Systems Security
Lab 2

Stjepan Picek
s.picek@tudelft.nl

Delft University of Technology, The Netherlands

May 12, 2021

TUDelft

Outline

Datasets

- https://drive.google.com/file/d/1p7ahj-r8vCTS9KLPGNvoUwNwqs6HSqOp/view?usp=sharing
- Optional: https://github.com/ANSSI-FR/ASCAD

Outline

TUDelft

Points of Interest

- Selecting good points of interest is very important since it can significantly change the computational complexity and performance of SCA.
- There are many ways how to select points of interest.
- Filter selection methods, wrapper selection methods, hybrid selection methods.

Points of Interest

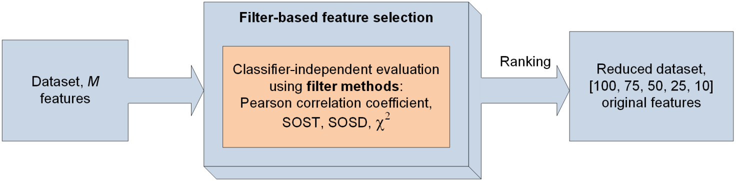


Figure: Filter methods

Points of Interest

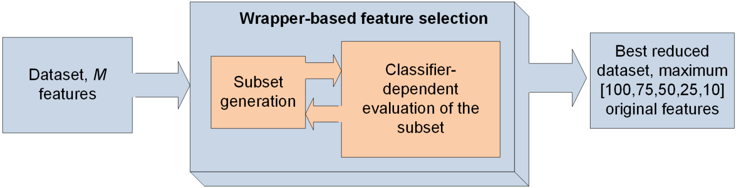


Figure: Wrapper methods

Points of Interest

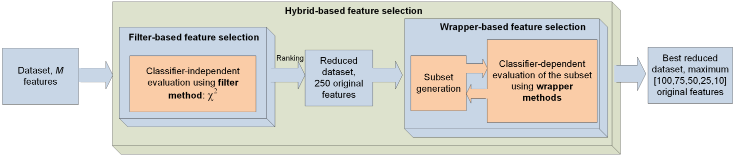


Figure: Hybrid methods

Points of Interest

- Pearson correlation coefficient measures linear dependence between two variables, x and y, in the range $[-1, 1]$, where 1 is the total positive linear correlation, 0 is no linear correlation, and -1 is the total negative linear correlation.

$$Pearson(x, y) = \frac{\sum_{i=1}^{N}((x_i - \bar{x})(y_i - \bar{y}))}{\sqrt{\sum_{i=1}^{N}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{N}(y_i - \bar{y})^2}}. \qquad (1)$$

- We calculate Pearson correlation for the target class variables HW and intermediate value, which consists of categorical values that are interpreted as numerical values.
- The features are ranked in descending order of the coefficient.

Points of Interest

- The sum of squared differences as a selection method, simply as:

$$SOSD(x, y) = \sum_{i,j>i} (\bar{x}_{y_i} - \bar{x}_{y_j})^2, \tag{2}$$

  where $\bar{x}_{y_i}$ is the mean of the traces where the model equals y_i.
- Because of the square, SOSD is always positive.
- Another advantage of using the square is to emphasize big differences in means.

Points of Interest

- SOST is the normalized version of SOSD and is thus equivalent to the pairwise Student's *t*-test:

$$SOST(x,y) = \sum_{i,j>i} \left((\bar{x}_{y_i} - \bar{x}_{y_j}) / \sqrt{\frac{\sigma_{y_i}^2}{n_{y_i}} + \frac{\sigma_{y_j}^2}{n_{y_j}}} \right)^2 \tag{3}$$

with n_{y_i} and n_{y_j} being the number of traces where the model equals to y_i and y_j, respectively.

Outline

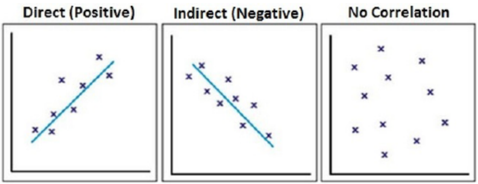
Correlation Power Analysis

- Write a leakage model for the power consumption.
- Obtain measurements of power consumption while device is running encryption over different plaintexts.
- Attack subparts of the key (divide and conquer approach):
    1. Consider all options for subkey. For each guess and trace, use plaintext and guessed subkey to calculate power consumption according to the model.
    2. Use the Pearson correlation to differentiate between the modeled and actual power consumption.
    3. Decide which subkey guess correlates best to the measured traces.
- Combine the best subkey guesses to obtain the secret key.

Pearson's Correlation

$$\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_x \sigma_y} = \frac{E[(X-\mu_x)(Y-\mu_y)]}{\sqrt{E[(X-\mu_x)^2]E[(Y-\mu_y)^2]}} \tag{4}$$

- If Y always increases when X increases, it is 1.
- If Y always decreases when X decreases, it is -1.
- If Y is independent of X, it is 0.



Pearson's Correlation

- Let us take N measurements m, where each measurement has z data points. $m_{n,j}$ is the data point j in trace n.
- We estimate the power consumption in each trace with the model.
- In total, there are S different subkeys to try.
- $h_{n,s}$ is the power estimate of the trace n with the subkey s.

Pearson's Correlation

- Next, we need to see how h and m correlate over N traces.

$$r_{s,j} = \frac{\sum_{n=1}^{N}\left[(h_{n,s} - \bar{h}_s)(m_{n,j} - \bar{m}_j)\right]}{\sqrt{\sum_{n=1}^{N}(h_{n,s} - \bar{h}_s)^2 \sum_{n=1}^{N}(m_{n,j} - \bar{m}_j)^2}} \tag{5}$$

Online CPA

- We do not need to take all the traces.
- Rather, we take a number of traces, observe results, add more traces, observe, etc.
- We do not want to calculate everything from scratch all the time.
- So, we run online CPA.

Online CPA

$$
\begin{aligned}
r_{i,j} &= \frac{\sum_{d=1}^{D}\left[\left(h_{d,i}-\overline{h_i}\right)\left(t_{d,j}-\overline{t_j}\right)\right]}{\sqrt{\sum_{d=1}^{D}\left(h_{d,i}-\overline{h_i}\right)^2 \sum_{d=1}^{D}\left(t_{d,j}-\overline{t_j}\right)^2}} \\
&= \frac{\sum_{d=1}^{D}\left[h_{d,i}t_{d,j}-t_{d,j}\overline{h_i}-h_{d,i}\overline{t_j}+\overline{t_j}\overline{h_i}\right]}{\sqrt{\sum_{d=1}^{D}\left(h_{d,i}{}^2-2\overline{h_i}h_{d,i}+\overline{h_i}^2\right)\sum_{d=1}^{D}\left(t_{d,j}{}^2-2\overline{t_j}t_{d,j}+\overline{t_j}^2\right)}} \\
&= \frac{\sum_{d=1}^{D}h_{d,i}t_{d,j}-\overline{h_i}\sum_{d=1}^{D}t_{d,j}-\overline{t_j}\sum_{d=1}^{D}h_{d,i}+D\overline{t_j}\overline{h_i}}{\sqrt{\left(\sum_{d=1}^{D}h_{d,i}{}^2-2\overline{h_i}\sum_{d=1}^{D}h_{d,i}+D\overline{h_i}^2\right)\left(\sum_{d=1}^{D}t_{d,j}{}^2-2\overline{t_j}\sum_{d=1}^{D}t_{d,j}+D\overline{t_j}^2\right)}} \\
&= \frac{\sum_{d=1}^{D}h_{d,i}t_{d,j}-\overline{h_i}\sum_{d=1}^{D}t_{d,j}-\frac{\sum_{d=1}^{D}t_{d,j}}{D}\sum_{d=1}^{D}h_{d,i}+D\frac{\sum_{d=1}^{D}t_{d,j}}{D}\frac{\sum_{d=1}^{D}h_{d,i}}{D}}{\sqrt{\left(\sum_{d=1}^{D}h_{d,i}{}^2-2\overline{h_i}\sum_{d=1}^{D}h_{d,i}+D\overline{h_i}\overline{h_i}\right)\left(\sum_{d=1}^{D}t_{d,j}{}^2-2\overline{t_j}\sum_{d=1}^{D}t_{d,j}+D\overline{t_j}\overline{t_j}\right)}} \\
&= \frac{\sum_{d=1}^{D}h_{d,i}t_{d,j}-\overline{h_i}\sum_{d=1}^{D}t_{d,j}}{\sqrt{\left(\sum_{d=1}^{D}h_{d,i}{}^2-2\overline{h_i}\sum_{d=1}^{D}h_{d,i}+D\overline{h_i}\frac{\sum_{d=1}^{D}h_{d,i}}{D}\right)\left(\sum_{d=1}^{D}t_{d,j}{}^2-2\overline{t_j}\sum_{d=1}^{D}t_{d,j}+D\overline{t_j}\frac{\sum_{d=1}^{D}t_{d,j}}{D}\right)}} \\
&= \frac{\sum_{d=1}^{D}h_{d,i}t_{d,j}-\overline{h_i}\sum_{d=1}^{D}t_{d,j}}{\sqrt{\left(\sum_{d=1}^{D}h_{d,i}{}^2-\overline{h_i}\sum_{d=1}^{D}h_{d,i}\right)\left(\sum_{d=1}^{D}t_{d,j}{}^2-\overline{t_j}\sum_{d=1}^{D}t_{d,j}\right)}} \\
&= \frac{\sum_{d=1}^{D}h_{d,i}t_{d,j}-\overline{h_i}\sum_{d=1}^{D}t_{d,j}}{\sqrt{\left(\sum_{d=1}^{D}h_{d,i}{}^2-\frac{\sum_{d=1}^{D}h_{d,i}}{D}\sum_{d=1}^{D}h_{d,i}\right)\left(\sum_{d=1}^{D}t_{d,j}{}^2-\frac{\sum_{d=1}^{D}t_{d,j}}{D}\sum_{d=1}^{D}t_{d,j}\right)}} \\
&= \frac{D\sum_{d=1}^{D}h_{d,i}t_{d,j}-\sum_{d=1}^{D}h_{d,i}\sum_{d=1}^{D}t_{d,j}}{D\sqrt{\frac{\left(\left(\sum_{d=1}^{D}h_{d,i}\right)^2-D\sum_{d=1}^{D}h_{d,i}^2\right)\left(\left(\sum_{d=1}^{D}t_{d,j}\right)^2-D\sum_{d=1}^{D}t_{d,j}^2\right)}{D^2}}} \\
&= \frac{D\sum_{d=1}^{D}h_{d,i}t_{d,j}-\sum_{d=1}^{D}h_{d,i}\sum_{d=1}^{D}t_{d,j}}{\sqrt{\left(\left(\sum_{d=1}^{D}h_{d,i}\right)^2-D\sum_{d=1}^{D}h_{d,i}^2\right)\left(\left(\sum_{d=1}^{D}t_{d,j}\right)^2-D\sum_{d=1}^{D}t_{d,j}^2\right)}}
\end{aligned}
$$

CPA Assignment (7 points)

- Implement guessing entropy metric.
- Attack the whole key with CPA.
- Implement online CPA.
- Implement guessing entropy metric.
- Attack the first key byte with online CPA.
- Use the Hamming weight leakage model.

Outline

Template Attack

- Using the copy of device, record a large number of measurements using different plaintexts and keys. We require information about every possible subkey value.
- Create a template of device's operation. A template is a set of probability distributions that describe what the power traces look like for many different keys.
- On device that is to be attacked, record a (small) number of measurements (called attack traces) using different plaintexts.
- Apply the template to the attack traces. For each subkey, record what value is the most likely to be the correct subkey.
- When using high-quality templates made from many traces, it is possible to attack a system with a single trace.
- Template attack can become unstable if there are more points of interest than measurements per value.

Template Attack

- It works under the assumption that the measurements are dependent among the D features given the target class.
- More precisely, given the vector of N observed attribute values for x, the posterior probability for each class value y is computed as:

$$p(Y = y|X = x) = \frac{p(Y = y)p(X = x|Y = y)}{p(X = x)}, \tag{6}$$

where $X = x$ represents the event that $X_1 = x_1 \wedge X_2 = x2 \wedge \ldots \wedge X_N = x_N$.

TUDelft

Template Attack

- Note that the class variable Y and the measurement X are not of the same type: Y is discrete while X is continuous. So, the discrete probability $p(Y = y)$ is equal to its sample frequency where $p(X = x|Y = y)$ displays a density function.
- Mostly in the state-of-the art, $p(X = x|Y = y)$ is assumed to rely on a (multivariate) normal distribution and is thus parameterized by its mean $\bar{x}_y$ and covariance matrix Σ_y:

$$p(X = x|Y = y) = \frac{1}{\sqrt{(2\pi)^D|\Sigma_y|}} e^{-\frac{1}{2}(x-\bar{x}_y)^T\Sigma_y^{-1}(x-\bar{x}_y)}. \tag{7}$$

TUDelft

Template Attack

- In practice, the estimation of the covariance matrices for each class value y can be ill-posed mainly due to an insufficient number of traces for each class.
- It is possible to use only one pooled covariance matrix to cope with statistical difficulties and thus a lower efficiency.
- Accordingly, Eq. (7) changes to

$$p(X = x|Y = y) = \frac{1}{\sqrt{(2\pi)^D|\Sigma|}} e^{-\frac{1}{2}(x-\bar{x}_y)^T\Sigma^{-1}(x-\bar{x}_y)}. \tag{8}$$

Pooled Template Attack Assignment (4 points)

- Implement pooled template attack.
- Show guessing entropy results for the whole key and the intermediate value leakage model.
- `https://wiki.newae.com/Template_Attacks#Applying_the_Template`
- `https://wiki.newae.com/Tutorial_B8_Profiling_Attacks_(Manual_Template_Attack)`
- Pooled template attack consist in averaging the covariance matrices of all models to create a unique covariance matrix.
- This technique generally results in a weaker attack, but helps to speed up computation when considering traces with many POIs.

Outline

Add Desync

Algorithm 2 Add Desynchronization.

function ADD_DESYNC($trace, desync\_level$)
  $new\_trace \leftarrow []$ ▷ container for new trace
  $level \leftarrow$randomNumber$(0, desync_level)$
  $i \leftarrow 0$
  while $i + level < len(trace)$ **do**
    $new\_trace[i] \leftarrow traces[i + level]$ ▷ add desynchronization to the trace
    $i \leftarrow i + 1$
  return new_trace

Machine Learning Attack Assignment (6 points)

- Use machine learning for profiling SCA on ChipWhisperer data.
- Use guessing entropy and attack all key bytes.
- Show results for both intermediate value and the Hamming weight leakage models.
- Add desynchronization countermeasure.
- Repeat machine learning attack.

Outline

TUDelft

Square-and-Multiply Algorithm

```
Input :  m, n ∈ ℕ, m < n, d = (d_{k−1} d_{k−2} . . . d_0)_2
Output : m^d mod n
a ← 1
while i = k − 1 to 0 do
    a ← a^2 mod n
    if d_i = 1 then
        a ← a × m mod n
    end if
end while
Return a
```

Square-and-Multiply Always Algorithm

Input : $m, n \in \mathbb{N}, m < n, d = (d_{k-1}d_{k-2}\dots d_0)_2$
Output : $m^d \, mod n$
$R_0 \leftarrow 1; R_1 \leftarrow m; i \leftarrow k-1; t \leftarrow 0$
while $i \geq 0$ **do**
    $R_0 \leftarrow R_0 \times R_t \ mod \ n$
    $t \leftarrow t \ XOR \ d_i; i \leftarrow i - 1 + t$
end while
Return R_0

Montgomery Ladder Algorithm

$Input: \ m, n \in \mathbb{N}, m < n, d = (d_{k-1} d_{k-2} \ldots d_0)_2$
$Output: m^d \, mod n$
$R_0 \leftarrow 1; R_1 \leftarrow m$
while $i = k - 1 \ to \ 0$ **do**
  $R_{1-d_i} \leftarrow R_0 \times R_1 \ mod \ n$
  $R_{d_i} \leftarrow R_{d_i}^2 \ mod \ n$
end while
$Return \ R_0$

Countermeasures Assignment (3 points)

- Implement square-and-multiply, square-and-multiply-always, and Montgomery ladder.
- Run timing tests on RSA.
- Report results.

General Rules

- Submit all the code.
- Write report.
- Again, you are in a role of security evaluator who is writing a report.
- Use only the key.npy file for the key (i.e., disregard other key files).