# Cryptographic Engineering

## Lecture 2: Side-channel analysis

## February 12, 2024

Assist. Prof. Guilherme Perin| Leiden

Universiteit Leiden

# Agenda

- What is side-channel analysis?
- How and why does side-channel analysis work?
- Cryptography: a short review (RSA, AES)
- Side-channel analysis on RSA
- Side-channel analysis on AES

- Take-away messages:
  - Embedded cryptography can be highly vulnerable to physical (side-channel) attacks.
  - It is very easy to steal cryptographic keys when no countermeasures (protections) are considered.
  - Side-channel analysis is a very simple process (and it can be very complex).

# What is side-channel analysis (SCA)?

- It is a technique that analyses the **information that leaks** from electronic devices, systems, software, etc.

- It is the **unintended leakage of information** carrying relevant data from hardware or software implementations. (Unintended = it happens regardless efforts to keep a product very secure).

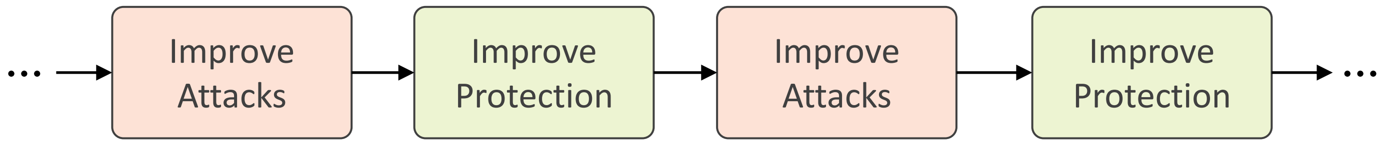
- Non-invasive attack: the victim does not know that it is being attacked.

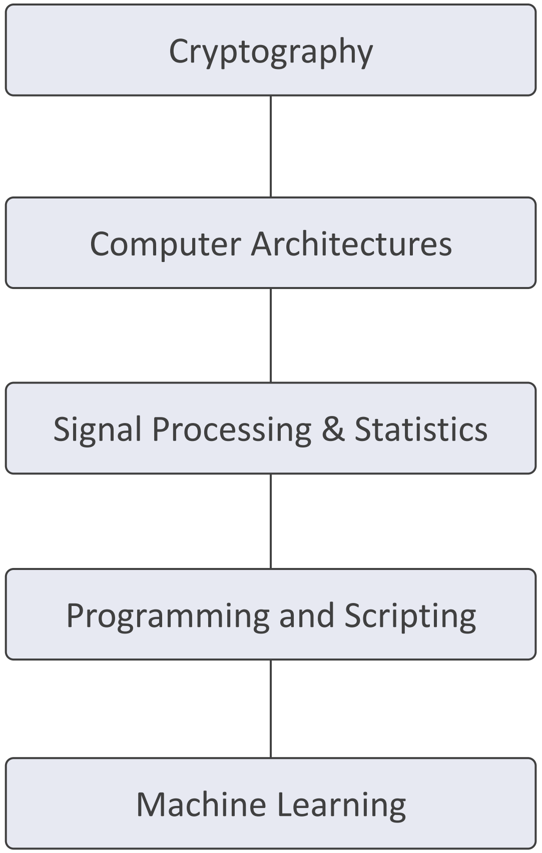
Remark:

- This course is mostly about cryptography and their implementation concepts: how to make crypto implementations secure and how to compromise their security.

# Side-channel analysis (SCA)

- Multi-disciplinary field.
- Research on side-channel analysis looks for most advanced and powerful attacks to know how to create highly secure products:



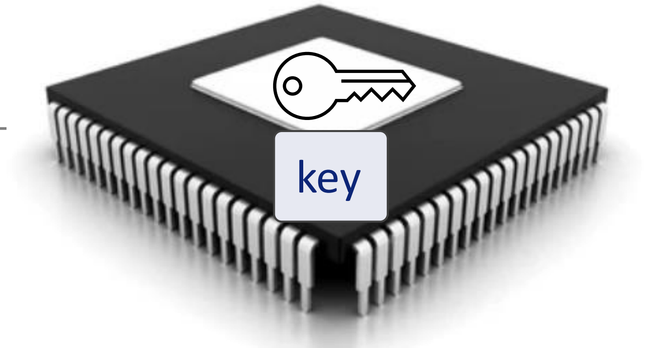



# Cryptographic implementations are secure. Are they?

Encryption/Decryption

Ciphertext / Plaintext

...00101001001000111 00...




Ciphertext / Plaintext

...11101011001101110110...

Physical Device

Power Consumption

Electromagnetic Emissions

Execution Time

Sound

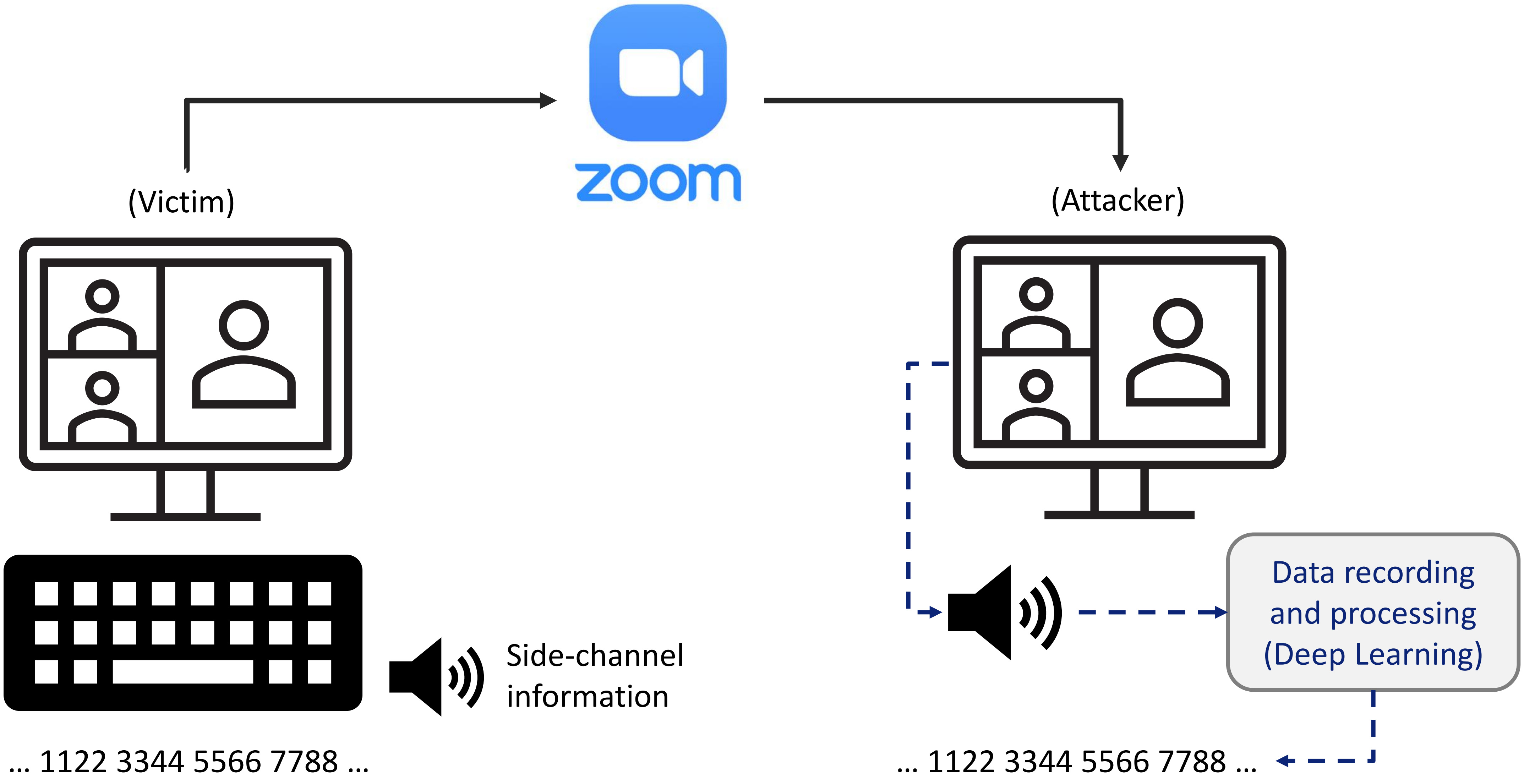
Temperature Variations

Photonic Emissions

Side-Channels

**In case of cryptographic implementations, side-channels carry information about the key.**

# When do side-channels happen?



# When do side-channels happen?

## A Practical Deep Learning-Based Acoustic Side Channel Attack on Keyboards

Joshua Harrison[1], Ehsan Toreini[2], and Maryam Mehrnezhad[3]

[1]Durham University, joshua.b.harrison@durham.ac.uk
[1]University of Surrey, e.toreini@surrey.ac.uk
[1]Royal Holloway University of London,
maryam.mehrnezhad@rhul.ac.uk

August 3, 2023

**Abstract**

With recent developments in deep learning, the ubiquity of microphones and the rise in online services via personal devices, acoustic side channel attacks present a greater threat to keyboards than ever. This paper presents a practical implementation of a state-of-the-art deep learning model in order to classify laptop keystrokes, using a smartphone integrated microphone. When trained on keystrokes recorded by a nearby phone, the classifier achieved an accuracy of 95%, the highest accuracy seen without the use of a language model. When trained on keystrokes recorded using the video-conferencing software Zoom, an accuracy of 93% was achieved, a new best for the medium. Our results prove the practicality of these side channel attacks via off-the-shelf equipment and algorithms. We discuss a series of mitigation methods to protect users against these series of attacks.

# Most important types of side-channel analyses

- **Timing**:
  - Can be done remotely (e.g., cache-based attacks).
  - Steal private data from data servers.
  - Very difficult to prevent.

- **Power consumption**:
  - Needs physical access to the device.
  - Easy and very powerful (lot of confidential information leaks through power consumption).

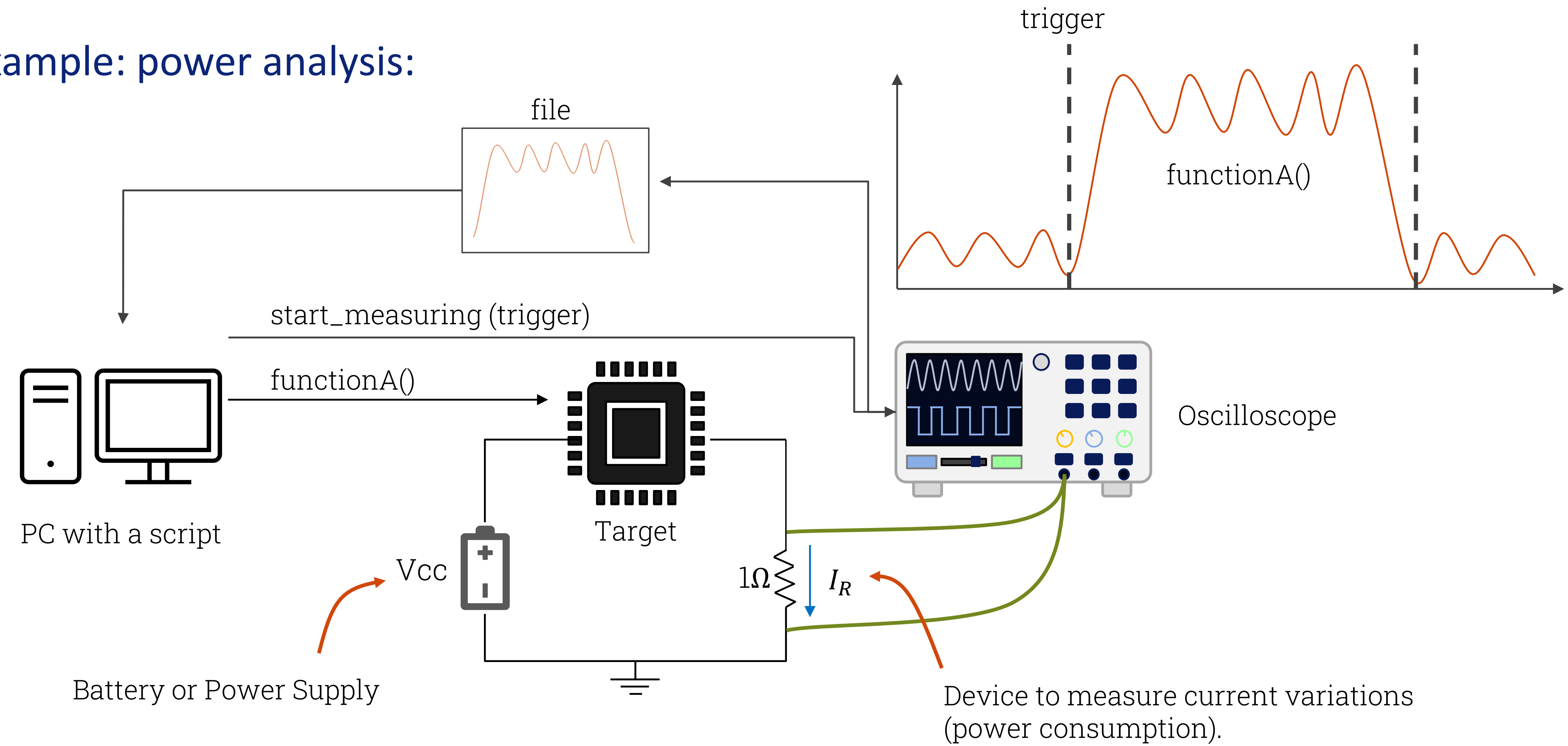
- **Electromagnetic emission**:
  - Needs less physical access to the device (antenna or near field measurements)
  - Very powerful
  - More complex (position of the measurement equipment is important).

# In summary…

- Side-channel analysis is a technique to extract secrets through physical means (sound, power, time, temperature, etc.).
- There is a target under evaluation (hardware, software, system) that process private information (cryptographic keys, passwords, user data, etc.).
- **Special equipment is necessary to measure side-channel information.**

# Measuring side-channels

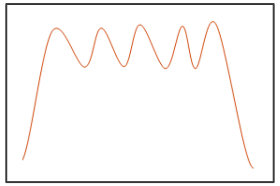
- Example: power analysis:

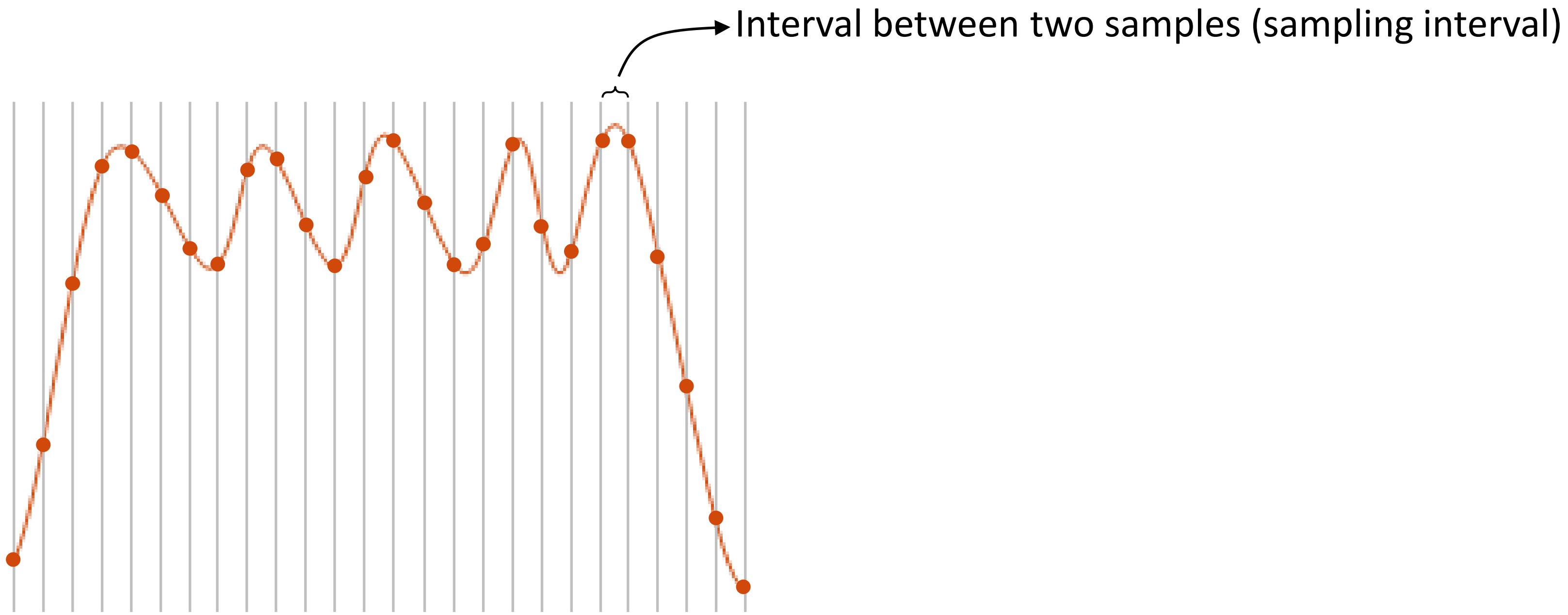


# Measuring side-channels

- Discrete values

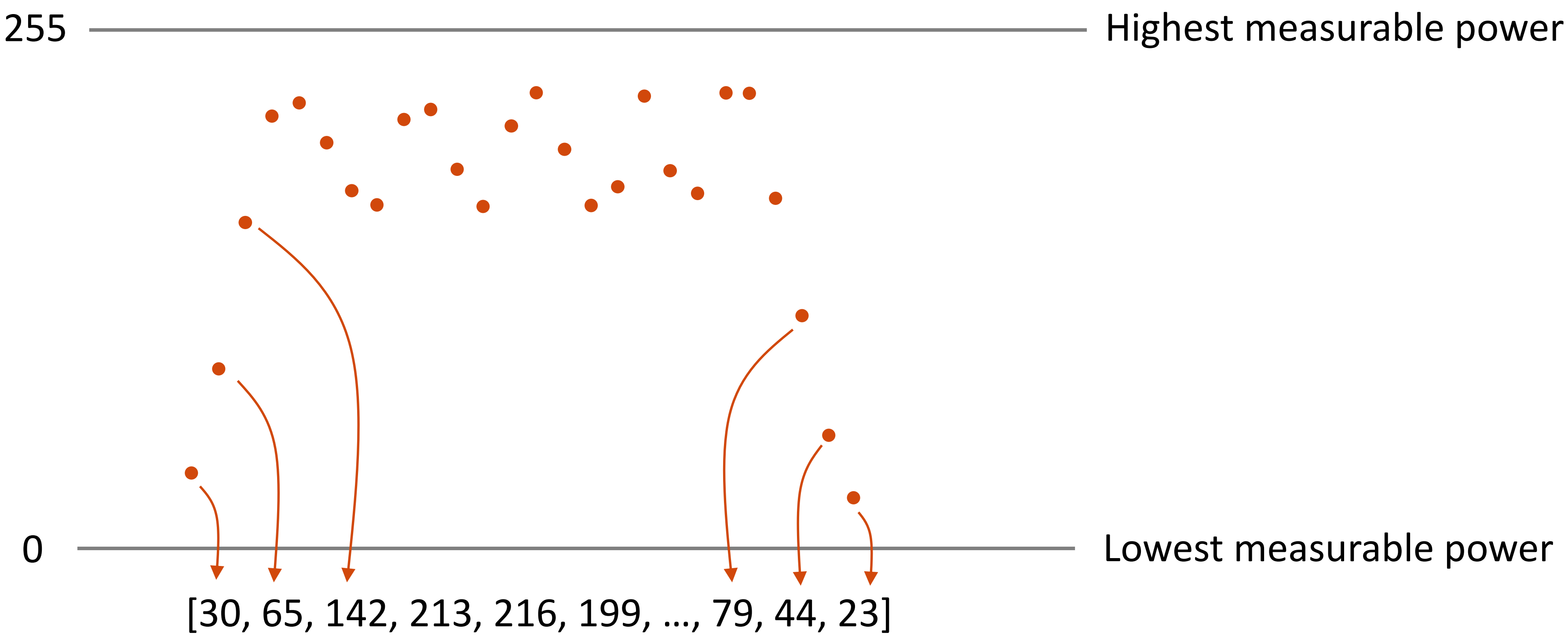
file





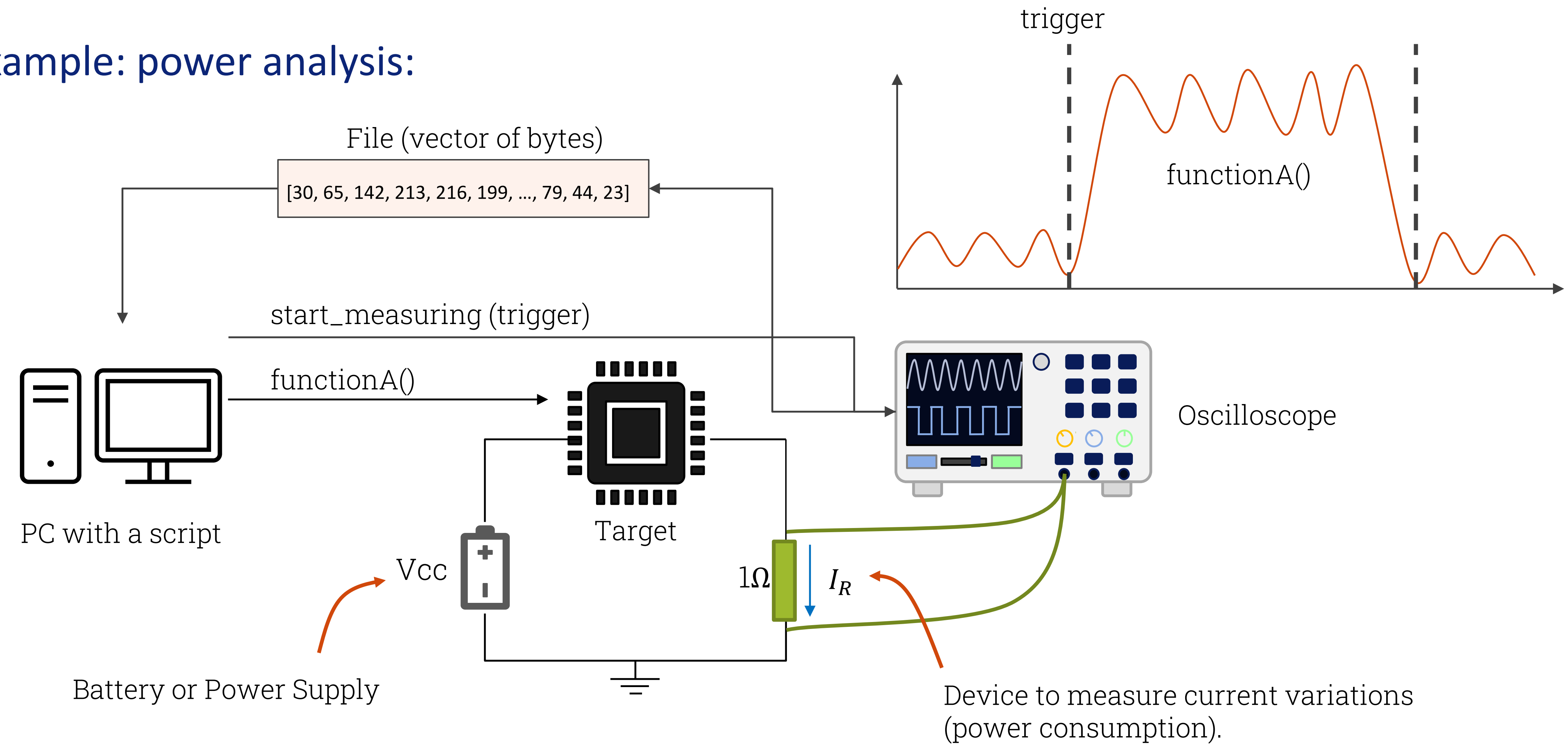
# Measuring side-channels

- Discrete values:
  - Every sample point is saved as a byte value $\in [0, 255]$.



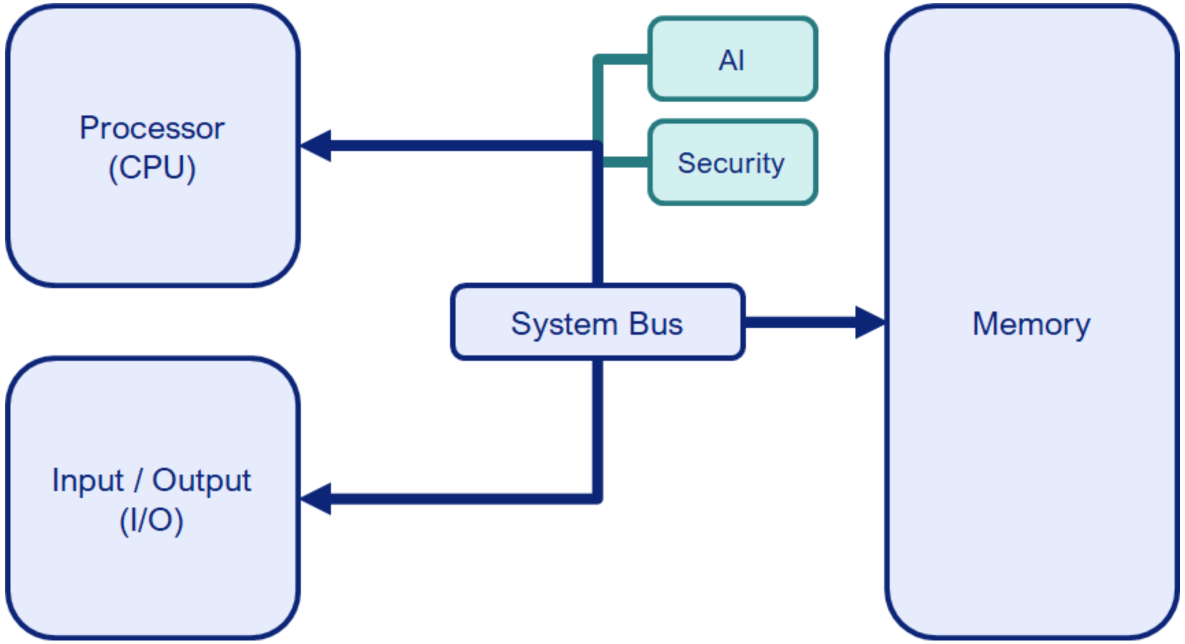
# Measuring side-channels

- Example: power analysis:

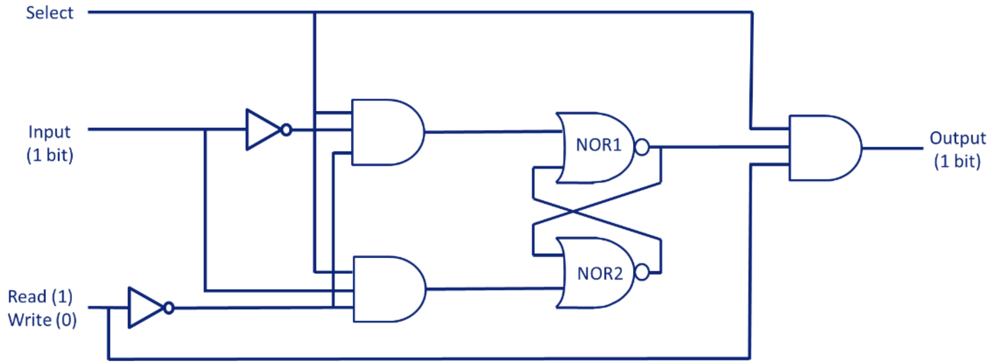


# Why do side-channels carry information about internal states of a device?

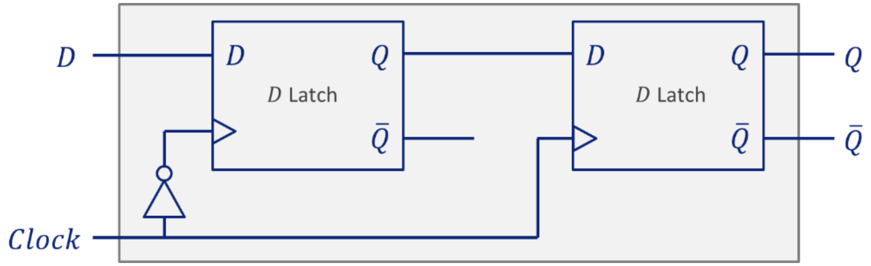
- Integrated circuits consist (mostly) of digital logic design: processors and memories.




Basic computer architecture.



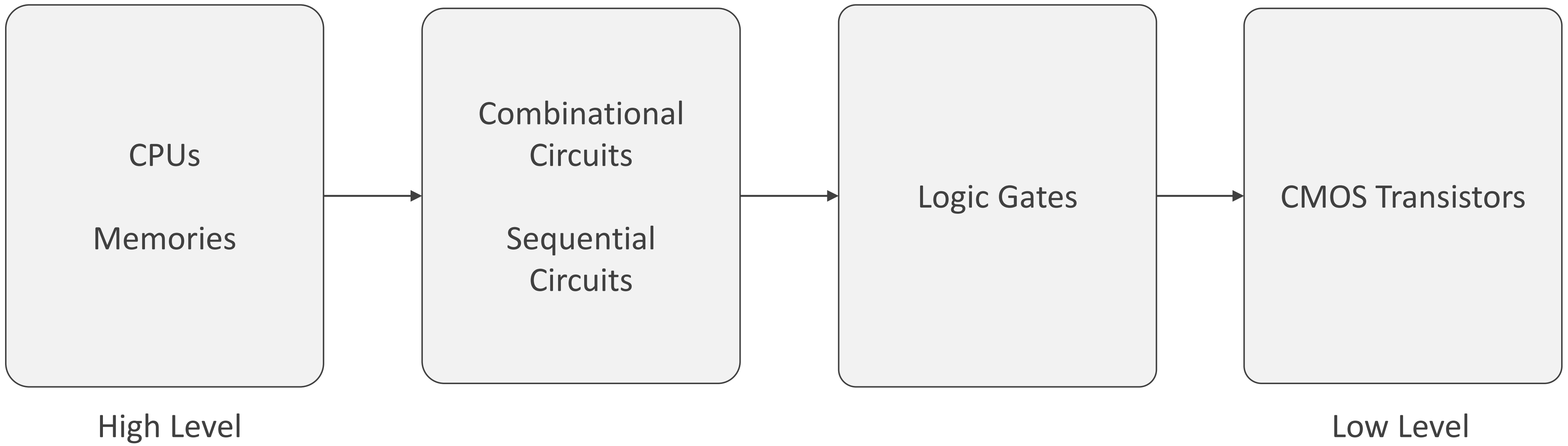

Basic 1-bit memory cell with a SR latch.




D-type flip-flop

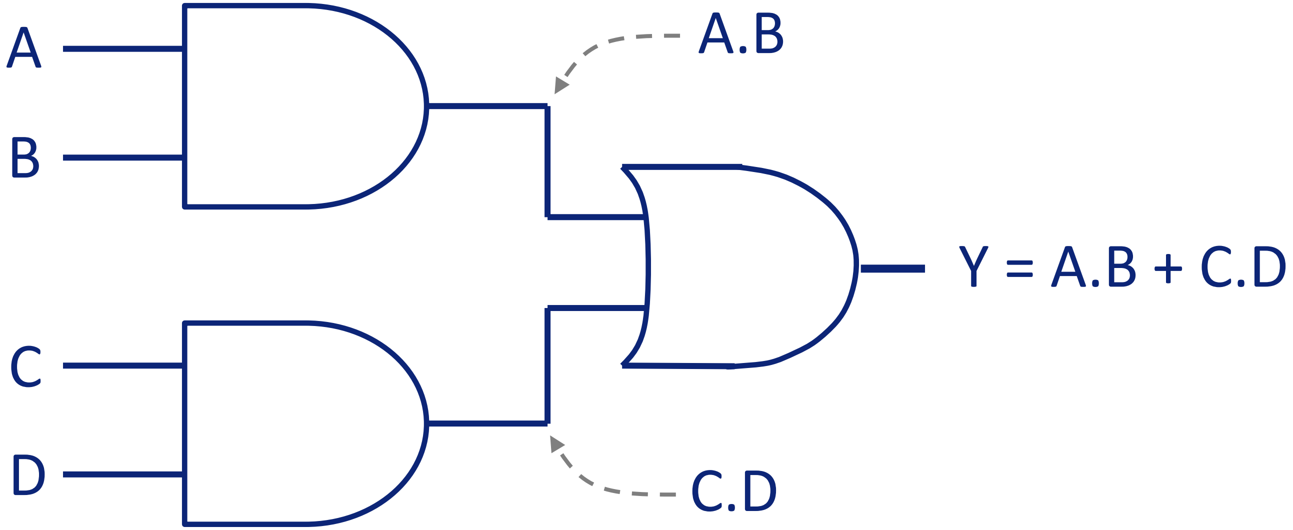
# Why do side-channels carry information about internal states of a device?

- Integrated circuits consist (mostly) of digital logic design: processors and memories.



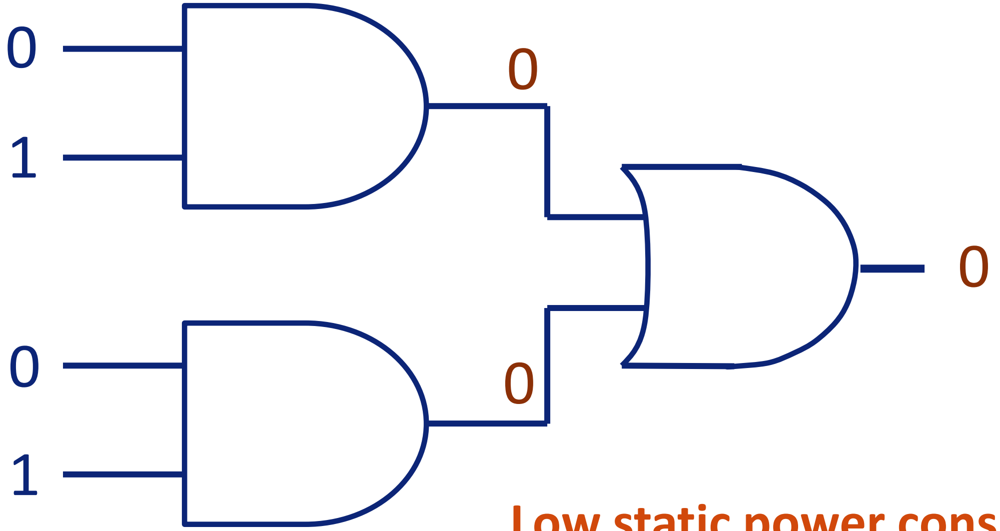
# Why do side-channels carry information about internal states of a device?

- Power consumption is proportional to the internal activity of a logic circuit.



# Why do side-channels carry information about internal states of a device?

- Power consumption is proportional to the internal activity of a logic circuit.



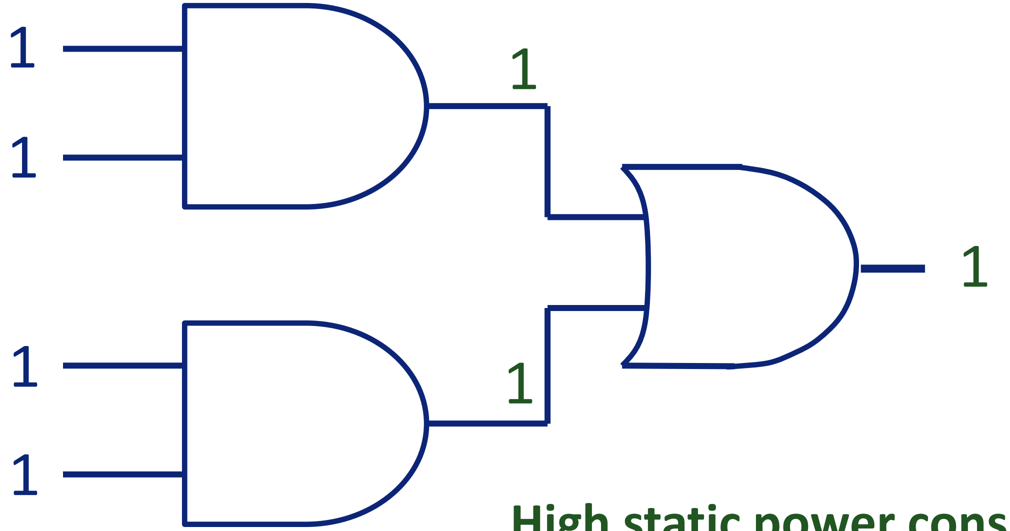

Low static power consumption.

**Static power consumption:**

- Power consumption proportional to the **current** logic values.

# Why do side-channels carry information about internal states of a device?

- Power consumption is proportional to the internal activity of a logic circuit.



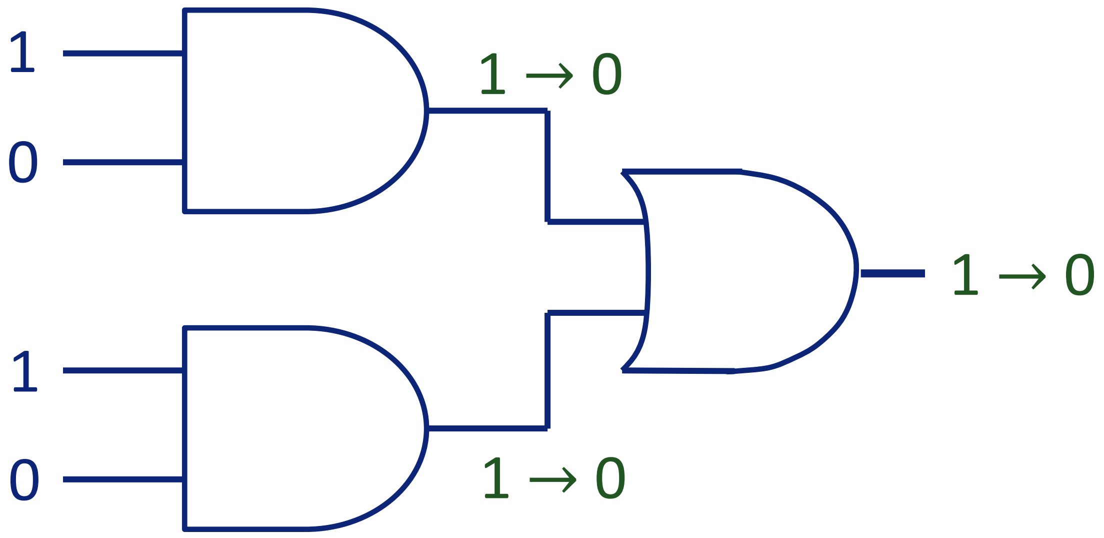

High static power consumption.

**Static power consumption:**

- Power consumption proportional to the **current** logic values.

# Why do side-channels carry information about internal states of a device?

- Power consumption is proportional to the internal activity of a logic circuit.



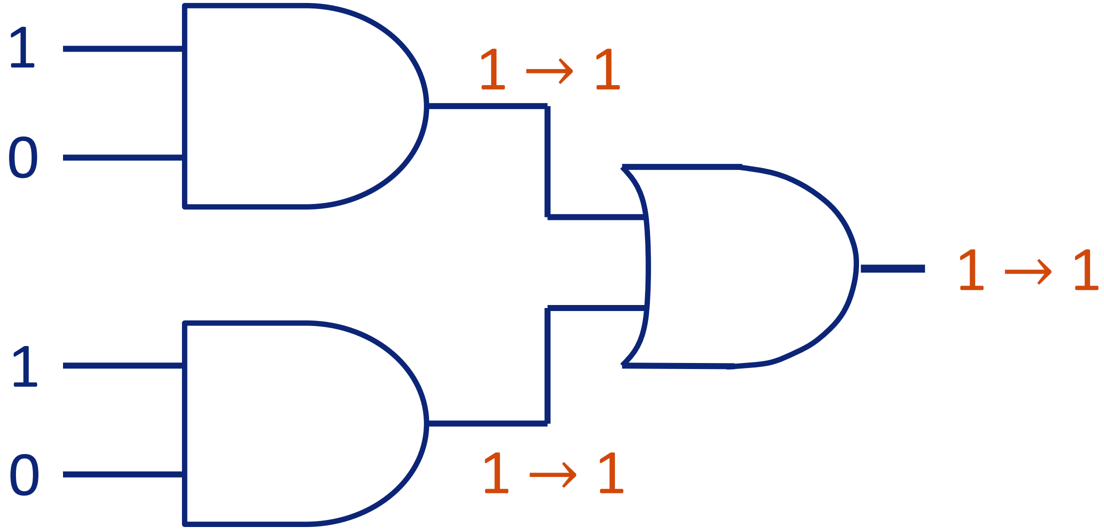

High dynamic power consumption.

**Dynamic power consumption:**

- Power consumption proportional to the **changes** in logic values.

# Why do side-channels carry information about internal states of a device?

- Power consumption is proportional to the internal activity of a logic circuit.




**Low dynamic power consumption.**

**Dynamic power consumption:**

- Power consumption proportional to the **changes** in logic values.

**Total power consumption: static + dynamic power consumption.**

# Measuring Power Consumption

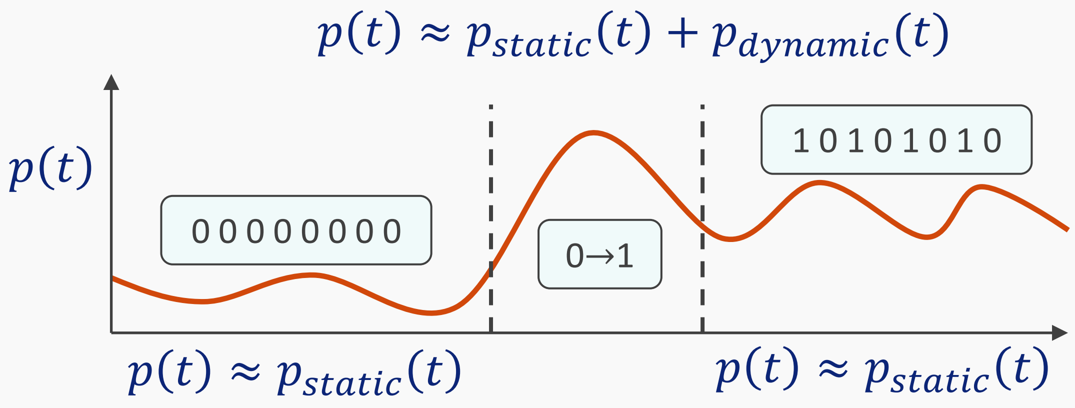
- **The overall power consumption of a circuit at time $t$ is the overall sum of all the power consumed by all the switching activities inside the circuit plus the static power.**

- Example: 8-bit register A in a CPU:

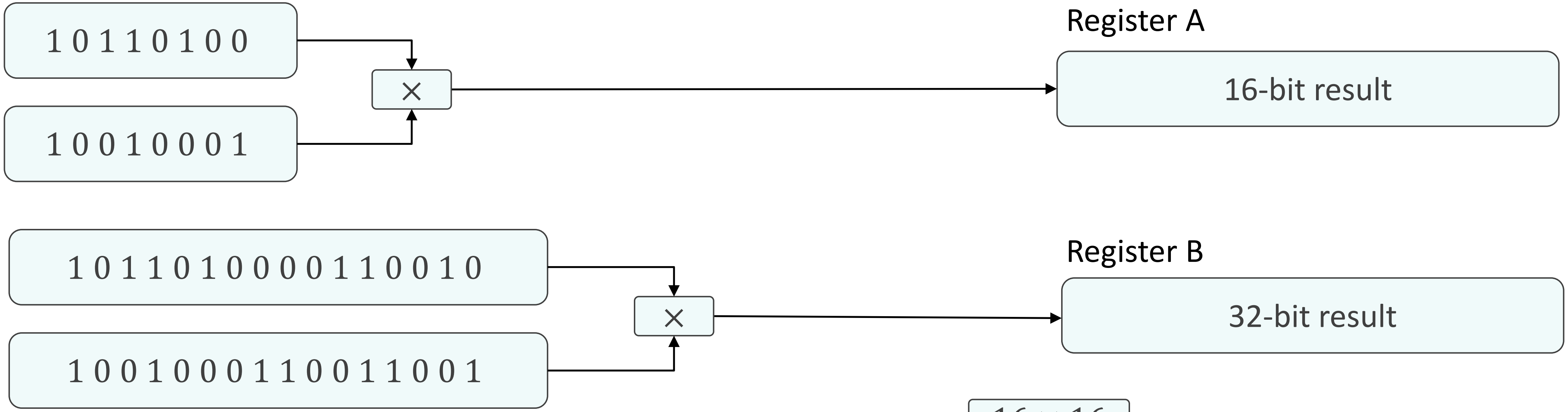
| Register A | | Register A |
|---|---|---|
| 0 0 0 0 0 0 0 0 | → Some Instruction → | 1 0 1 0 1 0 1 0 |

Power consumption:

$$p(t) \approx p_{static}(t) + p_{dynamic}(t)$$



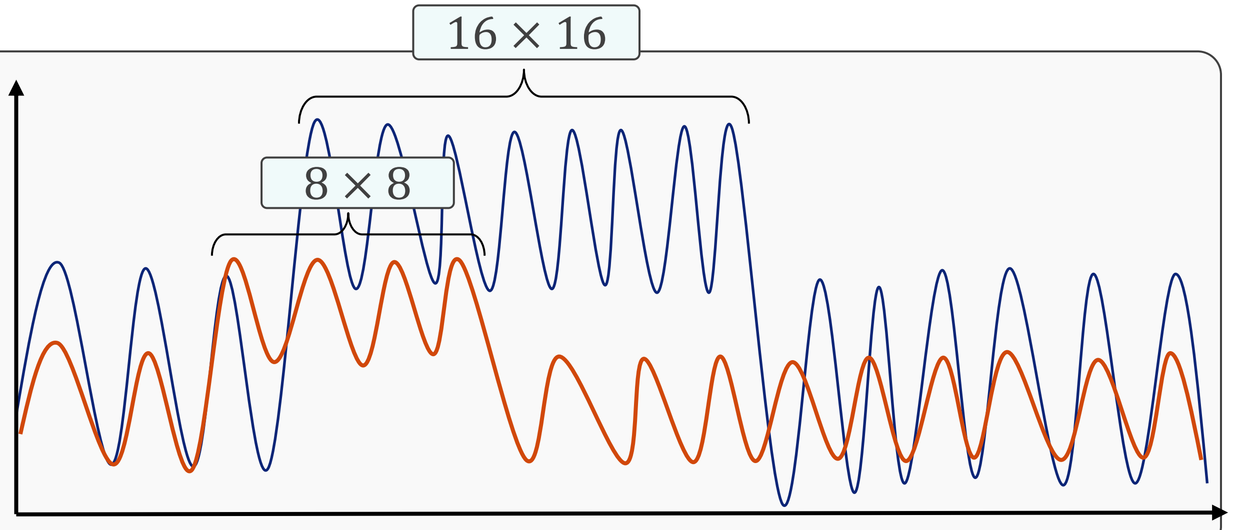
# Measuring Power Consumption




Power consumption:

$p_A(t) \approx p_{static\_A}(t) + p_{dynamic\_A}(t)$

$p_B(t) \approx p_{static\_B}(t) + p_{dynamic\_B}(t)$



# Timing Analysis

4-digit PIN verification method:

```
readPIN(pin);
PIN_verify(pin) {
    if pin[0] == 6:
        if pin[1] == 2:
            if pin[2] == 7:
                if pin[3] == 9:
                    pin_ok = 1
                end;
            end;
        end;
    end;
    return pin_ok;
}
```

digit by digit

What takes more time?

```
Pin_verify(6289)?
```

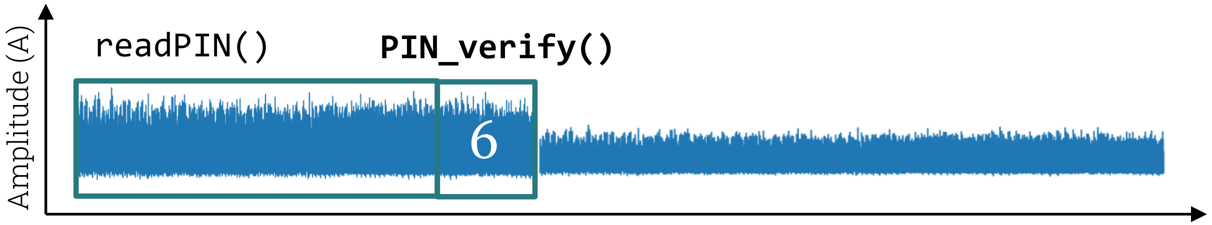
or

```
Pin_verify(6179)?
```

# Timing Analysis

4-digit PIN verification method:

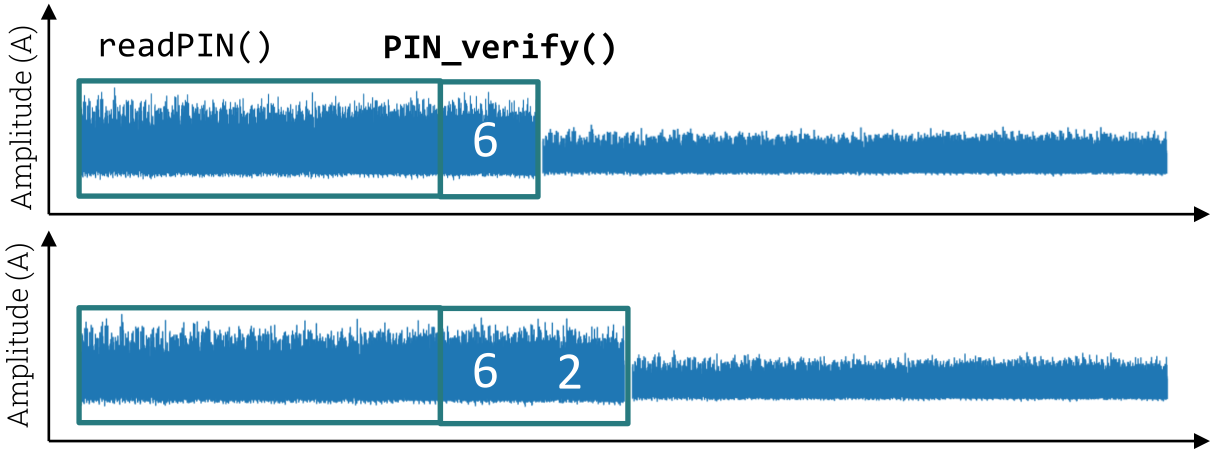
```
readPIN(pin);
PIN_verify(pin) {
    if pin[0] == 6:
        if pin[1] == 2:
            if pin[2] == 7:
                if pin[3] == 9:
                    pin_ok = 1
                end;
            end;
        end;
   end;
   return pin_ok;
}
```



# Timing Analysis

4-digit PIN verification method:

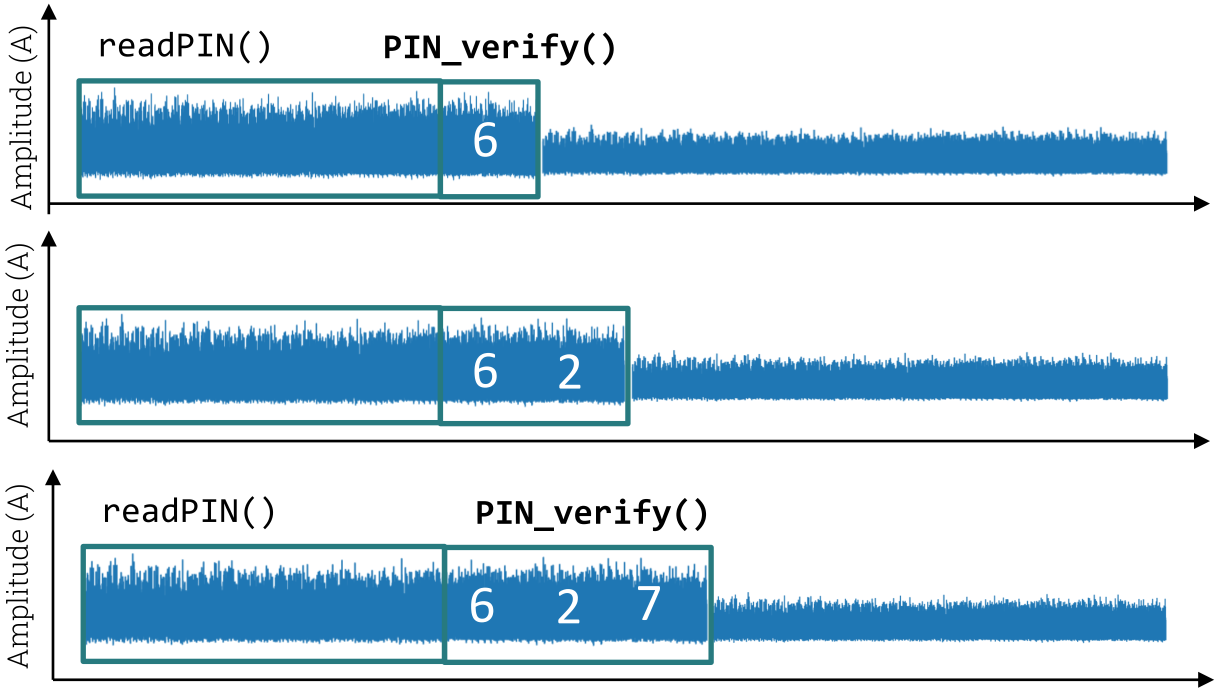
```
readPIN(pin);
PIN_verify(pin) {
    if pin[0] == 6:
        if pin[1] == 2:
            if pin[2] == 7:
                if pin[3] == 9:
                    pin_ok = 1
                end;
            end;
        end;
    end;
    return pin_ok;
}
```



# Timing Analysis

4-digit PIN verification method:

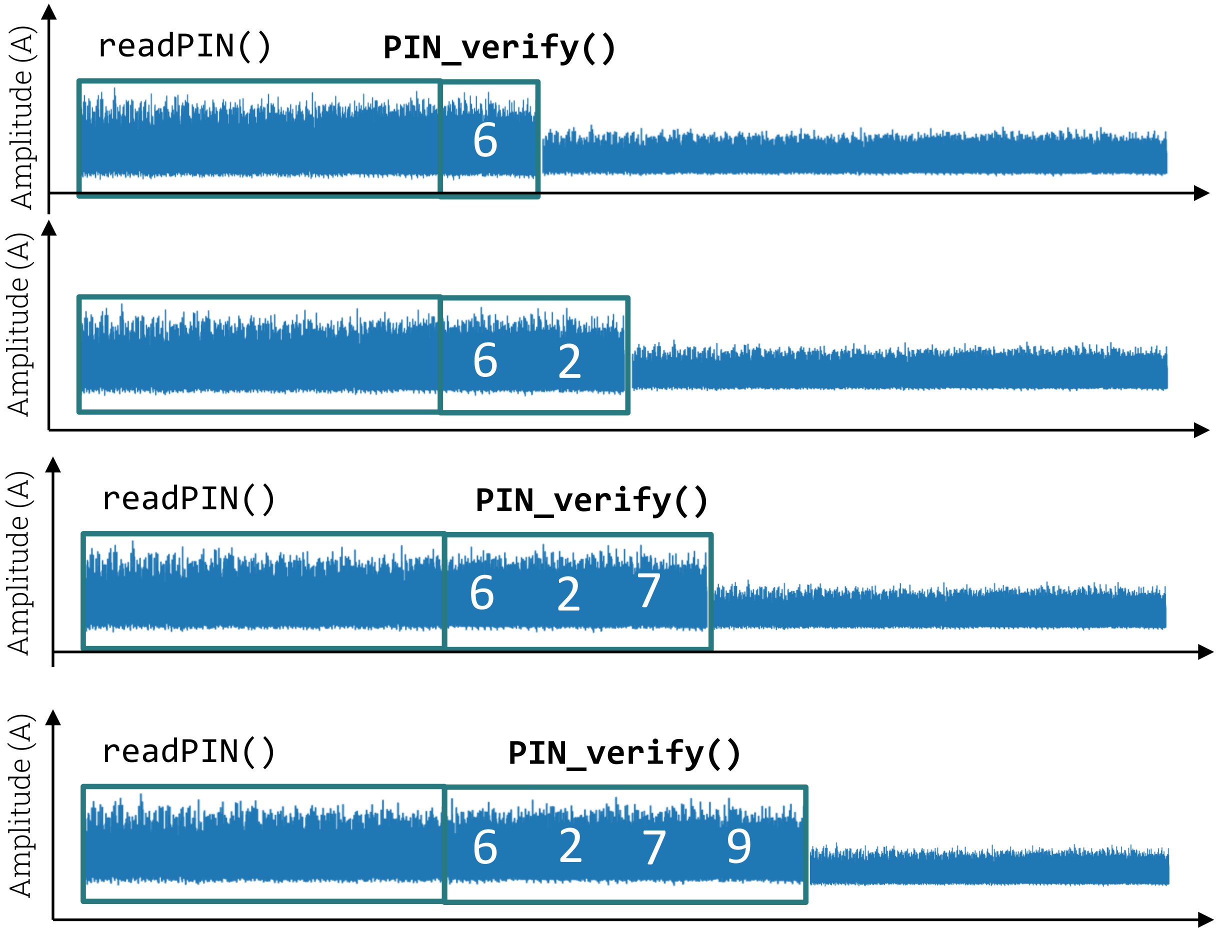
```
readPIN(pin);
PIN_verify(pin) {
    if pin[0] == 6:
        if pin[1] == 2:
            if pin[2] == 7:
                if pin[3] == 9:
                    pin_ok = 1
                end;
            end;
        end;
    end;
    return pin_ok;
}
```



# Timing Analysis

4-digit PIN verification method:

```
readPIN(pin);
PIN_verify(pin) {
    if pin[0] == 6:
        if pin[1] == 2:
            if pin[2] == 7:
                if pin[3] == 9:
                    pin_ok = 1
                end;
            end;
        end;
   end;
   return pin_ok;
}
```




To reveal PIN code, we need at most ? power measurements.

Protection: **only 3 attempts**.

# Cryptography: a review

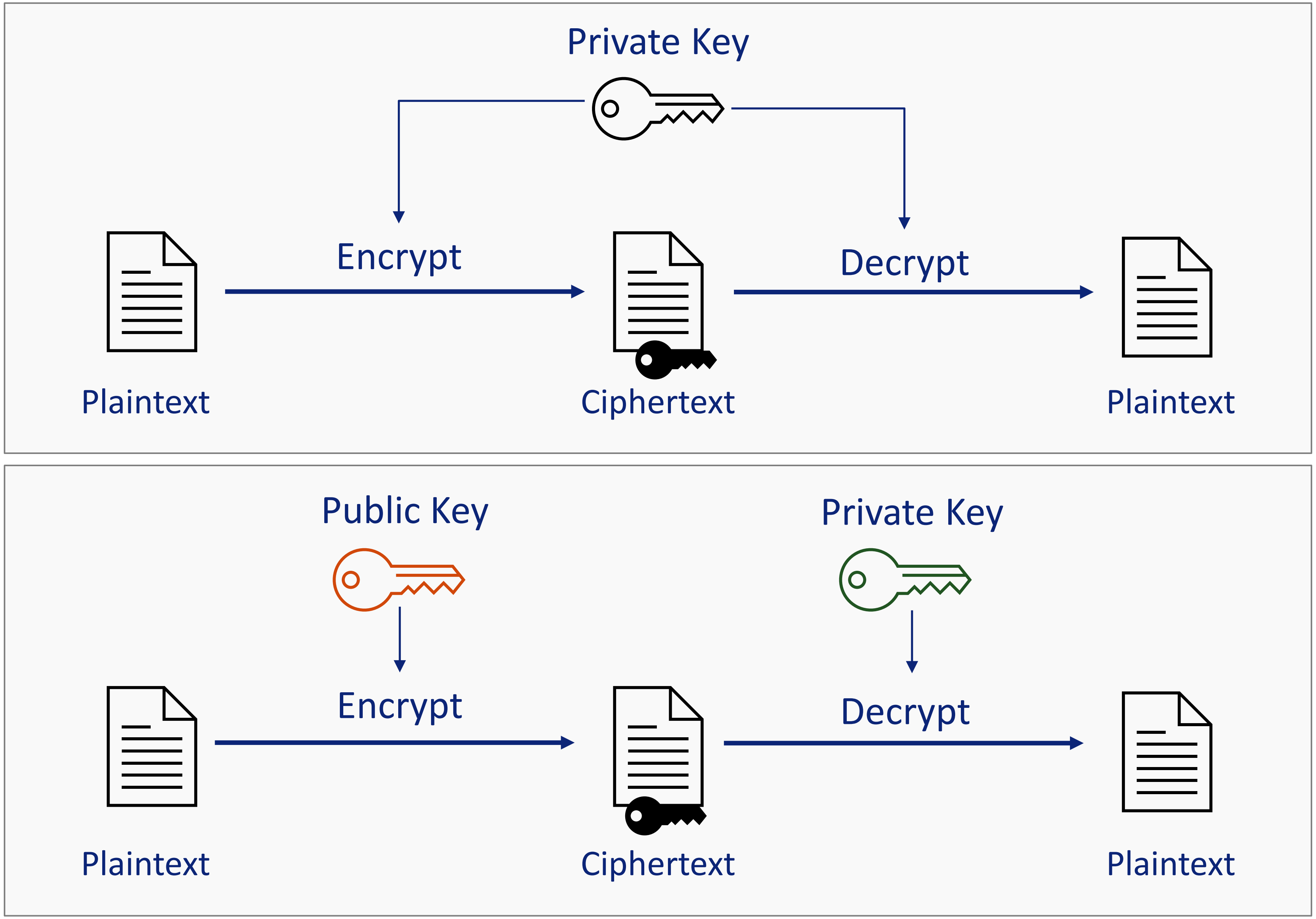
- Symmetric key cryptography:
  - Data encryption Standard (DES) – *becoming obsolete*
  - **Advanced Encryption Standard (AES)** – *widely applied everywhere*
  - Lightweight cryptography (PRESENT, ASCON) – *being adopted by industry (IoT)*

- Asymmetric key cryptography:
  - **RSA**
  - Elliptic Curves-based Cryptography
  - Future: post-quantum cryptography - *will make RSA, ECC obsolete*

# Symmetric vs Asymmetric Cryptography: main differences

Symmetric or private-key cryptography (1 key)

Asymmetric or public-key cryptography (2 keys)



# RSA (Rivest, Shamir, Adleman) - 1977

- **Key generation**:
  - Select two large prime numbers: $p$ and $q$
  - Multiply these two prime numbers: $n = pq$
  - Calculate Euler's Totient Function: $\varphi(n) = (p-1)(q-1)$
  - Choose a (small) public key that is relatively prime to $\varphi(n)$: gcd($\varphi(n)$,e) = 1 (co-primes)
  - Calculate private key: $d = e^{-1} mod\ n$
  - Public key pair: $e, n$
  - Private key pair: $d, n$
- **Encryption**:
  - $c = m^e mod\ n$
- **Decryption**:
  - $m = c^d mod\ n$

Modular exponentiation.

Example:

- **Key generation:**

$p = \mathbf{11}, q = \mathbf{13}$

$n = pq = 11 \times 13 = \mathbf{143}$

$\varphi(n) = (p-1)(q-1) = 10 \times 12 = \mathbf{120}$

$e = \mathbf{7}$ and $gcd(7, 120) = 1$

$d = e^{-1} mod\ n = 7^{-1} mod\ 143 = \mathbf{103}$

- **Encryption of $\boldsymbol{m = 9}$**

$c = 9^7 mod\ 143 = 48$

- **Decryption of $\boldsymbol{c = 48}$**

$m = 48^{103} mod\ 143 = 9$

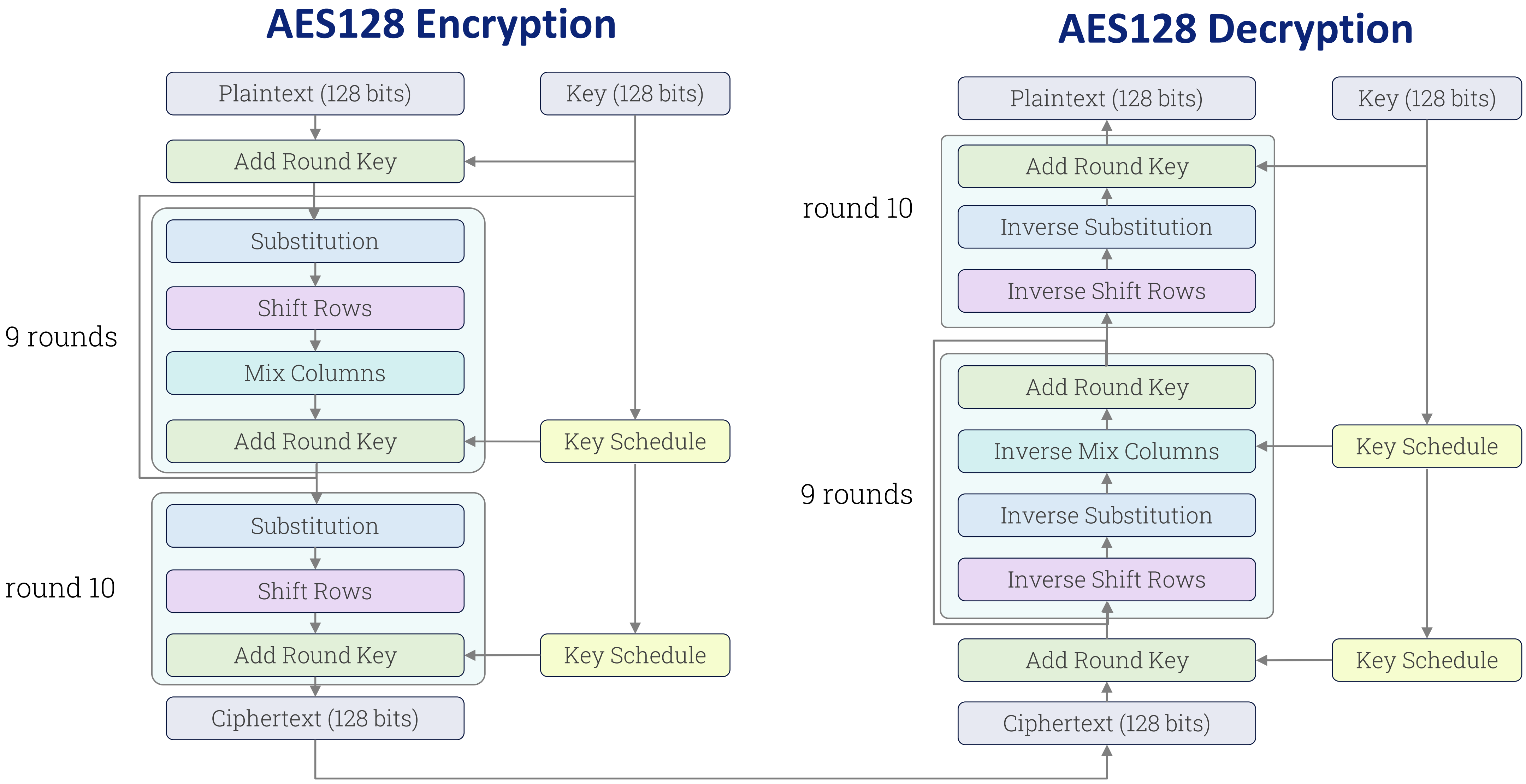
# Advanced Encryption Standard (AES)

Key sizes: **128-bit**, 192-bit, 256-bit keys.

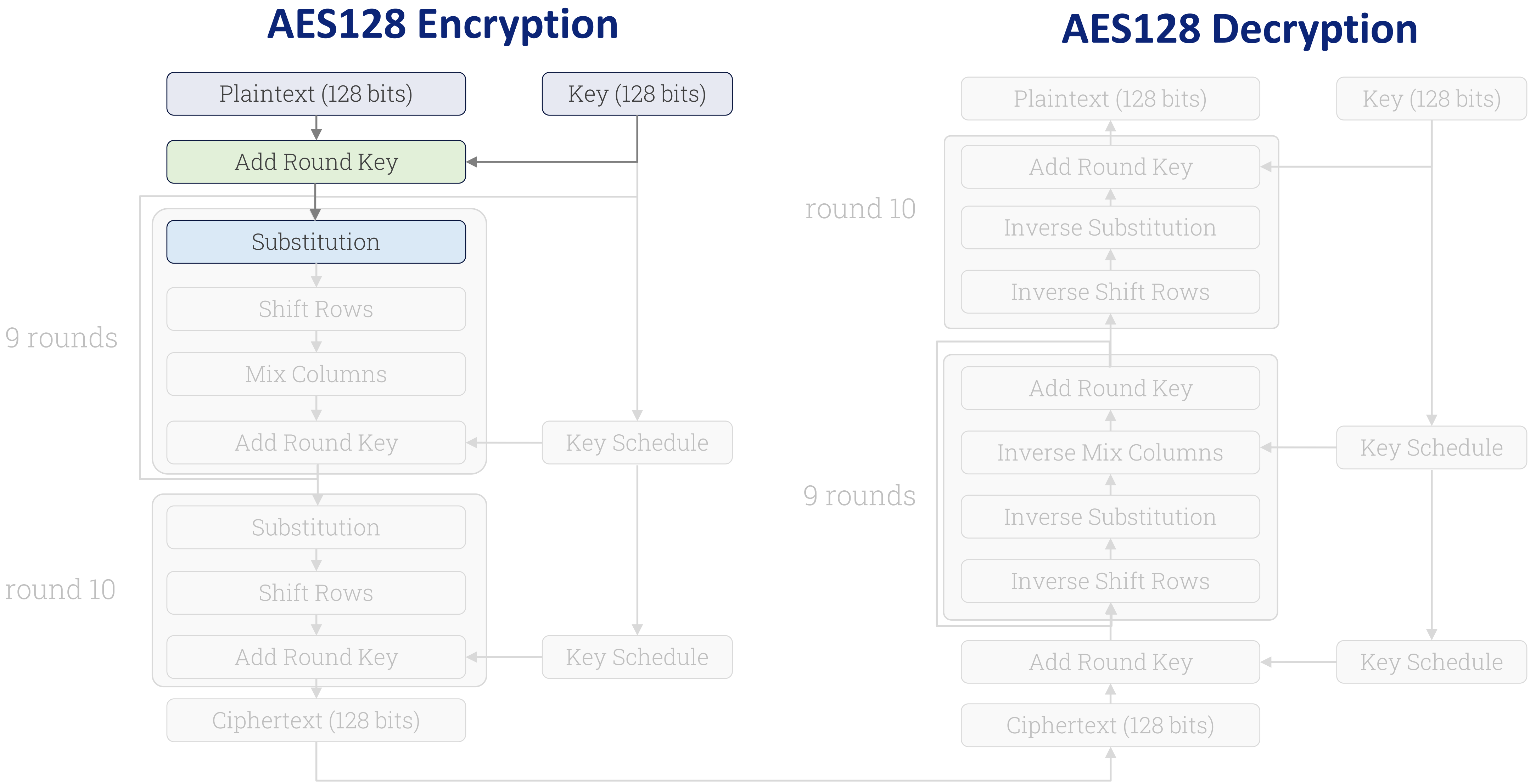
Three main steps:

- Key scheduling (the key is expanded to multiple *round keys*).
- Encryption(`AddRoundKey`, `SubBytes`, `ShiftRows`, `MixColumns`)
- Decryption (`AddRoundKey`, `InverseSubBytes`, `InverseShiftRows`, `InverseMixColumns`)
- Operations in Galois Field ($2^8$) (for instance, addition becomes xor operation)

# Advanced Encryption Standard (AES)

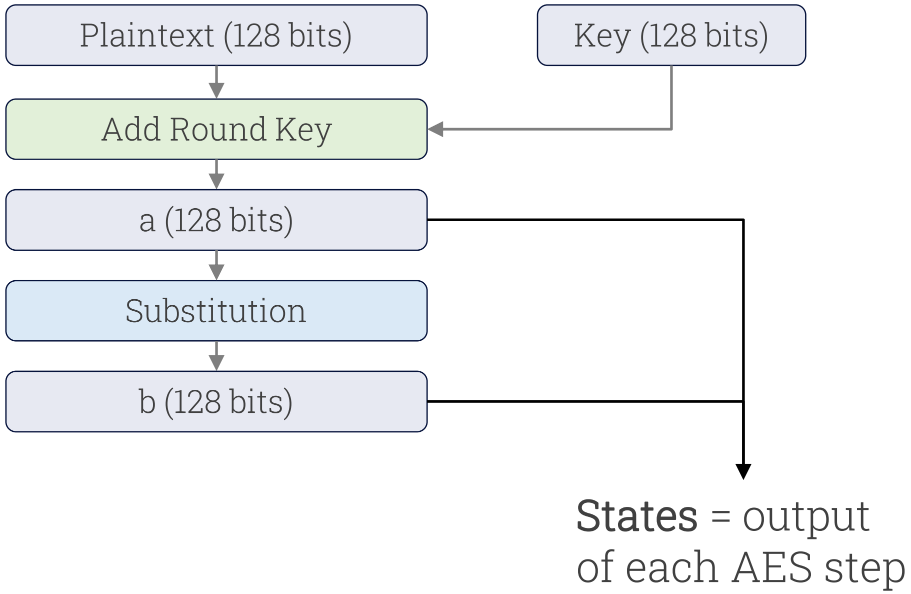


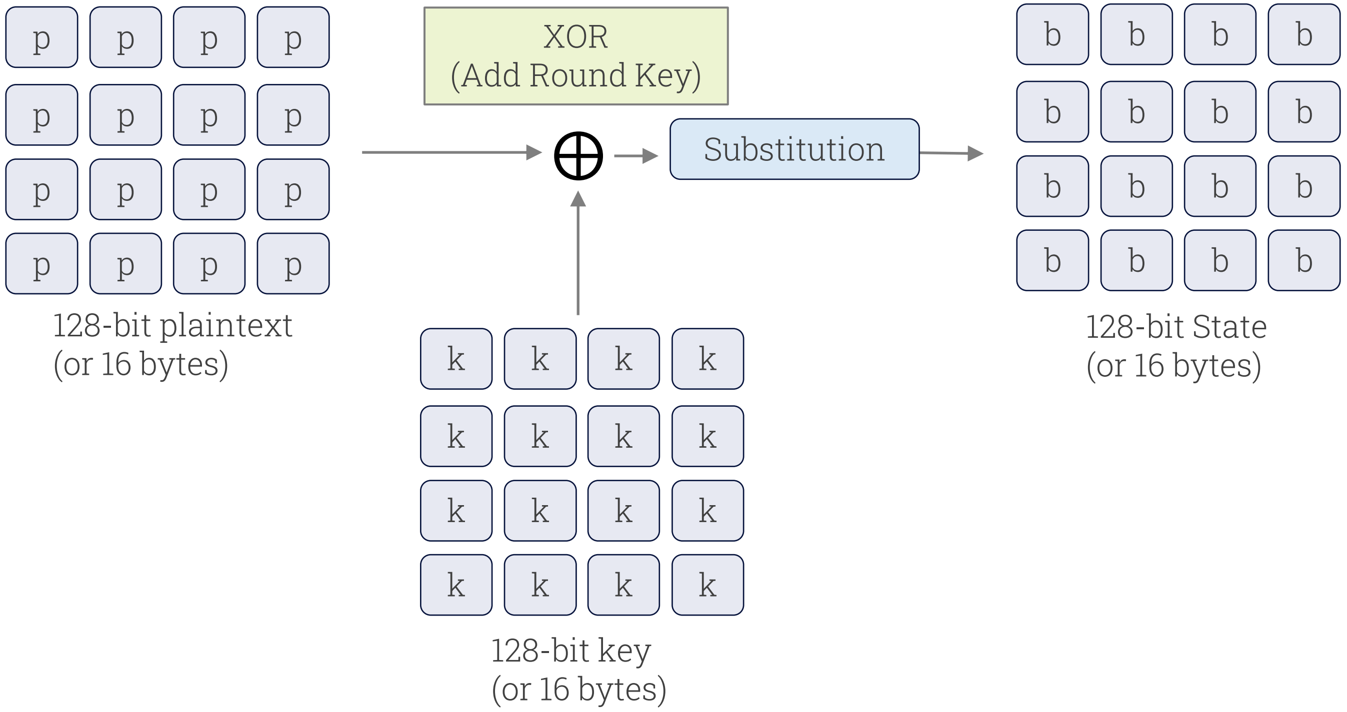
# Advanced Encryption Standard (AES)



# Advanced Encryption Standard (AES)

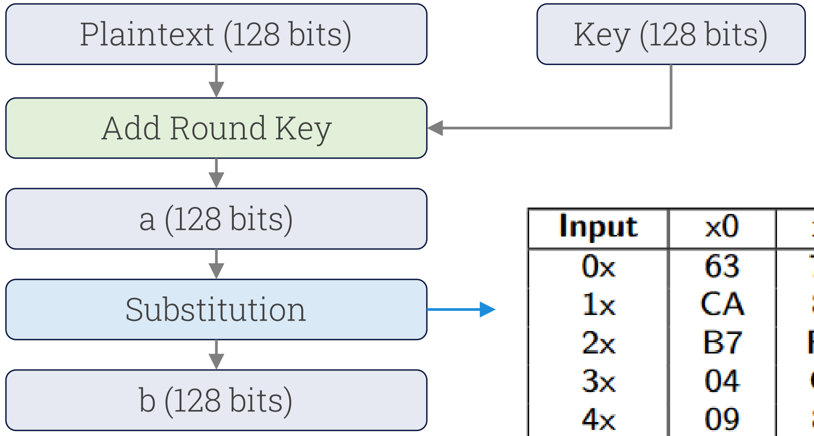
## AES128 Encryption



# Advanced Encryption Standard (AES)

## AES128 Encryption



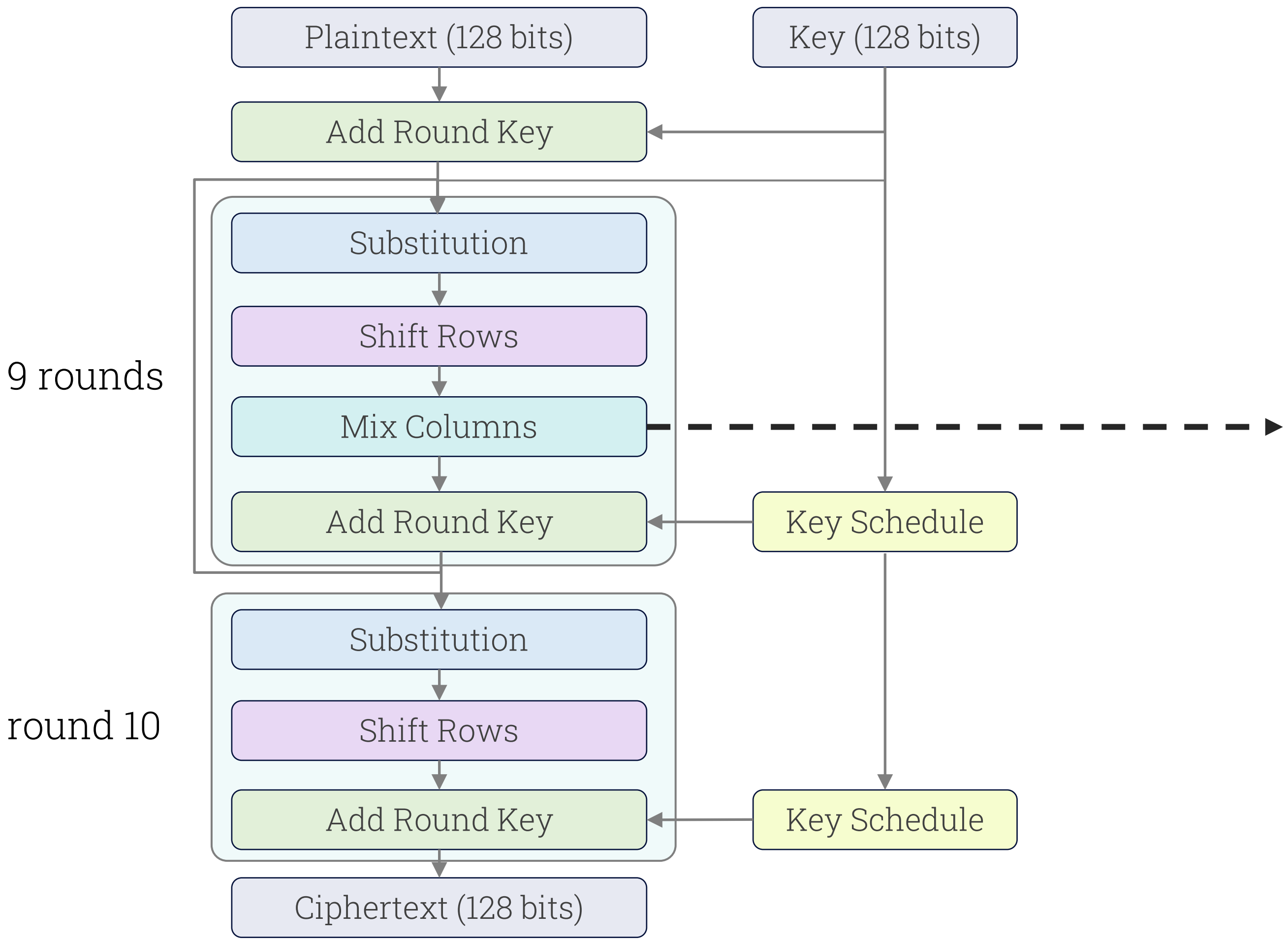

**0x9B** = row **9**, column **B** → SubBytes(**0x9B**) = **0x14**

| Input | x0 | x1 | x2 | x3 | x4 | x5 | x6 | x7 | x8 | x9 | xA | xB | xC | xD | xE | xF |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x | 63 | 7C | 77 | 7B | F2 | 6B | 6F | C5 | 30 | 01 | 67 | 2B | FE | D7 | AB | 76 |
| 1x | CA | 82 | C9 | 7D | FA | 59 | 47 | F0 | AD | D4 | A2 | AF | 9C | A4 | 72 | C0 |
| 2x | B7 | FD | 93 | 26 | 36 | 3F | F7 | CC | 34 | A5 | E5 | F1 | 71 | D8 | 31 | 15 |
| 3x | 04 | C7 | 23 | C3 | 18 | 96 | 05 | 9A | 07 | 12 | 80 | E2 | EB | 27 | B2 | 75 |
| 4x | 09 | 83 | 2C | 1A | 1B | 6E | 5A | A0 | 52 | 3B | D6 | B3 | 29 | E3 | 2F | 84 |
| 5x | 53 | D1 | 00 | ED | 20 | FC | B1 | 5B | 6A | CB | BE | 39 | 4A | 4C | 58 | CF |
| 6x | D0 | EF | AA | FB | 43 | 4D | 33 | 85 | 45 | F9 | 02 | 7F | 50 | 3C | 9F | A8 |
| 7x | 51 | A3 | 40 | 8F | 92 | 9D | 38 | F5 | BC | B6 | DA | 21 | 10 | FF | F3 | D2 |
| 8x | CD | 0C | 13 | EC | 5F | 97 | 44 | 17 | C4 | A7 | 7E | 3D | 64 | 5D | 19 | 73 |
| 9x | 60 | 81 | 4F | DC | 22 | 2A | 90 | 88 | 46 | EE | B8 | 14 | DE | 5E | 0B | DB |
| Ax | E0 | 32 | 3A | 0A | 49 | 06 | 24 | 5C | C2 | D3 | AC | 62 | 91 | 95 | E4 | 79 |
| Bx | E7 | C8 | 37 | 6D | 8D | D5 | 4E | A9 | 6C | 56 | F4 | EA | 65 | 7A | AE | 08 |
| Cx | BA | 78 | 25 | 2E | 1C | A6 | B4 | C6 | E8 | DD | 74 | 1F | 4B | BD | 8B | 8A |
| Dx | 70 | 3E | B5 | 66 | 48 | 03 | F6 | 0E | 61 | 35 | 57 | B9 | 86 | C1 | 1D | 9E |
| Ex | E1 | F8 | 98 | 11 | 69 | D9 | 8E | 94 | 9B | 1E | 87 | E9 | CE | 55 | 28 | DF |
| Fx | 8C | A1 | 89 | 0D | BF | E6 | 42 | 68 | 41 | 99 | 2D | 0F | B0 | 54 | BB | 16 |

Table 1: AES Substitution Lookup Table (Sbox).

# Advanced Encryption Standard (AES)

## AES128 Encryption




After each MixColumns operation, the output state bytes depends on **k + r** key bytes.

- After first MixColumns, each byte depends on 2 key bytes (16 bits of the key)
- After second MixColumns, each byte depends on 3 key bytes (24 bits of the key)
- ...

# Side-channel analysis on RSA

Main goal:

- Measure power consumption of a device while decryption ($c^d mod\ n$) is being computed.
- Recover $d$.

Threat model (assumptions):

- An attacker can decrypt ciphertexts.
- An attacker can measure the power consumption during decryption operation.

# Modular exponentiation: too complex?

Let us first analyse the modular exponentiation operation. Example: $\boxed{m = 48^{\mathbf{103}} mod\ 143 = 9}$

$$d = 103 \longrightarrow 48^{103} = \underbrace{48 \times 48 \times 48 \times \cdots \times 48}_{\text{102 multiplications!}}$$

Solution: the exponent $d$ is taken as a binary number:

**103 = 01100111**

- Initialize a temporary result as $T = 1$
- Scan the exponent bits from MSb to LSb.
- If bit is 1, perform a square followed by a multiplication by base:
- $T = T \times T$
- $T = T \times 48$
- If bit is 0, only perform a :
- $T = T \times T$

$$\left.\begin{aligned}
T &= 1 \\
T &= 1 \times 1 = 1 \\
T &= 1 \times 48 = 48 \\
T &= 48 \times 48 = 48^2 \\
T &= 48^2 \times 48 = 48^3 \\
T &= 48^3 \times 48^3 = 48^6 \\
T &= 48^6 \times 48^6 = 48^{12} \\
T &= 48^{12} \times 48^{12} = 48^{24} \\
T &= 48^{24} \times 48 = 48^{25} \\
T &= 48^{25} \times 48^{25} = 48^{50} \\
T &= 48^{50} \times 48 = 48^{51} \\
T &= 48^{51} \times 48^{51} = 48^{102} \\
T &= 48^{102} \times 48 = \mathbf{48^{103}}
\end{aligned}\right\}\ \text{12 multiplications!}$$

# Modular exponentiation: too complex?

Let us first analyse the modular exponentiation operation. Example: $m = 48^{\mathbf{103}} mod\ 143 = 9$

$$d = 103 \longrightarrow 48^{103} = \underbrace{48 \times 48 \times 48 \times \cdots \times 48}_{\text{102 multiplications!}}$$

Solution: the exponent $d$ is taken as a binary number:
**103 = 01100111**

- Initialize a temporary result as $T = 1$
- Scan the exponent bits from MSb to LSb.
- If bit is 1, perform a square followed by a multiplication by base:
- $T = T \times T$
- $T = T \times 48$
- If bit is 0, only perform a :
- $T = T \times T$

| | | |
|---|---|---|
| $T = 1$ | Initialize | |
| $T = 1 \times 1 = 1$ | Square | 1 |
| $T = 1 \times 48 = 48$ | Multiply | |
| $T = 48 \times 48 = 48^2$ | Square | 1 |
| $T = 48^2 \times 48 = 48^3$ | Multiply | |
| $T = 48^3 \times 48^3 = 48^6$ | Square | 0 |
| $T = 48^6 \times 48^6 = 48^{12}$ | Square | 0 |
| $T = 48^{12} \times 48^{12} = 48^{24}$ | Square | 1 |
| $T = 48^{24} \times 48 = 48^{25}$ | Multiply | |
| $T = 48^{25} \times 48^{25} = 48^{50}$ | Square | 1 |
| $T = 48^{50} \times 48 = 48^{51}$ | Multiply | |
| $T = 48^{51} \times 48^{51} = 48^{102}$ | Square | 1 |
| $T = 48^{102} \times 48 = \mathbf{48^{103}}$ | Multiply | |

# Left-to-Right Square-and-Multiply Algorithm

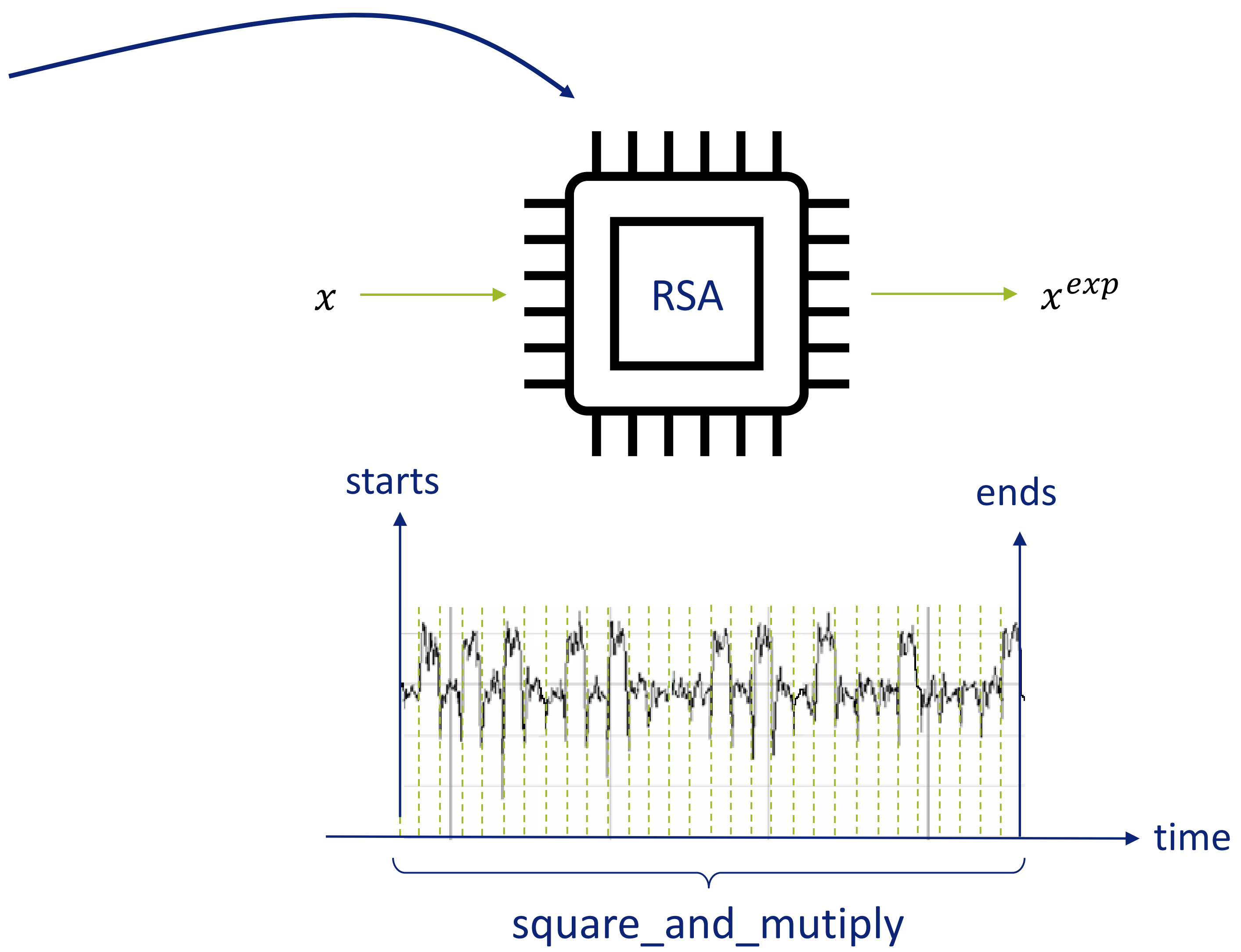
**Algorithm 1:** Left-to-right Square and Multiply Algorithm

**Input:** Base $b$, exponent $e$, modulus $n$
**Output:** $b^e \mod m$

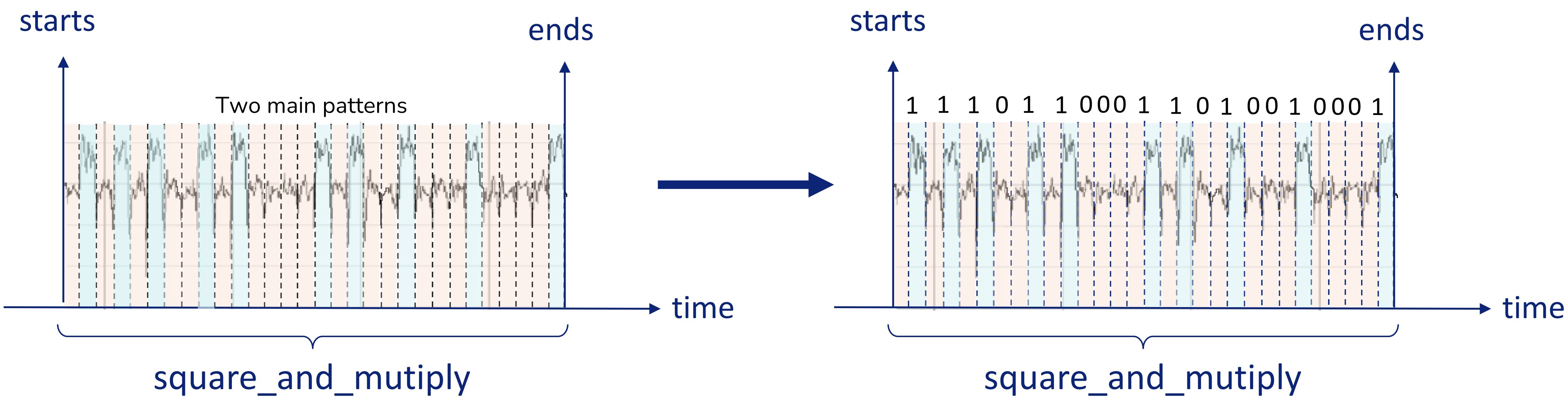
$t \leftarrow 1$;
binaryExp $\leftarrow$ Convert $e$ to binary;
**for** *bit in binaryExp* **do**
  $t \leftarrow t \times t \mod n$;
  **if** *bit is 1* **then**
    $t \leftarrow t \times$ b $\mod n$;
  **end**
**end**
**return** $t$;

# Running square-and-multiply algorithm in a device

```
def square(x):
    return x * x


def multiply(t, x):
    return t * x


def square_and_multiply(x, exp):
    t = 1
    exp_bits = bin(exp)[2:]

    for b in exp_bits:
        t = square(t)
        if b == '1':
            t = multiply(t, x)

    return t
```



# Side-channel analysis on RSA (square-and-multiply algorithm)




What is the private key d (in binary)?

# Side-channel analysis on RSA

- The private key can be recovered from a single power consumption measurements.
- Because the modular exponentiation is computed with an algorithm (square-and-multiply) that process the exponent bit-by-bit, the power consumption shows clear patterns where exponent bit is 1 (square followed by a multiply operation) or 0 (only square operation).

- Countermeasures:
  - Balance the power consumption to have the same operation if the exponent is either bit 1 or 0.

# SCA countermeasure

**Algorithm 2:** Left-to-right Square and Multiply **Always** Algorithm

**Input:** Base $b$, exponent $e$, modulus $n$

**Output:** $b^e \mod m$

$t \leftarrow 1$;

binaryExp $\leftarrow$ Convert $e$ to binary;

**for** *bit in binaryExp* **do**

    **if** *bit is 1* **then**

        $t \leftarrow t \times t \mod n$;

        $t \leftarrow t \times$ b $\mod n$;

    **else**

        $t \leftarrow t \times t \mod n$;

        $dummy \leftarrow t \times$ b $\mod n$;

    **end**

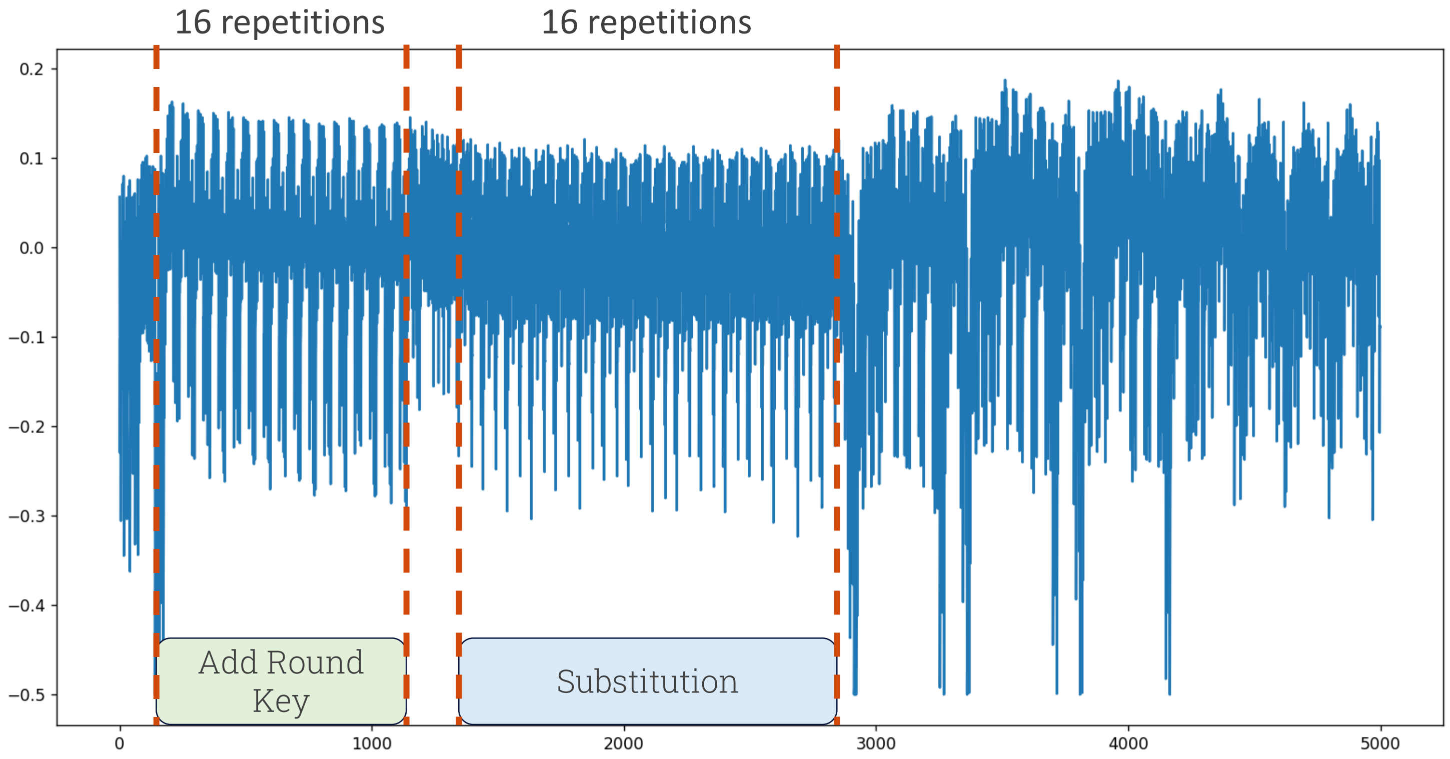
**end**

**return** $t$;

# Side-channel analysis on AES

Threat model (assumptions):

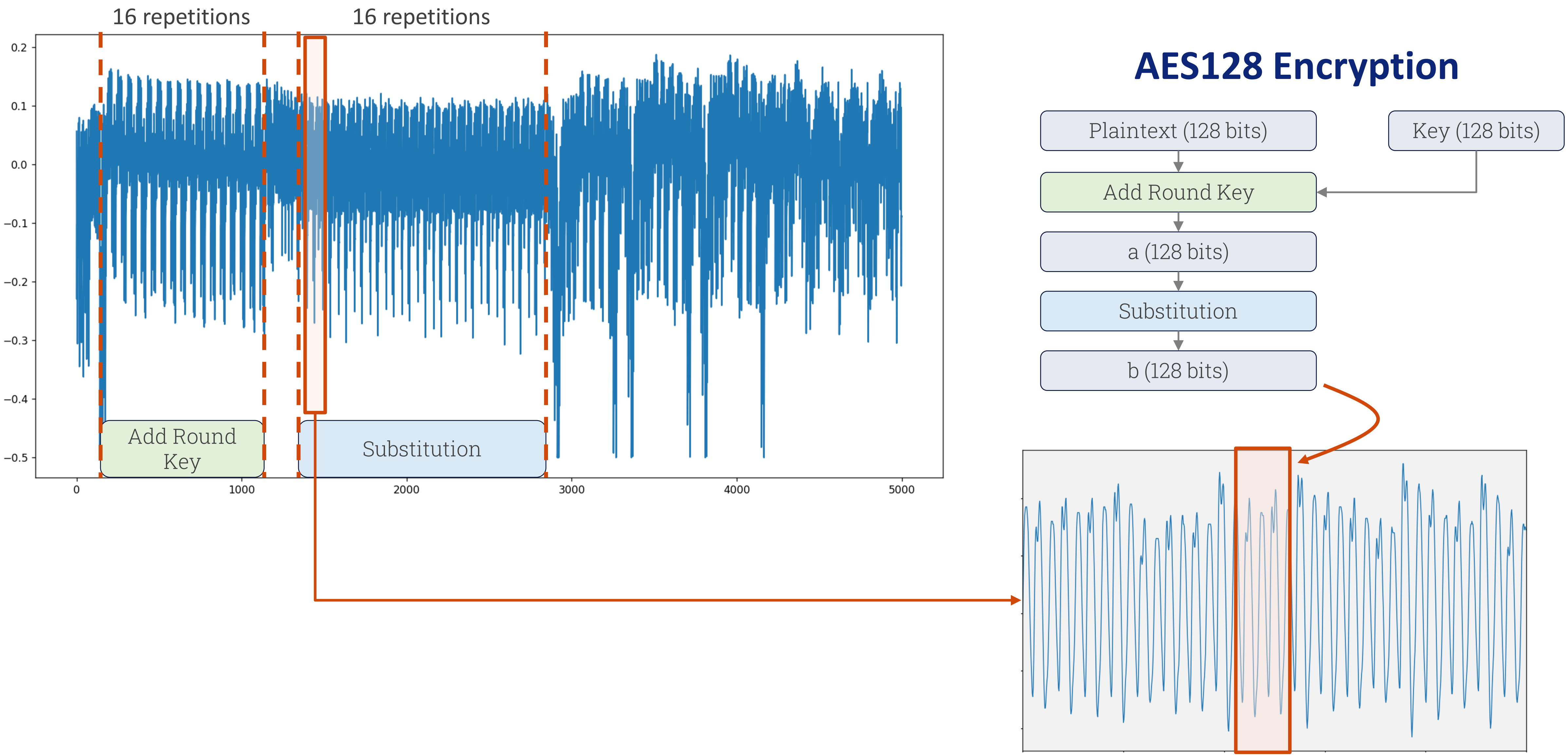
- An attacker can encrypt plaintexts
- An attacker can define the plaintexts
- An attacker can measure the power consumption during the encryption.




However, only one power consumption measurement cannot directly indicate the number of bits 1: we need some **statistics**.

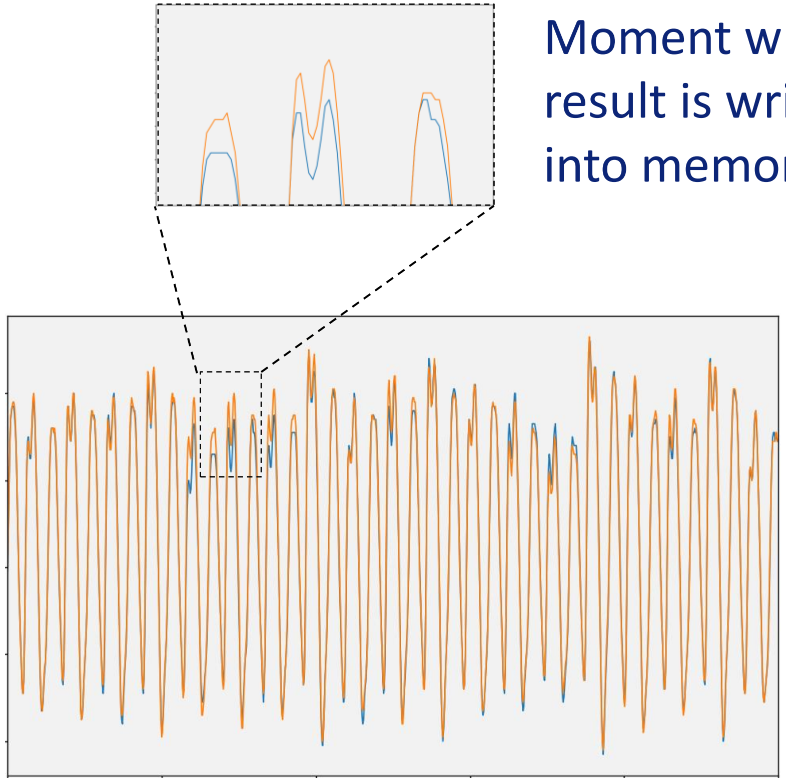
The amplitude of the power consumption could be ***proportional*** to the number of bits 1 in a register.

# Side-channel analysis on AES

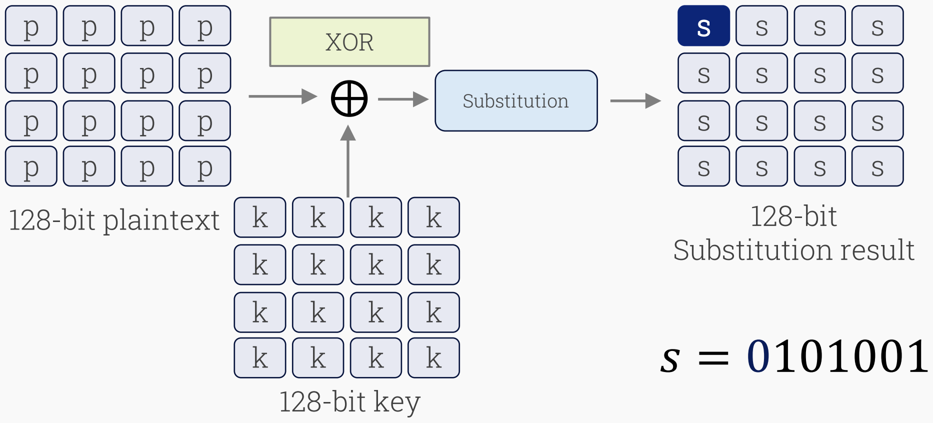


# Side-channel analysis on AES

Moment when ***s*** result is written into memory.

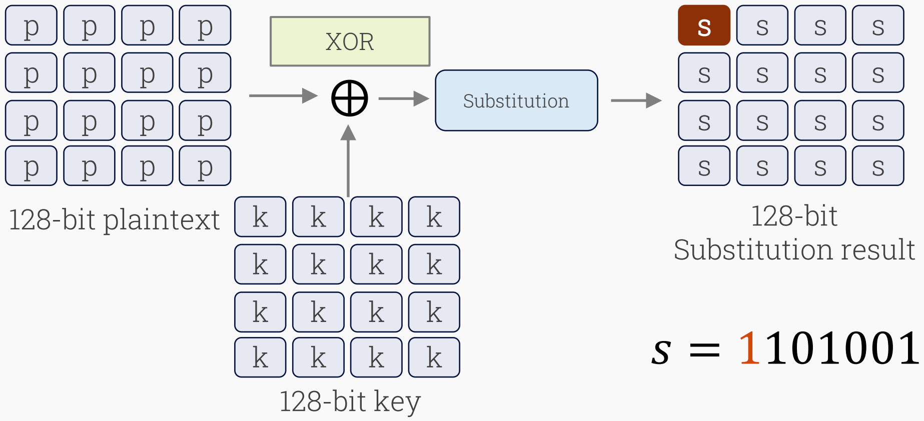


## Measurement 1




$$s = 01010011_2$$

## Measurement 2




$$s = 11010011_2$$

The amplitude of the power consumption could be ***proportional*** to the number of bits 1 in the result **s**.

# Side-channel analysis on AES

Back in 1999, researchers thought the following:

- If we measure several power consumption traces during AES encryption and we can split measurements into two groups:
  - Group 1: measurements when most significant bit of $\boldsymbol{s} = \mathbf{1}$
  - Group 2: measurements when most significant bit of $\boldsymbol{s} = \mathbf{0}$
- … the we can get the key (byte per byte).
- Why?

# Side-channel analysis on AES

Back in 1999, researchers thought the following:

- If we measure several power consumption traces during AES encryption and we can split measurements into two groups:
  - Group 1: measurements when most significant bit of $\boldsymbol{s} = \mathbf{1}$
  - Group 2: measurements when most significant bit of $\boldsymbol{s} = \mathbf{0}$
- Then we can average all measurements of group 1: $\widehat{M}_o$
- Then we can average all measurements of group 2: $M_1$
- And subtract the averaged groups: $\widehat{M}_o$- $\widehat{M}_1$
- If groups are correctly separated, then the subtraction should contain a peak where the most significant bit of $\boldsymbol{s}$ is processed. Otherwise, if groups are not correctly separated, there will no peak.
- However:
  - The bit values of $\boldsymbol{s}$ can be only be known if the key is known.
- Solution: Guess all key values: only the correct key guess should produce a peak in the subtraction.

# Differential Power Analysis (DPA) on AES

Demo

# Correlation Power Analysis (DPA) on AES

Side-channel information is correlated with data being processed by a device.

The output of first SubBytes operation is: $s_i = SubBytes(p_i \ xor \ k_i)$

| $k_i$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| Measurement 1 | 36 | 249 | 225 | 41 | 12 | 192 | 253 | 188 | 111 |
| Measurement 2 | 185 | 242 | 181 | 69 | 164 | 200 | 179 | 228 | 29 |
| Measurement 3 | 81 | 175 | 109 | 230 | 242 | 114 | 165 | 127 | 15 |
| Measurement 4 | 85 | 109 | 131 | 209 | 94 | 106 | 118 | 152 | 181 |
| Measurement 5 | 108 | 17 | 21 | 60 | 148 | 130 | 192 | 214 | 135 |
| Measurement 6 | 162 | 99 | 82 | 128 | 20 | 99 | 139 | 249 | 140 |
| Measurement 7 | 2 | 113 | 74 | 123 | 65 | 173 | 205 | 6 | 209 |
| Measurement 8 | 19 | 203 | 201 | 6 | 145 | 62 | 53 | 92 | 19 |
| Measurement 9 | 228 | 200 | 234 | 58 | 42 | 60 | 246 | 84 | 35 |
| Measurement 10 | 156 | 163 | 164 | 52 | 203 | 188 | 162 | 75 | 34 |

$x$

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| 20 | 19 | 13 | 6 | -2 | -10 | -19 | -29 | -39 | -47 |
| 22 | 20 | 15 | 8 | -1 | -9 | -19 | -30 | -39 | -48 |
| 24 | 22 | 18 | 12 | 3 | -6 | -15 | -24 | -34 | -43 |
| 24 | 23 | 18 | 11 | 2 | -6 | -16 | -26 | -37 | -45 |
| 21 | 20 | 16 | 9 | 1 | -6 | -16 | -24 | -34 | -43 |
| 17 | 18 | 15 | 9 | 1 | -6 | -14 | -23 | -32 | -40 |
| 24 | 26 | 22 | 16 | 8 | -2 | -12 | -22 | -32 | -42 |
| 19 | 18 | 14 | 7 | 0 | -8 | -17 | -27 | -36 | -43 |
| 21 | 21 | 16 | 10 | 2 | -6 | -15 | -25 | -35 | -44 |
| 21 | 22 | 17 | 12 | 4 | -4 | -13 | -23 | -32 | -42 |

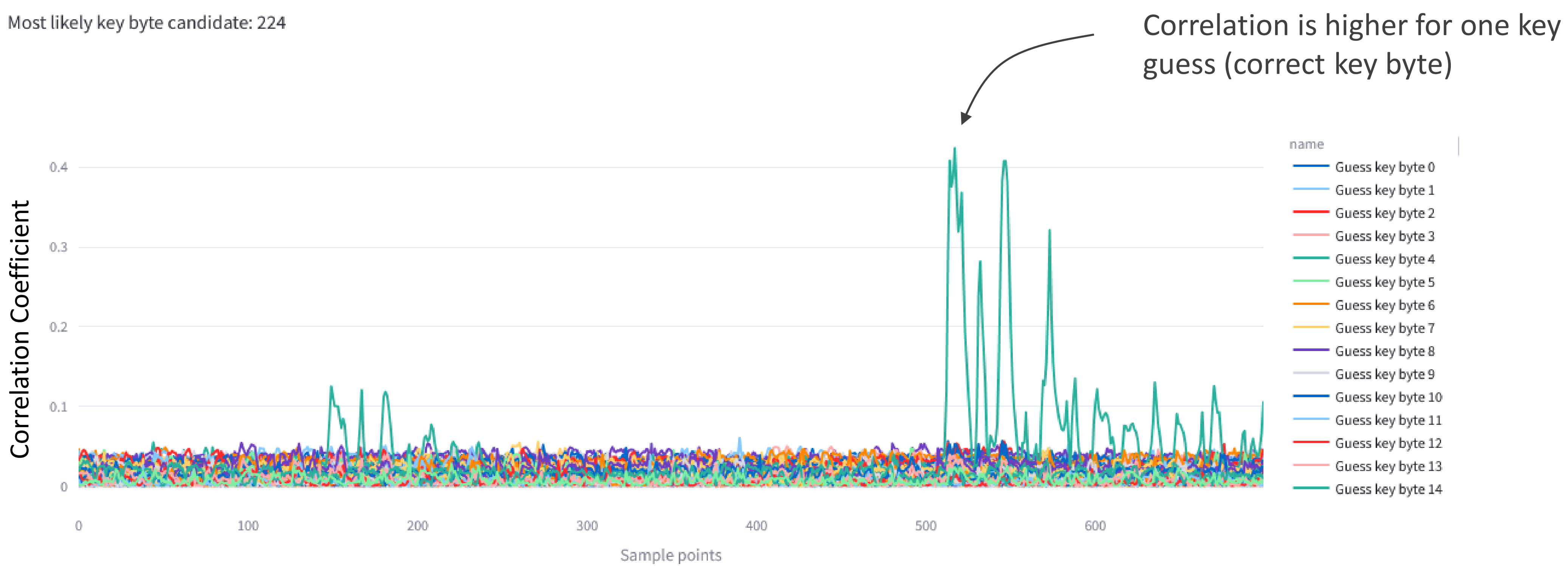
$y$

Pearson Correlation Coefficient

$$\rho(x,y) = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2 \sum(y_i - \bar{y})^2}}$$

# Correlation Power Analysis (DPA) on AES

Side-channel information is correlated with data being processed by a device.



# We continue next week with advanced SCA.

Thank you!