Recent advances in side-channel analysis using machine learning techniques

Annelie Heuser

with Stjepan Picek, Sylvain Guilley, Alan Jovic, Shivam Bhasin, Tania Richmond, Karlo Knezevic
In this talk...

• Short recap on side-channel analysis and datasets

• Evaluation metrics in SCA vs ML

• Redefinition of profiled side-channel analysis through semi-supervised learning

• Learning with imbalanced data

• New approach to compare profiled side-channel attacks: efficient attacker framework
Side-channel analysis

Invasive hardware attacks, proceeding in two steps:

1) During cryptographic operations capture additional side-channel information
   • power consumption/electromagnetic emanation
   • timing
   • noise, …

2) Side-channel distinguisher to reveal the secret
Profiled SCA

- strongest attacker model
- attacker processes two devices - profiling and attacking
- attention on devices and overfitting
Profiled SCA

- Profiling phase: building model
Profiled SCA

- Attacking phase: for each trace in the attacking phase, get the probability that the trace belongs to a certain class label
Profiled SCA

- Attacking phase: maximum likelihood principle to calculate that a set of traces belongs to a certain key

![Diagram showing traces and probabilities leading to key ranking]

# key guesses

Probabilities

key ranking
Template attack

- first profiled attack
- optimal from an information theoretical point of view
- may not be optimal in practice (limited profiling phase)
- often works with the pre-assumption that the noise is normal distributed
  - to estimate: mean and covariances for each class label
  - pooled version

Algorithm

Density estimation

MODEL

densities
Support Vector Machines

- one of first introduced machine learning algorithm to SCA
- shown to be effective when the number of profiling traces is not “unlimited”
- support vectors are estimated in profiling phase

Algorithm

SVM

MODEL

hyperplanes / support vectors
Random Forest

- one of first introduced machine learning algorithm to SCA
- shown to be effective when the number of profiling traces is not “unlimited”
- often less effective as SVM, but way more efficient in the training phase
Neural Networks

- new hype for side-channel analysis
- can be really effective in particular with countermeasures
- so far most investigated are CNN and MLP

Algorithm

CNN/MLP

MODEL

network design/weights
Guessing: labels vs keys

• Make “models” on:
  • secret key directly or
  • intermediate values related to the key

• Function between intermediate value and secret key
  • one-to-one (e.g. value = \( S_{\text{box}}[\text{plaintext} \oplus \text{secretkey}] \))
  • one-to-many (e.g. value = \( \text{HW}(S_{\text{box}}[\text{plaintext} \oplus \text{secretkey}]) \))
Dataset 1

- Low noise dataset - DPA contest v4 (publicly available)
- Atmel ATMega-163 smart card connected to a SASEBO-W board
- AES-256 RSM (Rotating SBox Masking)
- In this talk: mask assumed known
Leakage

- Correlation between HW of the Sbox output and traces
Leakage densities

- In low noise scenarios: HW easily distinguishable
Dataset 2

• High noise dataset (still unprotected!)

• AES-128 core was written in VHDL in a round based architecture (11 clock cycles for each encryption).

• The design was implemented on Xilinx Virtex-5 FPGA of a SASEBO GII evaluation board.

• publicly available on github: https://github.com/AESHD/AES HD Dataset
Leakage

• Correlation between HD of the Sbox output (last round) and traces
Leakage densities

• High noise scenario: densities of HWs
Dataset 3

- AES-128: Random delay countermeasure => misaligned
- 8-bit Atmel AVR microcontroller
- publicly available on github: https://github.com/ikizhvatov/randomdelays-traces
Leakage
Leakage densities

- High noise, random delay dataset
Evaluation metrics in SCA vs ML
Evaluation metrics

- common side-channel metrics
  - Success rate: Average estimated probability of success
  - Guessing entropy: Average secret key rank
    - depends on the number of traces used in the attacking phase
    - average is computed from $E$ number of experiments
Evaluation metrics

- **Accuracy**: commonly used in machine learning applications
- **average estimated probability (percentage) of correct classification**
- **averaged over the number of traces used in the attacking phase** (not over the experiments)
- **accuracy cannot be translated into guessing entropy/ success rate!**
- **is particularly important when the values to classify are not uniformly distributed**
- **indication**: high accuracy => good side-channel performance (not vice versa)
SR/GE vs acc

Label prediction vs fixed key prediction

- accuracy: each label is considered independently (along #measurements)

- SR/GE: computed regarding fixed key, accumulated over #measurements

- low accuracy may not indicate low SR/GE

- even accuracies below random guessing may lead to high SR/low GE for a large #measurements

- random guessing should lead to low SR/ GE around $2^{n/2}$ (n=#bits)
SR/GE vs acc

Global accuracy vs class accuracy

- only relevant for non-bijective function between class and key (e.g. class involved the HW)

- the importance to correctly classify more unlikely values in the class may be more significant than others

- accuracy is averaged over all class values

- recall may be more precise
Discussion

• May there be another ML metric which is better related to GE/SR?
  • In our experiments we could not find any other metric from the set of “usual” ML metrics…

• What to do about training? Can’t we just use GE/SR….
  • Not as straightforward, and integrating GE/SR will make the training extremely more expensive
  • not all ML techniques are outputting probabilities

• For DL recent advances with cross entropy…
Redefinition of profiled side-channel analysis through semi-supervised learning
Attacker models

• profiled (traditional view): attacker processes two devices - profiling and attacking
Attacker models

• profiled (more realistic?!):
  attacker processes two devices - profiling and attacking
Semi-supervised Learning

- Labeled data (profiling device)
- Unlabeled data (attacking device)
- Combined in the profiling phase to build more realistic model about the attacking device
Semi-supervised approach

• Settings: 25k traces total
  - (100+24.9k): \( l = 100 \), \( u = 24900 \) → 0.4% vs 99.6%
  - (500+24.5k): \( l = 500 \), \( u = 24500 \) → 2% vs 98%
  - (1k+24k): \( l = 1000 \), \( u = 24000 \) → 4% vs 96%
  - (10k+15k): \( l = 10000 \), \( u = 15000 \) → 40% vs 60%
  - (20k+5k): \( l = 20000 \), \( u = 5000 \) → 80% vs 20%

• the smaller the training set the higher the influence

• labeling strategies:
  - Self-training: classifier trained with labeled data, used to predict unlabelled data, label assigned when probability > threshold
  - label spreading: label spread according to their proximity
Semi-supervised approach

- Dataset 1: Low noise unprotected, HW model
Semi-supervised approach

- Dataset 2: High noise unprotected, HW model
Semi-supervised approach

- Dataset 2: High noise unprotected, HW model
Semi-supervised approach

- Dataset 3: High noise with random delay, intermediate value model
Observations

• works in cases of 9 and 256 classes and high and low noise!!
• self-training most effective in our studies
• the higher the noise in the dataset the more labeled data is required:
  • Dataset 1: improvements for 100 and 500 labeled data
  • Dataset 2: improvements mostly for 1k labeled data
  • Dataset 3: improvements for 20k labeled data
Learning with imbalanced data
Imbalanced data

• Hamming weight leakage model commonly used

• may not reflect realistic leakage model, but reduces the complexity of learning

• works (sufficiently good) in many scenarios for attacking

• for example, occurrences of Hamming weights for 8-bit variables:

<table>
<thead>
<tr>
<th>HW value</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Occurrences</td>
<td>1</td>
<td>8</td>
<td>28</td>
<td>56</td>
<td>70</td>
<td>56</td>
<td>28</td>
<td>8</td>
<td>1</td>
</tr>
</tbody>
</table>
Why do we care?

- most machine learning techniques are “designed” to maximise accuracy

- predicting always HW class 4 gives accuracy of 27%

<table>
<thead>
<tr>
<th>HW value</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Occurrences</td>
<td>1</td>
<td>8</td>
<td>28</td>
<td>56</td>
<td>70</td>
<td>56</td>
<td>28</td>
<td>8</td>
<td>1</td>
</tr>
</tbody>
</table>

- is not related to secret key value and therefore does not give any information for SCA

- in general: less populated classes give more information about key than higher populated
Data sampling techniques

- How to transform the data set size to achieve balancedness?
  - throw away => random under sampling
  - use data multiple times => random oversampling with replacement
  - add synthetic data => synthetic minority oversampling technique (SMOTE)
  - add synthetic data + clean “noisy” data: synthetic minority oversampling technique with edited nearest neighbour (SMOTE+ENN)
Experiments

• We do not use any specific knowledge about the implementation / dataset / distribution

• Varying number of training samples in the profiling phase
  
  • 1k, 10k, 50k for Dataset 1 & 3
  
  • 1k, 10k, 25k for Dataset 2
Data sampling techniques

- Dataset 1: Low noise unprotected
Data sampling techniques

- Dataset 2: High noise unprotected
Data sampling techniques

- Dataset 3: High noise with random delay
Further results

- additionally we tested SMOTE for CNN, MLP, TA:
  - also beneficial for CNN and MLP
  - not for TA (in our settings):
    - is not “tuned” regarding accuracy
    - may still benefit if #measurements is too low to build stable profiles
- in case available: perfectly “natural" balanced dataset leads to better performance
New approach to compare profiled side-channel attacks: efficient attacker model
Efficient Attacker Model

- $N$ traces in profiling phase
- commonly: $N$ as large as possible
- more interesting: what is the minimum #traces to still be able to attack
- real-world evaluations only have limited resources
Efficient Attacker Model

- Why?
  More traces is not always better…
Efficient Attacker Model

• Why?
  More traces is not always better…

• Realistic setting:
  • device 1: training
  • device 2: testing

• Overfitting
Efficient Attacker Model

• **Minimum number of traces** such that an **evaluation metric** is smaller than a threshold depending on the **number of attacking traces**

• certain threshold for example:
  
  • guessing entropy < 10,
  
  • success rate > 90%
  
  • accuracy > 10%
Efficient Attacker Model

- MLP vs TA (pooled) and HW vs value model:
  - only with value model single-trace attack possible
  - intermediate value require more traces in profiling
  - MLP requires less traces in profiling with value model
  - for HW model MLP and TA both perform similarly
Discussion

• Can be used to benchmark “anything”:
  • Leakage model: HW vs intermediate
  • Attacks: DL vs ML vs TA vs …. 
  • Datasets / implementations / designs 

• Future directions
  • include computational complexity / required resources of attacks as a further dimension
Conclusion

• Evaluation metrics in SCA vs ML:
  ➡ accuracy != GE or SR

• Redefinition of profiled side-channel analysis through semi-supervised learning:
  ➡ consider unlabelled data from testing device already in profiling phase

• Learning with imbalanced data
  ➡ Data sampling helps to improve GE/SR

• New approach to compare profiled side-channel attacks: efficient attacker model
  ➡ More realistic and meaningful benchmarking!
Looking for PostDocs...

- Always and currently looking for good candidates of postdocs in our team (TAMIS, IRISA (Inria, CNRS,…), Rennes, France)

- Research in
  - Side-channel analysis (particularly post-quantum crypto)
  - Formal methods
  - malware
  - code analysis
  - ….
Recent advances in side-channel analysis using machine learning techniques

Annelie Heuser

with Stjepan Picek, Sylvain Guilley, Alan Jovic, Shivam Bhasin, Tania Richmond, Karlo Knezevic