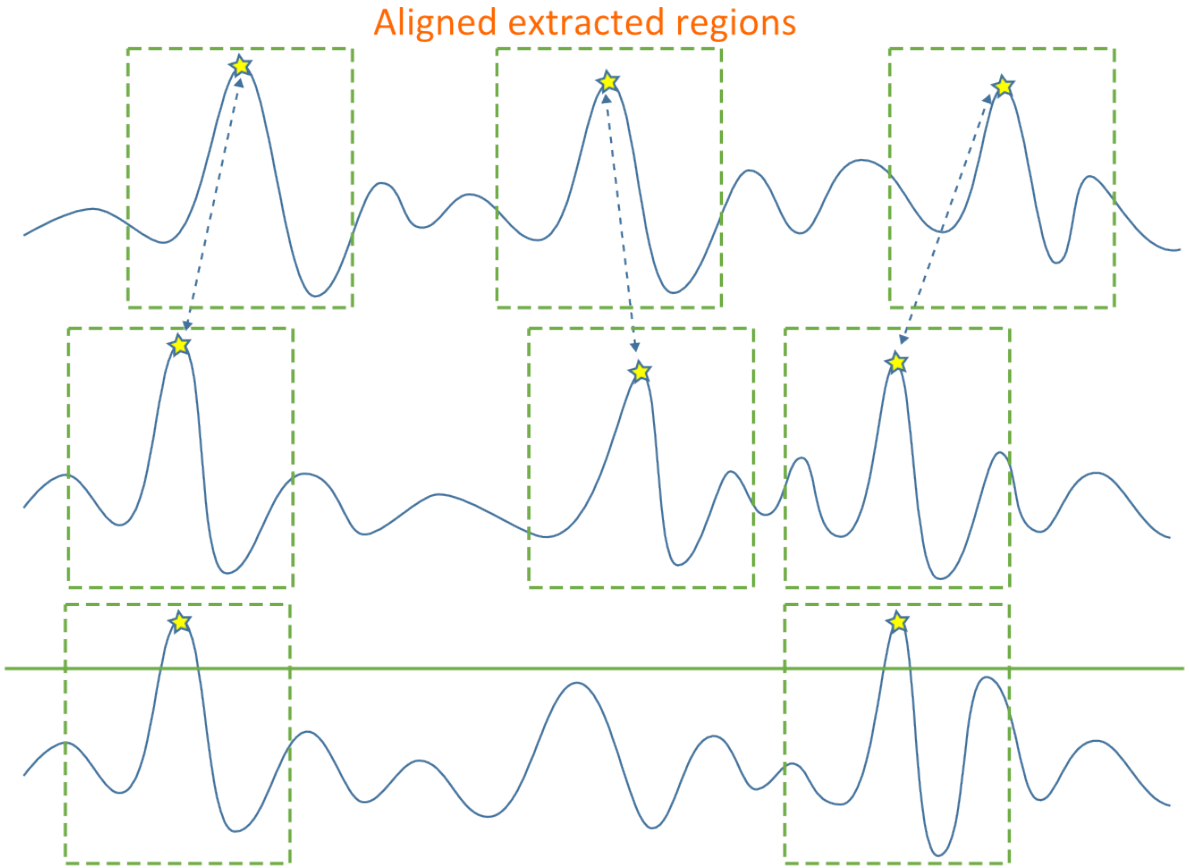
Risks of realignment

An example



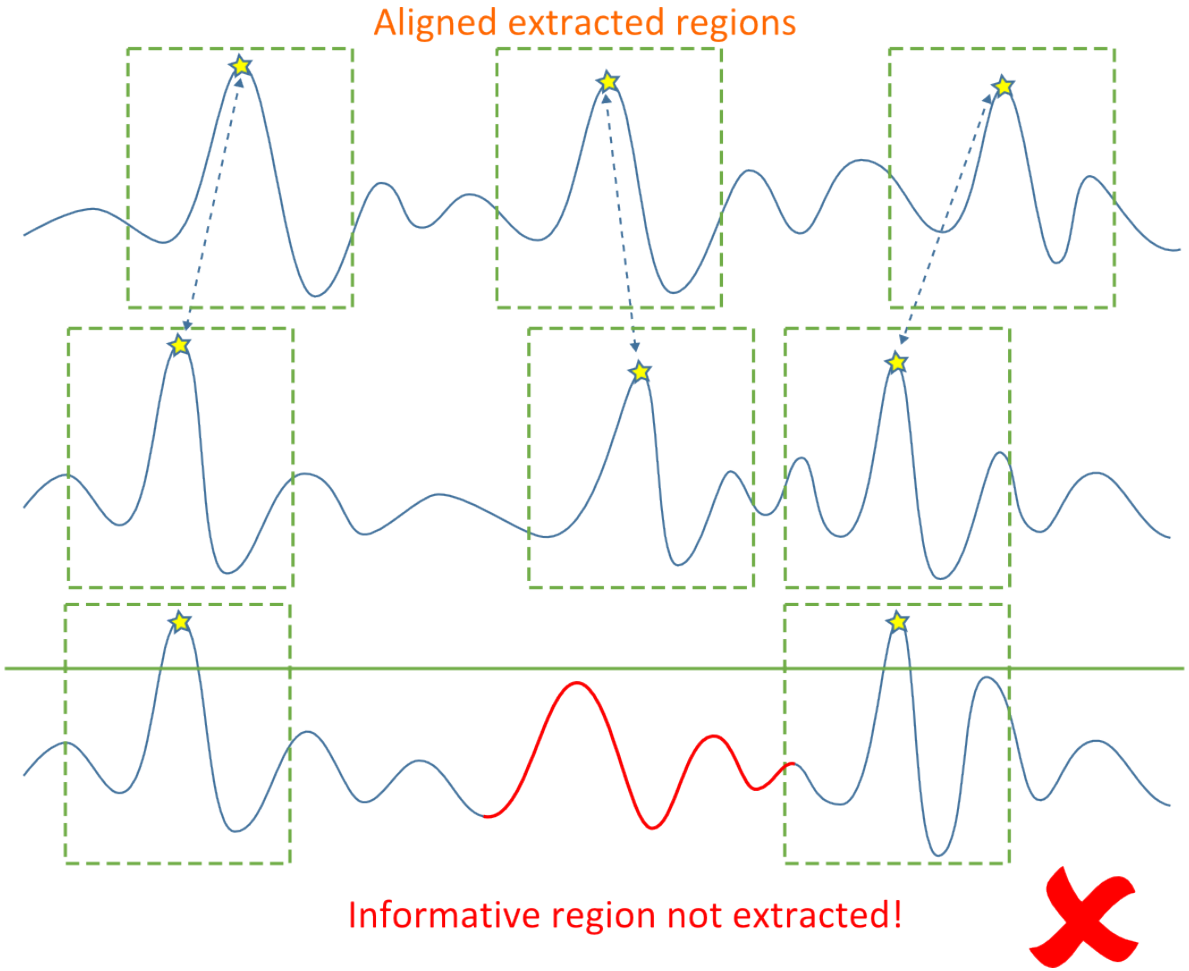
Risks of realignment

An example



Risks of realignment

An example



Motivating Conclusions

An evaluator can spend time to adjust realignment...

- try realignments based over other kind of patterns
- modify parameters...

Motivating Conclusions

An evaluator can spend time to adjust realignment...

- try realignments based over other kind of patterns
- modify parameters...

...but

- no prior knowledge about informative patterns
- no way to evaluate realignment without launching the afterwards attack

Motivating Conclusions

An evaluator can spend time to adjust realignment...

- try realignments based over other kind of patterns
- modify parameters...

...but

- no prior knowledge about informative patterns
- no way to evaluate realignment without launching the afterwards attack

Now:

- preprocessing to prepare data
- characterization to extract information

Our paper perspective:

- ~~preprocessing to prepare data~~
- direct information extraction

Motivating Conclusions

An evaluator can spend time to adjust realignment...

- try realignments based over other kind of patterns
- modify parameters...

...but

- no prior knowledge about informative patterns
- no way to evaluate realignment without launching the afterwards attack

Now:

- preprocessing to prepare data
- characterization to extract information

Our paper perspective:

- ~~preprocessing to prepare data~~
- direct information extraction

No preprocessing ⇒ No risks of information loss

Side Channel Attacks

Notations

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

Goal: make inference over $\mathbf{Z}$, observing $\mathbf{X}$ $\Pr[\mathbf{Z}|\mathbf{X}]$

Template Attacks

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

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

Side Channel Attacks with a Classifier

Notations

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

Goal: make inference over $\mathbf{Z}$, observing $\mathbf{X}$ $\Pr[\mathbf{Z}|\mathbf{X}]$

Template Attacks

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

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

Side Channel Attacks with a Classifier

Notations

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

Goal: make inference over $\mathbf{Z}$, observing $\mathbf{X}$ $\Pr[\mathbf{Z}|\mathbf{X}]$

Template Attacks

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

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

Side Channel Attacks with a Classifier

Notations

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

Goal: make inference over $\mathbf{Z}$, observing $\mathbf{X}$ $\Pr[\mathbf{Z}|\mathbf{X}]$

Template Attacks

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

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

Side Channel Attacks with a Classifier

Notations

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

Goal: make inference over $\mathbf{Z}$, observing $\mathbf{X}$ $\Pr[\mathbf{Z}|\mathbf{X}]$

Template Attacks

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

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

Side Channel Attacks with a Classifier

Notations

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

Goal: make inference over $\mathbf{Z}$, observing $\mathbf{X}$ $\Pr[\mathbf{Z}|\mathbf{X}]$

Template Attacks

- Training phase (using training traces under known $\mathbf{Z}$)
  - manage de-synchronization problem
  - mandatory dimensionality reduction
  - construct a classifier $F(\mathbf{x}) = \mathbf{y} \approx \Pr[\mathbf{Z}|\mathbf{X} = \mathbf{x}]$

  } Integrated approach
- Attack phase (N attack traces, e.g. with known plaintexts p_i)
  - Log-likelihood score for each key hypothesis k

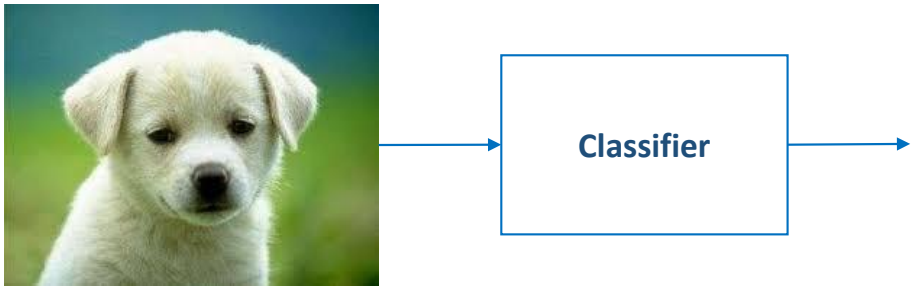
$$d_k = \sum_{i=1}^{N} \log F(\mathbf{x}_i)[f(p_i, k)]$$

Contents

Classification

Classification problem

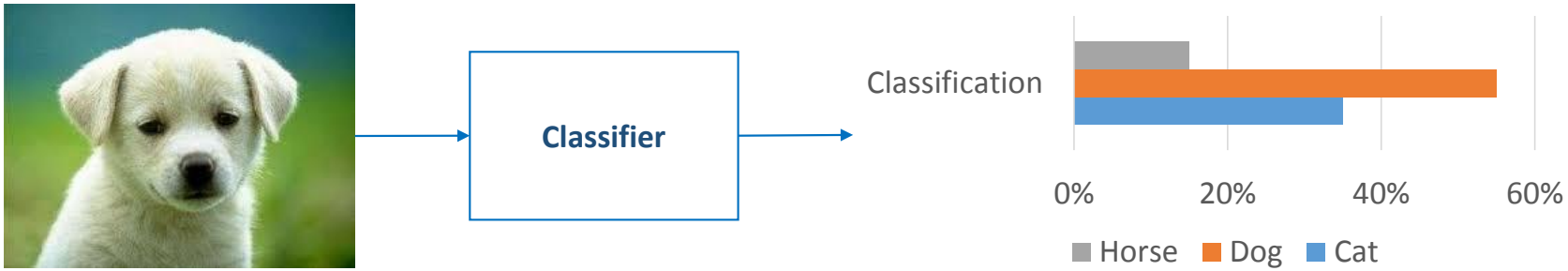
Assign to a datum **X** (e.g. an image) a label **Z** among a set of possible labels $\mathcal{Z} = \{\mathrm{Cat}, \mathrm{Dog}, \mathrm{Horse}\}$



Classification

Classification problem

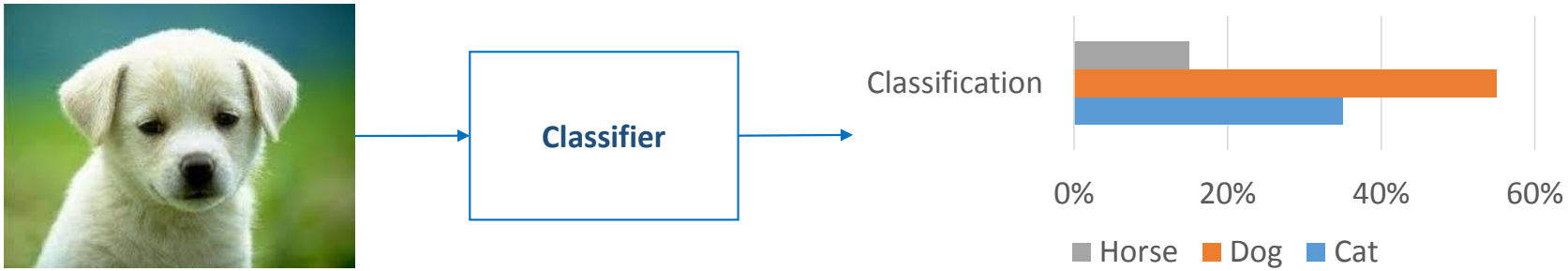
Assign to a datum **X** (e.g. an image) a label **Z** among a set of possible labels $\mathcal{Z} = \{\mathrm{Cat}, \mathrm{Dog}, \mathrm{Horse}\}$



Classification

Classification problem

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

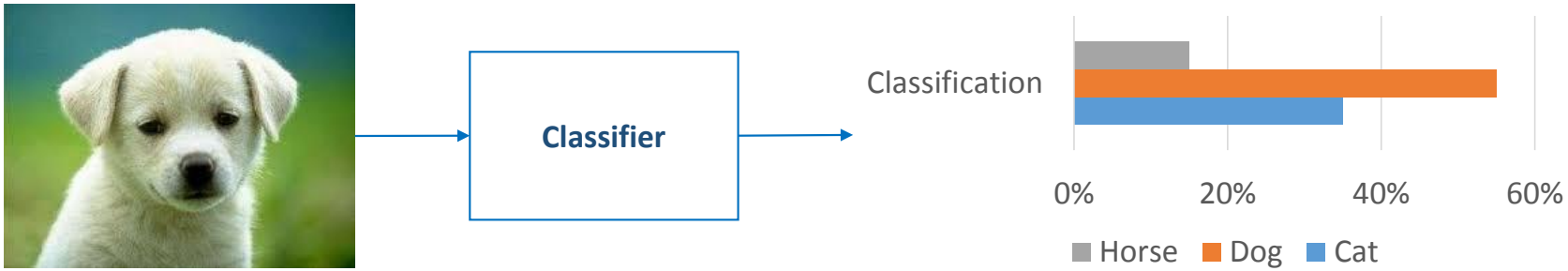


$\Pr[\mathbf{Z}|\mathbf{X}]$

Classification

Classification problem

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



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

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

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

Multi-Layer Perceptron (MLP)

$$F(\mathbf{x}, W) = \mathrm{s} \circ \lambda_n \circ \sigma_{n-1} \circ \lambda_{n-1} \circ \cdots \circ \lambda_1(\mathbf{x}) = \mathbf{y} \approx \Pr[\mathbf{Z}|\mathbf{X} = \mathbf{x}]$$

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

Multi-Layer Perceptron (MLP)

$$F(\mathbf{x}, W) = \mathrm{s} \circ \lambda_n \circ \sigma_{n-1} \circ \lambda_{n-1} \circ \cdots \circ \lambda_1(\mathbf{x}) = \mathbf{y} \approx \Pr[\mathbf{Z}|\mathbf{X} = \mathbf{x}]$$

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

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

Multi-Layer Perceptron (MLP)

$$F(\mathbf{x}, W) = \mathrm{s} \circ \lambda_n \circ \sigma_{n-1} \circ \lambda_{n-1} \circ \cdots \circ \lambda_1(\mathbf{x}) = \mathbf{y} \approx \Pr[\mathbf{Z}|\mathbf{X} = \mathbf{x}]$$

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

σ_i non-linear *activation* functions

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

Multi-Layer Perceptron (MLP)

$$F(\mathbf{x}, W) = \mathrm{s} \circ \lambda_n \circ \sigma_{n-1} \circ \lambda_{n-1} \circ \cdots \circ \lambda_1(\mathbf{x}) = \mathbf{y} \approx \Pr[\mathbf{Z}|\mathbf{X} = \mathbf{x}]$$

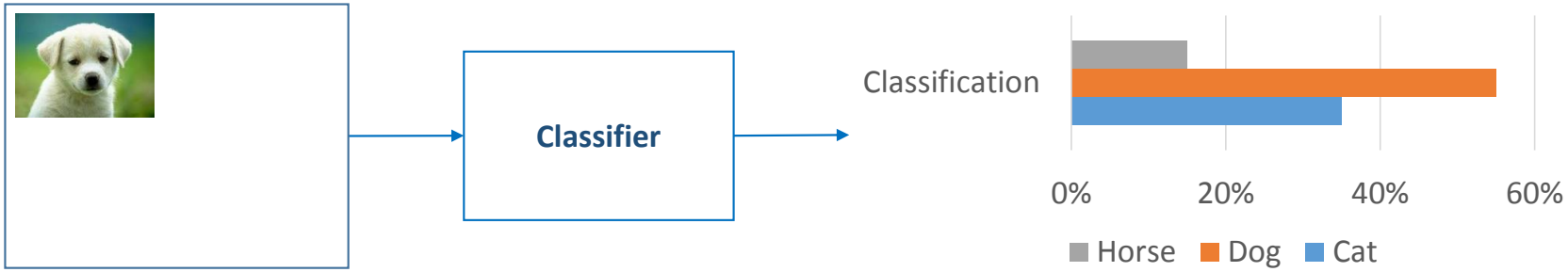
λ_i linear functions (linear combinations of time samples) depending on some **trainable weights** W

σ_i non-linear *activation* functions

s normalizing *softmax* function

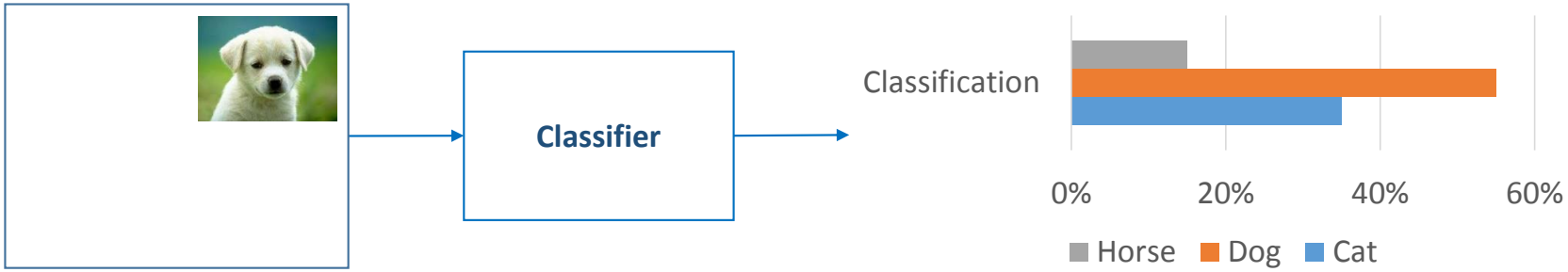
Convolutional Neural Networks

An answer to translation-invariance



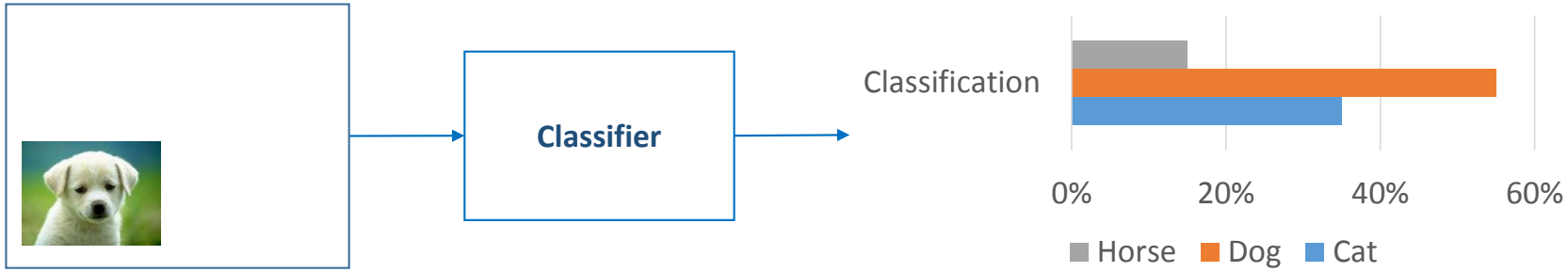
Convolutional Neural Networks

An answer to translation-invariance



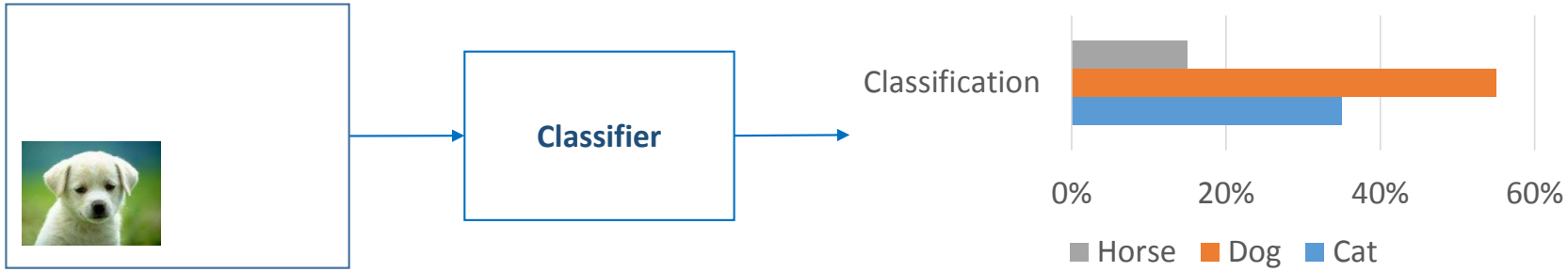
Convolutional Neural Networks

An answer to translation-invariance



Convolutional Neural Networks

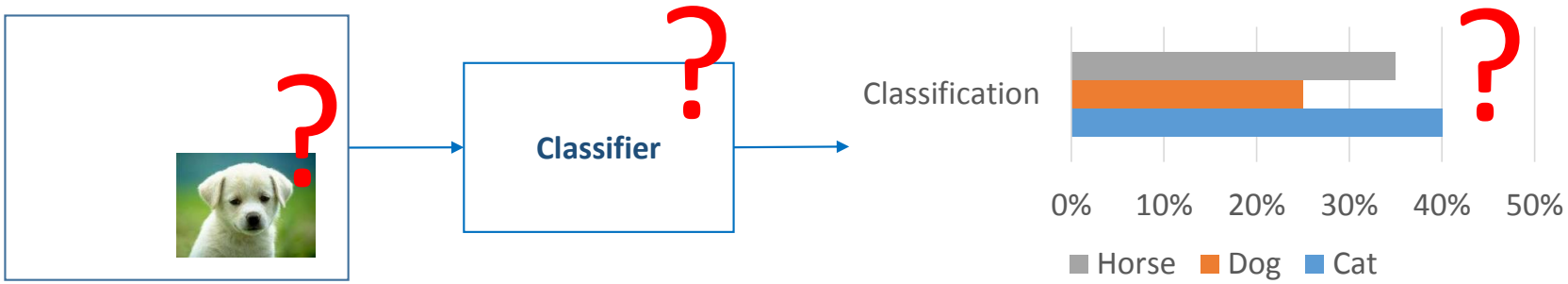
An answer to translation-invariance



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

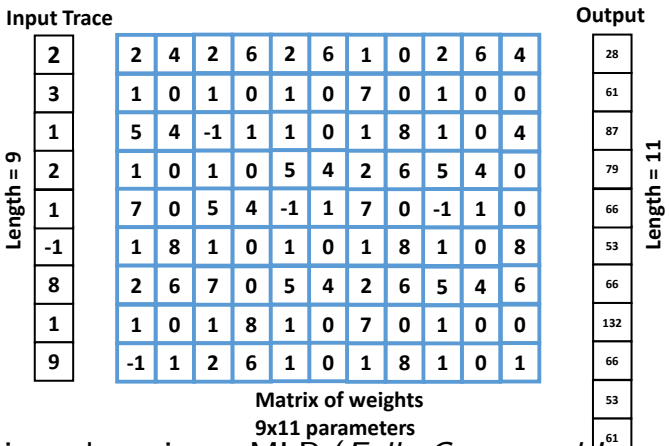
An answer to translation-invariance

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



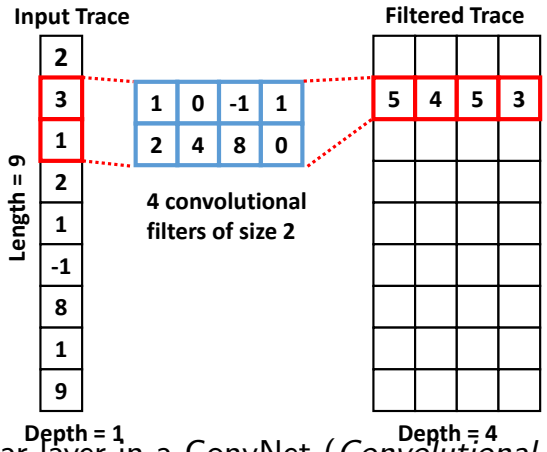
Linear layer in an MLP (*Fully Connected Layer*)

Convolutional Neural Networks

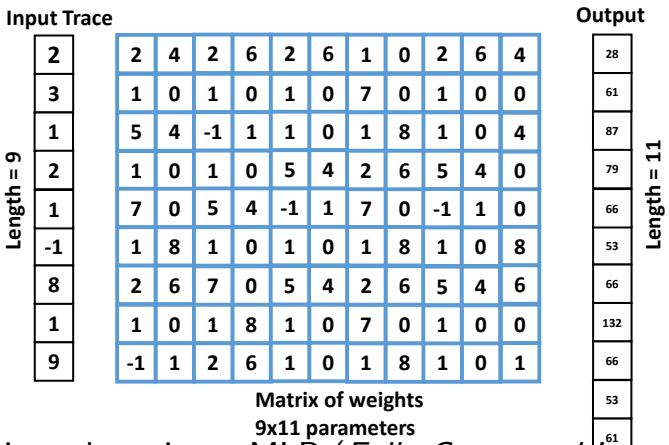
An answer to translation-invariance

It is important to explicit the data translation-invariance

Convolutional Neural Networks: share weights across space



Linear layer in a ConvNet (*Convolutional Layer*)



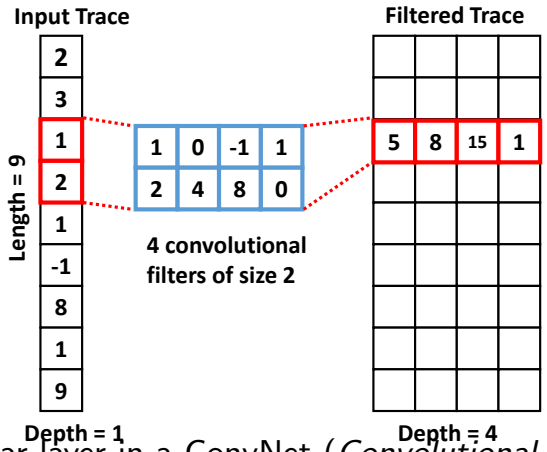
Linear layer in an MLP (*Fully Connected Layer*)

Convolutional Neural Networks

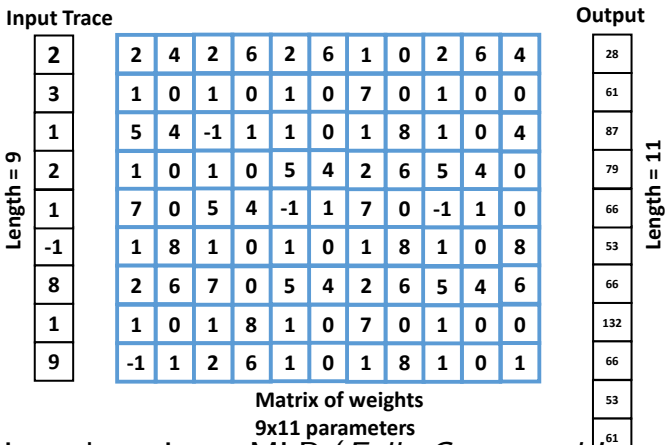
An answer to translation-invariance

It is important to explicit the data translation-invariance

Convolutional Neural Networks: share weights across space



Linear layer in a ConvNet (*Convolutional Layer*)



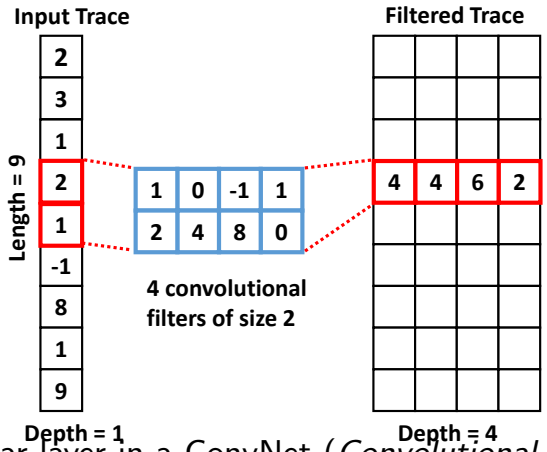
Linear layer in an MLP (*Fully Connected Layer*)

Convolutional Neural Networks

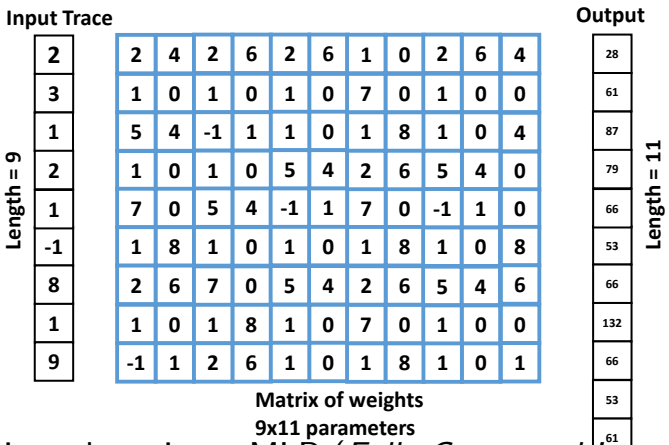
An answer to translation-invariance

It is important to explicit the data translation-invariance

Convolutional Neural Networks: share weights across space



Linear layer in a ConvNet (*Convolutional Layer*)



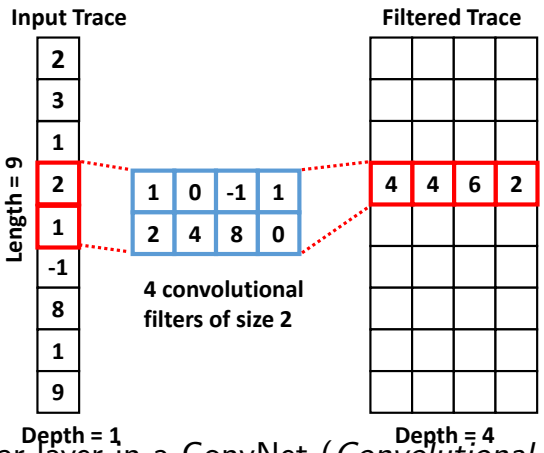
Linear layer in an MLP (*Fully Connected Layer*)

Convolutional Neural Networks

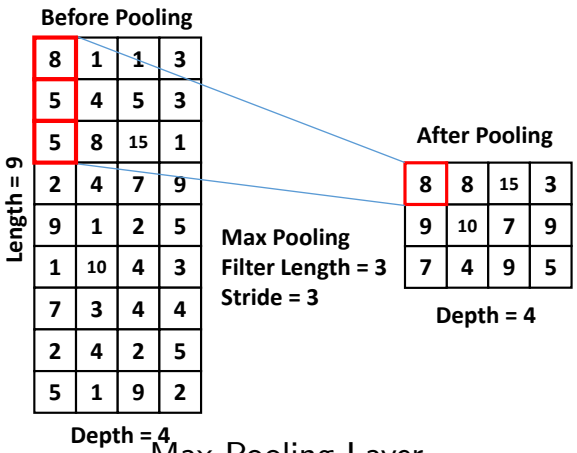
An answer to translation-invariance

It is important to explicit the data translation-invariance

Convolutional Neural Networks: share weights across space

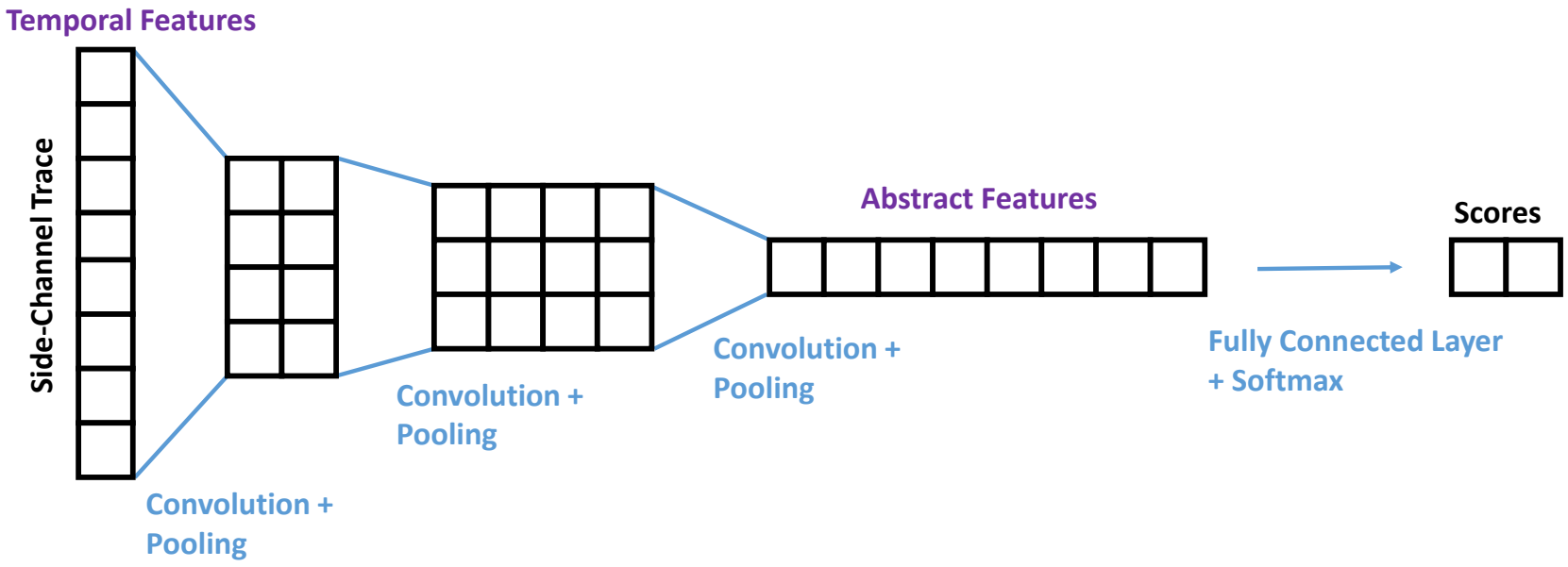


Linear layer in a ConvNet (*Convolutional Layer*)



Max Pooling Layer

ConvNet typical architecture



Training and overfitting

Training

Profiling set $\rightarrow \begin{cases} \text{Training set} \\ \text{Validation set} \end{cases}$

Randomly partition training set into batches

Iterative algorithm over batches (one weights update per processed batch)

Epoch:= one pass over the entire training set

Training and overfitting

Training

$$\text{Profiling set} \rightarrow \begin{cases} \text{Training set} \\ \text{Validation set} \end{cases}$$

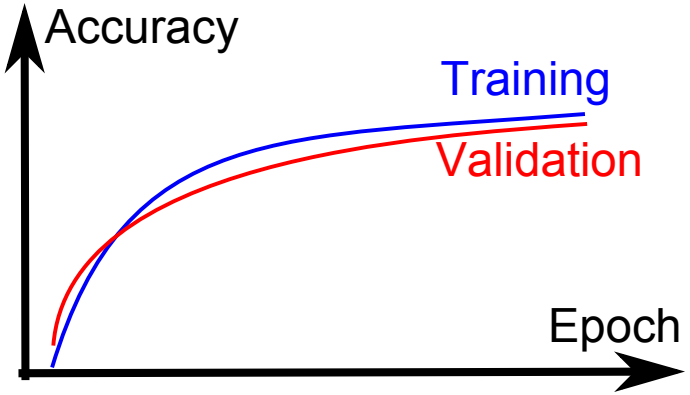
Randomly partition training set into batches

Iterative algorithm over batches (one weights update per processed batch)

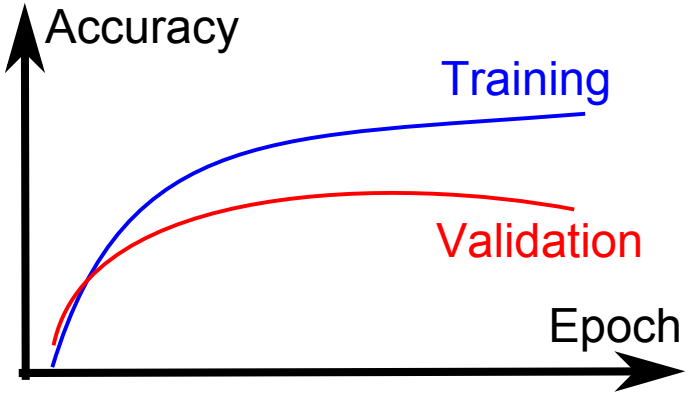
Epoch:= one pass over the entire training set

Evaluate and compare training and validation accuracy

Understand significant features



Learn by heart (**OVERFITTING**)



Training and overfitting

Training

Profiling set $\rightarrow \begin{cases} \text{Training set} \\ \text{Validation set} \end{cases}$

Randomly partition training set into batches

Iterative algorithm over batches (one weights update per processed batch)

Epoch:= one pass over the entire training set

Evaluate and compare training and validation accuracy

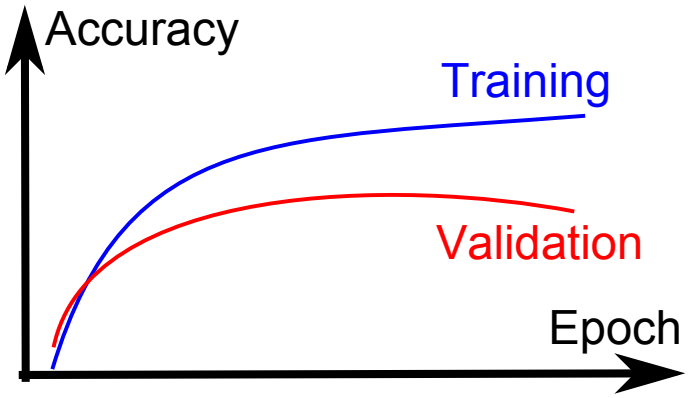
Why?

- Too complex model
- Not enough training data

Solution?

- Data augmentation

Learn by heart (**OVERFITTING**)



Contents

Data Augmentation

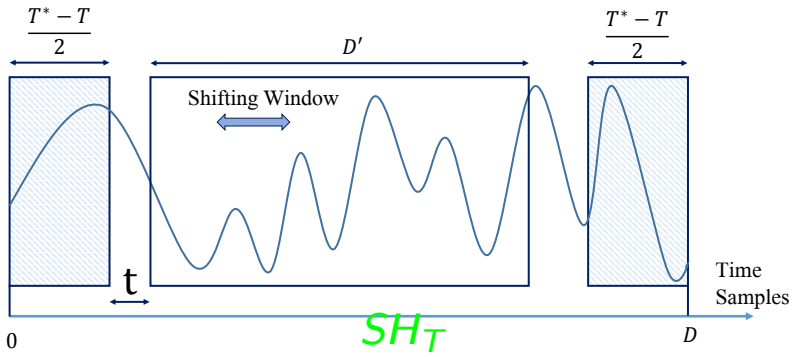
Data Augmentation

Artificially generate new training data by deforming those previously acquired,
Applying transformations that preserve the label **Z**

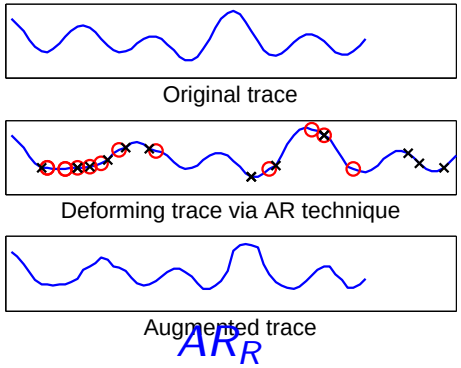
Countermeasure Emulation Idea

Emulate the effects of misaligning countermeasures to generate new traces

SHIFTING



ADD-REMOVE



Parameter T: ♯ of possible positions
Parameter R: ♯ of added and removed points
Data Augmentation techniques are applied online during training phase.

Data Augmentation

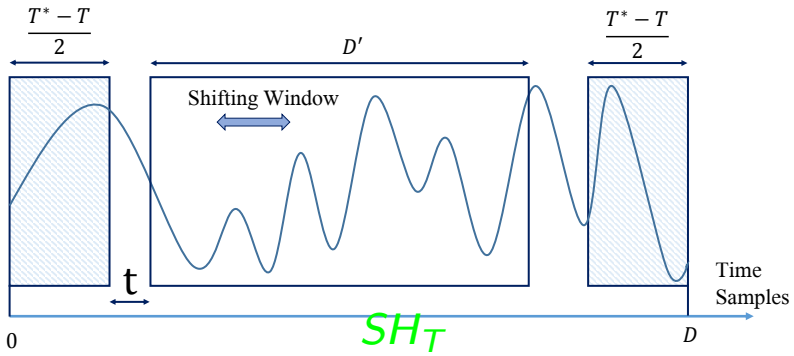
Data Augmentation

Artificially generate new training data by deforming those previously acquired,
Applying transformations that preserve the label **Z**

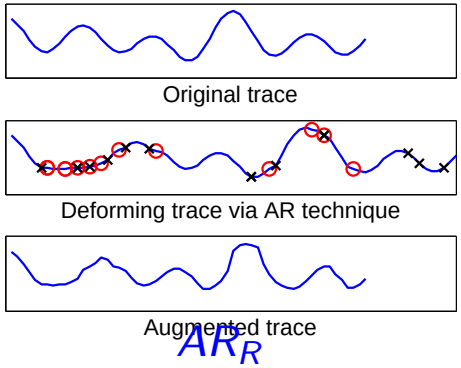
Countermeasure Emulation Idea

Emulate the effects of misaligning countermeasures to generate new traces

SHIFTING



ADD-REMOVE



Parameter T: ♯ of possible positions ⟶ new hyper-parameter

Parameter R: ♯ of added and removed points ⟶ new hyper-parameter

Data Augmentation techniques are applied online during training phase.

Data Augmentation (2)

Quantitative example

A training set of $10,000$ traces of $1,000$ time samples

$$\Downarrow SH_{10}AR_5$$

A training set of $\approx 10^{31}$ traces of $9,990$ time samples

Contents

Experimental Results

- Random delays
- Artificial Jitter
- Real Jitter

Experimental Results

- Random delays
- Artificial Jitter
- Real Jitter

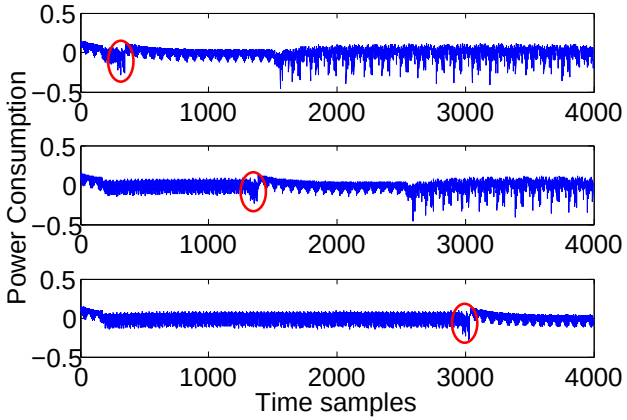
- Network architecture: $\mathrm{s} \circ [\lambda]^1 \circ [\delta \circ [\sigma \circ \gamma]^1]^4$
- Keras library with Tensorflow backend [Ker] (open source)

Experimental Results

- Random delays
- Artificial Jitter
- Real Jitter

- Network architecture: $\mathrm{s} \circ [\lambda]^1 \circ [\delta \circ [\sigma \circ \gamma]^1]^4$
- Keras library with Tensorflow backend [Ker] (open source)

Random delays



One leaking operation

Setup

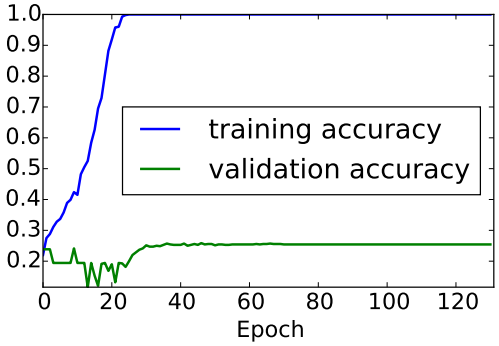
- Target Chip: Atmega328P
- Target Variable: $Z = \mathrm{HW}(\mathrm{Sbox}(P \oplus K))$
- Acquisition: through *ChipWhisperer*® platform, $\approx 4,000$ time samples
- Countermeasure: Random Delays - insertion of r *nop* operations, $r \in [0, 127]$ uniform random
- $1,000$ training traces

Random delays

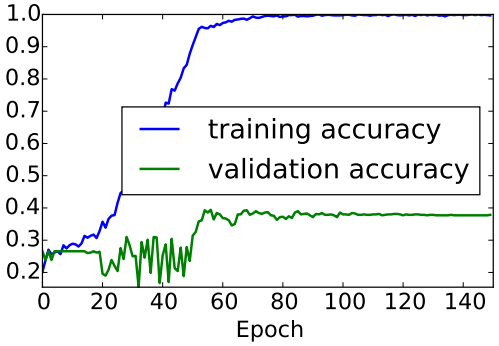
Data augmentation vs overfitting

Metrics

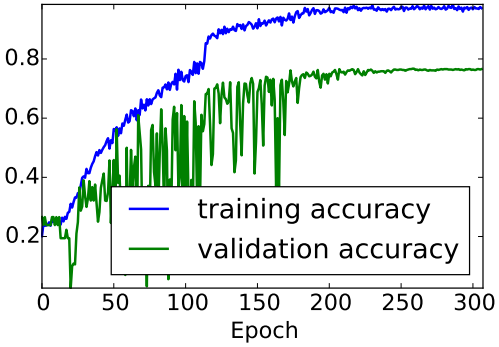
- Test accuracy: classification accuracy over the attack traces
- $N^\star$: minimum number of attack traces to make *guessing entropy* of the right key permanently equal to one ($N^\star$ estimated over 10 independent attacks)



SH_0



SH_{100}



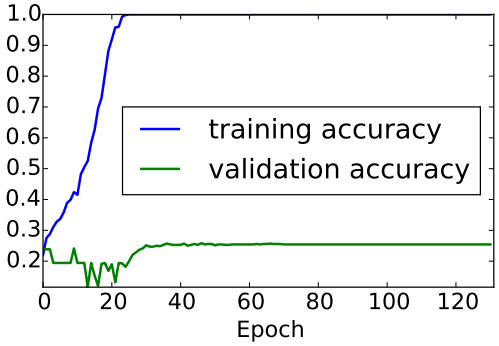
SH_{500}

Random delays

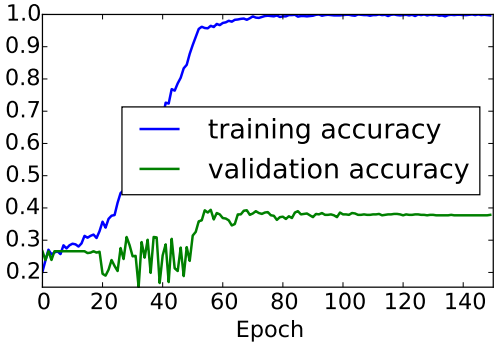
Data augmentation vs overfitting

Metrics

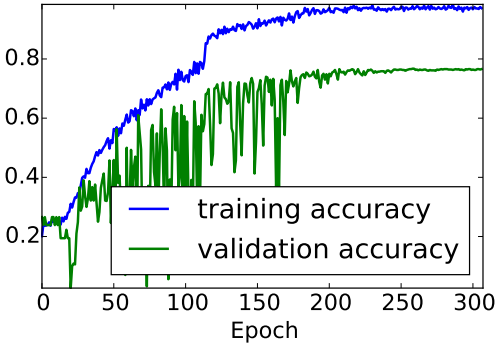
- Test accuracy: classification accuracy over the attack traces
- $N^\star$: minimum number of attack traces to make *guessing entropy* of the right key permanently equal to one ($N^\star$ estimated over 10 independent attacks)



SH_0



SH_{100}



SH_{500}

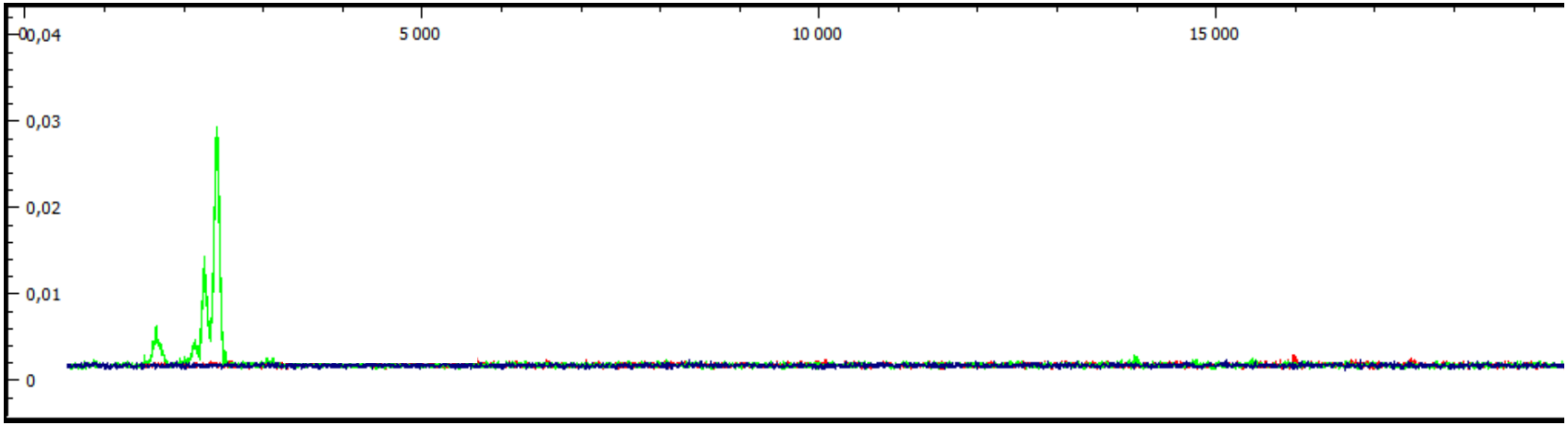
		SH_0		SH_{100}		SH_{500}	
Acc	$N^\star$	27.0%	$> 1,000$	31.8%	101	**78%**	**7**

Real Jitter (1)

Target

- AES hardware implementation
- strong jitter effect
- Target Variable: $Z = \mathrm{Sbox}(P \oplus K)$
- $2,500$ selected time samples
- $99,000$ training traces

SNR first Sbox

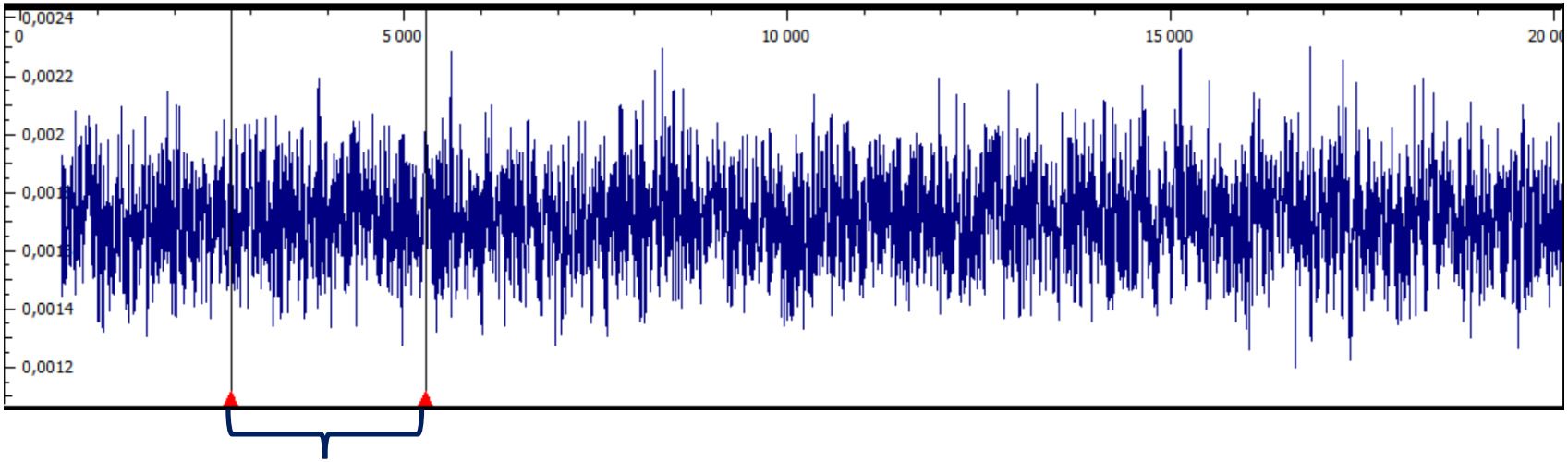


Real Jitter (1)

Target

- AES hardware implementation
- strong jitter effect
- Target Variable: $Z = \mathrm{Sbox}(P \oplus K)$
- $2,500$ selected time samples
- $99,000$ training traces

SNR second Sbox without realignment



Entry region for CNN (2,500 pts)

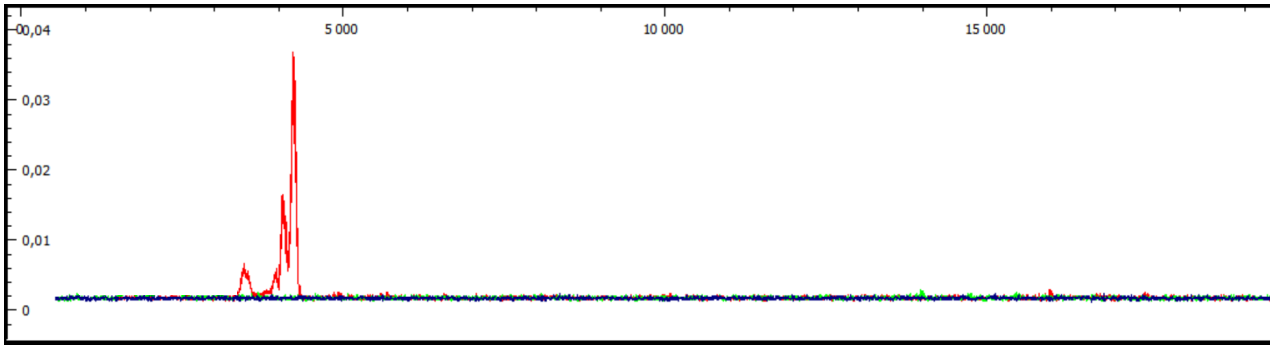
Real Jitter (2)

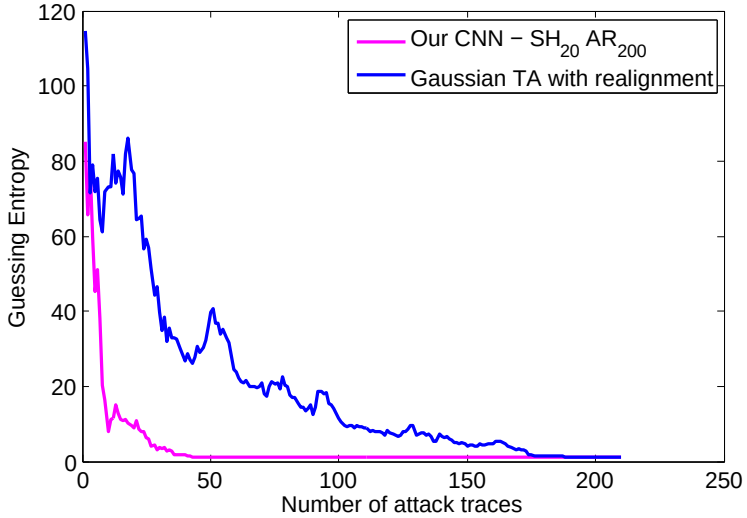
		SH_0AR_0		$SH_{10}AR_{100}$		$SH_{20}AR_{200}$	
Acc	$N^\star$	1.2%	137	1.3%	89	**1.8%**	**54**

Real Jitter (2)

		$SH_0 AR_0$		$SH_{10} AR_{100}$		$SH_{20} AR_{200}$	
Acc	$N^\star$	1.2%	137	1.3%	89	**1.8%**	**54**

SNR second Sbox with realignment





Contents

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
- ConvNets provide 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
- ConvNets provide an integrated approach to directly extract information from rough data (no preprocessing)
- ConvNet models can require plenty of data to be trained

Conclusions

- State-of-the-Art Template Attack separates resynchronization/dimensionality reduction from characterization
- ConvNets provide an integrated approach to directly extract information from rough data (no preprocessing)
- ConvNet models can require plenty of data to be trained
- Data Augmentation provides an answer to the lack of data

Conclusions

- State-of-the-Art Template Attack separates resynchronization/dimensionality reduction from characterization
- ConvNets provide an integrated approach to directly extract information from rough data (no preprocessing)
- ConvNet models can require plenty of data to be trained
- Data Augmentation provides an answer to the lack of data
- we proposed two Side-Channel-adapted Data Augmentation techniques

Conclusions

- State-of-the-Art Template Attack separates resynchronization/dimensionality reduction from characterization
- ConvNets provide an integrated approach to directly extract information from rough data (no preprocessing)
- ConvNet models can require plenty of data to be trained
- Data Augmentation provides an answer to the lack of data
- we proposed two Side-Channel-adapted Data Augmentation techniques
- we verified the effectiveness/efficiency of the ConvNet+Data Augmentation approach over different sets of misaligned data

Conclusions

- State-of-the-Art Template Attack separates resynchronization/dimensionality reduction from characterization
- ConvNets provide an integrated approach to directly extract information from rough data (no preprocessing)
- ConvNet models can require plenty of data to be trained
- Data Augmentation provides an answer to the lack of data
- we proposed two Side-Channel-adapted Data Augmentation techniques
- we verified the effectiveness/efficiency of the ConvNet+Data Augmentation approach over different sets of misaligned data

Thank You!

Questions?

References I

[Ker] *Keras library*. https://keras.io/.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira et al. Curran Associates, Inc., 2012, pp. 1097–1105. URL: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

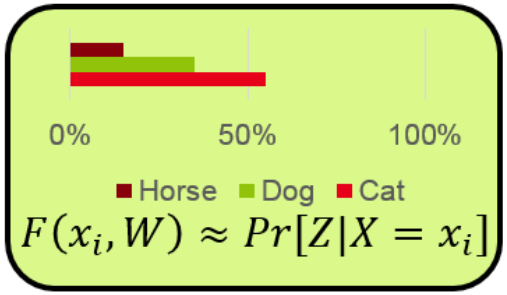
Cross-entropy Loss Function

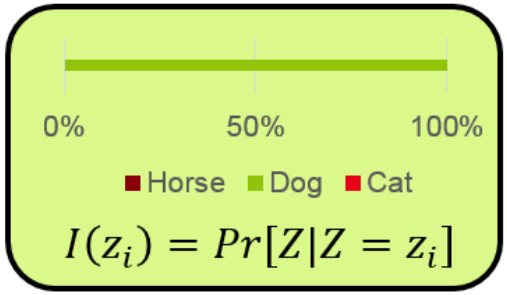
$$\mathrm{F(x,W)} = s \circ \lambda_n \circ \sigma_{n-1} \circ \lambda_{n-1} \circ \cdots \circ \lambda_1(x) = y \approx \Pr[Z|X = x]$$

- λ_i affine functions ($\lambda_i(x) = Ax + b$) depending on some parameters W (*weights*)
- σ_i non-linear functions *(activation functions)*
- s *softmax function* $s(x)[i] = \frac{e^{x[i]}}{\sum_j e^{x[j]}}$
- The weights W are *trained* on the basis of a *training set* $\{x_i, z_i\}_{i=1,..,N}$, by minimizing the *loss function*

$$L(W, x_i, z_i) = \frac{1}{N}\sum_{i=1}^{N} D(F(x_i, W), I(z_i))$$

where $I(z^i) = (0,0, \dots 0,1,0, \dots 0) = f_{Z|Z=z^i}$ (probability distribution of $Z|Z = z^i$)
and $D(f_X, f_Y) = -\sum_z \mathrm{f_Y(z)}\log(f_{X(z)})$ is the *cross-entropy* between two probability distributions f_X, f_Y (over the same probability space)





Cross-entropy measures dissimilarity between two probability distributions

Nowadays challenging image classification problems

Imagenet Large Scale Visual Recognition Challenge 2012 (ILSVRC 2012)

Classification and other tasks. 1000 categories.


Egyptian Cat



Tiger Cat

Convolutional Neural Networks (*AlexNet* [KSH12]) $\approx$ 84% of accuracy
Best of other submissions $\approx$ 74% of accuracy

Nowadays challenging image classification problems

Imagenet Large Scale Visual Recognition Challenge 2012 (ILSVRC 2012)

Classification and other tasks. 1000 categories.


Egyptian Cat



Tiger Cat

Convolutional Neural Networks (*AlexNet* [KSH12]) $\approx$ 84% of accuracy
Best of other submissions $\approx$ 74% of accuracy
At ILSVRC 2014 classification accuracy of winner *GoogLeNet* was $\approx$ 93% (close to human rate)

Nowadays challenging image classification problems

Imagenet Large Scale Visual Recognition Challenge 2012 (ILSVRC 2012)

Classification and other tasks. 1000 categories.


Egyptian Cat



Tiger Cat

Convolutional Neural Networks (*AlexNet* [KSH12]) $\approx$ 84% of accuracy
Best of other submissions $\approx$ 74% of accuracy
At ILSVRC 2014 classification accuracy of winner *GoogLeNet* was $\approx$ 93% (close to human rate)

Today, CNNs are the most diffused and best performing technique in many domains (image recognition, video analysis, natural language processing,...)

Nowadays challenging image classification problems

Imagenet Large Scale Visual Recognition Challenge 2012 (ILSVRC 2012)

Classification and other tasks. 1000 categories.


Egyptian Cat

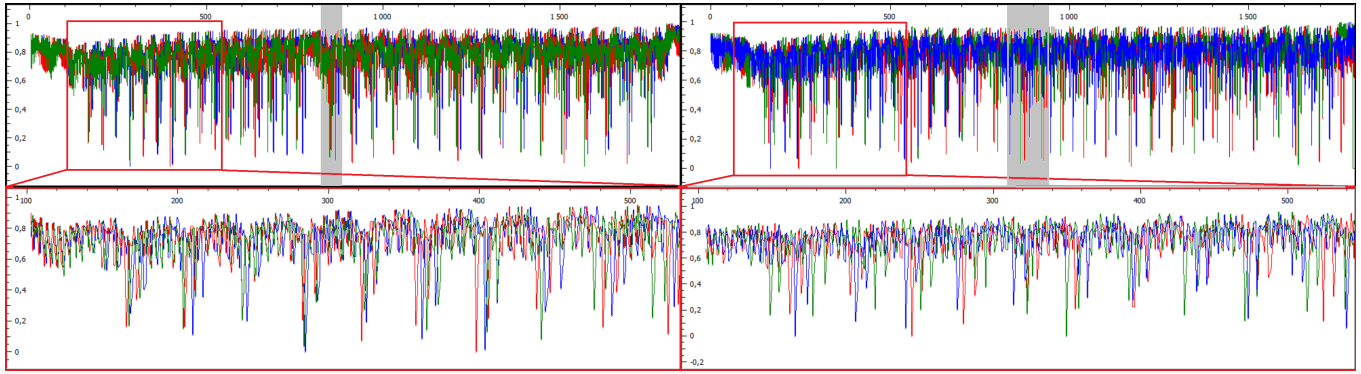


Tiger Cat

Convolutional Neural Networks (*AlexNet* [KSH12]) $\approx$ 84% of accuracy
Best of other submissions $\approx$ 74% of accuracy
At ILSVRC 2014 classification accuracy of winner *GoogLeNet* was $\approx$ 93% (close to human rate)

Today, CNNs are the most diffused and best performing technique in many domains (image recognition, video analysis, natural language processing,...)
It deals geometric deformation and de-synchronization (spacial or temporal)

Artificial Jitter



Low artificial jitter High artificial jitter

Target

- Target Variable: $Z = \mathrm{HW}(\mathrm{Sbox}(P \oplus K))$
- $\approx$ 2000 time samples
- Countermeasure: artificial signal treatment simulating clock jitter
- 10000 training traces

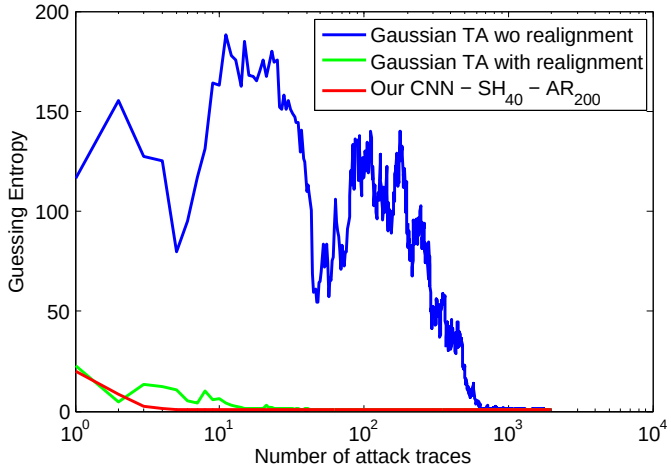
Artificial Jitter (2)

Low_jitter

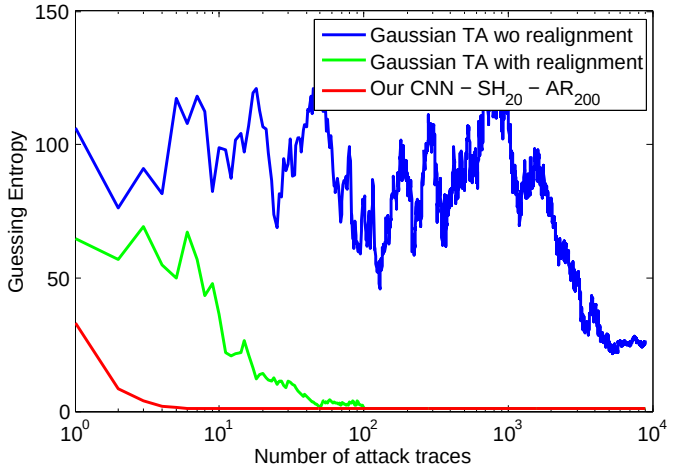
Acc	$N^\star$	SH_0		SH_{20}		SH_{40}	
AR_0		57.4%	14	82.5%	6	83.6%	6
AR_{100}		86.0%	6	87.0%	5	87.5%	6
AR_{200}		86.6%	6	85.7%	6	**87.7%**	**5**

High_jitter

Acc	$N^\star$	SH_0		SH_{20}		SH_{40}	
AR_0		40.6%	35	51.1%	9	62.4%	11
AR_{100}		50.2%	15	72.4%	11	73.5%	9
AR_{200}		64.0%	11	**75.5%**	**8**	74.4%	8

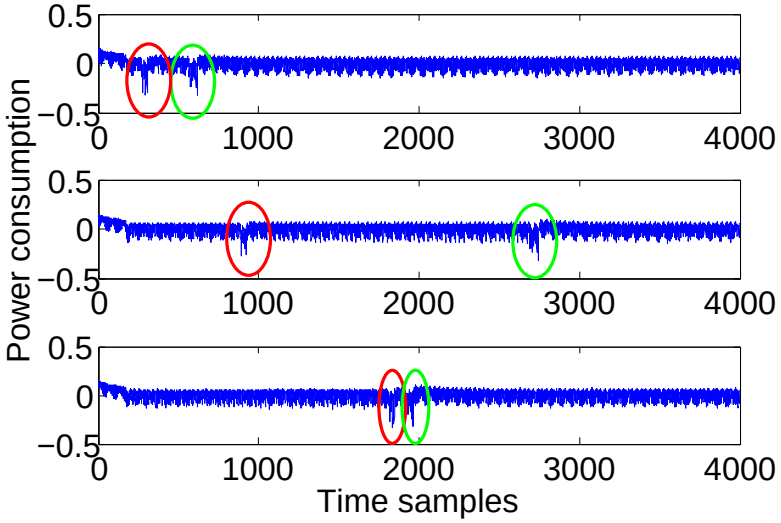


Low Jitter



High Jitter

Random Delays - Two Leaking Operations



Two leaking operations

First operation - Test acc: 76.8%, $N^\star = 7$
Second operation - Test acc: 82.5%, $N^\star = 6$

Artificial Jitter

DS_low_jitter									
a	*b*	SH_0		SH_{20}		SH_{40}		SH_{200}	
c	*d*								
AR_0		100.0%	68.7%	99.8%	86.1%	98.9%	84.1%		
		57.4%	14	82.5%	6	83.6%	6		
AR_{100}		87.7%	88.2%	82.4%	88.4%	81.9%	89.6%		
		86.0%	6	87.0%	5	87.5%	6		
AR_{200}		83.2%	88.6%	81.4%	86.9%	**80.6%**	**88.9%**		
		86.6%	6	85.7%	6	**87.7%**	**5**		
AR_{500}								85.0%	88.6%
								86.2%	5

DS_high_jitter									
a	*b*	SH_0		SH_{20}		SH_{40}		SH_{200}	
c	*d*								
AR_0		100%	45.0%	100%	60.0%	98.5%	67.6%		
		40.6%	35	51.1%	9	62.4%	11		
AR_{100}		90.4%	57.3%	76.6%	73.6%	78.5%	76.4%		
		50.2%	15	72.4%	11	73.5%	9		
AR_{200}		83.1%	67.7%	**82.0%**	**77.1%**	82.6%	77.0%		
		64.0%	11	**75.5%**	**8**	74.4%	8		
AR_{500}								83.6%	73.4%
								68.2%	11