Improving Side-channel Analysis though Semi-supervised Learning

Stjepan Picek[1] Annelie Heuser[2] Alan Jovic[3]
Karlo Knezevic[3] Tania Richmond[2] Axel Legay[4]

[1] Delft University of Technology, Delft, The Netherlands

[2] Univ Rennes, Inria, CNRS, IRISA, France

[3] University of Zagreb Faculty of Electrical Engineering and Computing, Croatia

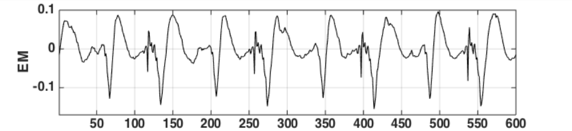
[4] UCLouvain, Belgium

Table of contents

Profiled side-channel analysis

Notations and Terminology

<u>traces</u> = measurements, <u>features</u> = points in time/interest



<u>classifier</u> = distinguisher = attack

E.g. Template attack, SVM attack

<u>label</u> = class = intermediate value = leakage model

E.g.: SBox($plaintext \oplus key$), HW(SBox($plaintext \oplus key$))

Profilied side-channel classifiers

Classifiers used in this work (1/2)

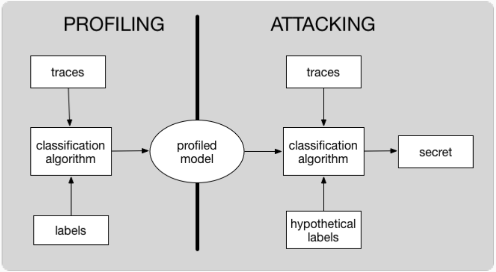
- Template attack:
  - profiling phase: estimate multivariate Gaussian distribution $\Rightarrow$ mean vector and covariance matrices for each label
  - attacking phase: given an unseen sample evaluate probability distribution
- Template attack pooled version:
  - profiling phase: estimate multivariate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

Profilied side-channel classifiers

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

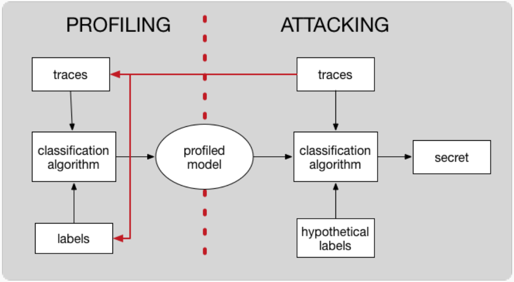
- Input: traces, profiled model, hypothetical labels
- Output: secret value

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)
- $\Rightarrow$ more accurate prediction about labels in attacking phase

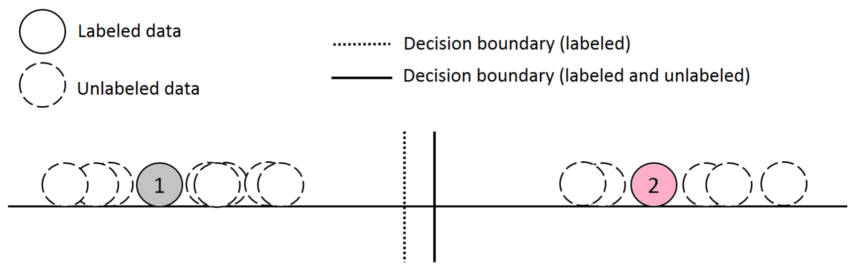
Disadvantages/Limitations

- May introduce wrongly predicted labels and therefore misclassification

Semi-supervised learning

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?

- $\Rightarrow$ 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 assumptions (1/2)

Semi-supervised smoothness assumption

- If two points x_1, x_2 in a high-density region are close, than 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)

Semi-supervised assumptions (2/2)

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 seperate data belowing to a manifold
- Should generally hold in SCA: features/points in trace typically have dependencies between each other $\Rightarrow$ lower dimensional manifold than in dimension # of features

Machine learning tools for label predication

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 reasearch 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 predication

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

- Scenarios:

labeled	unlabeled
100	12 900
250	12 750
500	12 500
1 000	12 000
3 000	10 000
5 000	8 000
7 000	6 000
10 000	3 000

- 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

Experimental validation

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

Size	TA			TA_p		
100/+12.9k	0.3	**58.9**	18.8	45.4	**67.6**	21.1
250/+12.75k	0.3	12.6	**61.4**	53	**75.2**	71.3
500/+12.5k	0.3	56.6	**58.8**	68.9	**76.9**	74.5
1k/+12k	1.3	**44.2**	7.1	73.1	**78.3**	76.6
3k/+10k	5.2	53	**66.6**	74.9	**78.1**	77.4
5k/+8k	2.8	**46.4**	3.2	75.8	**78.4**	78
7k/+6k	11.2	**75.6**	14.8	76.5	**78**	77.9
10k/+3k	0.4	**73.8**	49.6	77.2	77.9	**78**
13k	75.3			77.7		

Experimental results

Our experimental settings

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

Size	NB			SVM		
100/+12.9k	**61.5**	59	30	**69.1**	69	25
250/+12.75k	64.3	64.6	**65.3**	78.4	**78.2**	77.5
500/+12.5k	65.9	**66.2**	65.5	82.7	**82.8**	81.1
1k/+12k	64.8	**68.1**	67.7	86.6	**87.1**	84.1
3k/+10k	67.2	68.3	**68.7**	90.8	90.5	**91.8**
5k/+8k	67.9	68.1	**68.8**	92	**92.3**	91.8
7k/+6k	68	**68.4**	68.6	**92.8**	92.7	92.5
10k/+3k	68.1	68.7	68.7	93.3	**93.6**	93.5
13k	68.4			93.7		

Experimental results

Our experimental settings

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

Size	TA			TA_p		
100/+12.9k	0.3	0.3	0.3	0.4	**3.4**	2.6
250/+12.75k	0.3	0.3	0.3	3.3	**3.7**	3.5
500/+12.5k	0.4	**0.5**	0.4	6.4	**7.1**	7
1k/+12k	0.4	**0.5**	0.4	**10.2**	9.5	9
3k/+10k	0.1	**0.4**	0.3	**16.3**	15.5	14.8
5k/+8k	**0.2**	0.1	0.1	**19.2**	18.7	17.2
7k/+6k	**0.3**	0.1	0.1	20.6	**21**	20.1
10k/+3k	0	0.2	0.2	**22.5**	22.4	21.9
13k	0.1			23.7		

Experimental results

Our experimental settings

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

Size	NB			SVM		
100/+12.9k	1.5	**2.7**	1.7	**5.1**	4.2	3.7
250/+12.75k	2.2	**3.1**	3	**6.8**	6.4	6.1
500/+12.5k	4.9	5.7	5.7	**10.3**	8.5	7.9
1k/+12k	**10.5**	9.3	8.5	**13.6**	12.8	11
3k/+10k	**16.5**	15.6	15	**22.4**	21.7	18.7
5k/+8k	**18**	17.3	16	**27.4**	25.7	24.8
7k/+6k	**19.5**	18.4	17	**30**	29	26.9
10k/+3k	**20.1**	19.6	18.1	**33.3**	32.8	28.8
13k	20.2			34.9		

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?