This work is distributed under the CC BY-SA license, unless stated otherwise.

Hardware Security

10. Physical side-channel attacks 2 (passive)

Ján Jančár

Summer 2026

Today

- Defences (recap)
- Expanding on DPA
- Higher-order attacks
- Profiled attacks
- Horizontal attacks
- Metrics & Tools

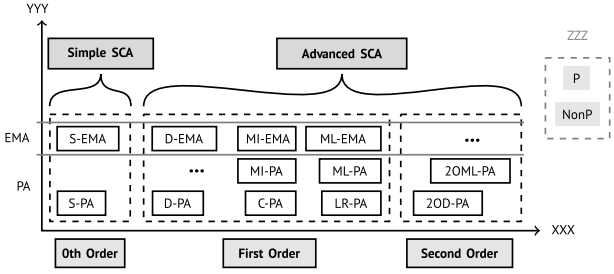
Defences

Last week

- Level
  - Algorithmic/Software
  - Hardware
  - Cell
- Type
  - Hiding
    - *Hide the leakage*
    - Time domain (mis-alignment)
    - Amplitude domain (higher noise, lower signal)
  - Masking
    - *Randomize the leakage by splitting*
    - Boolean
    - Arithmetic
  - Blinding
    - Randomize the leakage
    - Algorithm-specific

DPA

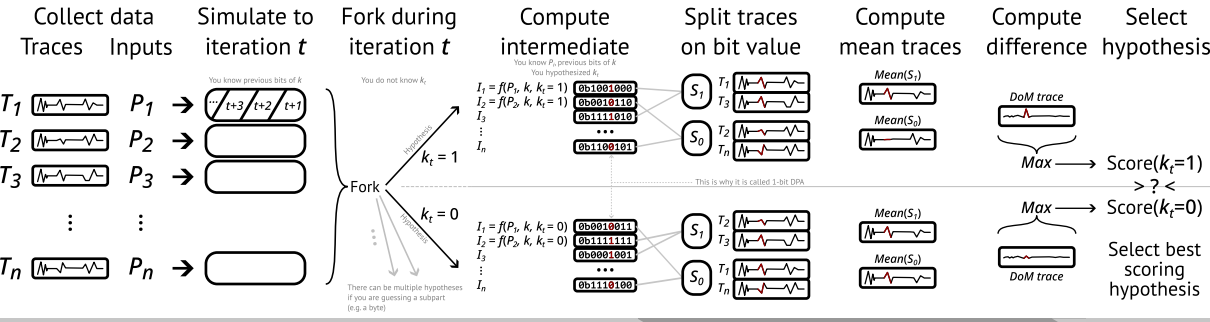
Extending DPA



- Single-bit DPA is the basic precursor to vertical advanced attacks
- Even higher-order attacks have origins in DPA

DPA

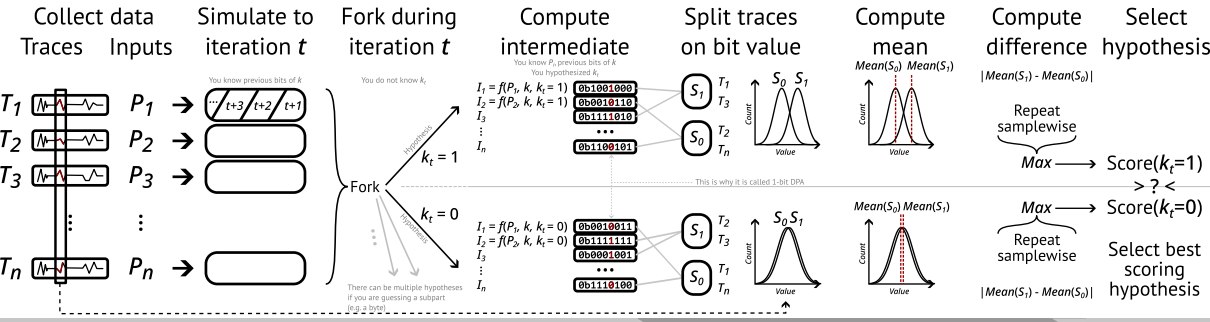
Single-bit DPA



DPA

Kocher99

Single-bit DPA (samplewise)



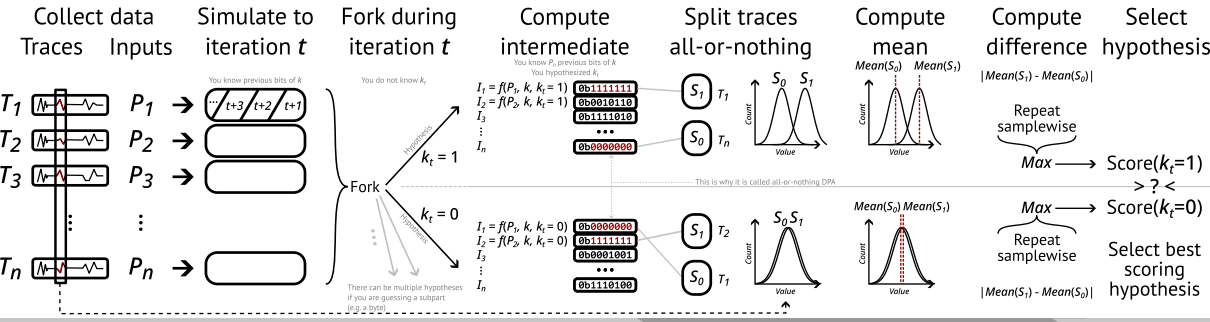
- Assumes linear leakage of a **single-bit**
  - Different bit value → Different mean
  - All other bits act as noise

- Uses difference-of-means distinguisher
- How could we include more bits in the split?
- Consider the extreme values `0xff` and `0x00`: all-or-nothing

DPA

Messerges99

All-or-nothing DPA (samplewise)

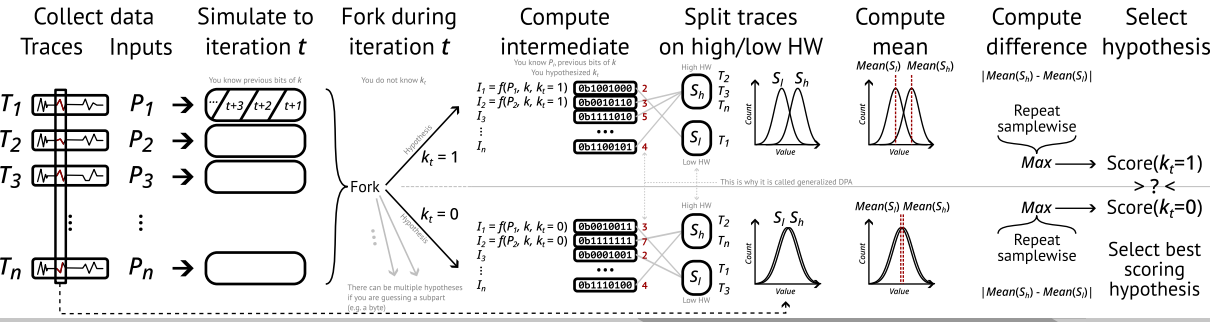


- Assumes linear leakage of all bits
  - Splits to all-one and all-zero groups
  - Ignores all the other Hamming weights
  - Less usable traces for mean computation
  - Other bits do not act as noise
- Uses difference-of-means distinguisher
- How could we waste less traces?
- Split based on mean-HW: above-mean and below-mean

DPA

Lemke04

Generalized DPA (samplewise)

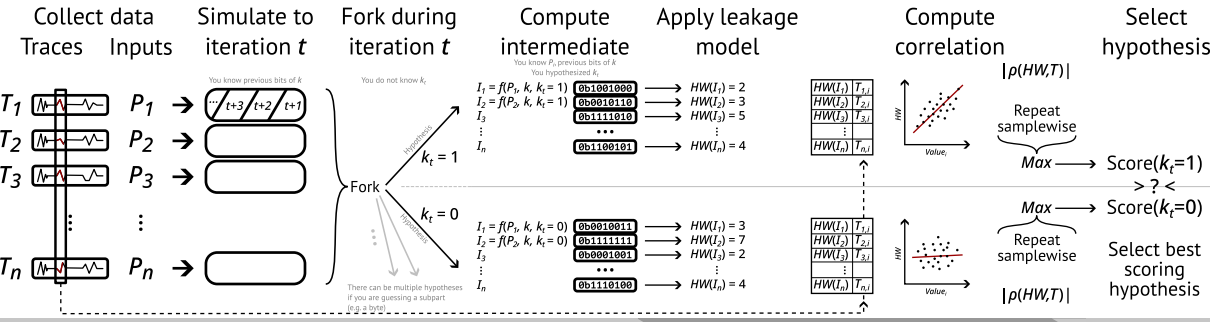


- Assumes linear leakage of all bits
  - Splits into above-mean HW and below-mean HW groups
  - Ignores mean-HW traces
  - Less waste than All-or-nothing DPA
  - Still some waste
- Uses difference-of-means distinguisher
- How can we utilize all the information, assuming a linear leakage of HW?
- What if we do not split but measure how well our predicted HW matches the leakage some other way?

CPA (Correlation Power Analysis)

Brier04

Samplewise

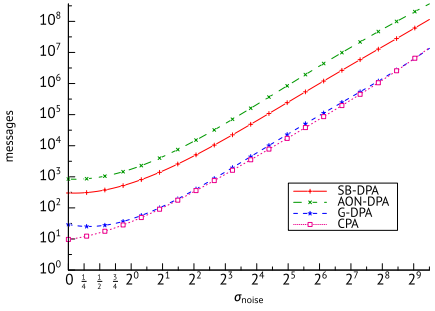


- Requires a leakage model
  - Assumption about the relationship between *intermediate value* and *observed leakage*
  - Hamming weight / Hamming distance
- Uses a correlation distinguisher
- Assumes linear relationship between prediction and leakage
- Computationally heavy
- Uses leakage more effectively, requires fewer traces than DPA

DPA and CPA

Comparing the attacks

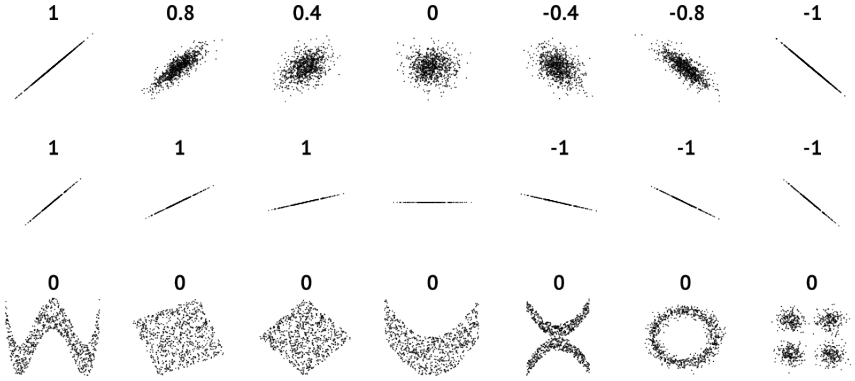
- Simulated $Leak = HW(x) + N(0, \sigma_{noise})$
- Asymptotically the same
- Difference most pronounced when noise is small
  - Graph shows the simulated noise
  - Does **not** show *algorithmic* noise (i.e. the other bits)
  - Algorithmic noise shown when $\sigma_{noise} = 0$



Messages necessary for success rate > 90%

CPA

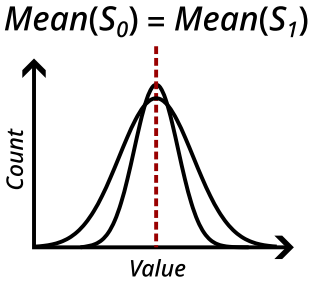
Limits



- All of the bottom row has correlation equal to zero
- All of the attacks so far targeted **linear** leakage
  - CPA could be used with a model that introduces non-linearity
  - But, attacker has to know such a model makes sense

MIA (Mutual Information Analysis)

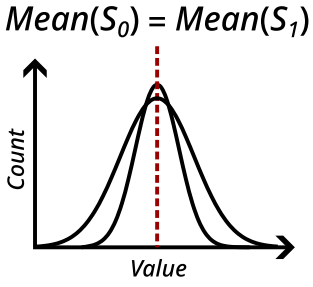
Non-linear leakage



- First-order boolean masked implementation
- $x = x_0 \oplus x_1$
- Implementation accidentally handles x_0 and x_1 at the same time
- $HW(x_0, x_1)$ leaks
  - $x = 0 \rightarrow HW(x_0, x_1) \in \{0, 2\} \rightarrow Mean = 1, Var = 1$
  - $x = 1 \rightarrow HW(x_0, x_1) = 1 \rightarrow Mean = 1, Var = 0$
- Means are the same, DPA/CPA will **not** work, what do we do?
- Leakage distributions are still different!

MIA (Mutual Information Analysis)

Non-linear leakage

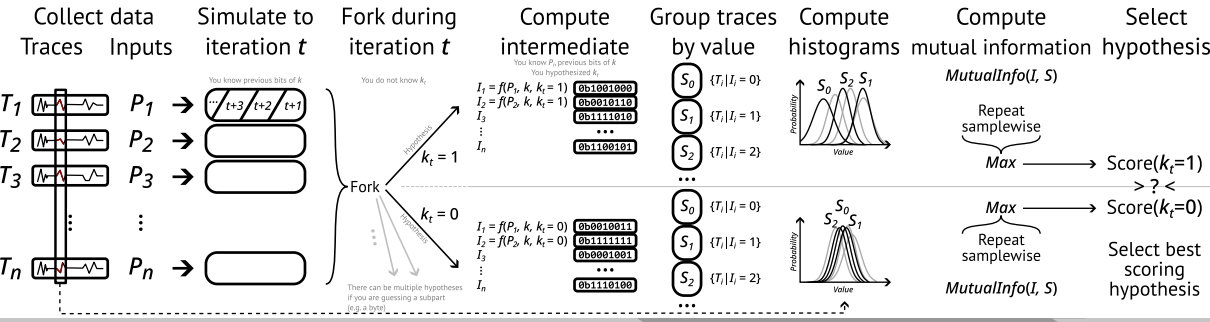


- First-order boolean masked implementation
- $x = x_0 \oplus x_1$
- Implementation accidentally handles x_0 and x_1 at the same time
- $HW(x_0, x_1)$ leaks
  - $x = 0 \rightarrow HW(x_0, x_1) \in \{0, 2\} \rightarrow Mean = 1, Var = 1$
  - $x = 1 \rightarrow HW(x_0, x_1) = 1 \rightarrow Mean = 1, Var = 0$
- Means are the same, DPA/CPA will **not** work, what do we do?
- We need a distinguisher that is more **generic**, captures **any** difference in distribution

MIA (Mutual Information Analysis)

Gierlichs08

Samplewise



- Most generic distinguisher
  - Does not assume anything about the leakage
- Computationally heavy

- Groups traces by intermediate value
  - Needs each group to be large enough for PDF-estimation (histogram/KDE)
  - Very trace-inefficient
- Uses a mutual information distinguisher

Higher-order attacks

Motivation

- First-order boolean masked implementation
- $x = x_0 \oplus x_1$
- Implementation correctly handles x_0 and x_1 at separate points in time (t and $t + i$)
  - $Leak(t) = HW(x_0) + N(0, \sigma) + \text{const}$
  - $Leak(t + i) = HW(x_1) + N(0, \sigma) + \text{const}$
- What do we do?
  - We need to somehow combine the leakage from the two time-points
- How?

Higher-order attacks

Motivation

- First-order boolean masked implementation
- $x = x_0 \oplus x_1$
- Implementation correctly handles x_0 and x_1 at separate points in time (t and $t + i$)
  - $Leak(t) = HW(x_0) + N(0, \sigma) + \text{const}$
  - $Leak(t + i) = HW(x_1) + N(0, \sigma) + \text{const}$
- What do we do?
  - Messerges00
- How?

Higher-order attacks

Motivation

- First-order boolean masked implementation
- $x = x_0 \oplus x_1$
- Implementation correctly handles x_0 and x_1 at separate points in time (t and $t + i$)
  - $Leak(t) = HW(x_0) + N(0, \sigma) + \text{const}$
  - $Leak(t + i) = HW(x_1) + N(0, \sigma) + \text{const}$
- What do we do?

- How?
  - Sum, product or difference?

Higher-order attacks

Motivation

- First-order boolean masked implementation
- $x = x_0 \oplus x_1$
- Implementation correctly handles x_0 and x_1 at separate points in time (t and $t + i$)
  - $Leak(t) = HW(x_0) + N(0, \sigma) + \text{const}$
  - $Leak(t + i) = HW(x_1) + N(0, \sigma) + \text{const}$
- What do we do?

- How?
  - Prouff09

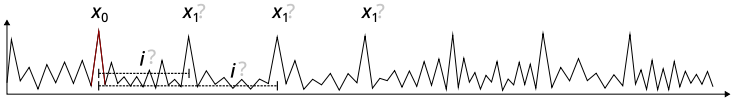
Higher-order attacks

Pre-processing function

- **Addition**: $Leak^{new}(t) = Leak(t) + Leak(t+i)$
  - $Leak^{new}(t) = HW(x_0) + HW(x_1) + 2\ N(0,\sigma) + 2$ const
  - Problem (No mean leakage)
    - Mean of $x = 0$ is the same as $x = 1$
    - We did not move the leakage into the means
- **Product**: $Leak^{new}(t) = Leak(t) \cdot Leak(t+i)$
  - $Leak^{new}(t) = (HW(x_0) + N(0,\sigma) + \text{const}) \cdot (HW(x_1) + N(0,\sigma) + \text{const})$
  - $HW(x_0) \cdot HW(x_1)$ term gives us leakage in the mean
  - Problem (Noise amplification)
    - $2\ N(0,\sigma) \cdot$ const will be very large (usually const $\gg$ data leakage)
- **Centered product**: $Leak^{new}(t) = (Leak(t) - Mean(t)) \cdot (Leak(t+i) - Mean(t+i))$
  - Centers trace by subtracting the mean trace, then does the product
  - Fixes the noise amplification by effectively subtracting the const
  - $N(0,\sigma)^2$ still remains
- **Centered Manhattan distance**:
  $Leak^{new}(t) = |Leak(t) - Mean(t) - Leak(t+i) + Mean(t+i)|$
  - Also centers trace, then does absolute difference
  - $Leak^{new}(t) = |HW(x_0) + N(0,\sigma) - HW(x_1) - N(0,\sigma)|$

Higher-order attacks

Explosion



- Attacker does not know the offset i
  - Worst-case: Has to combine all $n \times n$ samples and produce a matrix
- In practice, pick a bound b and compute an $n \times b$ matrix
- Correlation trace → Correlation matrix
- 👍 Break masking
- 👎 Computationally heavy

Profiled attacks

- **Scenario**: Attacker has access to a device that is the same model as the target, but can control the inputs (and key)
- Two phases
  1. Profiling phase (characterization, training, learning)
     - Attacker collects profiling traces
     - Attacker learns leakage distribution for all key subpart values
  2. Attack phase (matching)
     - Attacker collects attack traces (target device)
     - Attacker matches observed leakage to best fitting key subparts
- Two sub-categories
  - Template attacks
  - Machine learning & Deep learning

In assignment 2, you are actually in this scenario.

Profiled attacks

Chari02

Template attacks

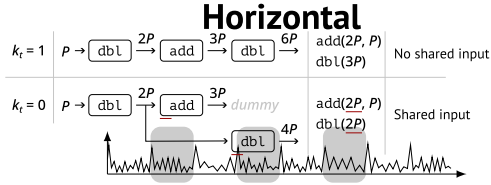
- Assumes leakage is a multivariate Gaussian distribution
- Learns Gaussian parameters
  - Mean trace and covariances for each plaintext p and key subpart k
- Maximum Likelihood strategy
  - Given observed attack sub-trace pick key hypothesis with maximum likelihood
- 👍 Theoretically optimal if assumption holds
- 👎 Has the multivariate Gaussian assumption
- ❓ Can we get rid of the assumption somehow?
- ❓ Model it as a classification problem?
  - ➡ Yes, then use ML techniques

Profiled attacks

Deep learning (DL/ML) attacks

- Prediction problem
  - Given (p, k) how will the observed leakage look
- Classification problem
  - Given observed leakage, what is the most likely (p, k)
- Template attacks learn distribution for prediction problem
  - It tends to be more complicated
- What if we use the classification problem directly
  - Just throw some deep neural network at it
  - Benadjila19
- 👍 No assumptions
- 👎 Need lots of data
- 👎 Black-box model, hard to interpret

Horizontal attacks



- Do not gather many traces, only one or few
- Instead, split a single trace into many sub-traces
  - Can be single-trace attacks
  - When is a single-trace attack advantageous?
  - Blinding/Ephemeral secret
- Usually more algorithm-specific
  - When you are looking at the whole trace you need to consider the algo
- Input collision in double-and-add always
  - If $k_t = 0$ then next iteration double computes on same input
  - Assumes attacker can detect field multiplication with repeated input (from a single trace)
- There are many more

Tools & Metrics

- Alignment
- Leakage assessment & analysis
  - SNR
  - Input, intermediate and output correlation
  - Trace average and standard deviation
  - TVLA
- Attack analysis
  - Success rate
  - Guessing entropy
  - Score traces (DoM traces, Correlation traces, ...)
  - Score evolution

Tools

Alignment

- Static
  1. Select a reference trace and a pattern in it
  2. Find pattern in all other traces
     - Sum of Absolute Differences, Correlation coefficient, Least squares, ...
  3. Align
- Dynamic
  - Dynamic Time Warp (Elastic Alignment)

Tools

Pre-processing

- Low/Band/High-pass filters
  - When you know where the leakage is (frequency-wise)
- Rolling mean (a.k.a. Moving average): window w
  - $T_i = \frac{1}{w} \sum_{j=0}^{w} T_{i-j}$
  - Visualization
- Downsampling
  - Average/Min/Max/Pick
  - Data efficiency
- Trace average and standard deviation
  - Visualization
  - Standard deviation shows noise (or misalignment)

Tools

Input, intermediate and output correlation

- Input and output correlation
  - Correlate traces with plaintext input & output
  - 👍 Target function must be in between
- Intermediate correlation
  - If you know the key
  - Known-key CPA
  - 👍 Structure of the trace & leakage

Tools

Test Vector Leakage Assesment (TVLA)

- Goodwill08 (Schneider15)
- Method for assesing the presence of leakage
- Assumes control over the target (key, plaintext, ciphertext, ...)
- **Idea**: Collect traces with two groups of different inputs, compare power consumption statistically
- Similar to DPA, uses Welch's t-test
  - Statistical test for equality of means (two-sample)
- Two variants
  - Non-specific test (fixed vs random)
    - One group: Fixed data
    - Other group: Random data
  - Specific test (fixed vs fixed)
    - Uses an intermediate value bit for the split
- Produces a TVLA trace
  - Like a DoM trace

Metrics

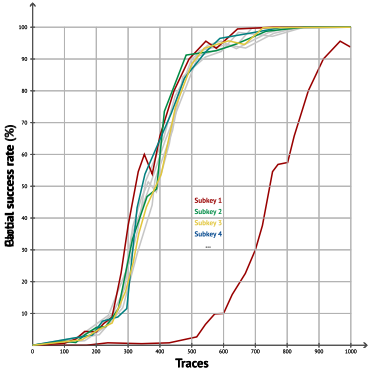
Signal-to-Noise Ratio (SNR)

- $SNR = \frac{Var(Signal)}{Var(Noise)}$
- Computed samplewise
- How much signal is there compared to the noise?
- Where is the signal (leakage)?
- In practice
  - Collect traces with random inputs
  - Compute intermediate value
  - Group by bit/all-or-nothing/HW
  - $Var(Signal)$ is the variance among mean group traces
  - $Var(Noise)$ is the variance within the group traces

Metrics

Success rate

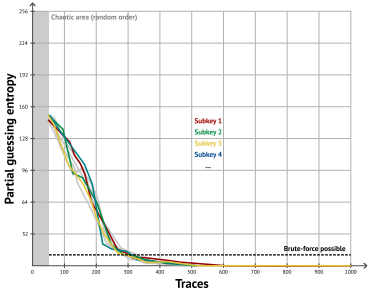
- Run attack many times, measure success rate
- Usually over number of traces
  - Sometimes over noise or other tested variables
- **Partial success rate**
  - Recover a secret subpart
- **Global success rate**
  - Completely recover the secret
  - Lower than partial success rate
  - Why?
  - Requires that **all** subparts are correct, probabilities multiply



Metrics

Guessing entropy

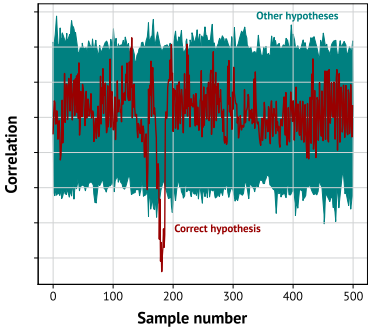
- Attack success/fail is too binary
- Distinguisher gives us a score (confidence) for each hypothesis
- Contains information even if top scoring hypothesis is not correct
- **Partial guessing entropy (PGE)**
  - Order by score
  - At which index is the correct key subpart found?
- Assumes a known-key scenario
- Runs attack many times
- *How much brute-forcing is left*
  - Have to brute-force up-to worst PGE
  - One subpart 1st, other subpart 3rd
    - Have to search all up-to three deep: Can only verify a full-key
- Usually over number of traces



Metrics

Score traces

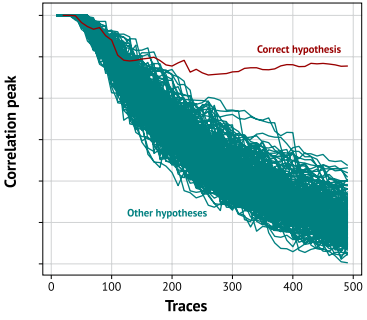
- Plot the score traces (DoM, Correlation) from the distinguisher for all hypotheses
- How tall is the (correct) peak above the rest?
- Where is the (correct) peak?
- Are there other (incorrect) peaks?



Metrics

Score evolution

- Assumes a known-key scenario
- Runs attack many times
- How does the peak height (max score) evolve with increasing number of traces?



Attack roadmap

There is no silver bullet.

- Mis-alignment → Re-alignment
- Low signal / High noise → Stronger first-order attacks
  - SB-DPA → G-DPA → CPA/MIA
  - Template attacks or Deep learning
- Masking → Higher-order attacks
- Blinding / Ephemeral secrets → Horizontal attacks
  - Horizontal / Collision / Algorithm-specific attacks

Takeaways

- How to break countermeasures with more advanced attacks
- There is no “fully secure” implementation
  - The goal is to make attacks impractical (hard & expensive)
- What tools and metrics can help in mounting attacks and analyzing targets

Recommended reading

- For **this** lecture
  - Skim Power Analysis Attacks - Chapter 8 - Attacks on Hiding
  - Skim Power Analysis Attacks - Chapter 10 - Attacks on Masking
- For the **next** lecture
  - Skim Embedded Cryptography 1 - Chapter 9 - Introduction to Fault Injection Attacks

References

- All-or-nothing DPA
- Generalized DPA
- CPA
- From DPA to CPA
- DCA (not covered)
- LRA (not covered)
- MIA
- Higher-order pre-processing
- Template attacks
- Efficient template attacks
- TVLA
- TVLA properties