Improving Side-channel Analysis though Semi-supervised Learning

Stjepan Picek\textsuperscript{1} Annelie Heuser\textsuperscript{2} Alan Jovic\textsuperscript{3}
Karlo Knezevic\textsuperscript{3} Tania Richmond\textsuperscript{2} Axel Legay\textsuperscript{4}

\textsuperscript{1} Delft University of Technology, Delft, The Netherlands
\textsuperscript{2} Univ Rennes, Inria, CNRS, IRISA, France
\textsuperscript{3} University of Zagreb Faculty of Electrical Engineering and Computing, Croatia
\textsuperscript{4} UCLouvain, Belgium
Table of contents

1. Profiled side-channel analysis

2. Semi-supervised leaning in profiled SCA

3. Experimental validation

4. Conclusion
Profiled side-channel analysis
Notations and Terminology

traces = measurements, features = points in time/interest

classifier = distinguisher = attack
E.g. Template attack, SVM attack

label = class = intermediate value = leakage model
E.g.: $\text{SBox}(\text{plaintext} \oplus \text{key})$, $\text{HW}(\text{SBox}(\text{plaintext} \oplus \text{key}))$
Classifiers used in this work (1/2)

- Template attack:
  - **profiling phase**: estimate multivariate Gaussian distribution $\Rightarrow$ mean vector and covariance matrices for each label
  - **attacking phase**: given an unseen sample evaluate probability distribution

- Template attack pooled version:
  - **profiling phase**: estimate multivariate Gaussian distribution $\Rightarrow$ mean vector for each label, and one covariance matrix
  - **attacking phase**: given an unseen sample evaluate probability distribution
Classifiers used in this work (2/2)

- Naive Bayes
  - **profiling phase**: estimate probability distribution assuming it is Gaussian and univariate $\Rightarrow$ mean value and variance for each feature and label
  - **attacking phase**: given an unseen sample evaluate probability distribution

- Support vector machines (SVM)
  - **profiling phase**: estimate hyperplane seperations between labels
  - **attacking phase**: evaluate one which side of the hyperplanes the unseen lies
Profiled side-channel analysis

Profiling phase
- Input: traces and known labels
- Output: profiled model

Attacking phase
- Input: traces, profiled model, hypothetical labels
- Output: secret value
Profiled side-channel analysis

- Traditional setup: only information transferred between phases is the profiled model
- If profiling phase is *unlimited* traditional setup reasonable, since profiling phase should yield ’optimal’ result
- If attacker is more *restricted* in resources, why relying on strict separation between phases?
Semi-supervised leaning in profiled SCA
Semi-supervised leaning in profiled SCA

Profiling phase
- Input: traces, known labels, traces (attack), predicted labels (attack)
- Output: profiled model

Attacking phase
- Input: traces, profiled model, hypothetical labels
- Output: secret value
Semi-supervised leaning in profiled SCA

Suitable

• Attacker has limited/restricted resources
  • device on which he has full knowledge: only a limited amount of profiling data
  • device with secret values: ability to gain more or equal amount of data
• Noise level is reasonably low (reasonably needs to be evaluated)

Advantages

• Additional information about leakage distribution (in the attacking phase)
• → more accurate prediction about labels in attacking phase

Disadvantages/Limitations

• May introduce wrongly predicted labels and therefore misclassification
Why helpful?

- Intuitive simplified example
- Decision boundary enhanced through unlabeled data
When is semi-supervised learning meaningful?

When is it advantageous to consider unlabeled data?

• ⇒ Distribution of data samples is of importance, and certain assumptions should hold

• Two main important assumptions in semi-supervised learning:
  • Smoothness assumption
  • Manifold assumption

• If one assumption is valid, then specific semi-supervised labeling techniques are exploitable
Semi-supervised smoothness assumption

• If two points $x_1, x_2$ in a high-density region are close, then their labels $y_1, y_2$ should be close

• Accordingly, if they are separated by a low-density region, then their labels do not need to be close

• Note that, similar assumptions also need to hold for supervised learning

• Should generally hold in SCA: measurements are related to the activity of device (labels)
Manifold assumption

- The (high-dimensional) data lie (roughly) in a low-dimensional manifold
- Classifier is then able to operate on the corresponding low dimension and separate data belonging to a manifold
- Should generally hold in SCA: features/points in trace typically have dependencies between each other ⇒ lower dimensional manifold than in dimension # of features
How to predict labels of unlabeled data?

Two common prediction algorithms for semi-supervised learning:

- Self-training of a classifier
- Graph-based learning with label spreading

... many more out there, active research direction in machine learning.
Machine learning tools for label predication

Self-training of a classifier with added threshold value

Idea

1. Train classifier on labeled data
2. Predict unlabeled data
3. if probability of prediction is $\geq$ threshold value:
   - add label to data
   - else keep it unlabeled
4. Repeat from 1 until classification accuracy decreases, or no samples exists with probability $\geq$ threshold

Drawbacks

- Depends on classifier
- Possible mistakes reinforce themselves, noise amplifies
Machine learning tools for label predication

Self-training of a classifier with added threshold value

Idea

1. Train classifier on labeled data
2. Predict unlabeled data
3. if probability of prediction is $\geq$ threshold value:
   - add label to data
   - else keep it unlabeled
4. Repeat from 1 until classification accuracy decreases, or no samples exists with probability $\geq$ threshold

In our experiments, classifier:

- SVM: relies on manifold assumption
- Naive Bayes: relies on smoothness assumption
Machine learning tools for label prediction

Graph-based learning using label spreading

Idea

• Represent data as a graph
• Vertices are traces: labeled if exists, otherwise unlabeled
• Edges are labeled with distances of neighbor nodes (euclidean distance)
• Idea:
  • Vertix labels propagate through graph
  • \( k \)-NN neighbors as a technique to assign labels
• Depends on the manifold assumption

Drawback

• Data should be represenable in a graph structure/ problem if graph does not fit the task
Experimental validation
Experimental validation

Our experimental settings

<table>
<thead>
<tr>
<th>labeled</th>
<th>unlabeled</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>12 900</td>
</tr>
<tr>
<td>250</td>
<td>12 750</td>
</tr>
<tr>
<td>500</td>
<td>12 500</td>
</tr>
<tr>
<td>1 000</td>
<td>12 000</td>
</tr>
<tr>
<td>3 000</td>
<td>10 000</td>
</tr>
<tr>
<td>5 000</td>
<td>8 000</td>
</tr>
<tr>
<td>7 000</td>
<td>6 000</td>
</tr>
<tr>
<td>10 000</td>
<td>3 000</td>
</tr>
</tbody>
</table>

- Scenarios:
  - Dataset: Dpav4 contest (turned into unmasked scenario), in paper additionally Dpav2 contest (very noisy/ usual drifting noise)
  - Results given in accuracy: percentage of correctly classified labels in the attacking set
Our experimental settings

- Classifiers:
  - Naïve Bayes (Bayes theorem, independent points of interest)
  - Support vector machine (hyperplane classification)
  - Template attack (Bayes theorem, dependent points of interest)
  - Template attack pooled (only one covariance matrix for all label classes)

- Leakage model: S-box output (256 classes), Hamming weight of S-box output (9 classes)

- Self-learning uses Naive Bayes and SVM (RBF kernel) as a classifier
Experimental results

Our experimental settings

- DPAv4 contest, 9 classes
- Accuracy: supervised | self-learning | label spreading

<table>
<thead>
<tr>
<th>Size</th>
<th>TA</th>
<th>$T_A$</th>
<th>$T_A^p$</th>
</tr>
</thead>
<tbody>
<tr>
<td>100/+12.9k</td>
<td>0.3</td>
<td>58.9</td>
<td>45.4</td>
</tr>
<tr>
<td>250/+12.75k</td>
<td>0.3</td>
<td>12.6</td>
<td>53</td>
</tr>
<tr>
<td>500/+12.5k</td>
<td>0.3</td>
<td>56.6</td>
<td>68.9</td>
</tr>
<tr>
<td>1k/+12k</td>
<td>1.3</td>
<td>44.2</td>
<td>73.1</td>
</tr>
<tr>
<td>3k/+10k</td>
<td>5.2</td>
<td>53</td>
<td>74.9</td>
</tr>
<tr>
<td>5k/+8k</td>
<td>2.8</td>
<td>46.4</td>
<td>75.8</td>
</tr>
<tr>
<td>7k/+6k</td>
<td>11.2</td>
<td>75.6</td>
<td>76.5</td>
</tr>
<tr>
<td>10k/+3k</td>
<td>0.4</td>
<td>73.8</td>
<td>77.2</td>
</tr>
<tr>
<td>13k</td>
<td>75.3</td>
<td>77.7</td>
<td>78</td>
</tr>
</tbody>
</table>
Experimental results

Our experimental settings

- DPAv4 contest, 9 classes
- Accuracy: supervised | self-learning | label spreading

<table>
<thead>
<tr>
<th>Size</th>
<th>NB</th>
<th>SVM</th>
<th>SVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>100/+12.9k</td>
<td>61.5</td>
<td>69.1</td>
<td>69</td>
</tr>
<tr>
<td>250/+12.75k</td>
<td>64.3</td>
<td>78.4</td>
<td>78.2</td>
</tr>
<tr>
<td>500/+12.5k</td>
<td>65.9</td>
<td>82.7</td>
<td>82.8</td>
</tr>
<tr>
<td>1k/+12k</td>
<td>64.8</td>
<td>86.6</td>
<td>87.1</td>
</tr>
<tr>
<td>3k/+10k</td>
<td>67.2</td>
<td>90.8</td>
<td>90.5</td>
</tr>
<tr>
<td>5k/+8k</td>
<td>67.9</td>
<td>92</td>
<td>92.3</td>
</tr>
<tr>
<td>7k/+6k</td>
<td>68</td>
<td>92.8</td>
<td>92.7</td>
</tr>
<tr>
<td>10k/+3k</td>
<td>68.1</td>
<td>93.3</td>
<td>93.6</td>
</tr>
<tr>
<td>13k</td>
<td>68.4</td>
<td>93.7</td>
<td></td>
</tr>
</tbody>
</table>
Experimental results

Our experimental settings

- DPAv4 contest, 256 classes
- Accuracy: supervised | self-learning | label spreading

<table>
<thead>
<tr>
<th>Size</th>
<th>TA</th>
<th>TA$p$</th>
</tr>
</thead>
<tbody>
<tr>
<td>100/+12.9k</td>
<td>0.3</td>
<td>0.4</td>
</tr>
<tr>
<td>250/+12.75k</td>
<td>0.3</td>
<td>3.3</td>
</tr>
<tr>
<td>500/+12.5k</td>
<td>0.4</td>
<td>6.4</td>
</tr>
<tr>
<td>1k/+12k</td>
<td>0.4</td>
<td>10.2</td>
</tr>
<tr>
<td>3k/+10k</td>
<td>0.1</td>
<td>16.3</td>
</tr>
<tr>
<td>5k/+8k</td>
<td>0.2</td>
<td>19.2</td>
</tr>
<tr>
<td>7k/+6k</td>
<td>0.3</td>
<td>20.6</td>
</tr>
<tr>
<td>10k/+3k</td>
<td>0</td>
<td>22.5</td>
</tr>
<tr>
<td>13k</td>
<td>0.1</td>
<td>23.7</td>
</tr>
</tbody>
</table>
Experimental results

Our experimental settings

- DPAv4 contest, 256 classes
- Accuracy: supervised | self-learning | label spreading

<table>
<thead>
<tr>
<th>Size</th>
<th>NB</th>
<th>SVM</th>
<th>SVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>100/+12.9k</td>
<td>1.5</td>
<td>2.7</td>
<td>5.1</td>
</tr>
<tr>
<td>250/+12.75k</td>
<td>2.2</td>
<td>3.1</td>
<td>6.8</td>
</tr>
<tr>
<td>500/+12.5k</td>
<td>4.9</td>
<td>5.7</td>
<td>10.3</td>
</tr>
<tr>
<td>1k/+12k</td>
<td>10.5</td>
<td>9.3</td>
<td>13.6</td>
</tr>
<tr>
<td>3k/+10k</td>
<td>16.5</td>
<td>15.6</td>
<td>22.4</td>
</tr>
<tr>
<td>5k/+8k</td>
<td>18</td>
<td>17.3</td>
<td>27.4</td>
</tr>
<tr>
<td>7k/+6k</td>
<td>19.5</td>
<td>18.4</td>
<td>30</td>
</tr>
<tr>
<td>10k/+3k</td>
<td>20.1</td>
<td>19.6</td>
<td>33.3</td>
</tr>
<tr>
<td>13k</td>
<td>20.2</td>
<td></td>
<td>34.9</td>
</tr>
</tbody>
</table>
Conclusion
Conclusion

- Explored the concept of semi-supervised learning when the attacker is ‘restricted’ (not unlimited power) in the profiling phase
- Self-learning and label spreading was used in our experiments
- For DPAcontest v4 (low noise scenario) we observed:
  - Semi-supervised learning helped mostly throughout all classifiers for 9 classes
  - Semi-supervised learning for 256 classes mostly failed (for accuracy, maybe advantages in guessing entropy?)
  - Self-learning more advantageous than label spreading
  - Biggest advantage for template attack
- In paper: also DPAcontest v2, unusual noise scenario, naturally here semi-supervised learning did not help the attacker
- In final version:
  - Comparison using guessing entropy
  - Additional two datasets: medium/high noise, random delay countermeasure
Questions?