

Power Analysis Attacks

Kostas Papagiannopoulos

University of Amsterdam

kostaspap88@gmail.com // kpcrypto.net

Contents

Introduction

The Power Side-Channel

Correlation Power Analysis

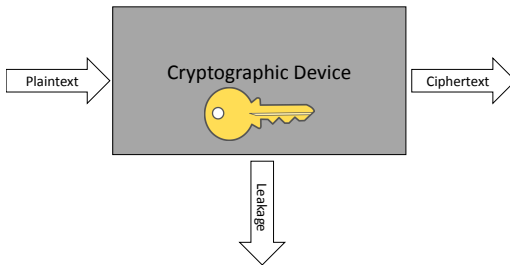
Introduction

Introduction



- ▶ The cryptographic algorithm (e.g. AES) is a black box, parameterized with key K , that turns plaintext P to ciphertext C
- ▶ Analyzing the cipher's security in the **blackbox scenario** relates to classical cryptanalysis techniques like linear and differential cryptanalysis
- ▶ "Can you recover the secret key by observing plaintext/ciphertext pairs?"

Introduction



- ▶ "Can you derive the secret key by observing plaintext/ciphertext pairs **and a side-channel?**"
- ▶ Secure algorithms under the blackbox scenario may not be secure under the greybox scenario

e.g. AES and DES are resistant to differential cryptanalysis but can be broken easily with a side-channel attack

The Power Side-Channel

Power side-channel: CMOS leakage

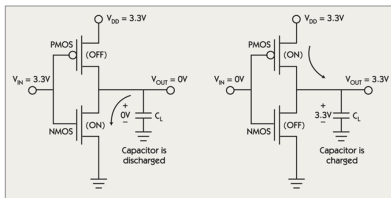


Fig. 1 Dynamic power in a CMOS inverter.

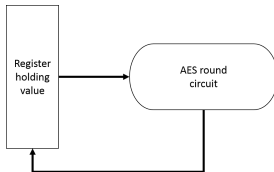
- ▶ CMOS is the most popular circuit style and exhibits several types of leakage
- ▶ The most relevant for side-channel attacks is the charge and discharge of the CMOS load capacitance a.k.a dynamic power consumption
- ▶ Dynamic power consumption (P_{dyn}) is produced by CMOS transitions from state 0 to 1 and from state 1 to 0
- ▶ Thus a power analysis attack explores the fact that the dynamic power consumption relates to the bitflips i.e. transitions $0 \rightarrow 1$ and $1 \rightarrow 0$

Power side-channel: Modeling the leakage

- ▶ **Starting point:** dynamic power consumption depends on bit transitions thus we use Hamming distance (HD) and the Hamming weight (HW) to model the leakage
- ▶ $HD(a, b)$ = no. of transitions from a to b
 $HD(10011, 01101) = 4$
- ▶ $HW(a)$ = no. 'ones' in a
 $HW(10011) = 3$
- ▶ $HD(a, b) = HW(a \oplus b)$

Power side-channel: Modeling the leakage

- ▶ The **Hamming distance** (HD) model counts the number of $0 \rightarrow 1$ and $1 \rightarrow 0$ transitions, assuming that they are equivalent
- ▶ **Example 1:** Assume a hardware register R storing the result of an AES round. The register initially contains value v_0 and gets overwritten with value v_1

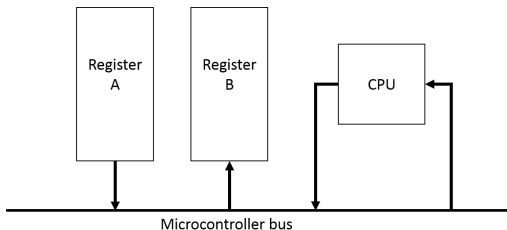


- ▶ The power consumption because of the register transition $v_0 \rightarrow v_1$ is related to the number of bit flips that occurred
- ▶ Thus it can be modeled as $HD(v_0, v_1)$
- ▶ It's common to see the HD model in hardware implementations (FPGA, ASIC)

Power side-channel: Modeling the leakage

- ▶ **Example 2:** In a microcontroller, assume register A with value v_0 and an assembly instruction that moves the contents of register A to register B

`mov rB, rA`

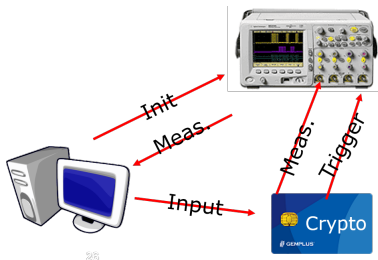


- ▶ In general-purpose processors the instruction will transfer value v_0 from register A to B via the CPU, using the bus
- ▶ In several cases the bus is very leaky component and it is also precharged at all bits being zeros or all being ones
- ▶ The power consumption of the assembly instruction can be modeled as:

$$HD(\text{precharge}, v_0) = HW(\text{precharge} \oplus v_0) = HW(0 \oplus v_0) = HW(v_0)$$

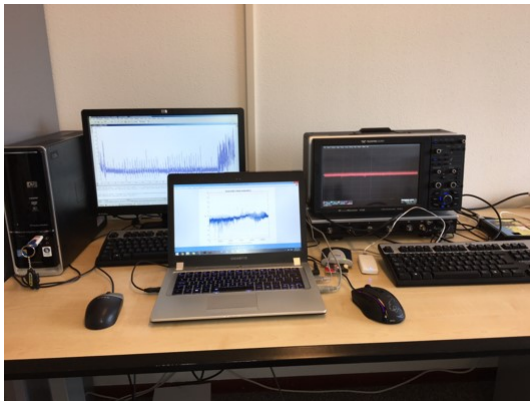
- ▶ It's common to see HW leakages in software implementations (AVR/ARM/RISC-V microcontrollers)

Power side-channel: Measurement setup



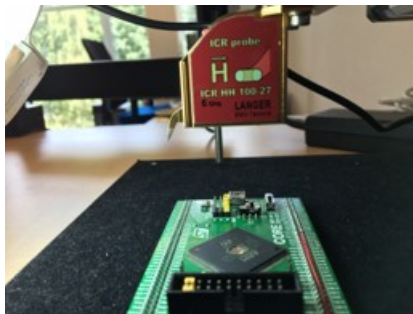
- Usually power measurements requires physical proximity to the device and customized measurement equipment (resistor, oscilloscope)

Power side-channel: Measurement setup



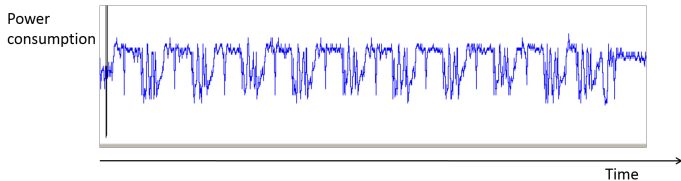
- Power measurement setup

Power side-channel: Measurement setup



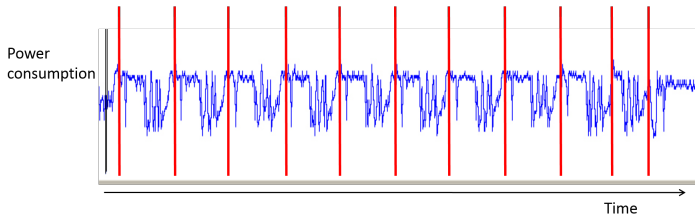
- ▶ Electromagnetic emission measurement setup, using microprobe and decapsulated chip

Power side-channel: AES rounds



- ▶ Power consumption leakage of an AES cipher implementation on an AVR microcontroller
- ▶ How many rounds are executed?

Power side-channel: AES rounds



- ▶ We see 10 repeating patterns thus it's AES-128
- ▶ We can even notice that the last round is smaller due to the lack of MixColumns
- ▶ In general, we can use the power side-channel to reverse engineer certain implementation details such as the cipher, its version, assembly instructions used etc.

Power side-channel: RSA Square and Multiply

- ▶ Software code for RSA modular exponentiation

Input: integers x , e , n , e has length l

Output: $x^e \bmod n$

Variable e is the secret

Process ModularExponentiation(x , e , n , l)

```
r = 1;
```

```
for j=l-1 down to 0
```

```
    r = r^2 mod n // square
```

```
    if (bit j of e) == 1 // if the key bit is 1
```

```
        r = r*x mod n //multiply
```

```
    endif
```

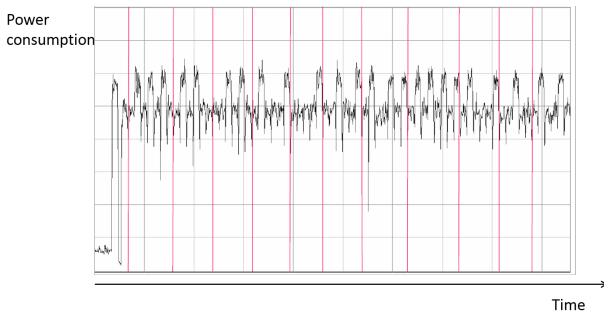
```
end
```

```
return r
```

```
EndProcess
```

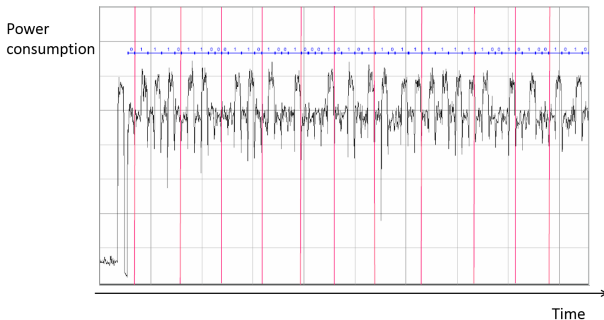
- ▶ Do you already see a timing attack? Yes! The exponent-dependent branch is causing it!
- ▶ Does this also imply a power attack?

Power side-channel: RSA Square and Multiply



- Can you find the exponent bits by visual inspection of the patterns?

Power side-channel: RSA Square and Multiply



- ▶ Square and Multiply (bit==1) are lengthier operations than Square only (bit==0)
- ▶ Multiplications are often more power consuming compared to Squarings

Correlation Power Analysis

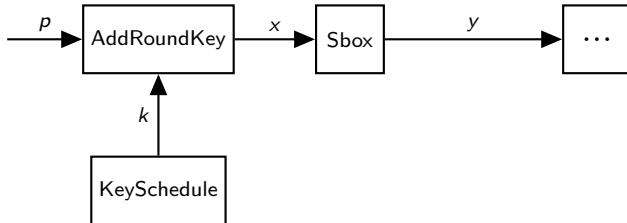
Correlation Power Analysis (CPA)

- ▶ One of the most popular side-channel attacks:
used by academics, secure chip manufacturers and evaluation labs
- ▶ Tries to recover the secret key of a cipher using a large number of repeated power measurements
- ▶ Uses simple statistical correlation to distinguish the correct key from incorrect key candidates
- ▶ CPA is carried out in 5 steps

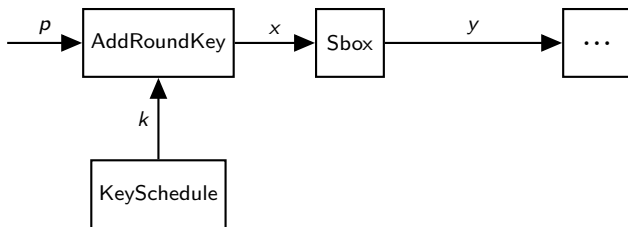
CPA: Step 0

Step 0: To perform CPA we must know the cipher's structure

- ▶ The figure below shows a small part of the 1st round of AES-128
- ▶ All values (p , k , x , y) have size of 8 bits
- ▶ $x = p \oplus k$
- ▶ $y = S(x)$



CPA: Step 1



Step 1: Choose an intermediate value v of the cipher to attack

1. The value v must be a function of the plaintext and the key i.e. $v = f(p, k)$
A common choice for v is the Sbox output i.e. $v = y = S(p \oplus k)$
Note that recovering y is equivalent to recovering k since:
 $x = S^{-1}(y)$, the Sbox S is one-to-one and p is known, thus $k = x \oplus p$
2. The targeted value v must be fairly small
This part of AES works with 8-bit values which is manageable
3. Throughout the attack the key k on the device must remain constant
4. Throughout the attack the plaintext p is random

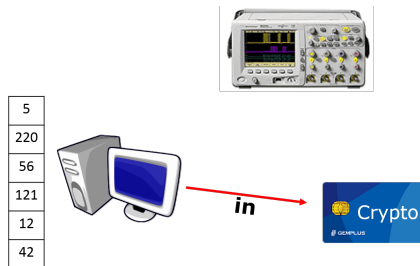
CPA: Step 2

Step 2: Measure the power consumption of the device

- ▶ In the measurement setup we generate randomly n 8-bit plaintexts p_i
Typically the number of traces n is large i.e. thousands to millions
- ▶ Store the plaintexts in the input vector

$$\mathbf{p} = [p_1, p_2, p_3, \dots, p_n]$$

- ▶ Send the plaintexts to our device

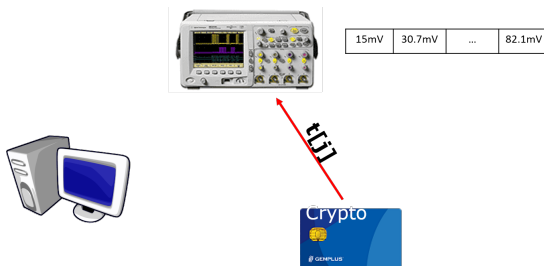


CPA: Step 2

Step 2: Measure the power consumption of the device

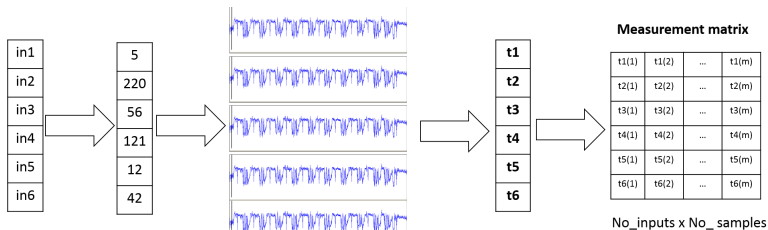
- ▶ While encrypting every p_i , we measure the power consumption
- ▶ Every encryption results in a digitized signal over time i.e. the trace \mathbf{t}_i
- ▶ Every trace \mathbf{t}_i contains m time samples i.e.

$$\mathbf{t}_i = [t_i[1], t_i[2], t_i[3], \dots t_i[m]]$$



CPA: Step 2

- ▶ Capturing n power traces with m time samples per trace results in the **measurement matrix M**



- ▶ Notation: measurements vs. traces vs. samples vs. features

CPA: Step 2

Measurement matrix M

$$\begin{array}{c} \text{no. traces} \end{array} \begin{bmatrix} t_1[1] & t_1[2] & t_1[3] & \dots & t_1[m] \\ t_2[1] & t_2[2] & t_2[3] & \dots & t_2[m] \\ t_3[1] & t_3[2] & t_3[3] & \dots & t_3[m] \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ t_n[1] & t_n[2] & t_n[3] & \dots & t_n[m] \end{bmatrix} \begin{array}{c} \text{no. samples} \end{array}$$

- ▶ At some point in time the device should process the intermediate $v = y = S(p \oplus k_{true})$ that we target
- ▶ Thus some time samples should be useful for the attack while other time samples will be unrelated
- ▶ Thus just a few columns of the measurement matrix are useful for the attack

CPA: Step 3

Step 3: Compute the hypothetical intermediate values

- ▶ In our device $v = y = S(in \oplus k_{true})$, but k_{true} is unknown
- ▶ For any plaintext p_i we can compute the value y_i for all possible key candidates $k \in \{0, 1, \dots, 255\}$

Compute the hypothetical intermediates

for i = 1 until n

 for j = 1 until 256

 k = j - 1

 VP(i,j) = S(p(i) xor k)

Value-prediction matrix

in1		k=0	k=1	...	k=255
in2		Sbox(in1 XOR 0)	Sbox(in1 XOR 1)	...	Sbox(in1 XOR 255)
in3		Sbox(in2 XOR 0)	Sbox(in2 XOR 1)	...	Sbox(in2 XOR 255)
in4		Sbox(in3 XOR 0)	Sbox(in3 XOR 1)	...	Sbox(in3 XOR 255)
in5		Sbox(in4 XOR 0)	Sbox(in4 XOR 1)	...	Sbox(in4 XOR 255)
in6		Sbox(in5 XOR 0)	Sbox(in5 XOR 1)	...	Sbox(in5 XOR 255)
		Sbox(in6 XOR 0)	Sbox(in6 XOR 1)	...	Sbox(in6 XOR 255)

No_inputs x No_keys

CPA: Step 3

Value-prediction matrix VP

$$\begin{array}{c} \text{no. traces} \end{array} \begin{bmatrix} S(p_1 \oplus 0) & S(p_1 \oplus 1) & \dots & S(p_1 \oplus 255) \\ S(p_2 \oplus 0) & S(p_2 \oplus 1) & \dots & S(p_2 \oplus 255) \\ S(p_3 \oplus 0) & S(p_3 \oplus 1) & \dots & S(p_3 \oplus 255) \\ \vdots & \vdots & \ddots & \vdots \\ S(p_n \oplus 0) & S(p_n \oplus 1) & \dots & S(p_n \oplus 255) \end{bmatrix}$$

no. key candidates

- ▶ Only one of the 256 key candidates is correct
- ▶ Thus only one of the columns in the value-prediction matrix is correct
- ▶ Do you see why we need to target fairly small intermediate values?
⇒ the size of the value affects the number of the columns in the VP matrix

CPA: Step 4

Step 4: Apply the leakage model

- ▶ We map the hypothetical intermediate values to hypothetical power consumption values, producing the **power-prediction** matrix

Power-prediction matrix PP

$$\begin{array}{c} \text{no. traces} \end{array} \begin{bmatrix} \text{HW}(S(p_1 \oplus 0)) & \text{HW}(S(p_1 \oplus 1)) & \dots & \text{HW}(S(p_1 \oplus 255)) \\ \text{HW}(S(p_2 \oplus 0)) & \text{HW}(S(p_2 \oplus 1)) & \dots & \text{HW}(S(p_2 \oplus 255)) \\ \text{HW}(S(p_3 \oplus 0)) & \text{HW}(S(p_3 \oplus 1)) & \dots & \text{HW}(S(p_3 \oplus 255)) \\ \vdots & \vdots & \ddots & \vdots \\ \text{HW}(S(p_n \oplus 0)) & \text{HW}(S(p_n \oplus 1)) & \dots & \text{HW}(S(p_n \oplus 255)) \end{bmatrix}$$

no. key candidates

- ▶ A common choice for the leakage model is the Hamming weight of a value
- ▶ Various models may be applicable: Hamming distance, linear models, etc.

CPA: Step 5

Step 5: Compare the power-prediction matrix PP to the measurement matrix M

- ▶ We will use the Pearson's correlation coefficient for this comparison
- ▶ We will compare the columns of the power-prediction matrix PP with columns of the measurement matrix M

$$\rho_{a,b} = \frac{\sum_{i=1}^n \left(a_i - \frac{1}{n} \sum_{i=1}^n a_i \right) \left(b_i - \frac{1}{n} \sum_{i=1}^n b_i \right)}{\sqrt{\sum_{i=1}^n \left(a_i - \frac{1}{n} \sum_{i=1}^n a_i \right)^2} \sqrt{\sum_{i=1}^n \left(b_i - \frac{1}{n} \sum_{i=1}^n b_i \right)^2}}$$

- ▶ Uncorrelated columns will result in $\rho_{a,b} \approx 0$
- ▶ Correlated columns will result in $\rho_{a,b} \approx \pm 1$
- ▶ CPA does not care whether the correlation is positive or negative thus it uses absolute correlation $|\rho_{a,b}|$

CPA: Step 5

- ▶ Say that we know already that the correct key k_{true} is equal to 42
- ▶ Say that we know already that the intermediate value v is processed and leaks during time sample 500
- ▶ Let's correlate column 1 of PP and column 1 of M

$$PP = \begin{bmatrix} \text{HW}(S(p_1 \oplus 0)) & \text{HW}(S(p_1 \oplus 1)) & \dots & \text{HW}(S(p_1 \oplus 255)) \\ \text{HW}(S(p_2 \oplus 0)) & \text{HW}(S(p_2 \oplus 1)) & \dots & \text{HW}(S(p_2 \oplus 255)) \\ \text{HW}(S(p_3 \oplus 0)) & \text{HW}(S(p_3 \oplus 1)) & \dots & \text{HW}(S(p_3 \oplus 255)) \\ \vdots & \vdots & \ddots & \vdots \\ \text{HW}(S(p_n \oplus 0)) & \text{HW}(S(p_n \oplus 1)) & \dots & \text{HW}(S(p_n \oplus 255)) \end{bmatrix}$$

$$M = \begin{bmatrix} t_1[1] & t_1[2] & t_1[3] & \dots & t_1[m] \\ t_2[1] & t_2[2] & t_2[3] & \dots & t_2[m] \\ t_3[1] & t_3[2] & t_3[3] & \dots & t_3[m] \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ t_n[1] & t_n[2] & t_n[3] & \dots & t_n[m] \end{bmatrix}$$

- ▶ We tried to correlate the wrong key candidate ($k = 0$) with the wrong point in time (sample = 1)
- ▶ Thus $|\rho_{pp1, m1}| \approx 0$

CPA: Step 5

- ▶ Let's correlate column 43 of PP and column 200 of M
- ▶ i.e. correlate the correct key candidate ($k = 42$) with the wrong point in time (sample = 200)

$$PP = \begin{bmatrix} \text{HW}(S(p_1 \oplus 0)) & \dots & \text{HW}(S(p_1 \oplus 42)) & \dots & \text{HW}(S(p_1 \oplus 255)) \\ \text{HW}(S(p_2 \oplus 0)) & \dots & \text{HW}(S(p_2 \oplus 42)) & \dots & \text{HW}(S(p_2 \oplus 255)) \\ \text{HW}(S(p_3 \oplus 0)) & \dots & \text{HW}(S(p_3 \oplus 42)) & \dots & \text{HW}(S(p_3 \oplus 255)) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \text{HW}(S(p_n \oplus 0)) & \dots & \text{HW}(S(p_n \oplus 42)) & \dots & \text{HW}(S(p_n \oplus 255)) \end{bmatrix}$$

$$M = \begin{bmatrix} t_1[1] & \dots & t_1[200] & \dots & t_1[m] \\ t_2[1] & \dots & t_2[200] & \dots & t_2[m] \\ t_3[1] & \dots & t_3[200] & \dots & t_3[m] \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ t_n[1] & \dots & t_n[200] & \dots & t_n[m] \end{bmatrix}$$

- ▶ Thus $|\rho_{pp_{43}, m_{200}}| \approx 0$

CPA: Step 5

- ▶ Let's correlate column 1 of PP and column 500 of M
- ▶ i.e. correlate the wrong key candidate ($k = 0$) with the correct point in time (sample = 500)

$$PP = \begin{bmatrix} \text{HW}(S(p_1 \oplus 0)) & \text{HW}(S(p_1 \oplus 1)) & \dots & \text{HW}(S(p_1 \oplus 255)) \\ \text{HW}(S(p_2 \oplus 0)) & \text{HW}(S(p_2 \oplus 1)) & \dots & \text{HW}(S(p_2 \oplus 255)) \\ \text{HW}(S(p_3 \oplus 0)) & \text{HW}(S(p_3 \oplus 1)) & \dots & \text{HW}(S(p_3 \oplus 255)) \\ \vdots & \vdots & \ddots & \vdots \\ \text{HW}(S(p_n \oplus 0)) & \text{HW}(S(p_n \oplus 1)) & \dots & \text{HW}(S(p_n \oplus 255)) \end{bmatrix}$$

$$M = \begin{bmatrix} t_1[1] & \dots & t_1[500] & \dots & t_1[m] \\ t_2[1] & \dots & t_2[500] & \dots & t_2[m] \\ t_3[1] & \dots & t_3[500] & \dots & t_3[m] \\ \vdots & \dots & \vdots & \ddots & \vdots \\ t_n[1] & \dots & t_n[500] & \dots & t_n[m] \end{bmatrix}$$

- ▶ Thus $|\rho_{pp_1, m_{500}}| \approx 0$

CPA: Step 5

- ▶ Let's correlate column 43 of PP and column 500 of M
- ▶ i.e. correlate the correct key candidate ($k = 42$) with the correct point in time (sample = 500)

$$PP = \begin{bmatrix} \text{HW}(S(p_1 \oplus 0)) & \dots & \text{HW}(S(p_1 \oplus 42)) & \dots & \text{HW}(S(p_1 \oplus 255)) \\ \text{HW}(S(p_2 \oplus 0)) & \dots & \text{HW}(S(p_2 \oplus 42)) & \dots & \text{HW}(S(p_2 \oplus 255)) \\ \text{HW}(S(p_3 \oplus 0)) & \dots & \text{HW}(S(p_3 \oplus 42)) & \dots & \text{HW}(S(p_3 \oplus 255)) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \text{HW}(S(p_n \oplus 0)) & \dots & \text{HW}(S(p_n \oplus 42)) & \dots & \text{HW}(S(p_n \oplus 255)) \end{bmatrix}$$

$$M = \begin{bmatrix} t_1[1] & \dots & t_1[500] & \dots & t_1[m] \\ t_2[1] & \dots & t_2[500] & \dots & t_2[m] \\ t_3[1] & \dots & t_3[500] & \dots & t_3[m] \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ t_n[1] & \dots & t_n[500] & \dots & t_n[m] \end{bmatrix}$$

- ▶ Thus $|\rho_{PP_{43}, M_{500}}| \approx 1$

CPA: Step 5

CPA attack

```
# Compute the VP and PP
for i = 1 until n
    for j = 1 until 256
        k = j - 1
        VP(i,j) = S(p(i) xor k)
        PP(i,j) = HW(VP(i,j))

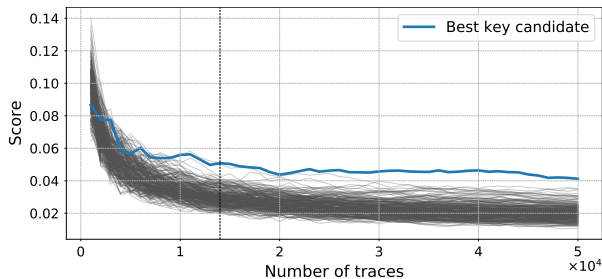
# Compute the column-wise correlation
for i = 1 until m
    for j = 1 until 256
        score(i,j) = abs(correlation(M(:,i), PP(:,j)))

# Compute the highest score per key
for j = 1 until 256
    key_score(j) = max(score(:,j))

# Sort the keys using their scores and select the top candidate
[sorted_scores, sorted_indexes] = sort(key_score)
top_candidate = sorted_indexes - 1
```

CPA: Step 5

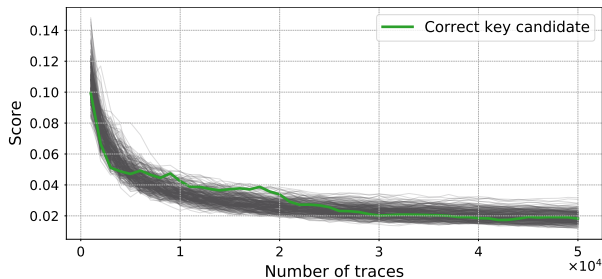
Score vs. number of traces plot



- Observe that increasing the traces converges to a top key candidate
- We highlight the top key with blue
- This is a successful attack

CPA: Step 5

Score vs. number of traces plot



- ▶ Observe that increasing the traces does not help key recovery
- ▶ We highlight the correct (unknown) key with green (evaluation setting)
- ▶ This is an unsuccessful attack

Advantages of CPA

- ▶ Easy and quick to implement:
⇒ just code two nested `for` loops
- ▶ Relies on simple statistics:
⇒ estimating correlation needs the mean, variance and covariance, all of which can be computed efficiently
- ▶ The workload can be partitioned and parallelized:
⇒ split the dataset into blocks of traces and/or samples, distribute the computation and aggregate the results
- ▶ The attack can be incremental and memoryless:
⇒ memory bottlenecks are avoided by custom formulas for online computation of statistical moments
- ▶ The attack is fairly generic:
⇒ the simple Hamming weight leakage model assumes very little and can be applied when little information is known about the device

Disadvantages of CPA

- ▶ The generic Hamming weight model may not exploit all the available leakage and thus overestimates security:
 - ⇒ if CPA recovered the key with 100k traces, then an attack that models the leakage with a neural network may succeed with 10k traces
- ▶ CPA decides which key candidate is correct using a single time sample, the one with the highest absolute correlation:
 - ⇒ what if a better decision can be reached using multiple time samples i.e. using a *multivariate* attack
- ▶ CPA requires the key to remain constant during the attack:
 - ⇒ re-keying the cipher can stop attacks like CPA; yet a *profiled* attack still work

Countermeasures:

- ▶ Add noise to the chip, typically by performing operations in parallel to the encryption
- ▶ Shuffle the order of operations
e.g. reorder the 16 AES sbox operations during every round
- ▶ Randomize the values during computation (masking)
- ▶ Instead of standard CMOS electronics, use special electronics like dual-rail-logic