Strength in Numbers: Improving Generalization with Ensembles in Machine Learning-based Profiled Side-Channel Analysis

Guilherme Perin, Lukasz Chmielewski, Stjepan Picek
Contributions

• Analysis of output class probabilities (predictions)

• Using proper metrics for profiled SCA with deep learning

• Improving generalization in DL-based profiled SCA:
  • Ensembles: combining multiple NN models into a stronger model
DL-based profiled SCA

Device A (AES) -> Profiling Traces (*known key*)

Device B (AES) -> Attack Traces (*unknown key*)

(learning algorithm - DNN)

Validation Traces (*known key*)

(learning algorithm - DNN)

Recovered key

Good (enough) generalization
“... Improving Generalization ...”

- If (n-order) SCA leakages are there, we can improve generalization by:
  - Using a small NN model (implicitly regularized)
  - Using a large NN model and add (explicit) regularization (dropout, data augmentation, noise layers, batch normalization, weight decay, etc.)
  - Being precise in training time/epochs (early stopping)
  - Or, using ensembles.
DL-based SCA is (mostly) about hyperparameters

- No points of interest selection
- Less sensitive to trace desynchronization (CNN)
- Implement high-order profiled SCA
- Allow visualization techniques

- Work in progress:
  - Create a good DL model is difficult: efficient and automated hyperparameters tuning not solved yet for SCA
  - SCA is already costly by itself: adding hyperparameters tuning can render the DL-based SCA impractical
DL-based SCA is (also) about metrics

- Accuracy, Loss, Recall, Precision: *not very consistent for SCA (multiple test traces)*
- Success Rate
- Guessing Entropy
  - Custom loss/error function in Keras/TensorFlow

What can we learn here?

SCA Traces → Predictions → Key Rank (GE, SR)
Results on Masked AES (MLP)
Attacking 1 key byte with HW model
**Output Class Probabilities**

Example: HW model of 1 byte on AES (S-box output)

\[
P = \begin{bmatrix}
    p_{0,0} & p_{0,1} & p_{0,2} & p_{0,3} & p_{0,4} & p_{0,5} & p_{0,6} & p_{0,7} & p_{0,8} \\
    p_{1,0} & p_{1,1} & p_{1,2} & p_{1,3} & p_{1,4} & p_{1,5} & p_{1,6} & p_{1,7} & p_{1,8} \\
    p_{2,0} & p_{2,1} & p_{2,2} & p_{2,3} & p_{2,4} & p_{2,5} & p_{2,6} & p_{2,7} & p_{2,8} \\
    \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
    p_{N-1,0} & p_{N-1,1} & p_{N-1,2} & p_{N-1,3} & p_{N-1,4} & p_{N-1,5} & p_{N-1,6} & p_{N-1,7} & p_{N-1,8}
\end{bmatrix}
\]

\[p_{i,j} = \text{probability that trace } i \text{ contains label (HW) } j\]

\[j = S_{box}(pt_i \oplus k_i) \text{ (leakage or selection function)}\]
Summation: Key Rank

Label according to key guess $k$

$Label(0) = Sbox(pt_0 \oplus k) = 3$
$Label(1) = Sbox(pt_1 \oplus k) = 6$
$Label(2) = Sbox(pt_2 \oplus k) = 2$

... $Label(N-1) = Sbox(pt_{N-1} \oplus k) = 4$

$P(k) =$

$$P(k) = \sum_{i=0}^{N-1} \log p_{i,j} = \log p_{0,3} + \log p_{1,6} + \log p_{2,2} + \cdots + \log p_{N-1,4}$$

Recovered key: $\arg\max_k [P(0), P(1), \ldots, P(255)]$
# Summation: Key Rank

Test Accuracy is **100%**

\[ P(k) = \sum_{i=0}^{N-1} \log p_{i,j} = \log 0.40 + \log 0.35 + \log 0.53 + \cdots + \log 0.40 \]

Always the **highest** value per row
**Summation: Key Rank**

Test Accuracy is 27%

\[ P(k) = \sum_{i=0}^{N-1} \log p_{i,j} = \log 0.25 + \log 0.15 + \log 0.53 + \cdots + \log 0.25 \]

**NOT** Always the highest value per row
Rank of Class Probabilities

Ordering keys by accuracy

\[ \max(p_{i,0}, \ldots, p_{i,8}) \]

\[ \min(p_{i,0}, \ldots, p_{i,8}) \]
Rank of Class Probabilities

Ordering keys by accuracy

Low Ranks: summation for $k$ is pushed up

High Ranks: summation for $k$ is pushed down

Correct key candidate
Incorrect key candidates

large influence on correct $P(k)$

small influence on correct $P(k)$
Results on Leaky AES (MLP)
Attacking 1 key byte with HW model

Output class probabilities are pushed towards ranks 1 and 2.

No test traces with high ranked probabilities.
Results on Masked AES (MLP)
Attacking 1 key byte with HW model

Successful key recovery

Output class probabilities are pushed towards ranks 1 and 2

Few test traces with high ranked probabilities
Two CNN models on masked AES

- CNN with 4 hidden layers
- CNN with 3 hidden layers
Deep learning analysis requires a large amount of hyperparameters experiments:

\[ h_{\text{best}} = \arg\min_{m \in M} \text{Loss}(\lambda_m, t_{\text{train}}, t_{\text{val}}) \]

Select a proper metric

\[ h_{\text{best}} = \arg\min_{m \in M} \text{GE}(\lambda_m, t_{\text{train}}, t_{\text{val}}) \]

From multiple models, we elect a best one. Why not benefit from multiple models instead of a best single model?
Ensembles

- Boosting
- Stacking
- Bootstrap Aggregating (Bagging)

\[ P(k) = \sum_{m=0}^{M-1} \sum_{i=0}^{N-1} \log p_{i,j,m} \]

Select best models based on GE: \( M_{\text{best}} < M \)

\[ \arg\min_m GE(\lambda_m, t_{\text{train}}, t_{\text{val}}) \]

- hyperparameters
- train traces
- validation traces
Ensembles

Ensemble \((M_{\text{best}} = 10, M = 50)\)

Single Best Model = \(\arg\min_m GE(\lambda_m, t_{\text{train}}, t_{\text{val}})\)

\[
P(k) = \sum_{m=0}^{M-1} \sum_{i=0}^{N-1} \log p_{i,j,m}
\]

\[
P(k) = \sum_{i=0}^{N-1} \log p_{i,j}
\]
## Datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Training</th>
<th>Validation</th>
<th>Test</th>
<th>Features</th>
<th>Countermeasures</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pinata SW AES</td>
<td>6,000 (fixed key)</td>
<td>1,000</td>
<td>1,000</td>
<td>400</td>
<td>No</td>
</tr>
<tr>
<td>DPAv4</td>
<td>34,000 (fixed key)</td>
<td>1,000</td>
<td>1,000</td>
<td>2,000</td>
<td>RSM</td>
</tr>
<tr>
<td>ASCAD</td>
<td>200,000 (random keys)</td>
<td>500</td>
<td>500</td>
<td>1,400</td>
<td>Masking</td>
</tr>
<tr>
<td>CHES CTF 2018</td>
<td>43,000 (fixed key)</td>
<td>1,000</td>
<td>1,000</td>
<td>2,000</td>
<td>Masking</td>
</tr>
</tbody>
</table>
## Range of Hyperparameters

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>MIN</th>
<th>MAX</th>
<th>STEP</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MLP</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Learning Rate</td>
<td>0.0001</td>
<td>0.001</td>
<td>0.0001</td>
</tr>
<tr>
<td>Mini-batch</td>
<td>100</td>
<td>1000</td>
<td>100</td>
</tr>
<tr>
<td>Dense Layers</td>
<td>2</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>Neurons</td>
<td>100</td>
<td>1000</td>
<td>100</td>
</tr>
<tr>
<td>Activation Function</td>
<td>Tanh, ReLU, ELU or SELU</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>MIN</th>
<th>MAX</th>
<th>STEP</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CNN</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Learning Rate</td>
<td>0.0001</td>
<td>0.001</td>
<td>0.0001</td>
</tr>
<tr>
<td>Mini-batch</td>
<td>100</td>
<td>1000</td>
<td>100</td>
</tr>
<tr>
<td>Convolution Layers (i)</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Filters</td>
<td>8*i</td>
<td>32*i</td>
<td>4</td>
</tr>
<tr>
<td>Kernel Size</td>
<td>10</td>
<td>20</td>
<td>2</td>
</tr>
<tr>
<td>Stride</td>
<td>5</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>Dense Layers</td>
<td>2</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>Neurons</td>
<td>100</td>
<td>1000</td>
<td>100</td>
</tr>
<tr>
<td>Activation Function</td>
<td>Tanh, ReLU, ELU or SELU</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*optimal ranges based on literature
Results on ASCAD (Hamming Weight)
Results on ASCAD (Identity)

MLP

CNN
Conclusions

• Output class probabilities are a valid distinguisher for side-channel analysis.
• Output class probabilities are sensitive to small changes in hyperparameters: ensembles remove the effect of small variations, improving generalization results.
• Ensembles do not replace hyperparameters search. Ensembles relax the fine tuning of hyperparameters: GE or SR of ensemble tends to be superior to GE or SR of a single best model.
• Ensembles do not improve learnability: they improve what single models already learn.
• Limited amount of models can be enough to build a strong ensemble.

As future works:
• Explore another ensemble methods (e.g., stacking).
• Verify how ensembles work in combination with other regularization methods and other metrics (SR, MI).
• Formalize the density distribution of output class probabilities (a new metric).
Thank you!

- Our code is available at: https://github.com/AISyLab/EnsembleSCA