Lowering the Bar: Deep Learning for Side Channel Analysis

Guilherme Perin, Baris Ege, Jasper van Woudenberg
December 4, 2018
Before

Signal processing → Leakage modeling → Key
After
Helping security

Implementation flaws
Vulnerabilities
Source of leakages

Fixes / Improvement

Activation paths
Secure Product
Faster certification

Metrics
Power / EM side channel analysis
Power analysis

Some crypto algorithm
Example (huge) leakage
Signal processing

Raw trace

Processed trace
Misalignment
AES-128 first round attack

Key Addition

Unknown

Known

Leakage model,
Power prediction
Points of interest selection

Correlation, T-test, Difference of Means

Samples showing statistical dependency between intermediate (key-related) data and power consumption.
Concept of Template Analysis

Ciphertext Keys → Open Sample → Measure → Learn (Profiling) Phase → Leakage Model

Closed Sample → Fixed Key → Measure → Attack (Exploitation) Phase → Analysis
Key recovery

AES key bytes 0-15

Number of traces

Key Byte Rank

riscure
The actual process

- Setup
- Acquisition
- Analysis
- Processing
Deep learning background
Deep Learning

Data with labels

cat

dog

cat

dog

cat

dog

cat

dog
Deep Learning

Data with labels

Train a machine to classify these data

Machine

Error function

BACK-PROPAGATION ALGORITHM

Cat (%)

Dog (%)

Train a machine to classify these data.
Deep Learning

Data with labels

Train a machine to classify these data

Test the machine on new data

Trained machine

Cat (%) Dog (%)
Deep Learning

- **Data with labels**
- **Train a machine to classify these data**
- **Test the machine on new data**
- **Is classification accuracy good enough?**
  - **No**
  - **Change parameters**
  - **Yes**
    - **We are done!**
- **Trained machine**
- **Machine = Deep Neural Network**
- **Cat**
Convolutional Neural Networks (CNNs)

**Input Layer**
(the size is equivalent to the number of samples)

Conv. Layers
(feature extractor + encoding)

**Output Layer**
(the size is equivalent to the number of classes)

Dense Layers
(classifiers)

The **convolutional layers** are able to detect the features independently of their positions.
Creating training/test/validation data sets

- Features
- Label
- HW = 5
- HW = 7
- HW = 3
- HW = 4

Leakage model
Classification

Trained Model

Trace (samples)

Key enumeration using output probabilities (Bayes)

Softmax ($\sum p_i = 1$)

- HW = 4: 0.05
- HW = 5: 0.15
- HW = 6: 0.65
- HW = 7: 0.08

Bayes

- $p_i$ probabilities
- 0.65
- 0.15
- 0.05
- 0.08
- 0.01
- 0.02
- 0.02
- 0.02
- 0.01

Trained Model

- Softmax
- $\sum p_i = 1$
Deep learning on side channels in practice
Step 1: Define initial hyper-parameters
Step 2: Make sure it’s capable of learning

- Increase the number of training traces and observe the training and validation accuracy
- Overfitting too fast?
  - Training accuracy: 100% | Validation accuracy: low
  - Neural network is too big for the number of traces and samples
Step 3: Make it generalize

Make sure the training accuracy/recall is increasing

Validation recall stays above the minimum threshold value = model is generalizing

0.111 = 1/9 (9 is the number of classes – HW of a byte)
Step 3: Make it generalize

Regularization techniques:

- L1, L2 (penalty applied to the weights)
- Dropout
- Data Augmentation (+traces)
- Early Stopping
Step 4: Key Recovery

In this analysis, we only need slightly-above coin flip accuracy!
Getting keys from the thingz!
Piñata AES-128 with misalignment
Bypassing Misalignment with CNNs

Neural Network: Input Layer > ConvLayer > 36 > 36 > 36 > Output Layer
Training/validation/test sets: 90000/5000/5000 traces of 500 samples
Leakage Model: HW of S-Box Out (Round 1) → 9 classes

Results for key byte 0:

Use Data Augmentation as regularization technique to improve generalization
Breaking protected ECC on Piñata

Supervised deep learning attack:
- Curve25519, Montgomery ladder, scalar blinding
- Messy signal
- Brute-force methods for ECC are needed if test accuracy < 100%
- Need to get (almost) all bits from one trace!
Breaking protected ECC

Unsupervised/Supervised Horizontal Attack: 60% success rate
Deep learning: 90% success rate
Deep learning (+ data augmentation): 99.4% success rate
Data augmentation: 25k → 200k traces.

Input (4000) 3 Conv Layers (10 filters) 4 Dense Layers (100 Neurons) Output (2 Classes)

RELU TANH SOFTMAX
Breaking AES with First-Order Masking

- Target published in 2013 (http://www.dpacontest.org/v4/)
- 40k traces available
- AES-256 (Atmel ATMega-163 smart card)
- Countermeasure: Rotating S-box Masking (RSM)
How does DPA contest V4 masking work?

- Masking is expensive in performance and memory
- Rotating mask helps by pre-computing masked S-boxes
Second order attack on masked implementations

• We cannot predict $Y_{Mj}$, but we can predict $Y_j$
• We cannot measure $Y_j$, but we can measure $Y_{Mj}$

$YM_1 = M_{(i+1)} \oplus Y_1$
$YM_2 = M_{(i+1)} \oplus Y_2$

$YM_1 \oplus YM_2 = Y_1 \oplus Y_2$

• By measuring two S-box output leakage points ($YM_1$ and $YM_2$), and subtracting their values, we get a value that corresponds to the leakage of $Y_1 \oplus Y_2$

→ second order attack

Cost:
• Must know or guess position of $YM_j$ leakage
• Attacking two S boxes → 2 sub keys → quadratic complexity
Breaking AES with First-Order Masking

**Neural Network:** Input Layer > ConvLayer > 50 > 50 > 50 > Output Layer

**Training/validation/test sets:** 36000/2000/2000 traces

**Leakage Model:** HW of S-Box Out (Round 1) → 9 classes

**Results for key byte 0:**

The processing of 8 traces is sufficient to recover the key.
1\textsuperscript{st} cool thing

DL is up there with dozens of SCA research teams
2nd cool thing

This shouldn’t work... why?
Identifying leakage
Where is the leak?

- Correlation Analysis
- Template Analysis
- Deep Learning

Correlation

POI

Visualization Techniques

?
Visualization

Object detection in images
Visualizing what neural networks learn from input data (proposed by Keras’ creator):
- Observe effect of ‘occlusion’ (input blocking)
- Create heat maps of class activations

Feature location
Activation path (illustration)

Input Data → Conv. → Pooling → Conv. → Pooling → Feature Map → Dense Layers → Output

Feature Extraction + Dimensionality Reduction

Feature Combination + Classification

HW = 5
Our method

Input Data → Conv. → Pooling → Conv. → Pooling → Feature Map → Dense Layers → Output

Feature Extraction + Dimensionality Reduction

Feature Combination + Classification

HW = 5
Results (unprotected target)

Raw trace

T-test (first round key byte)

Our visualization method

CPA succeeds

CPA fails
Digging deeper
Leakage Assessment (White-box)

40k Traces

- HW (Masked S-Box Out)
- ID (Masked S-Box Out)
- HW (S-Box Out)
- ID (S-Box Out)
Visualize the learned features (CNN)

Validation accuracy: 0.119

Key Byte 0 rank: 16
Optimized results

**Overfitting**
Very small generalization

<table>
<thead>
<tr>
<th>Validation Recall</th>
<th>Training Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.11987649</td>
<td>0.9889999</td>
</tr>
</tbody>
</table>

**No Overfitting**
Significant generalization

<table>
<thead>
<tr>
<th>Validation Recall</th>
<th>Training Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.21620668</td>
<td>0.3048745</td>
</tr>
</tbody>
</table>

- Key byte 0 found (rank 1) after: **9 traces!**
- Helping DL by sample selection improves quality
Confusion Matrix

<table>
<thead>
<tr>
<th></th>
<th>HW 0</th>
<th>HW 1</th>
<th>HW 2</th>
<th>HW 3</th>
<th>HW 4</th>
<th>HW 5</th>
<th>HW 6</th>
<th>HW 7</th>
<th>HW 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>HW 0</td>
<td>12</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>HW 1</td>
<td>0</td>
<td>20</td>
<td>152</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>HW 2</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>67</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>HW 3</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>36</td>
<td>152</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>HW 4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>HW 5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>HW 6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>HW 7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>HW 8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Expected Predicted

HW 0 → HW 2, HW 4, HW6
HW 1 → HW 3, HW 5
HW 2 → HW 2, HW 4, HW6
HW 3 → HW 3, HW 5
HW 4 → HW 2, HW 4, HW6
HW 5 → HW 3, HW 5
HW 6 → HW 2, HW 4, HW6
HW 7 → HW 3, HW 5
HW 8 → HW 2, HW 4, HW6

Imperfect leakage, but good enough
Wrapping up
Thoughts on Spectre & friends

• Spectre relies on 1d measurement: time
  • Plain old statistics probably better than DL

• Speculation: DL could be useful for an attacker that combines multiple micro-architectural side channels
Key takeaways

• If SCA is a concern, DL can exploit and identify leakage
• DL does SCA art + science and scales
• DL still requires humans, the bar is low, not yet at 0
• More automation needed to put a dent in insecurity
I want to learn more!

Deeplearningbook.org  riscure.com/training  bookstores  nostarch
References

- [http://www.deeplearningbook.org/](http://www.deeplearningbook.org/)
- S. Haykin, “Neural Networks and Learning Machines”.