CSE 5095 & ECE 4451 & ECE 5451 – Spring 2017

Lecture 8a

- Slide deck originally based on some material by Chenglu and Ha during ECE 6095 Spring 2017 on Secure Computation and Storage, a precursor to this course

# Side Channel Analysis
# Power Side Channel

**Marten van Dijk**
**Syed Kamran Haider, Chenglu Jin, Phuong Ha Nguyen**

Department of Electrical & Computer Engineering
University of Connecticut

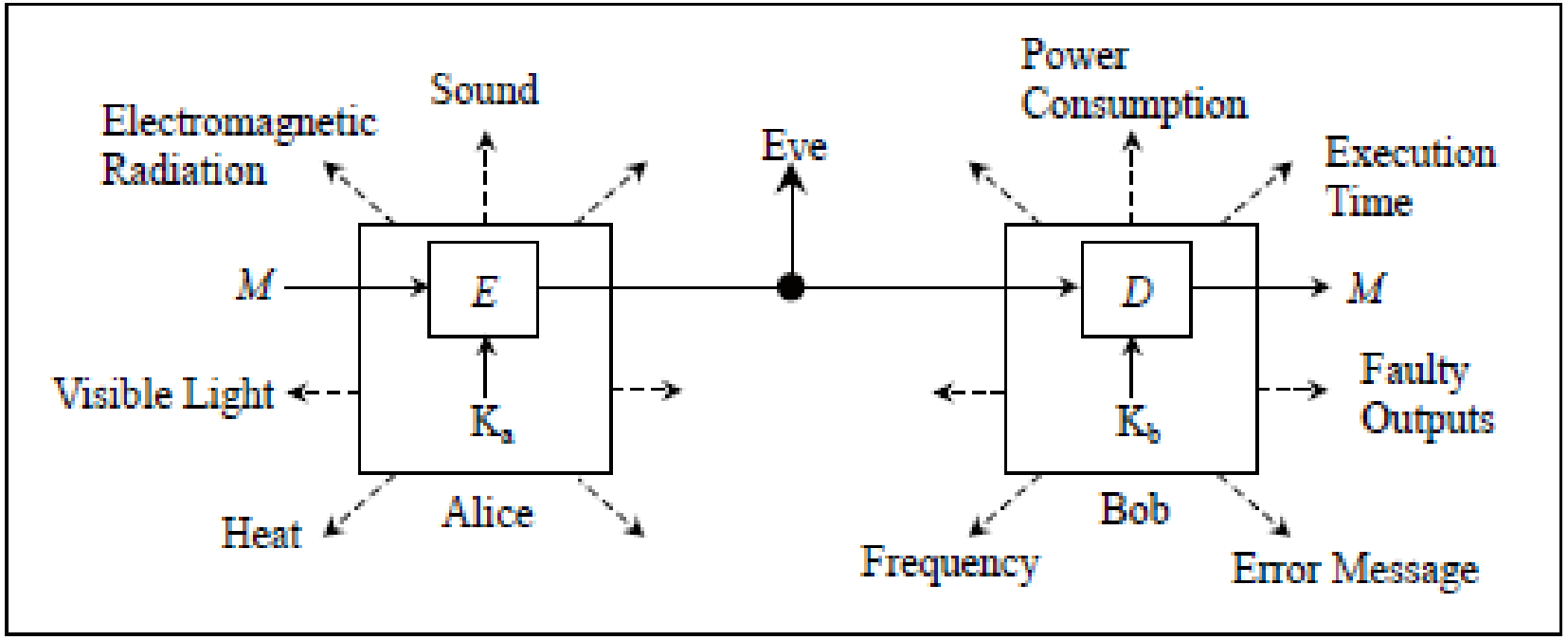
UCONN

# Outline

1. Introduction
   1. Timing channel
   2. Power channel
   3. EM radiation channel
   4. Acoustic channel
   5. Photonic emission channel
2. Power Side Channel Attacks
   1. Attacks
      1. Simple Power Analysis, Differential Power Analysis, Correlation Power Analysis, Template Attack
   2. Countermeasures
      1. Algorithm Modification, Dummy Operations, Operation Shuffling, Hiding, Masking, Threshold Implementation

# Introduction

- In cryptography, a side-channel attack is an attack based on information gained from the physical implementation of a cryptosystem, rather than brute force or theoretical weaknesses in the algorithms (compare cryptanalysis).

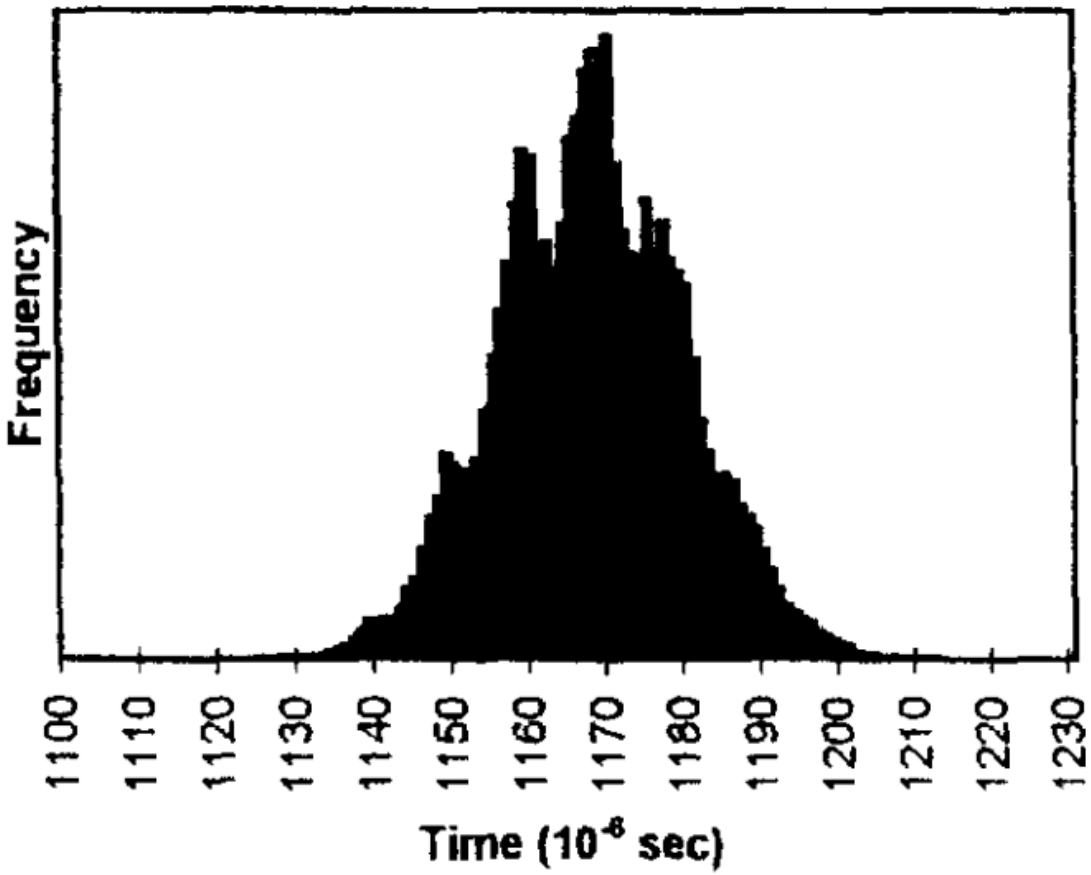



https://en.wikipedia.org/wiki/Side-channel_attack

# Timing Side Channel

- The execution time of e.g. RSA depends on the value of secret data/key.
- By timing the execution for a particular public input, one obtains information about the secret.
- RSA: By adaptively modifying the public input and measuring the execution times one can uncover the key bit by bit.
- On the right: Statistics of the termination channel – its variation can be exploited.

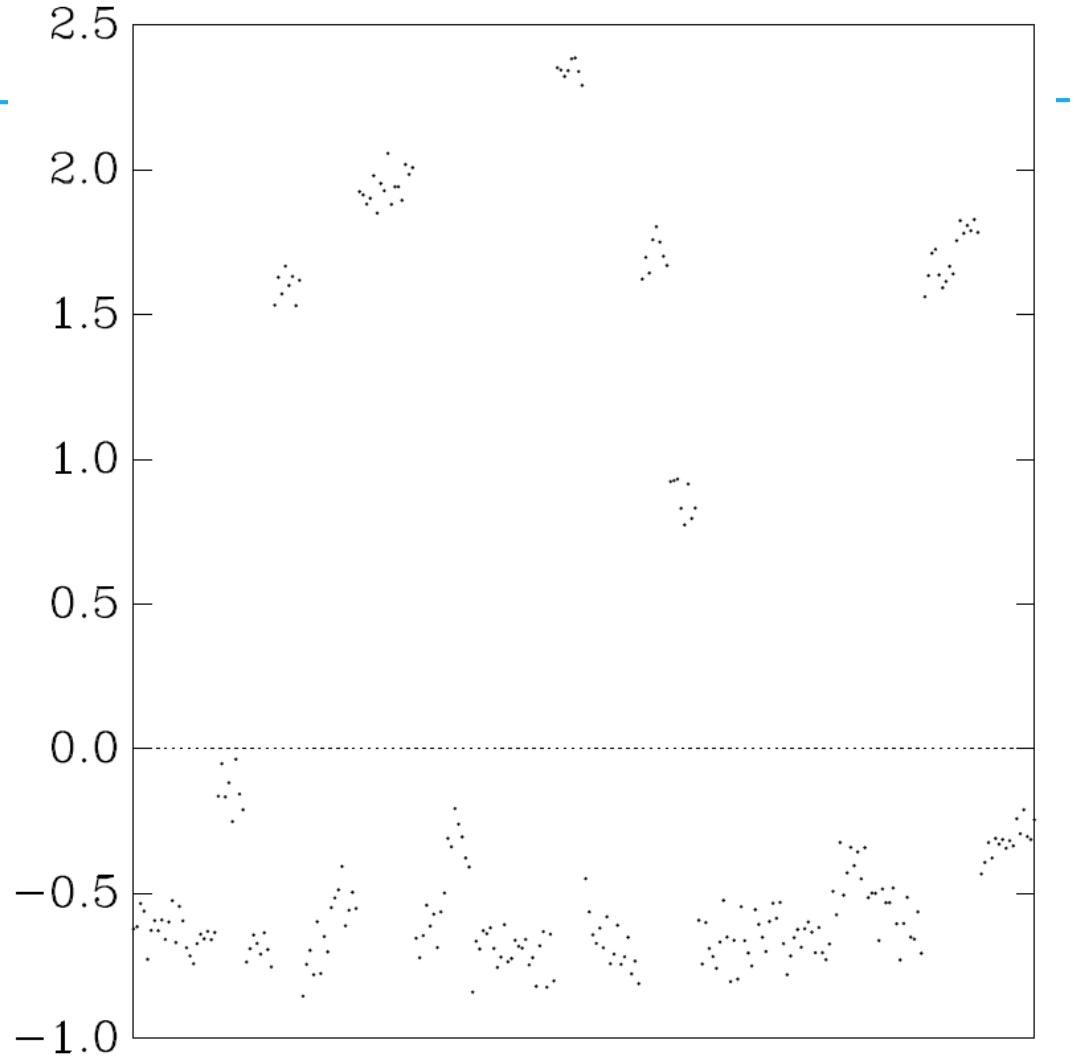
**FIGURE 1**: RSAREF Modular Multiplication Times




Paul Kocher. "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems". Crypto'96

# Cache Timing Channel

- Target application, e.g. AES, runs together with a malicious monitoring application on top of the same processor.
- They share LLC: They push one another out of the LLC and this interference can be monitored by the malicious app and reveal security sensitive data of the target app.
- AES: Different keys in the target app lead to different data access patterns for the monitor app (cache hit or cache miss). Since cache hit and miss have huge timing differences, the monitor app can extract information about the key used in the target app.




Daniel J. Bernstein. “Cache-timing attacks on AES”. 2005

# Power Channel

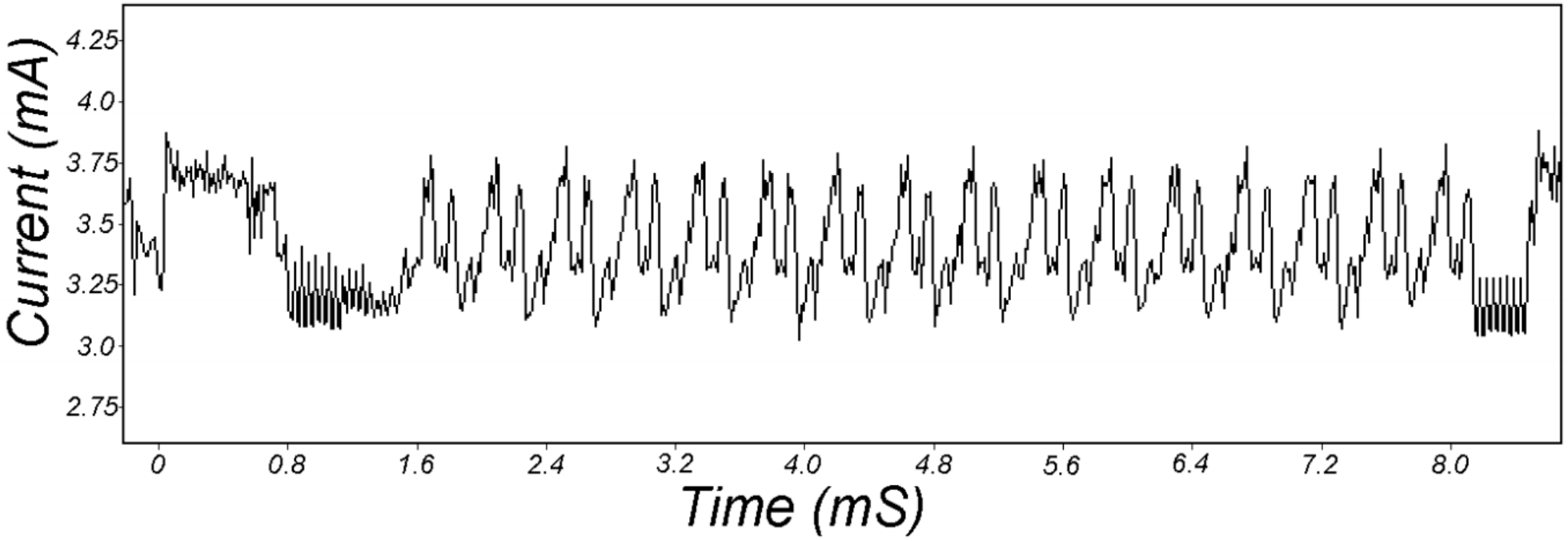
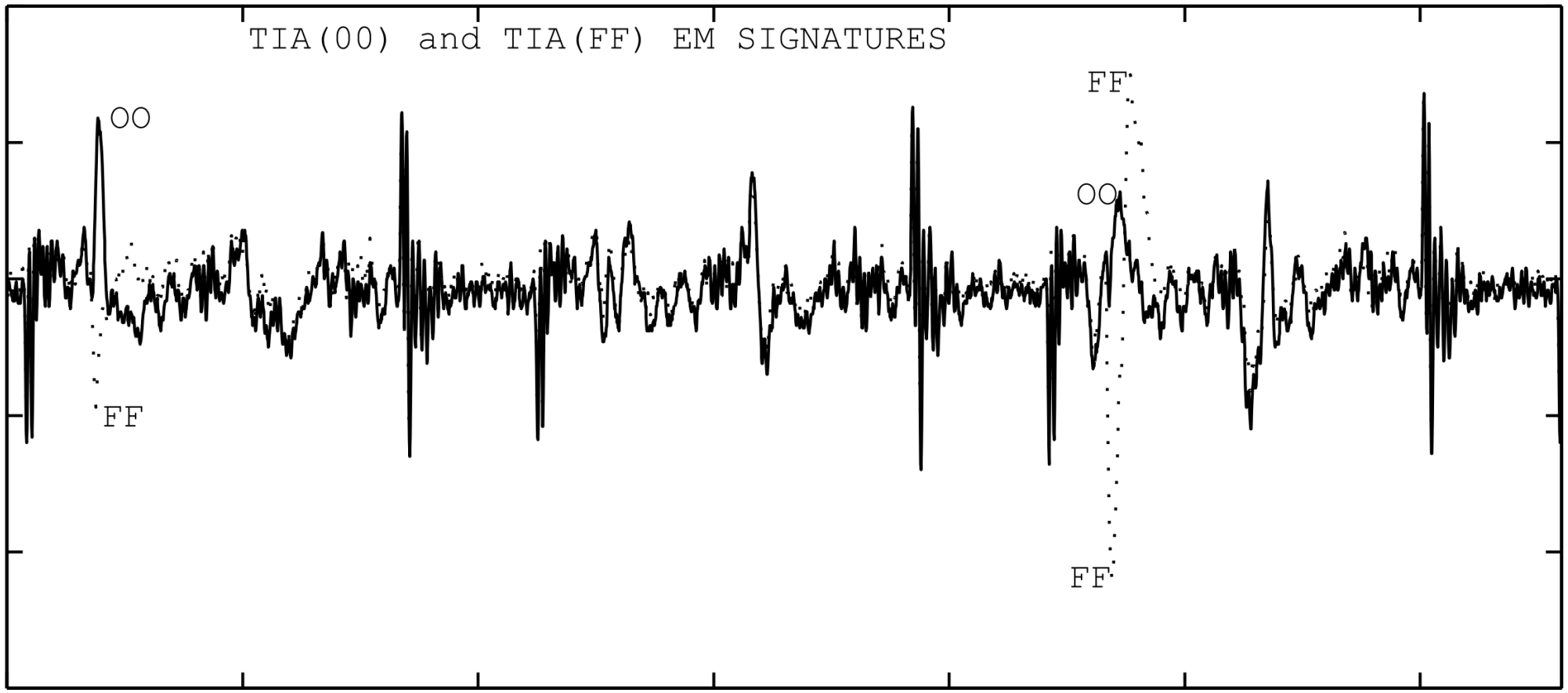



**Figure 1:** SPA trace showing an entire DES operation.

The power consumption of a chip depends on the secret data that is used in the computation on the chip. One is able to uncover the secret data by measuring the power consumption of the entire chip.

Paul Kocher, Joshua Jaffe and Benjamin Jun. "Differential Power Analysis". Crypto'99

# EM Radiation Channel




EM radiation depends on the secret data that is being processed.

Karine Gandolfi, Christophe Mourtel, and Francis Olivier. "Electromagnetic Analysis: Concrete Results". CHES'01

# Acoustic Channel

Acoustic frequency from different motherboard components leak information about the instructions performed by the target's CPU

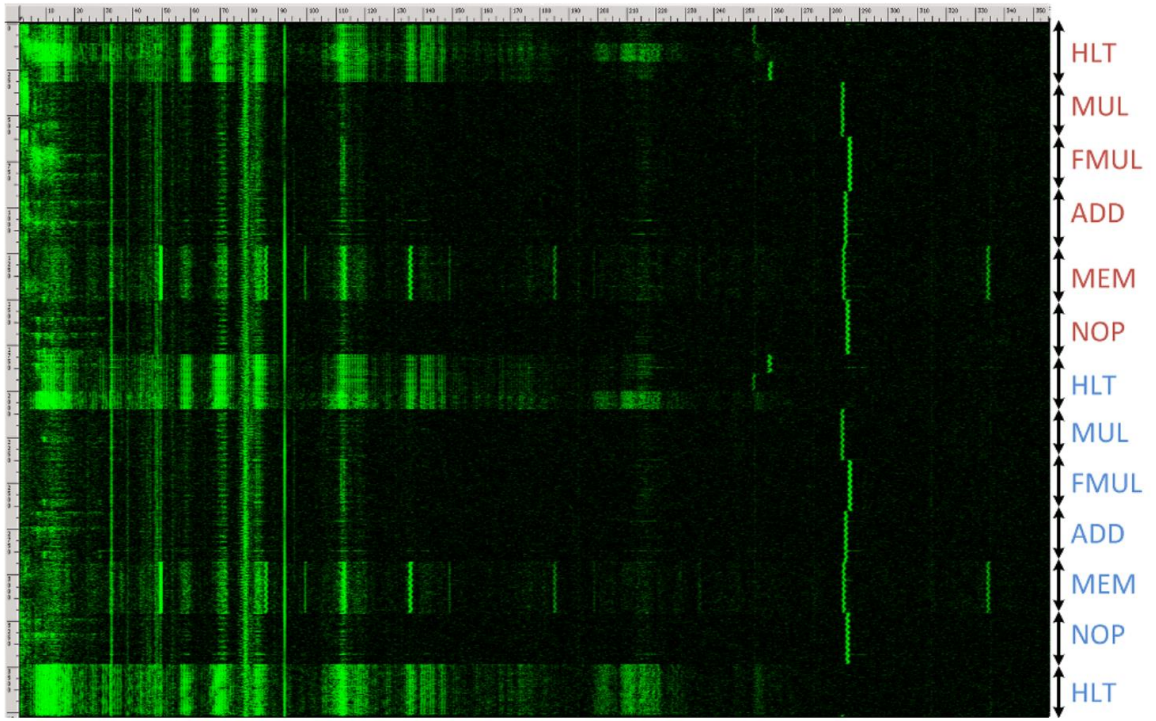
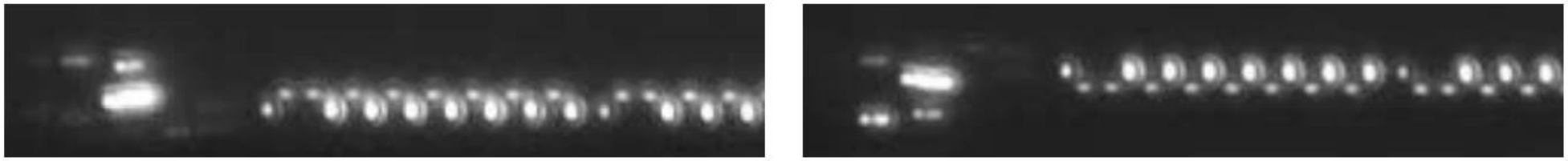



Figure 7: Acoustic measurement frequency spectrogram of a recording of different CPU operations using the Brüel&Kjær 4939 microphone capsule. The horizontal axis is frequency (0–310 kHz), the vertical axis is time (3.7 sec), and intensity is proportional to the instantaneous energy in that frequency band.

Daniel Genkin, Adi Shamir and Eran Tromer. "RSA Key Extraction via Low-Bandwidth Acoustic Cryptanalysis". CRYPTO'14

# Photonic Emission Channel

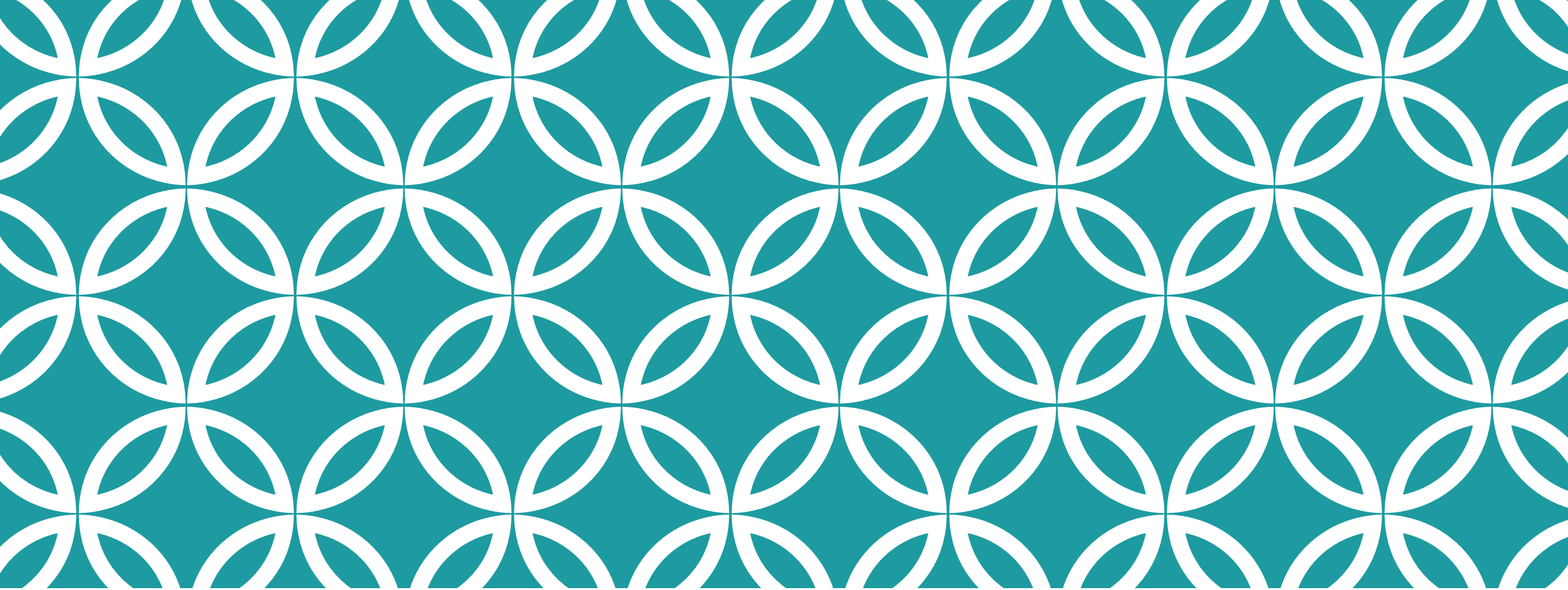


(a) Access to 0x300 (b) Access to 0x308

**Fig. 2.** 120 s emission images of memory accesses to two adjacent memory rows obtained with the Si-CCD detector

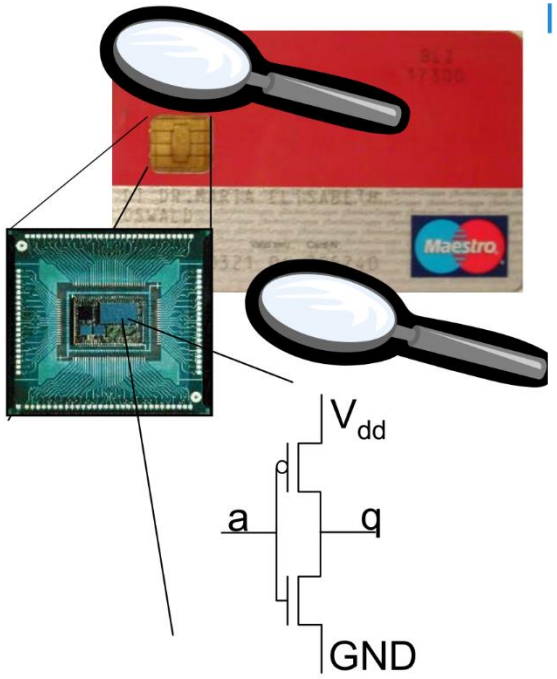
Photonic emission pattern is data dependent, so it can also be used to extract the secret data.

Alexander Schlösser, Dmitry Nedospasov, Juliane Krämer, Susanna Orlic and Jean-Pierre Seifert. "Simple Photonic Emission Analysis of AES Photonic Side Channel Analysis for the Rest of Us". CHES'12



# Power Side Channel Attack

# Why does the power consumption reflect the data?



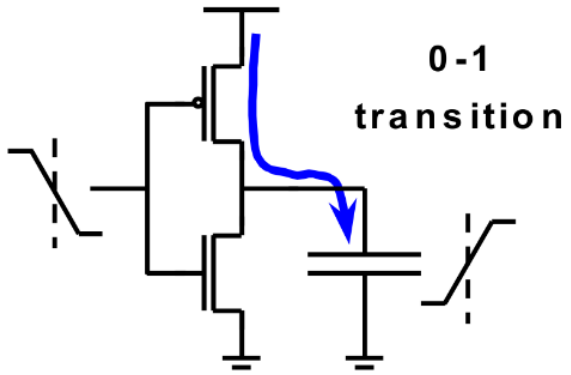

CMOS Inverter

1. CMOS technology is the common technology for implementing crypto devices
2. Power analysis attacks exploit the fact that the instantaneous power consumption of a device built in CMOS technology depends on the data it processes and the operations it performs

- CMOS inverter:

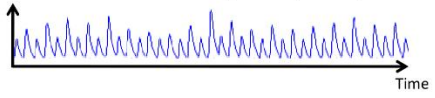
a=>a, b=>b Capacity

| Input | Output | Current |
|---|---|---|
| 0 → 0 | 1 → 1 | Low |
| 0 → 1 | 1 → 0 | Discharge |
| 1 → 0 | 0 → 1 | Charge |
| 1 → 1 | 0 → 0 | Low |

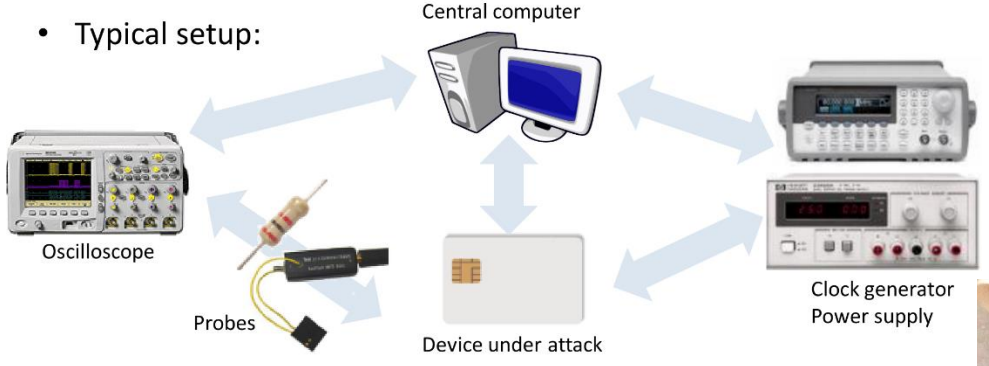


# Setup for Power Analysis

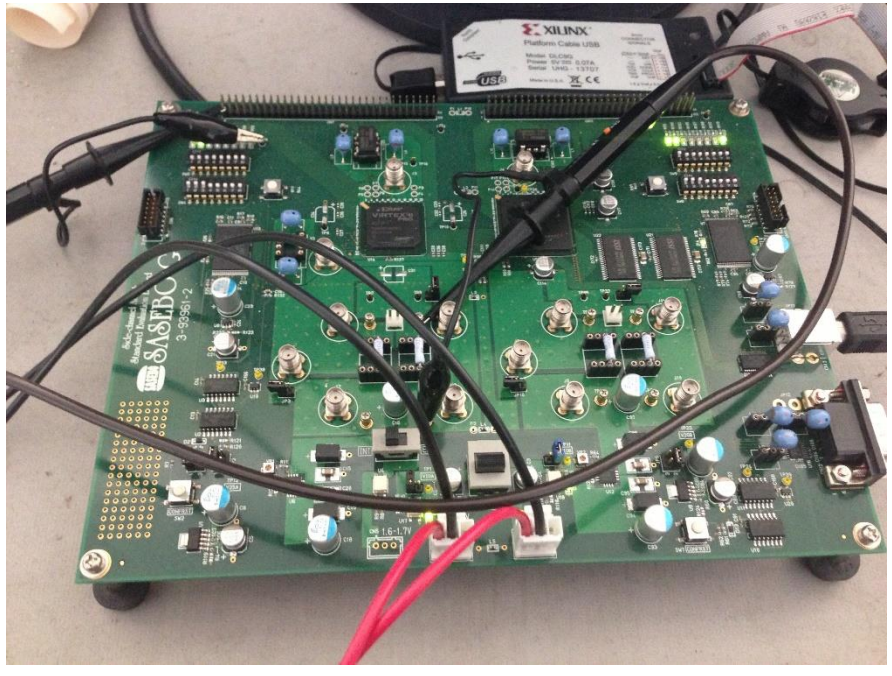
- Instantaneous power over time
  - Trace or curve, many samples

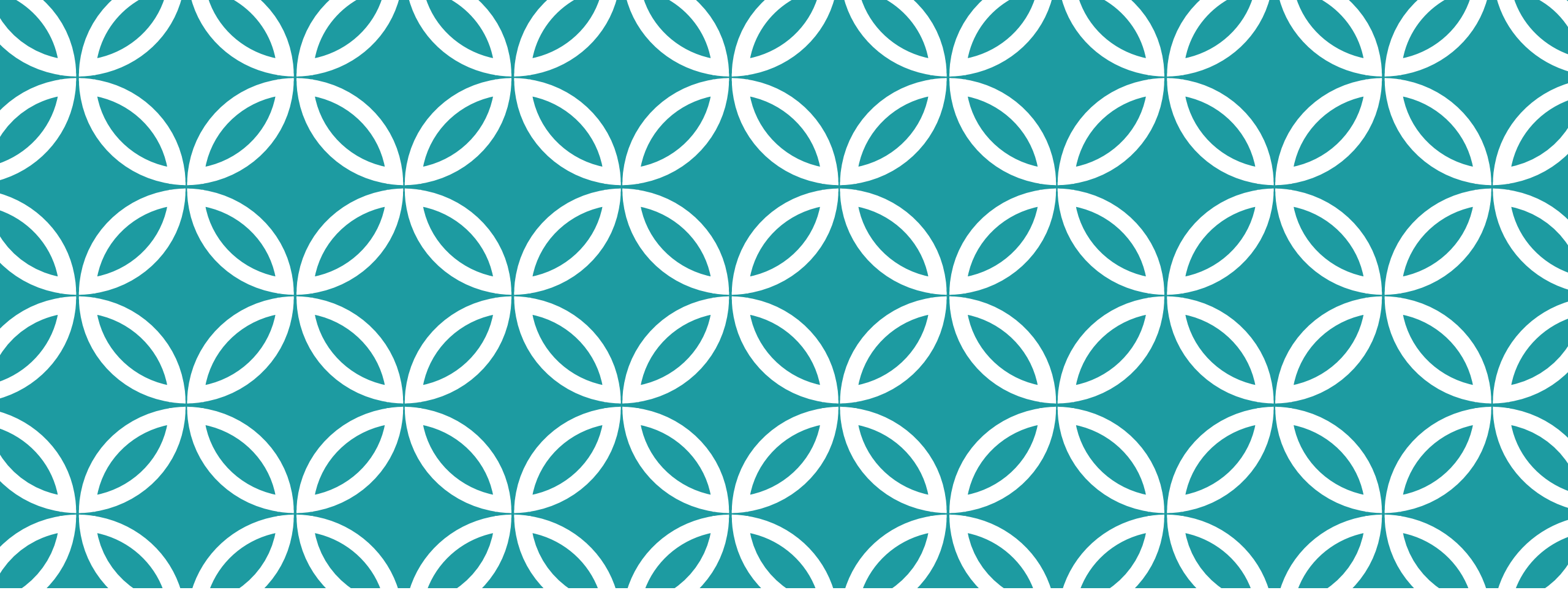



- Typical setup:




- We usually use oscilloscope to measure voltage
- Measure in VDD or GND line
  - Connect a resistor (Ohm's law: V = R x I)
  - Measure V over resister
  - Probe -> Voltage





# Simple Power Analysis

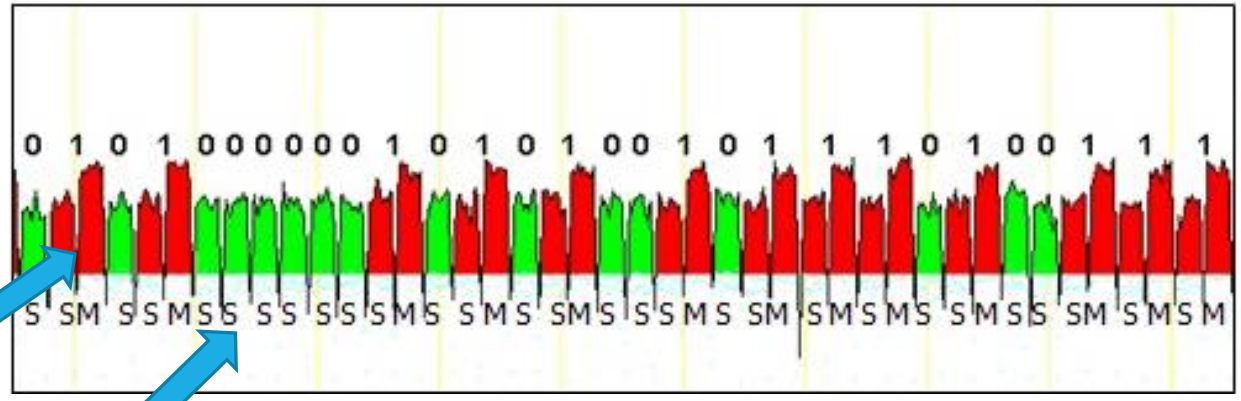
# RSA Background

- RSA algorithm requires modular exponentiation
- How to do modular exponentiation efficiently?
- Short answer: repeated squaring
- Example: we want to compute $a^{18}$
- Notice that 18 = 2 x 9 = 2 x (8+1) = 2 x (2 x 2 x 2 +1) relates to 18 = 0b10010
- Do 4 squaring $((((a)^2)^2)^2a)^2) = a^{18}$

# SPA on RSA

**Algorithm 1** Multiply and Square Algorithm

```
1:  procedure Mul − Squ(g,K)
2:      Convert K into binary representation k_0, k_1, ...k_n, where k_0 = 1
3:      if K == 0 then
4:          Result = 1
5:          return Result
6:      else
7:          Result = g
8:          for  doi ← 1, n
9:              if k_i == 1 then
10:                 Result = M(Result, Result)
11:                 Result = M(Result, g)
12:             else
13:                 Result = M(Result, Result)
14:             end if
15:         end for
16:         return Result
17:     end if
18: end procedure
```

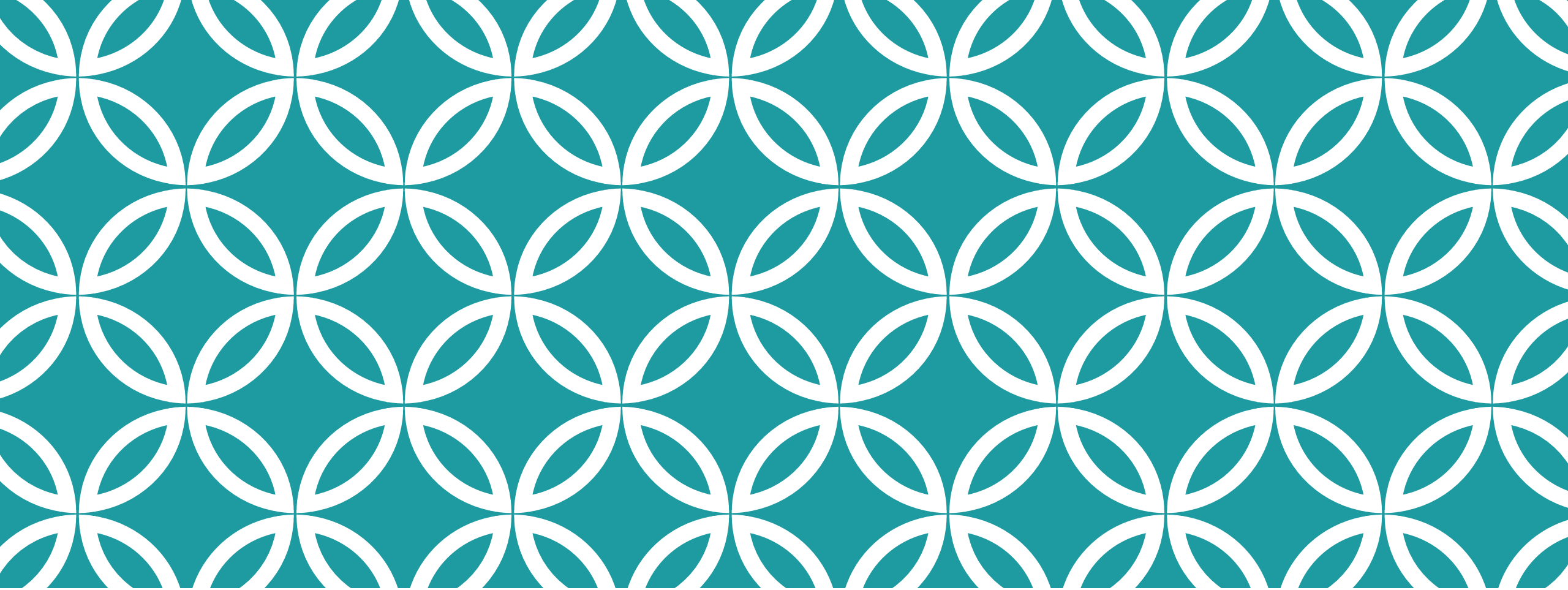


# Countermeasure

- Montgomery Ladder
  - Add a dummy multiplication when the secret bit is 0
  - Have a constant execution time
  - Prevent SPA perfectly, because SPA reveals the instruction flow.

- $N = n_{k-1} \dots n_0$
- $x_1 = x^{Rep(n_{k-1} \dots n_{i+1})}$ and $x_2 = x^{1+Rep(n_{k-1} \dots n_{i+1})}$
- $x_1 x_2 = x^{1+2Rep(n_{k-1} \dots n_{i+1})} = x^{Rep(n_{k-1} \dots n_{i+1} 1)}$
- $x_1^2 = x^{2Rep(n_{k-1} \dots n_{i+1})} = x^{Rep(n_{k-1} \dots n_{i+1} 0)}$
- $x_2^2 = x^{2+2Rep(n_{k-1} \dots n_{i+1})} = x^{1+Rep(n_{k-1} \dots n_{i+1} 1)}$

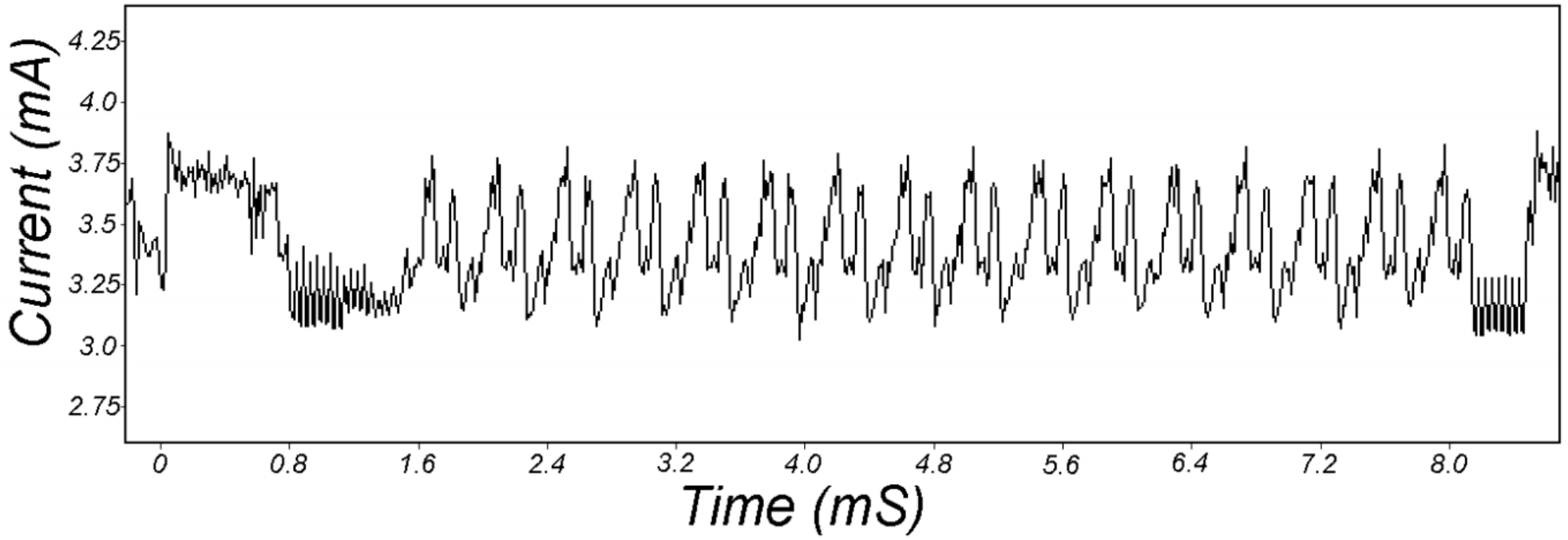
```
x1=x; x2=x^2
for i=k-2 to 0 do
  If ni=0 then
    x2=x1*x2; x1=x1^2
  else
    x1=x1*x2; x2=x2^2
return x1
```



# Differential Power Analysis

# DPA: Differential Power Analysis

- DPA extracts some statistical features from a large amount of power traces to distinguish a correct key guess from a bunch of wrong key guesses.
- Power traces: records current (=voltage/resistance) variations over time
- Power traces are sampled by oscilloscope, and are stored as an array of data.

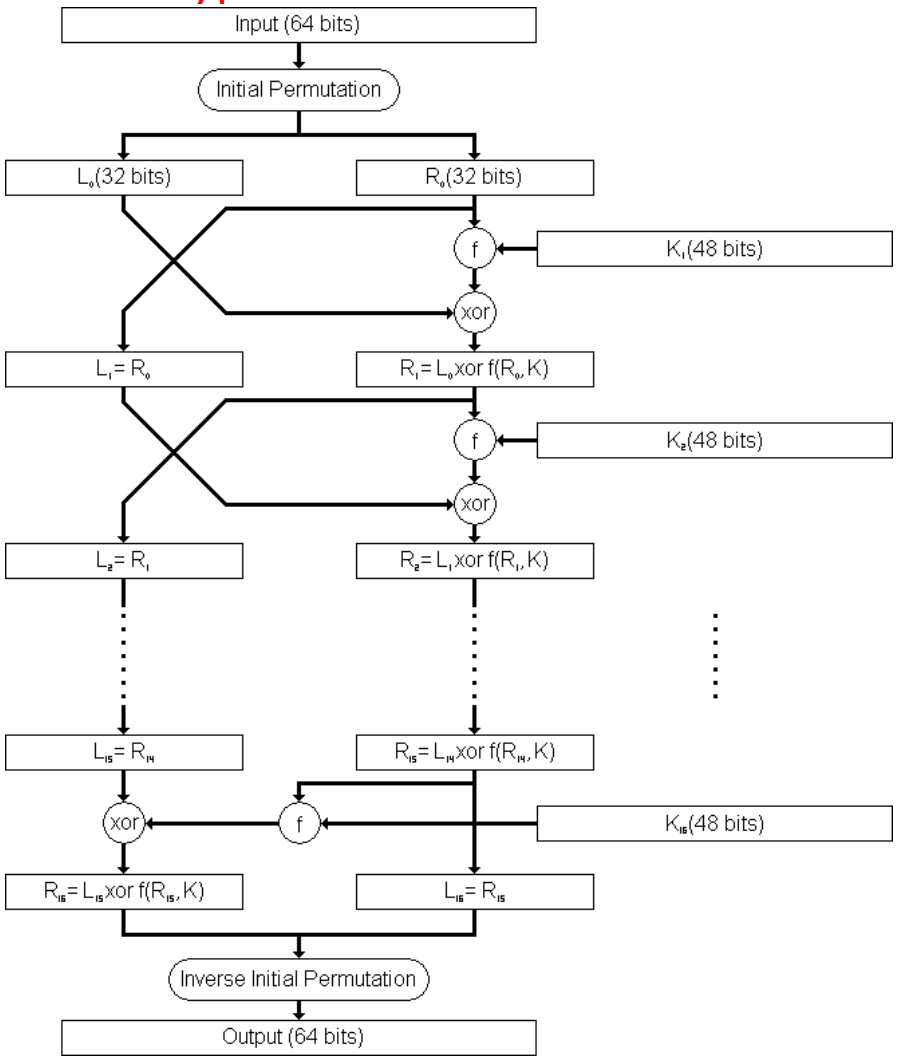


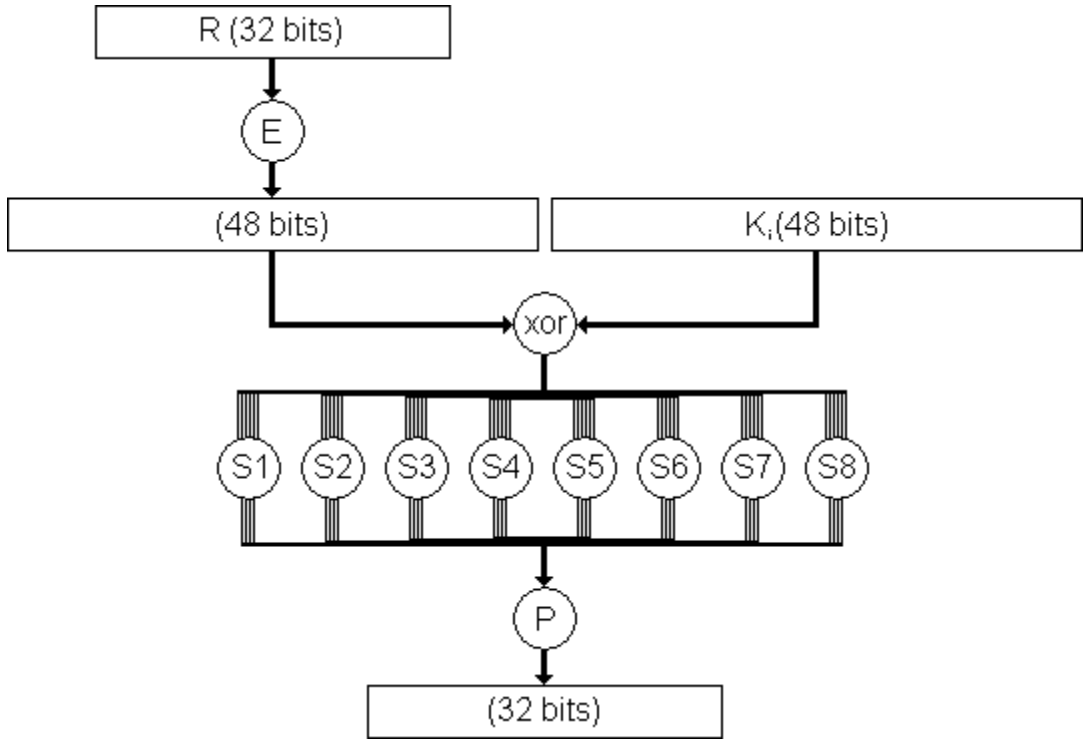
# DPA vs SPA

1. While SPA uses one/few traces to recover the secret key, DPA uses many traces in order to exploit statistical features.
2. SPA exploits the instruction dependency, but DPA exploits the data dependency.
3. In SPA, we may not need to know the data (i.e., plaintext or ciphertext), because the instruction flow is enough for extracting the secret. However, the data (plaintexts or ciphertexts) in DPA is required.

# DES encryption data path

Data Encryption Standard = DES




Function f (also called mangler function)

# Attack Steps Overview

1. Measure the power traces and record the corresponding plaintext or ciphertext.
2. Use the known data (plaintext or ciphertext) with a guessed key to predict an intermediate value
3. Check if the intermediate value is reflected in the power trace or not. If yes, guess is correct. Otherwise, guess is wrong.

# Attack Strategy

- If the attack is simply guess and check, what is the different from a brute force attack?
- Principle: Divide and Conquer
- We need to use some statistical features to distinguish a correct subkey guess from a wrong subkey guess (the space for subkey guesses should not be prohibitively large).
- In the structure of DES, it allows us to guess 6-bit subkey per attack, so the guess space is $2^6 = 64$.
- The complexity of brute force is $2^{56}$, but the complexity of DPA is $2^6 * 8 + 2^8$.
- Note: DES key is 64 bit long, including 56 bit effective key and 8 bit checksum. 48 effective key bits will be used to derive 48 bit first round key.

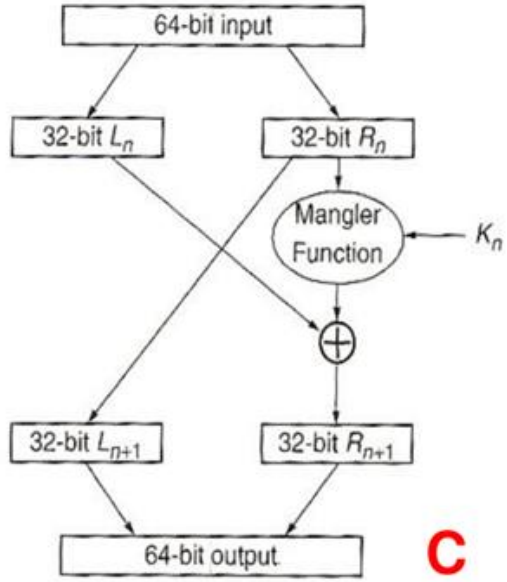
# Attack Step 1

- Capture m power traces $T_{1...m}$ [1…k], where k is the number of sampling points in each trace. In the meantime, record m ciphertexts $C_{1...m}$

K sampling points

M traces

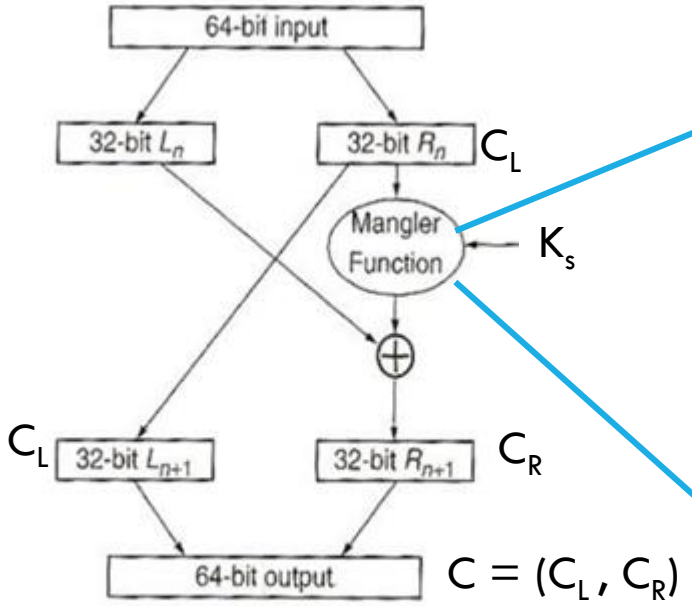
| | | | | |
|---|---|---|---|---|
| $T_1[1]$ | $T_1[2]$ | $T_1[3]$ | … | $T_1[k]$ |
| $T_2[1]$ | $T_2[2]$ | $T_2[3]$ | … | $T_2[k]$ |
| … | | | | |
| $T_m[1]$ | $T_m[2]$ | $T_m[3]$ | … | $T_m[k]$ |

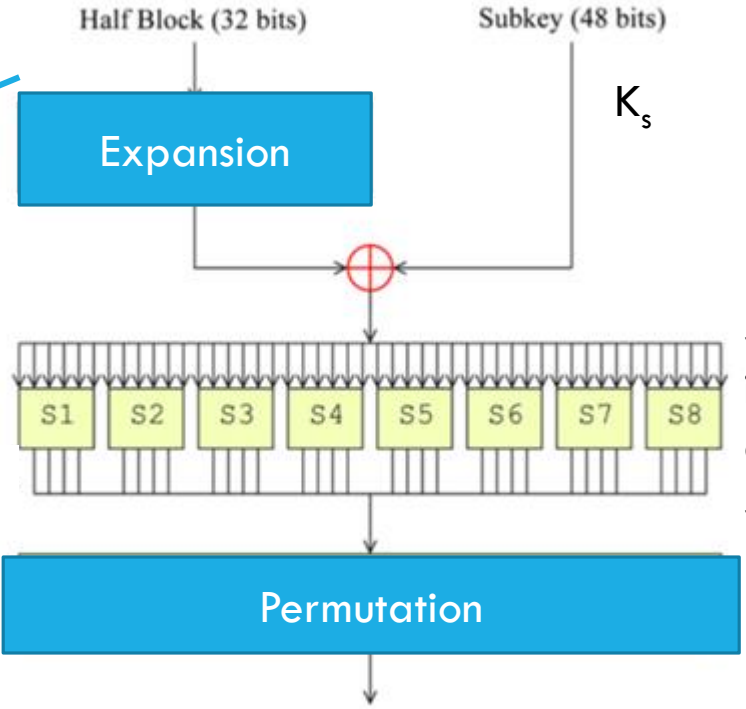



16th DES round

# Attack Step 2

- Guess a 6-bit subkey and compute the output of one SBOX in DES $S_i(K_s + C_L)$
- Select a bit b from the 4-bit output of the i-th SBOX
- Given a guess for the 6-bit subkey
  - We compute the bit value b from $S_i(K_s+C_L)$
  - We partition all the power traces in two sets $Tr_0$ and $Tr_1$ according to bit b



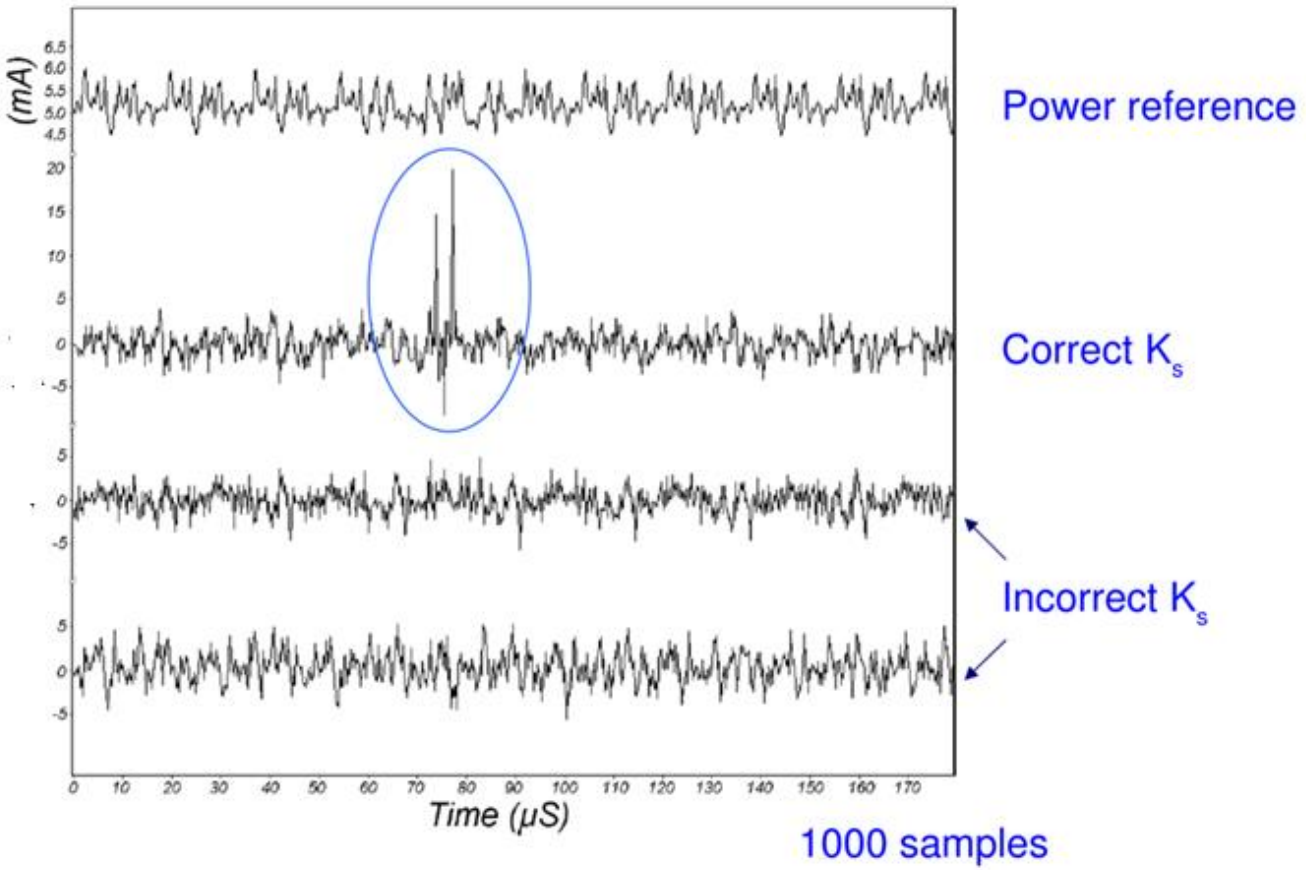

$S_i(K_s+C_L)$ takes the 6 input bits from $K_s+C_L$ and applies the i-th SBOX

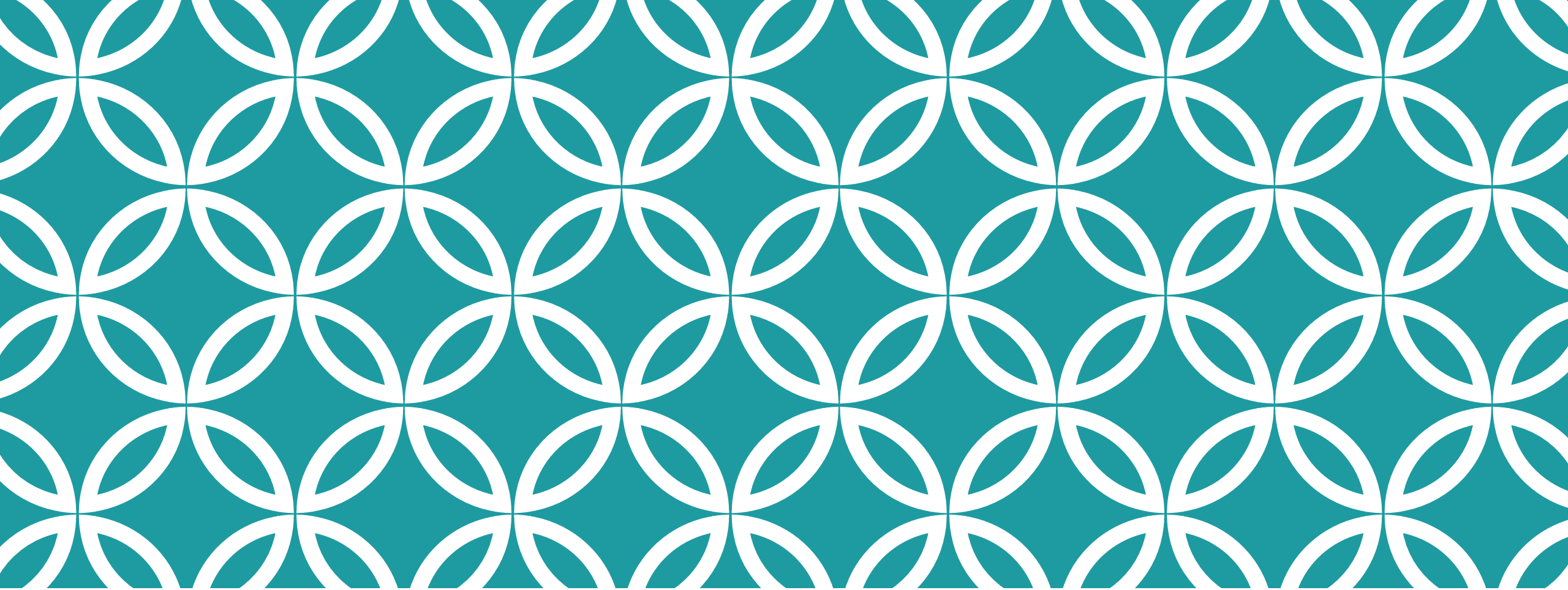
# Attack step 3

- For each sample point j, compute

$$\Delta_D[j] = \frac{\sum Tr_1[j]}{|Tr_1|} - \frac{\sum Tr_0[j]}{|Tr_0|}$$

- If the key guess is wrong, then this delta will be close to zero, because the traces are uncorrelated with our computed value of b.
- If the key guess is correct, then all the traces will be partitioned into two sets correctly, then we can see a large spike in delta because there is a power consumption difference between producing a 0 vs 1 as intermediate value in the mangler function.



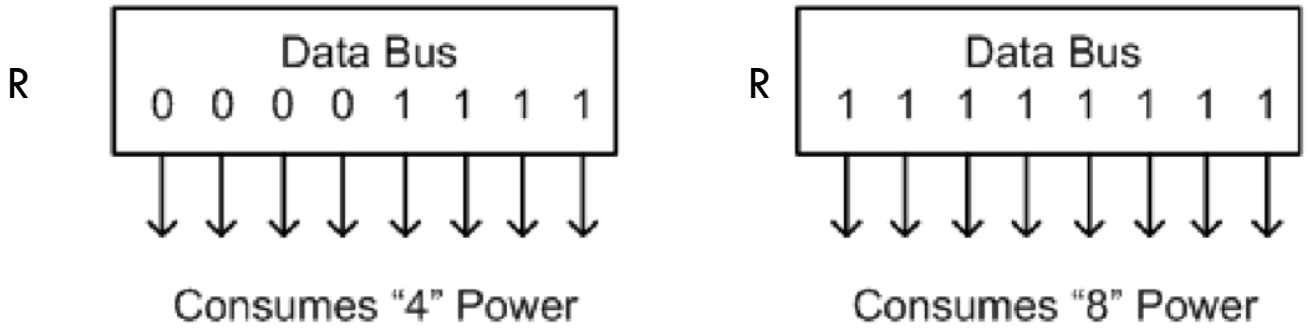


# Correlation Power Analysis

# CPA: Correlation Power Analysis

- Instead of the function for computing delta in DPA, CPA uses Pearson correlation as metric to distinguish a correct key guess from wrong key guesses.
- Compared to DPA, since CPA combines leakage of multiple bits, its efficiency is better in terms of the number of power traces
- In order to understand the leakage of multiple bits, we need to have a power model.

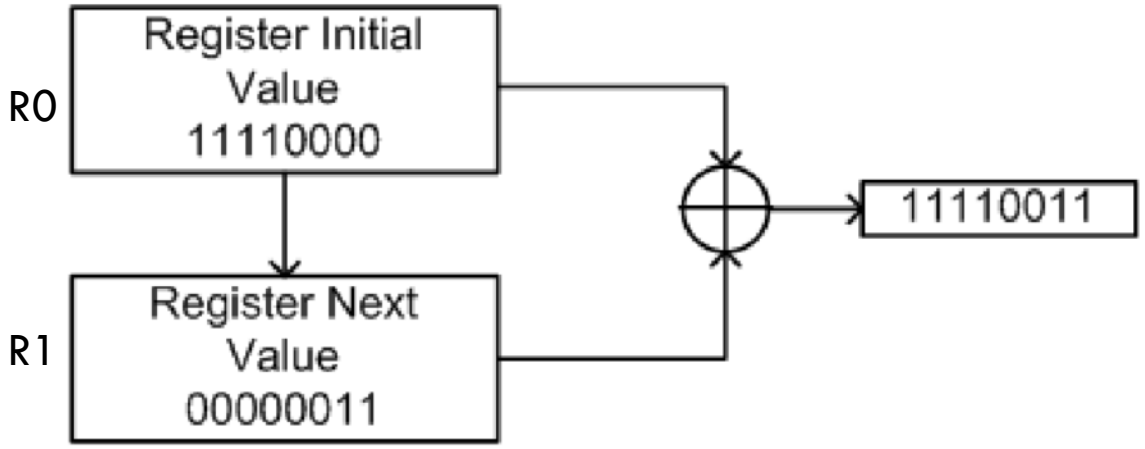
# Power Models: HW and HD




Hamming Weight HW:

P ~ HW(R)

An example showing the Hamming weight model. The Hamming weight of the value on the data bus is taken to be proportional to the power consumption of the bus.

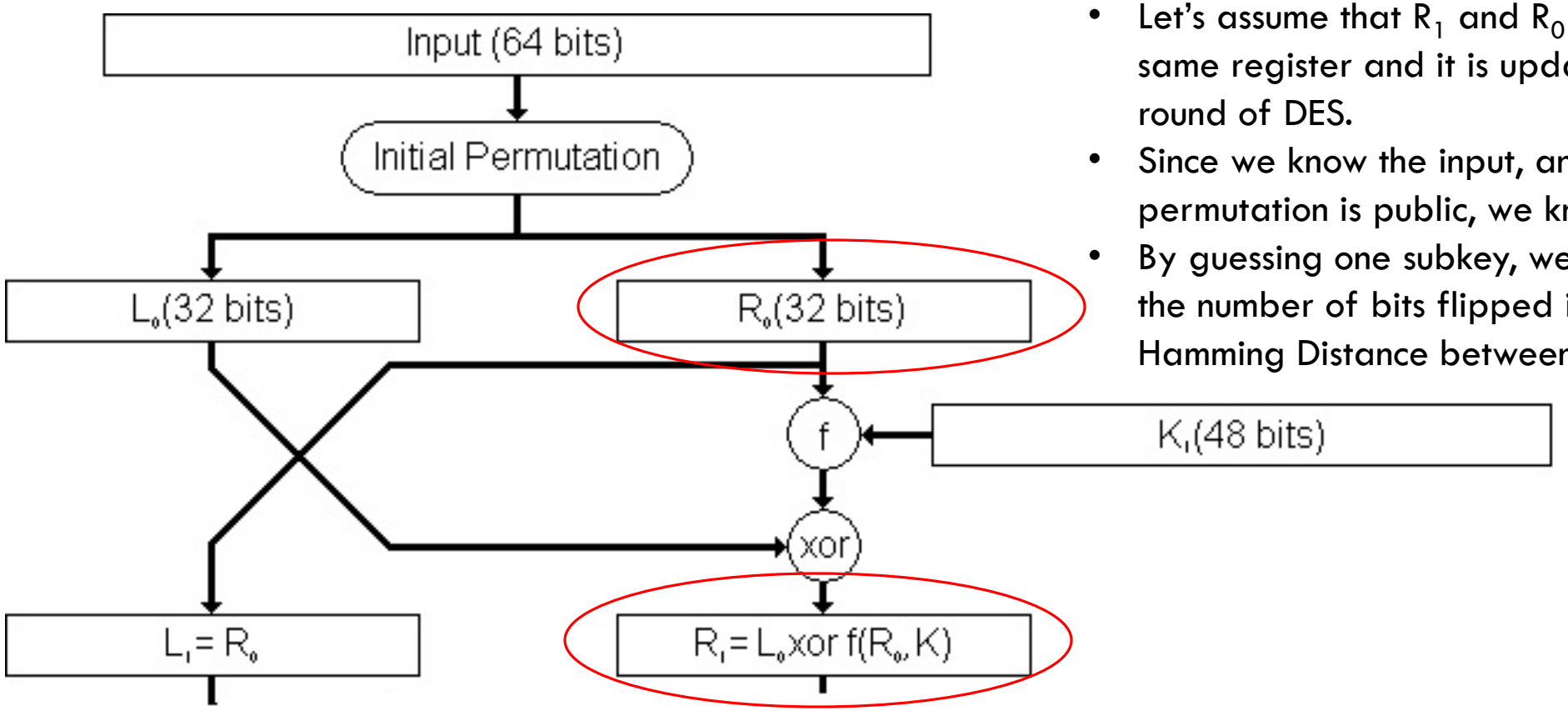



An example showing the Hamming Distance model.

Hamming Distance HD:

P ~ HD(R0,R1)=HW(R0 XOR R1)

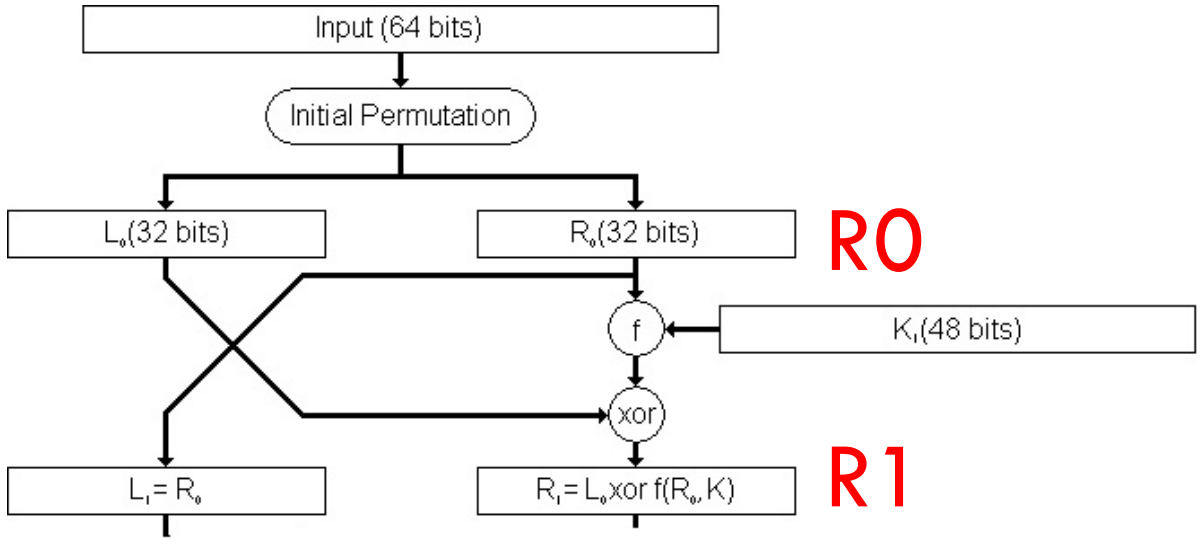
# DES Implementation

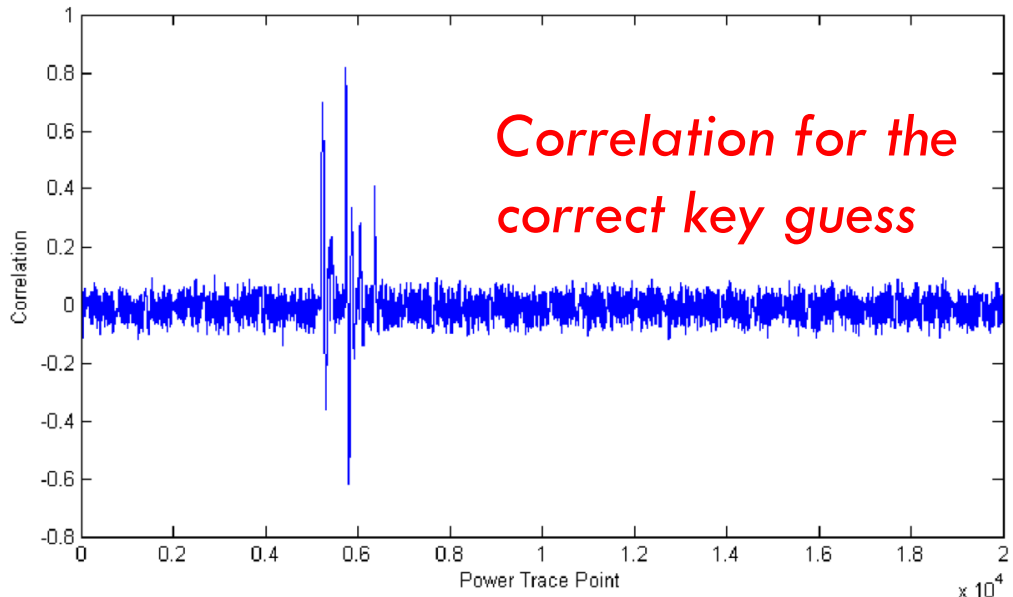



- Let's assume that $R_1$ and $R_0$ are stored in the same register and it is updated after one round of DES.
- Since we know the input, and the initial permutation is public, we know correct $R_0$.
- By guessing one subkey, we are able to predict the number of bits flipped in the register: Hamming Distance between $R_0$ and $R_1$.

Attack applies to first or last round.

# HD model for DES




*Correlation for the correct key guess*

$$X_i = HD(R0, R1) = HW(R_0 \oplus f(R_0 \oplus K) \oplus L_0)$$

*We guess K (we know R0, L0 representing the plaintext corresponding to the i-th power trace)*

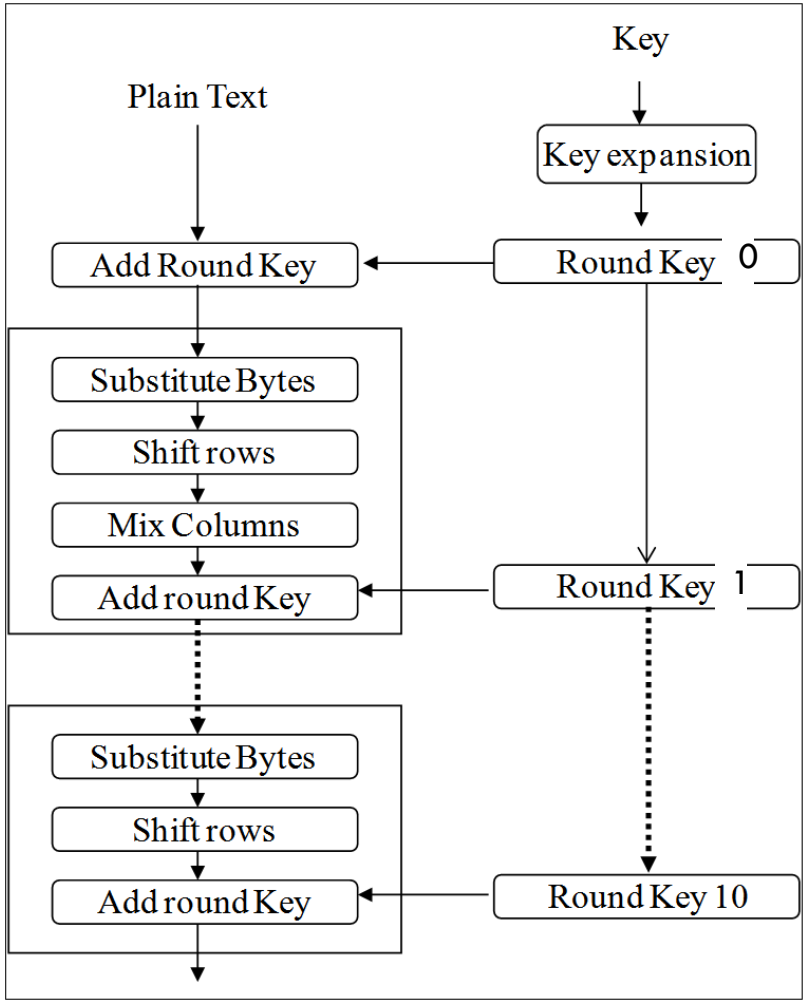
$$Y_{i,j} = T_i[j], j = 1, \ldots, k$$

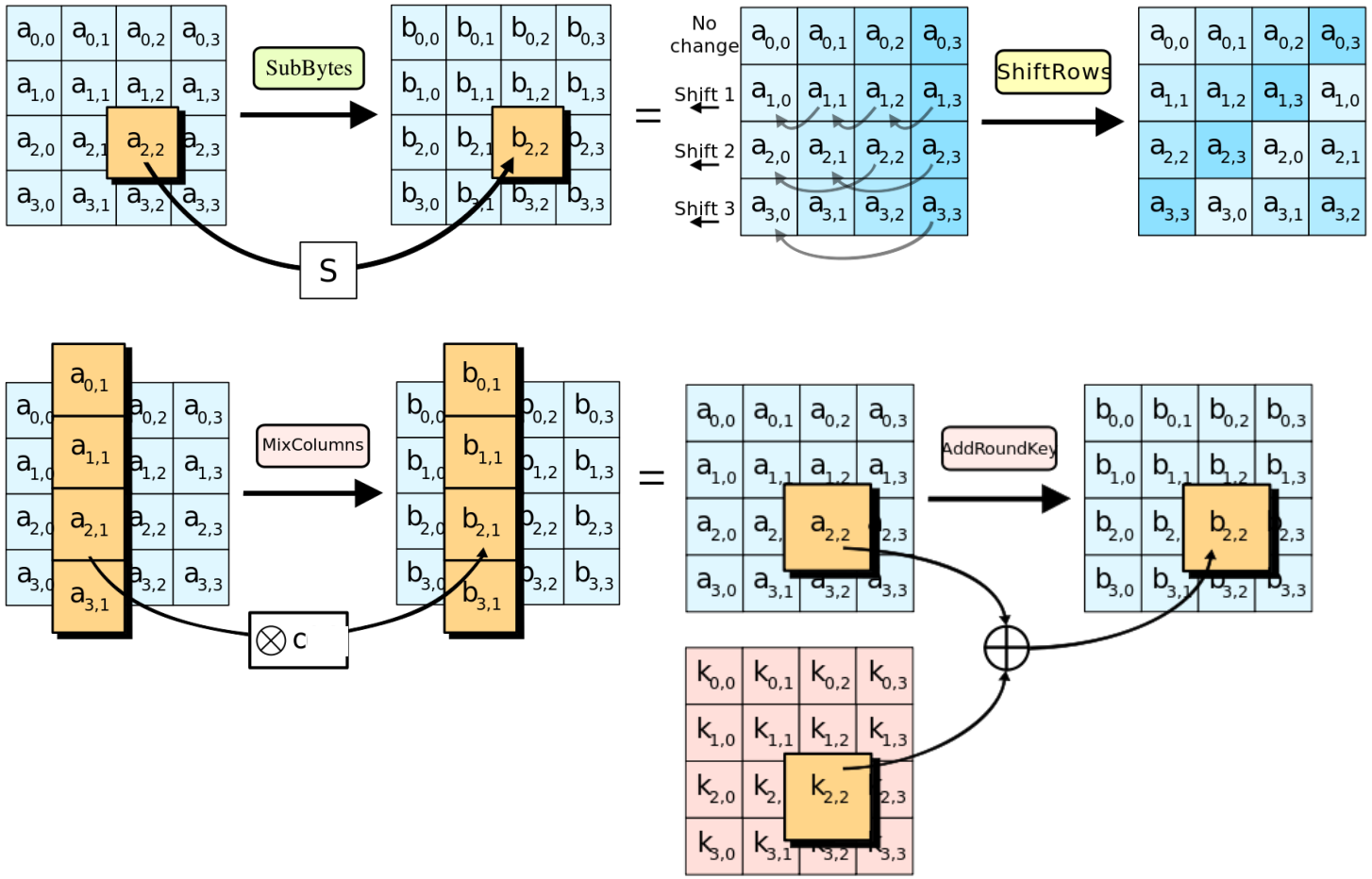
$$\bar{X} = \sum_{i=1}^{m} X_i/m$$

$$\bar{Y_j} = \sum_{i=1}^{m} Y_{i,j}/m$$

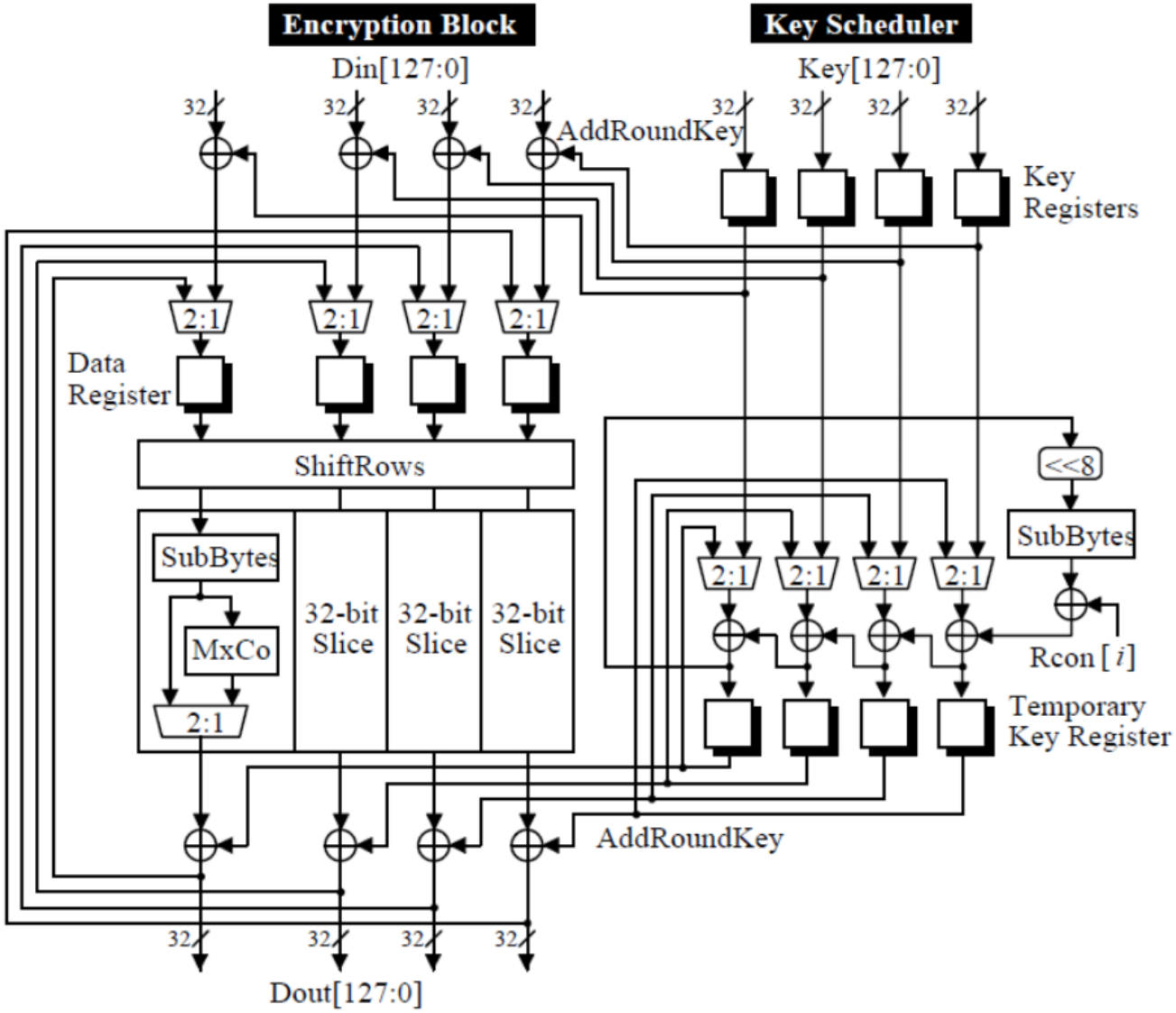
$$r_j = \frac{\sum_{i=1}^{m}(X_i - \bar{X}) \times (Y_{i,j} - \bar{Y_j})}{(\sqrt{\sum_{i=1}^{m}(X_i - \bar{X})^2}) \times (\sqrt{\sum_{i=1}^{m}(Y_{i,j} - \bar{Y_j})^2})}$$

# AES


Key
Plain Text
Key expansion
Add Round Key
Round Key 0
Substitute Bytes
Shift rows
Mix Columns
Add round Key
Round Key 1
Substitute Bytes
Shift rows
Add round Key
Round Key 10

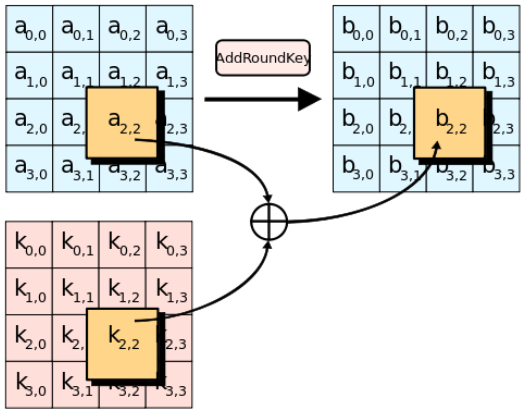

SubBytes
S
No change
Shift 1
Shift 2
Shift 3
ShiftRows
MixColumns
c
AddRoundKey

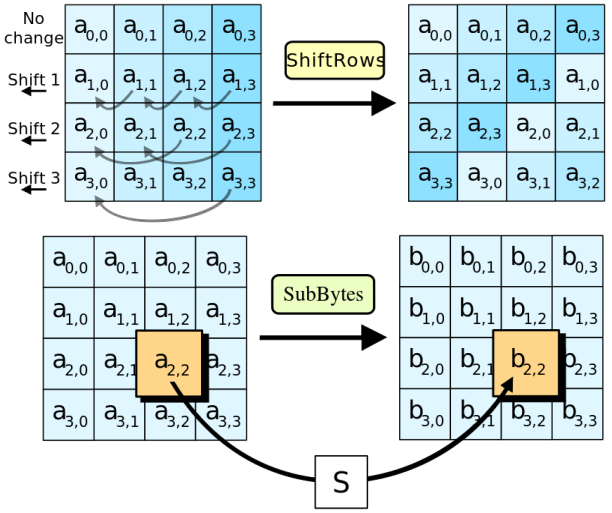
# AES implementation

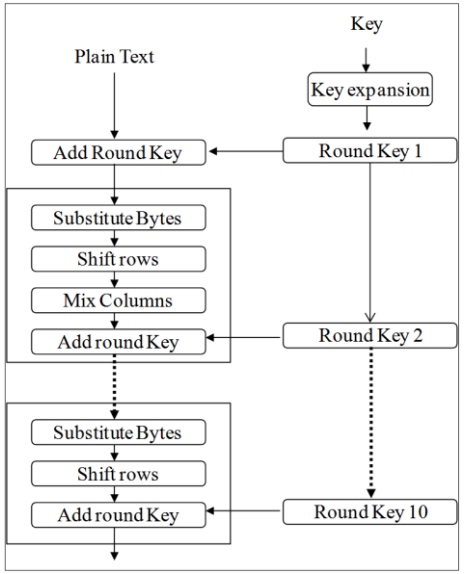


# Target of the attack

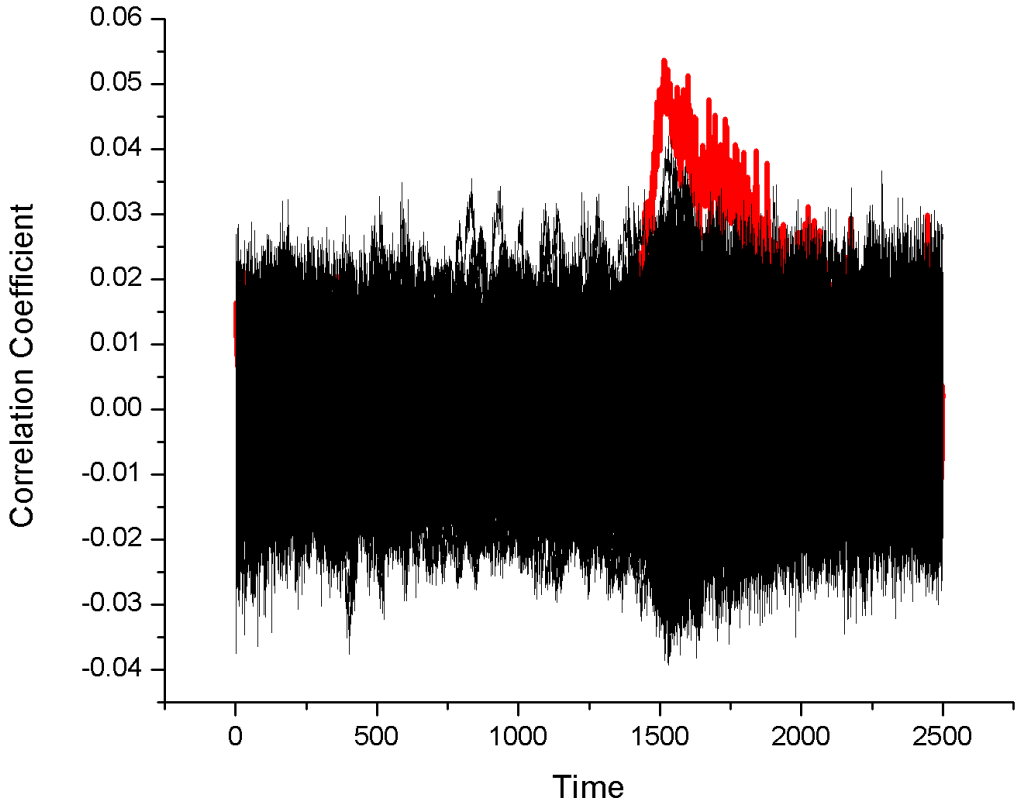
- According to the AES implementation above, the key K0 is added to the plaintext which is the input of the first round. The output of the first round will be added with key K1 and then it is stored in the register. Thus, we have two unknown keys K0 and K1 in the intermediate state after first round stored in the register. It means that, in this implementation we can not use the hamming distance model at the first round
- In the last round, before round 10, the intermediate state stored in register R9 is the input of the round 10. The intermediate C=R10= ShiftRows(SubBytes(R9)) xor K10 is the ciphertext and it is stored in the register. This can be reformulated as R9 = $SubBytes^{-1}$ ($ShiftRows^{-1}$(C xor K10)), where only key K10 is unknown (our target).
- Note that, in the round 10, there is no MixColumn operation and thus, we can do the Divide-and-Conquer attack, recover the secret key byte by byte



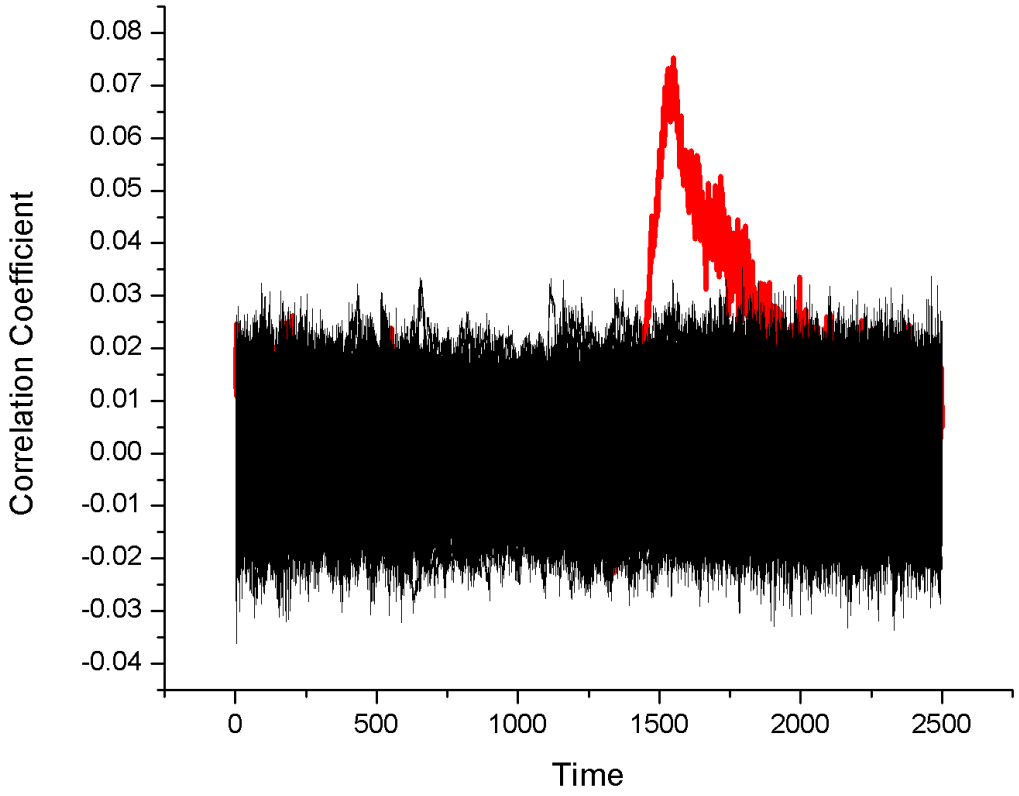



# Results byte 0 of roundkey 10 =0x13

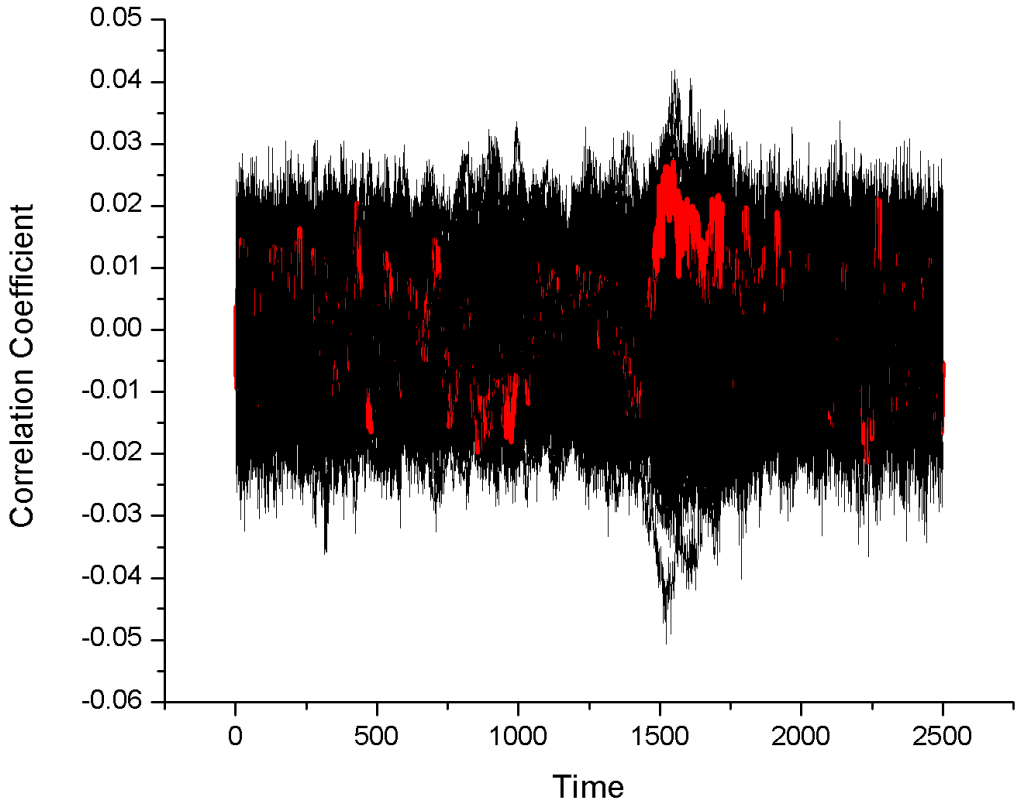


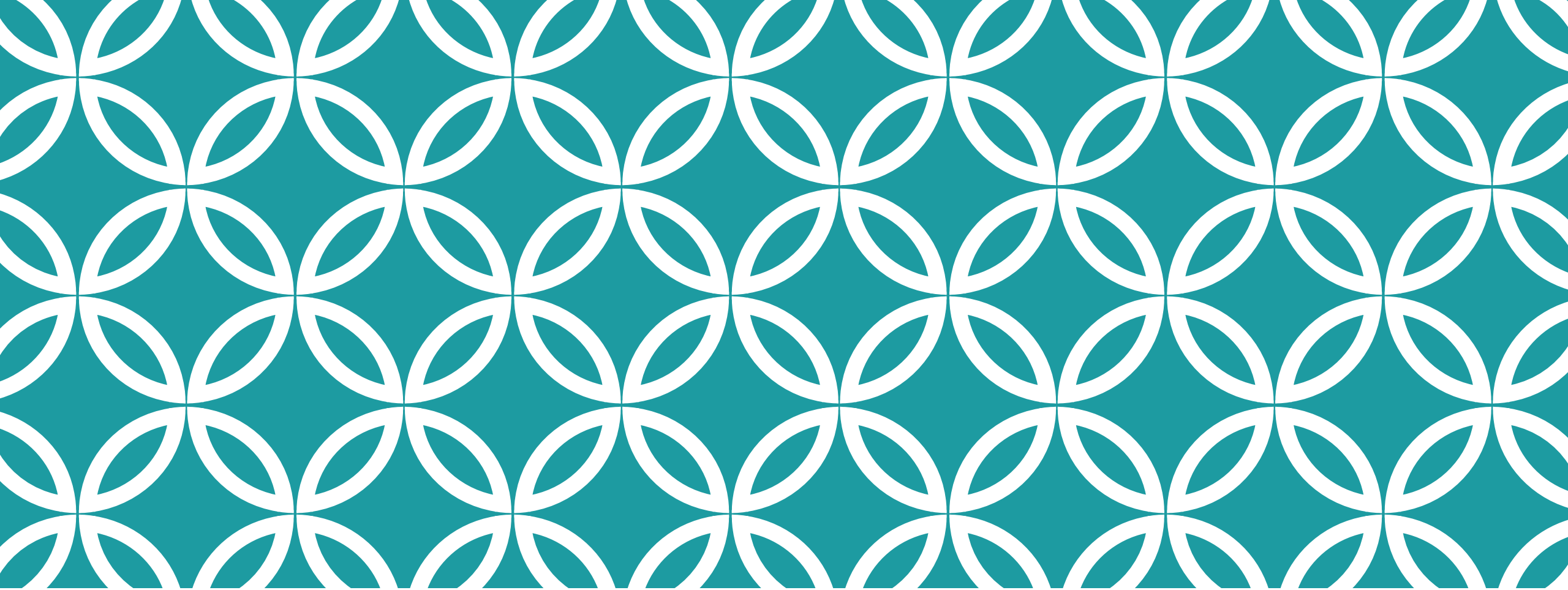

Results after analyzing10000 power traces

# Results byte 1 of roundkey 10 =0x11



# Results byte 8 of roundkey 10 =0xF3





# Template Attack

# Template Attack

- Is the Hamming Weight / Hamming Distance model always a good fit to the circuit?
  - Maybe not
- How can we find a good model for any circuit/implementation?
- Template attack
- The key idea is to first characterize the implementation with known keys, then use the created model to attack an implementation with an unknown key.
- A good template can make the template attack very efficient, i.e. only needs one power trace.

# Create a Template

- Two metrics: a mean vector and a covariance matrix: (m, **C**)
- On the device we are characterizing, we execute the target function with different data $d_i$ and keys $k_j$
- For each pair ($d_i$, $k_j$), record several power traces from repeated executions. Then, estimate the mean vector m and the covariance matrix **C** which models the multivariate normal distribution representing power trace samples.
- As a result, we obtain a template for every pair of data and key ($d_i$, $k_j$) => $(m,C)_{i,j}$

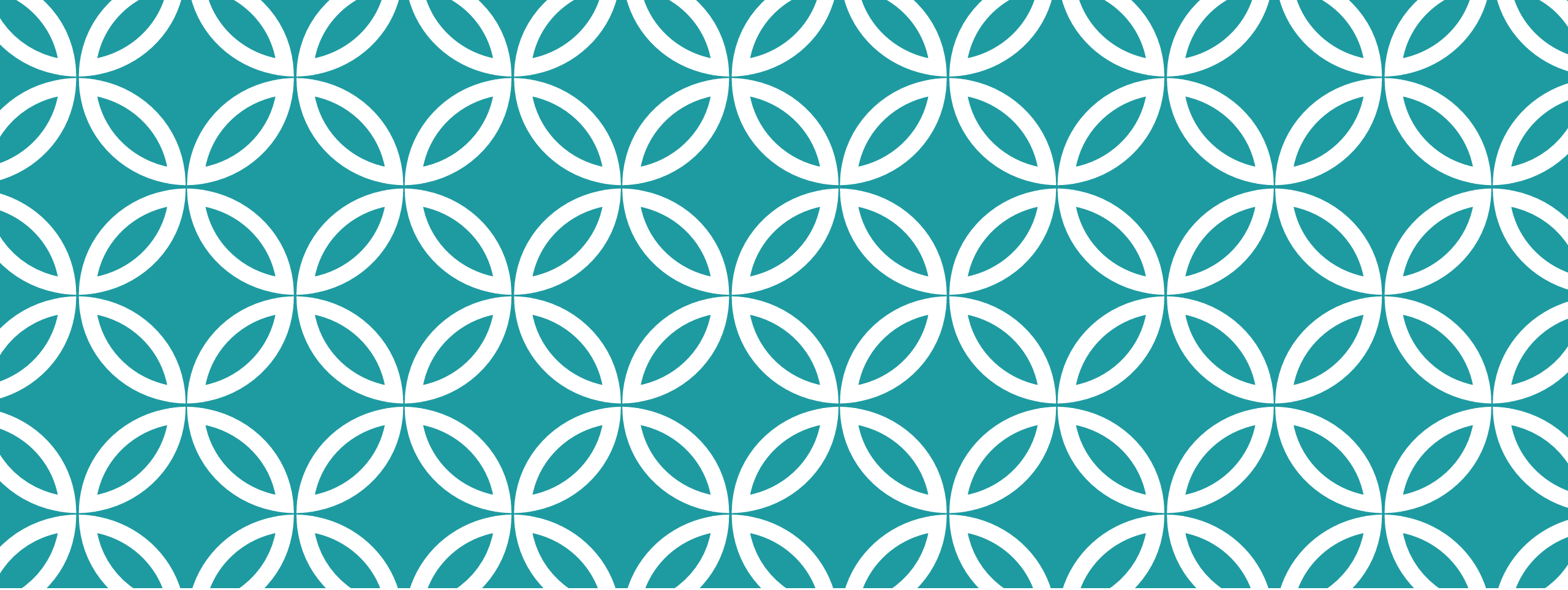
# Attacking Phase

- Given one power trace t, evaluate the following probability function with every template.

$$p(\mathbf{t}; (\mathbf{m}, \mathbf{C})_{d_i, k_j}) = \frac{exp\left(-\frac{1}{2} \cdot (\mathbf{t} - \mathbf{m})' \cdot \mathbf{C}^{-1} \cdot (\mathbf{t} - \mathbf{m})\right)}{\sqrt{(2 \cdot \pi)^T \cdot det(\mathbf{C})}}$$

Number of sampling points.

- According to maximum likelihood rule, the highest probability indicates the correct key.



# Countermeasures

# SCA Countermeasures: General Strategy

- Remove any execution time dependence on data and key, e.g., remove conditional execution that depends on key (Effective for SPA)
- Randomly insert instructions with no effect on the algorithm (Dummy operations/ instructions). Use different instructions that are hard to distinguish in a trace
- Shuffling, randomly shuffle the order of operations.
- Hiding – Hide sensitive information in trace
- Masking – Randomize the relationship between trace and data

# Remove instruction dependency

- Montgomery Ladder
  - Add a dummy multiplication when the secret bit is 0
  - Have a constant execution time
  - Prevent SPA perfectly, because SPA reveals the instruction flow.

$x_1=x;\ x_2=x^2$
for i=k-2 to 0 do
  If $n_i=0$ then
    $x_2=x_1*x_2;\ x_1=x_1^2$
  else
    $x_1=x_1*x_2;\ x_2=x_2^2$
return $x_1$

# Dummy Operations

- If we randomly insert D dummy operations, then we increase the difficulty of attacking by D + 1
- Do not insert NOP as dummy operation, because it is easy to distinguish in the trace.

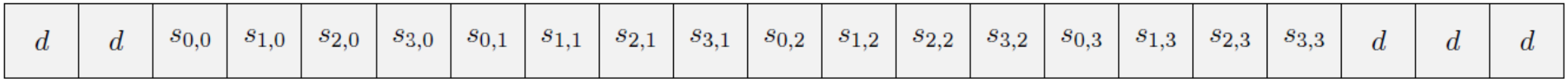
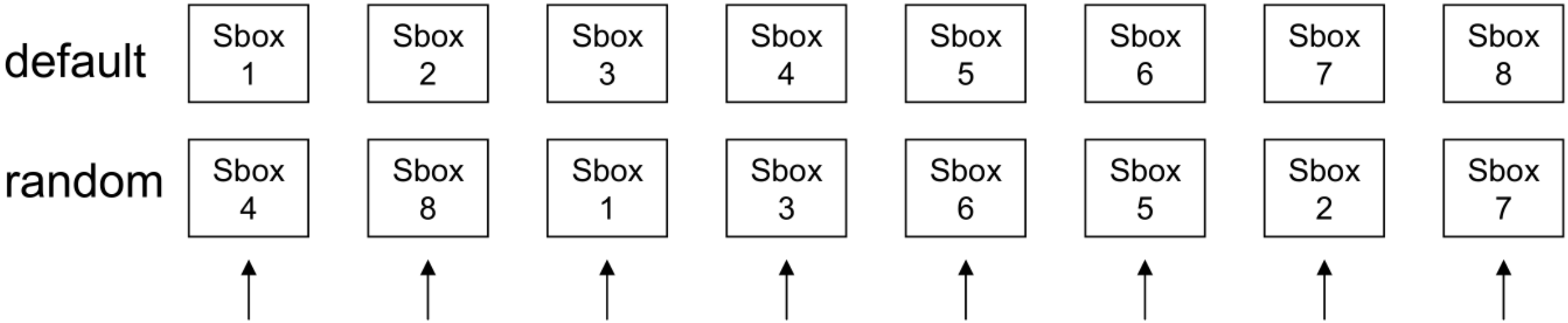



Figure 4.5: AES-State with dummy values

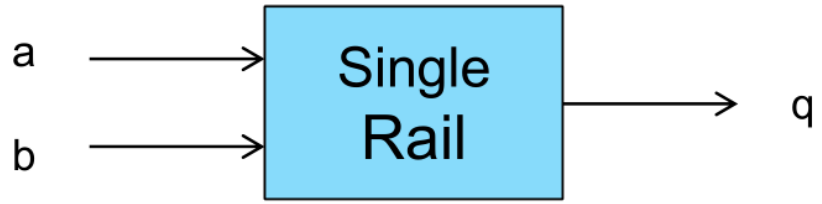
# Shuffling

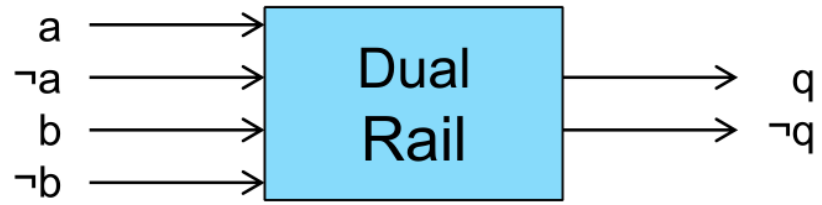
- Changing order of independent operations
- E.g. If we have 8 independent operations, we should have 8! shuffling possibilities, but this only increase the attack difficulty by 8. Therefore, we do not need to randomly shuffle the operation, we just need to randomly select a starting point and follow the original order.

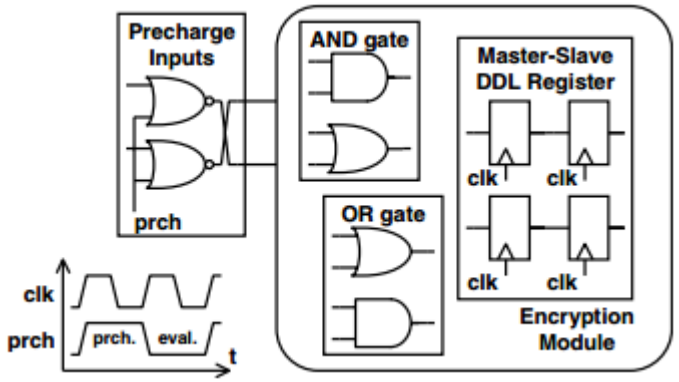


# Hiding

- Balancing the power consumption
  - Dual-rail Precharge Logic
  - Two wires carry one bit of information
- Precharging:
  - The clock is divided into two phases:
    - Precharging phase
      - Set the two wires q and not(q) into the same value
    - Evaluation phase
      - Set the two wires q and not(q) into the output of the dual rail gate

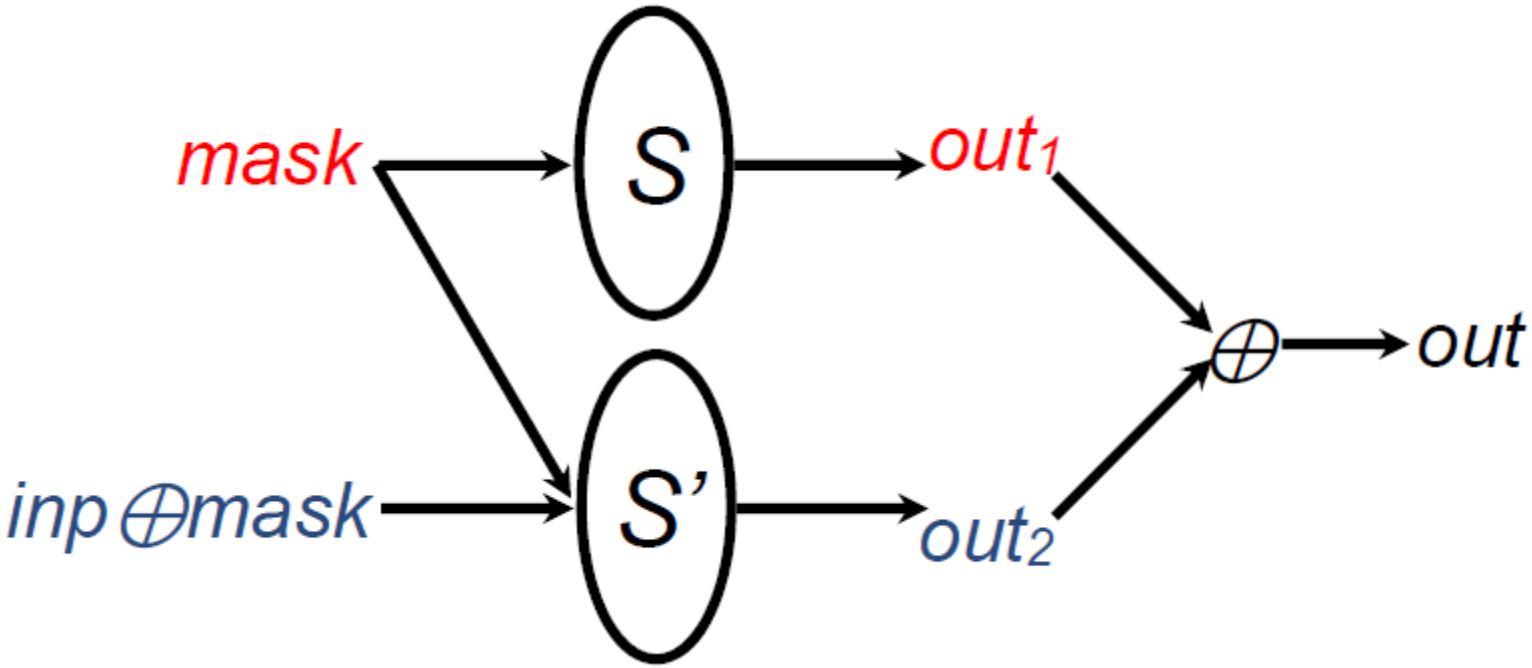


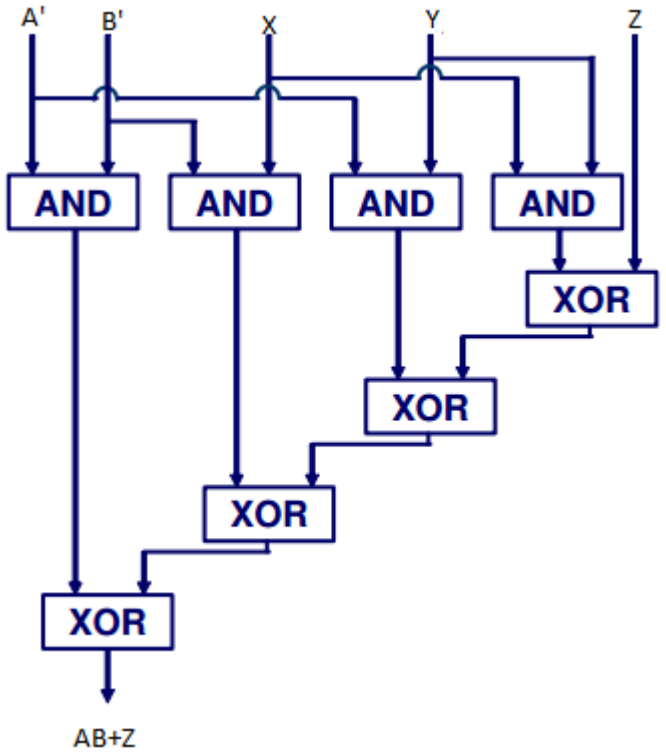
# Masking

- Only input and output value appear in the implementation, all the intermediate value is masked (XORed) with random number.



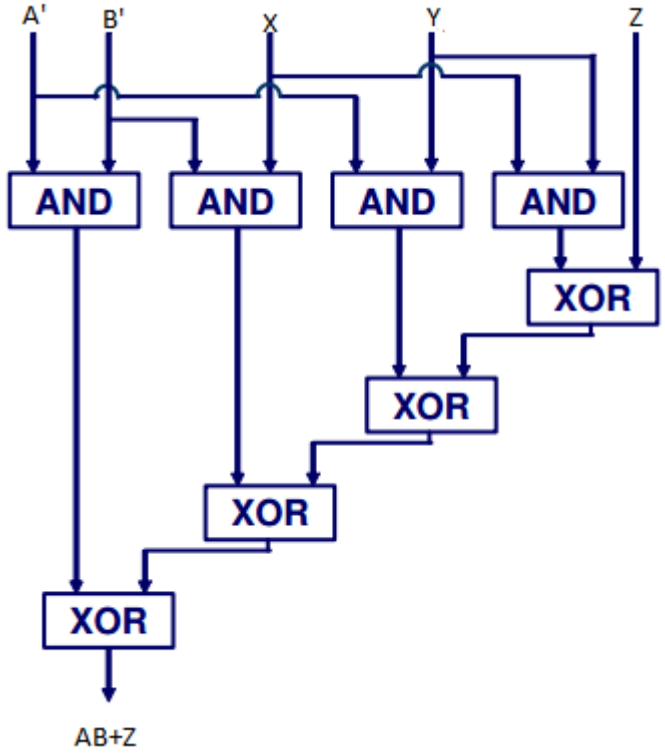
# Gate Level Masking

- Data A and B, Mask X and Y.
- A = A' + X
- B = B' + Y
- XOR operation under masking
  - A + B = (A' + B') + (X + Y)
- AND operation under masking
  - A B = (Z + (A' B') + (A' Y) + (B' X) + (X Y) ) + Z = (Z + AB) + Z
  - The order of + matters, we should + Z (new mask) first



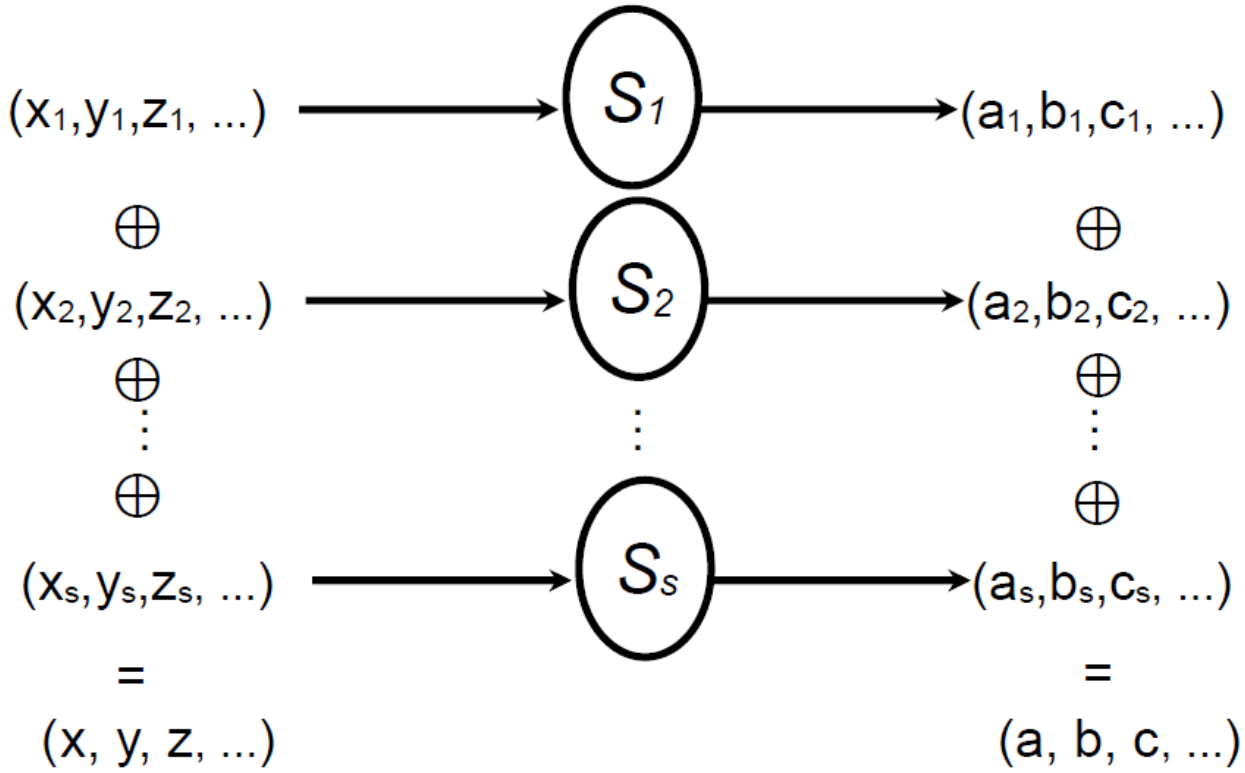
# Attack on Masking

- The order of XOR matters, but in hardware the propagation of a signal cannot be controlled precisely, due to the delay of different paths.
- If the propagation of Z is slower than the other four terms, the unmasked result will appear at the output. This phenomenon is called glitch.
- Masking is not secure when glitches can happen.
  - What is the remedy? Threshold implementation.



# Threshold Implementation

- Three properties of threshold implementation
  - Correctness
  - Non-completeness
  - Uniformity




Correctness

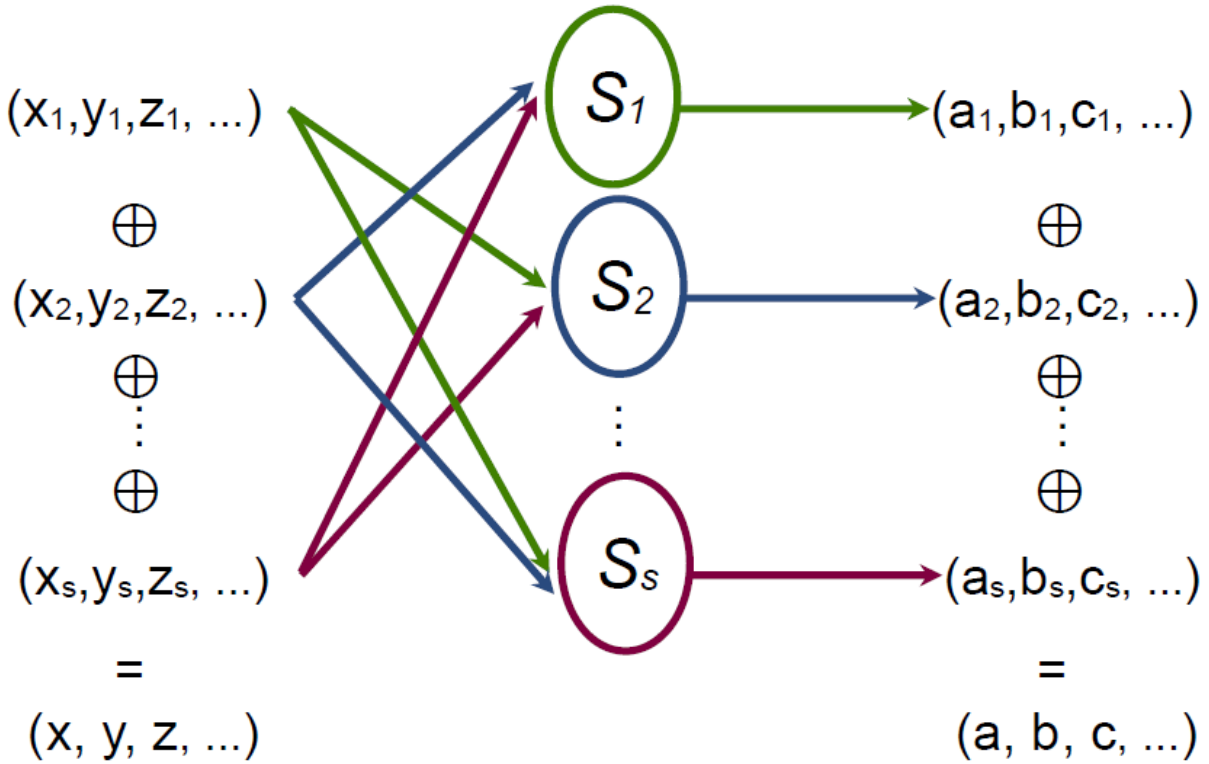
Natural, each masking should satisfy it.

# Threshold Implementation (TI)

- Three properties of threshold implementation
  - Correctness
  - Non-completeness
  - Uniformity

$$S(x,y,z) = x \oplus yz$$
$$= (x_1 \oplus x_2 \oplus x_3) \oplus (y_1 \oplus y_2 \oplus y_3)(z_1 \oplus z_2 \oplus z_3)$$

$$S_1(x_2,x_3,y_2,y_3,z_2,z_3) = x_2 \oplus y_2z_2 \oplus y_2z_3 \oplus y_3z_2$$
$$S_2(x_1,x_3,y_1,y_3,z_1,z_3) = x_3 \oplus y_3z_3 \oplus y_3z_1 \oplus y_1z_3$$
$$S_3(x_1,x_2,y_1,y_2,z_1,z_2) = x_1 \oplus y_1z_1 \oplus y_1z_2 \oplus y_2z_1$$

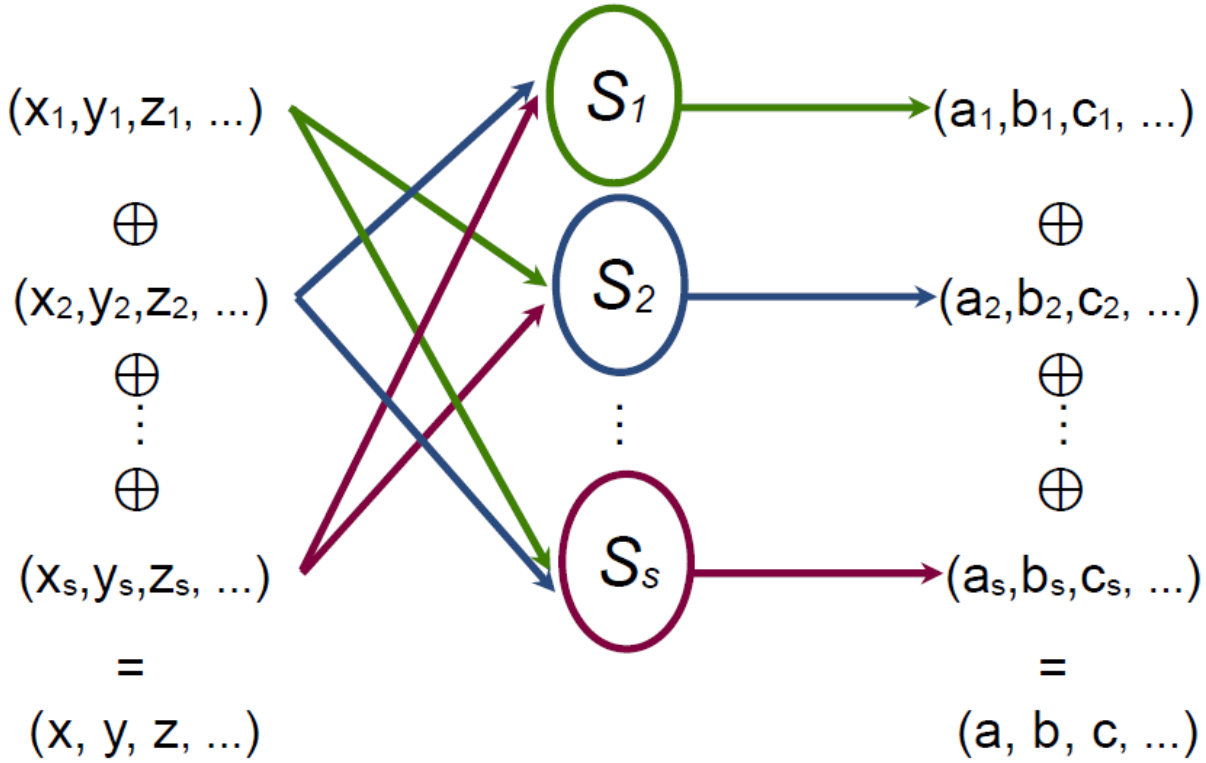



Non-completeness

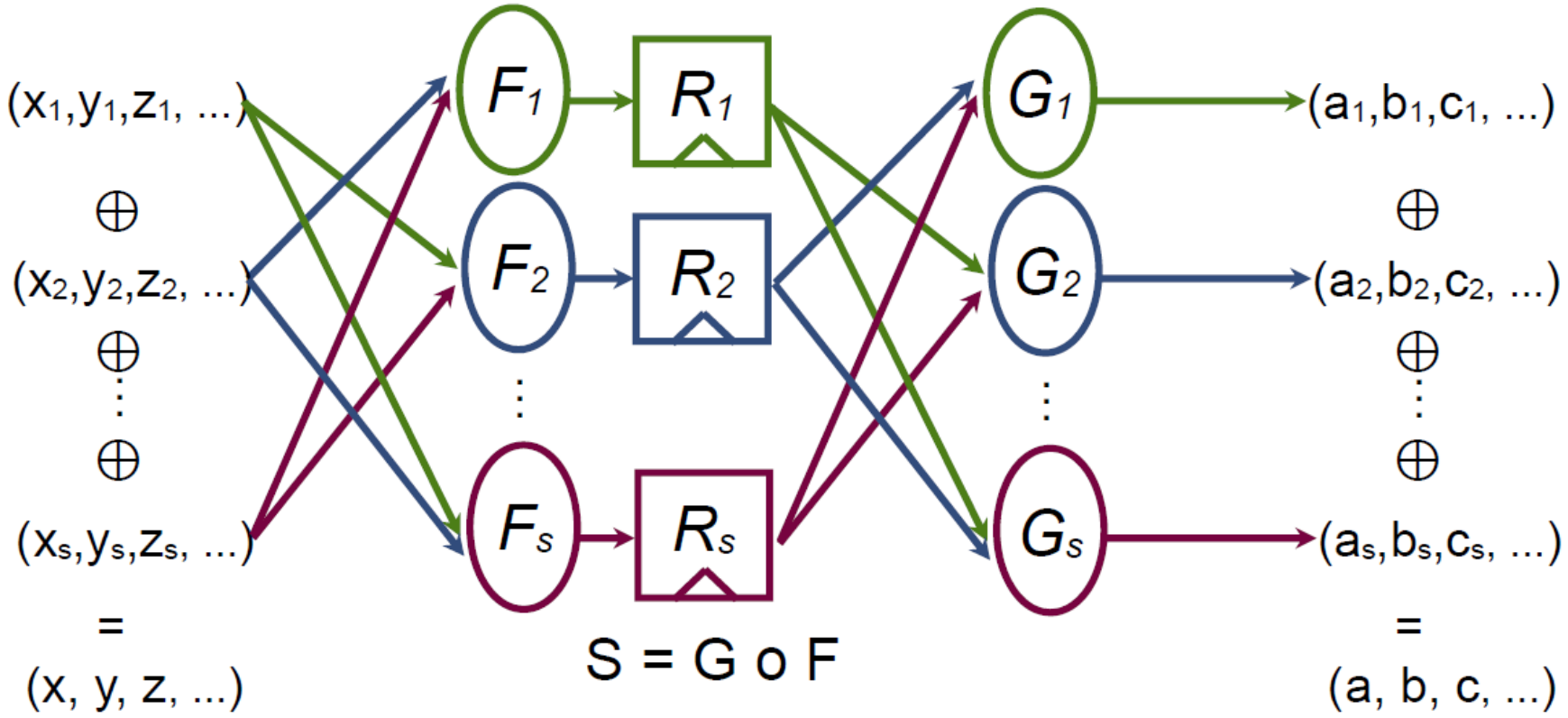
The most important TI property

# Threshold Implementation

- Three properties of threshold implementation
  - Correctness
  - Non-completeness
  - Uniformity
- Uniformity means the value of each share is uniformly distributed.



# Registers between nonlinear functions




Separate non-linear functions with registers!

Registers are important to stop the glitches.

# References [1]


1. http://ca.olin.edu/2005/fpga_dsp/fpga.html
2. http://ca.olin.edu/2005/fpga_dsp/fpga.html
3. https://upload.wikimedia.org/wikipedia/commons/5/5c/Side_channel_attack.png
4. https://en.wikipedia.org/wiki/Power_analysis
5. http://people.eku.edu/styere/Encrypt/JS-DES.html
6. http://www.tutorialspoint.com/cryptography/data_encryption_standard.htm
7. Paul Kocher, Joshua Jaff and Benjamin Jun: Differential Power Analysis
8. Farinaz Koushanfar: Physical Security and Side Channel Attacks (presentation)
9. http://www.embedded.com/print/4199399
10. Differential Power Analysis presented by Italo Dacoseta (presentation)

# References [2]


11. Differential Power Analysis Side Channel Attacks in Cryptography (Bachelor Thesis of William Hnath)
12. http://pubs.sciepub.com/iscf/3/1/1/
13. https://en.wikipedia.org/wiki/Advanced_Encryption_Standard
14. SIDE-CHANNEL ATTACKS ON HARDWARE IMPLEMENTATIONS OF CRYPTOGRAPHIC ALGORITHMS by Sıddıka Berna Ors Yal¸ cın (presentation)
15. Side Chanel Attack and Countermeasures for Embedded System by Job de Haas (presentation)
16. Overview of Countermeasures against Implementation Attack by Macel Medwed (presentation)
17. SPA and DPA attacks. Pascal Paillier
18. Lecture7 –More on Attacks Farinaz Koushanfar
19. Power Analysis Attacks. Elisabeth Oswald
20. Power Analysis – an overview. Benedikt Gierlichs

# References [3]


21. https://cryptography.gmu.edu/fobos/
22. https://en.wikipedia.org/wiki/Side-channel_attack
23. Paul Kocher. “Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems”. Crypto’96
24. Daniel J. Bernstein. “Cache-timing attacks on AES”. 2005
25. Paul Kocher, Joshua Jaffe and Benjamin Jun. ”Differential Power Analysis”. Crypto’99
26. Karine Gandolfi, Christophe Mourtel, and Francis Olivier. “Electromagnetic Analysis: Concrete Results”. CHES’01
27. Daniel Genkin, Adi Shamir and Eran Tromer. “RSA Key Extraction via Low-Bandwidth Acoustic Cryptanalysis”. CRYPTO’14
28. Alexander Schlösser, Dmitry Nedospasov, Juliane Krämer, Susanna Orlic and Jean-Pierre Seifert. “Simple Photonic Emission Analysis of AES Photonic Side Channel Analysis for the Rest of Us”. CHES’12
29. David Brumleya and Dan Boneh, “Remote timing attacks are practical”. Computer Networks’05.
30. Preneel, Bart and Paar, Christof and Pelzl, Jan. “Understanding cryptography: a textbook for students and practitioners”. Springer 2009

# References [4]


31. Bellare M, Rogaway P. Optimal asymmetric encryption EUROCRYPT'94
32. https://en.wikipedia.org/wiki/Optimal_asymmetric_encryption_padding
33. https://en.wikipedia.org/wiki/Montgomery_modular_multiplication
34. https://en.wikipedia.org/wiki/Karatsuba_algorithm
35. https://github.com/stoutbeard/crypto
36. Osvik D A, Shamir A, Tromer E. Cache attacks and countermeasures: the case of AES[M]//Topics in Cryptology–CT-RSA 2006. Springer Berlin Heidelberg, 2006: 1-20.
37. Chongxi Bao and Ankur Srivastava. "3D Integration: New Opportunities in Defense Against Cache-timing Side-channel Attacks". ICCD'15.
38. Gorka Irazoqui, Thomas Eisenbarth and Berk Sunar, "S$A: A Shared Cache Attack that Works Across Cores and Defies VM Sandboxing—and its Application to AES", Oakland'15
39. Yarom Y, Falkner K. Flush+ reload: a high resolution, low noise, L3 cache side-channel attack. USENIX Security 14
40. Gruss D, Maurice C, Wagner K. Flush+ Flush: A Stealthier Last-Level Cache Attack. arXiv preprint arXiv:2015

# References [5]


41. Roberto Guanciale, Hamed Nemati, Christoph Baumann, and Mads Dam, “Cache Storage Channels: Alias-Driven Attacks and Verified Countermeasures”, S&P’16
42. Zhenghong Wang and Ruby B. Lee “New cache designs for thwarting software cache-based side channel attacks”, ISCA 2007.
43. Kong J, Aciicmez O, Seifert J P, et al. Deconstructing new cache designs for thwarting software cache-based side channel attacks, ACM workshop on Computer security architectures. 2008
44. Costan V, Lebedev I, Devadas S. Sanctum: Minimal Hardware Extensions for Strong Software Isolation[J].
45. Danfeng Zhang, Yao Wang, G. Edward Suh and Andrew C. Myers. “A Hardware Design Language for Timing-Sensitive Information-Flow Security”. ASPLOS’15
46. Mangard S, Oswald E, Popp T. Power analysis attacks: Revealing the secrets of smart cards[M]. Springer Science & Business Media, 2008.
47. Costan V, Devadas S. Intel SGX Explained[J]. IACR Cryptology ePrint Archive, 2016, 2016: 86.
48. Hoheisel A. Side-Channel Analysis Resistant Implementation of AES on Automotive Processors[D]. Master’s thesis, Ruhr-University Bochum, Germany (June 2009), 2009.
49. Tiri K, Verbauwhede I. A logic level design methodology for a secure DPA resistant ASIC or FPGA implementation[C]//Proceedings of the conference on Design, automation and test in Europe-Volume 1. IEEE Computer Society, 2004: 10246.
50. Trichina E. Combinational Logic Design for AES SubByte Transformation on Masked Data[J]. IACR Cryptology ePrint Archive, 2003, 2003: 236.

# References [6]


51. John T M, Haider S K, Omar H, et al. Connecting the Dots: Privacy Leakage via Write-Access Patterns to the Main Memory[J]. arXiv preprint arXiv:1702.03965, 2017.

52. Mangard S, Pramstaller N, Oswald E. Successfully attacking masked AES hardware implementations[C]//International Workshop on Cryptographic Hardware and Embedded Systems. Springer Berlin Heidelberg, 2005: 157-171.

53. Nikova S, Rechberger C, Rijmen V. Threshold implementations against side-channel attacks and glitches[C]//International Conference on Information and Communications Security. Springer Berlin Heidelberg, 2006: 529-545.

54. Ventzi Nikov, presentation on “Threshold Implementation”18.03.2016

55. Kutzner S, Nguyen P H, Poschmann A. Enabling 3-share threshold implementations for all 4-bit s-boxes[C]//International Conference on Information Security and Cryptology. Springer International Publishing, 2013: 91-108.