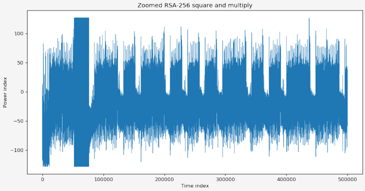
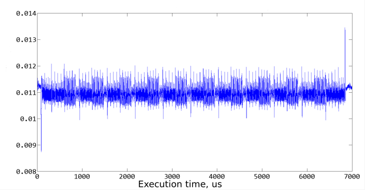
Lecture 2: AES-128

Hardware attacks

- Hardware attacks extends the threat model: an adversary can measure and influence on the computation device
- Successful hardware attacks don't mean that an underlying cryptographic algorithm is bad
- Successful hardware attacks mean that this concrete algorithm implementation on that concrete device is vulnerable

Side-channel leakage



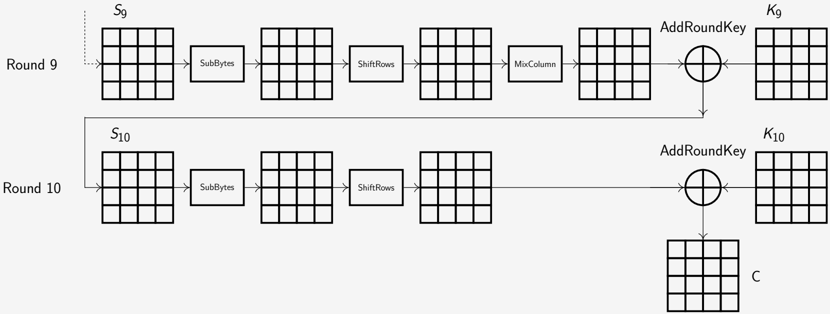


- You worked on simple RSA attack
- Simple side-channel attacks use visually distinguishable patterns
- In case of RSA, the leakage was "horizontal": long signal - 1, short signal - 0

- We will work with AES-128
- We will learn how to perform correlation power analysis
- Contrary to previous RSA example, the leakage is "vertical" (we will work with amplitudes)

AES Example

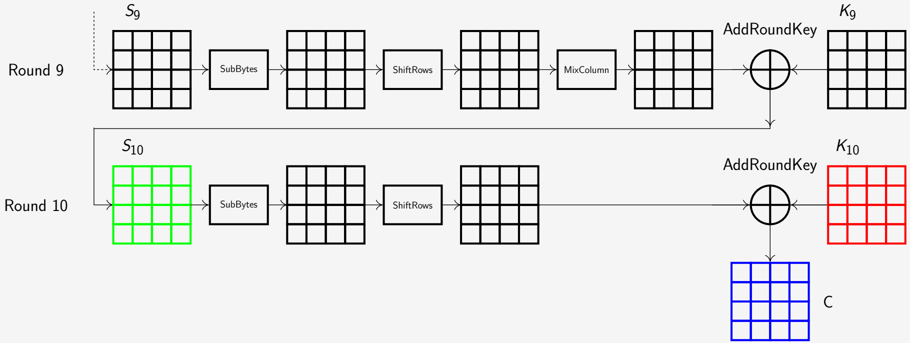
Advanced Encryption Standard algorithm (AES-128)



Advanced Encryption Standard

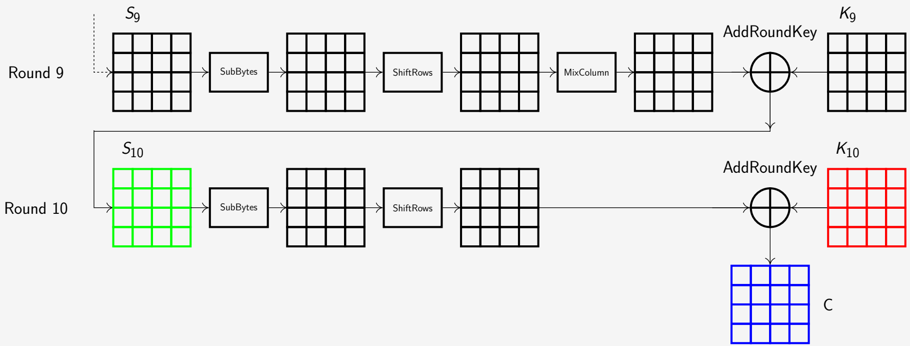
- All operations are reversable for Key Expansion, Encryption and decryption
- A master key can be re-computed from any round key (in your tasks this function is available)
- The number of rounds depend on the key size. We will work only with 128-bit keys (10 rounds).
- In the tasks don't forget about ShiftRows

AES-128 Assignment 1



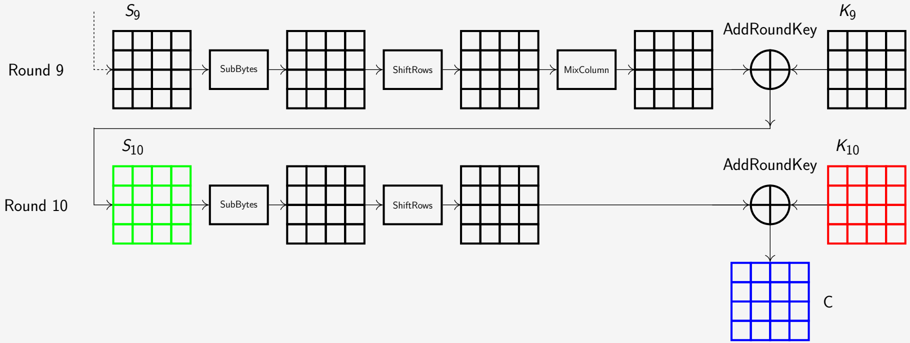
- A developer left a debug option that prints the state 10 (S_{10} shown in green) and a ciphertext C shown in blue. What can be wrong with that?

AES-128 Assignment 1



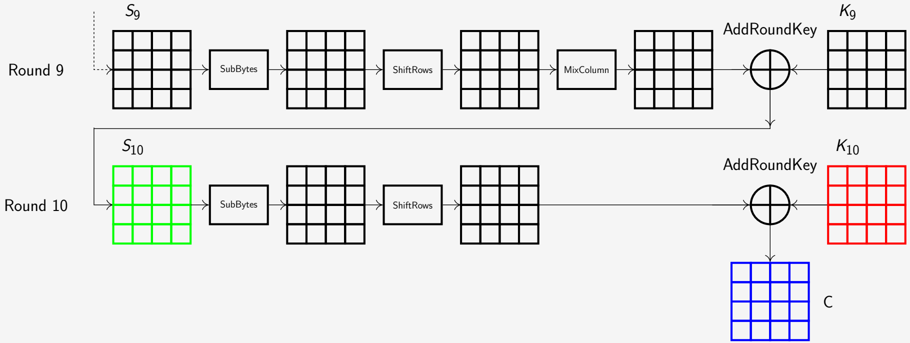
- A developer left a debug option that prints the state 10 (S_{10} shown in green) and a ciphertext C shown in blue. What can be wrong with that?
- Find K_{10} and then get the master key using the provided code

AES-128 Assignment 1



- A developer left a debug option that prints the state 10 (S_{10} shown in green) and a ciphertext C shown in blue. What can be wrong with that?
- Find K_{10} and then get the master key using the provided code
- $C = ShiftRows(SubBytes[S_{10}]) \oplus K_{10}$

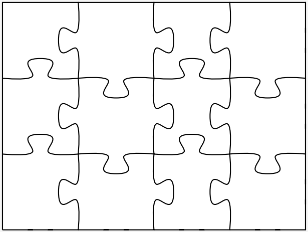
AES-128 Assignment 1



- A developer left a debug option that prints the state 10 (S_{10} shown in green) and a ciphertext C shown in blue. What can be wrong with that?
- Find K_{10} and then get the master key using the provided code
- $C = ShiftRows(SubBytes[S_{10}]) \oplus K_{10}$
- $K_{10} = C \oplus ShiftRows(SubBytes[S_{10}])$

Puzzle 2: Law of large numbers

A bit of mathematics... who said side-channels are easy



Law of large numbers

- This is the cornerstone law for side-channel attacks

Law of large numbers

- This is the cornerstone law for side-channel attacks

The average of the results obtained from a large number of trials should be close to the expected value and tend to become closer to the expected value as more trials are performed

$$\lim_{x \to \inf} \sum_{i=0}^{n} \frac{X_i}{n} = \bar{X}$$

Law of large numbers

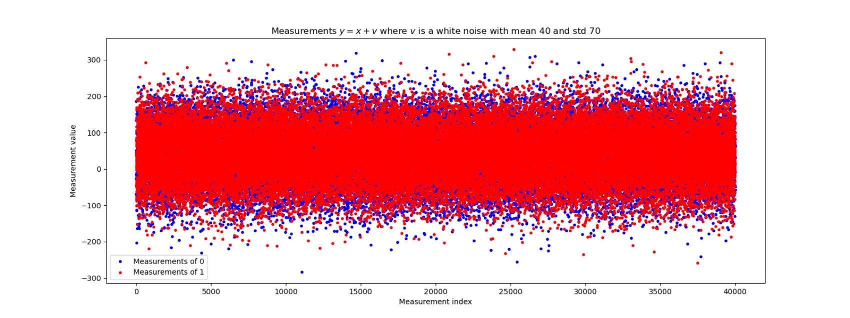
- This is the cornerstone law for side-channel attacks

The average of the results obtained from a large number of trials should be close to the expected value and tend to become closer to the expected value as more trials are performed

$$\lim_{x \to \inf} \sum_{i=0}^{n} \frac{X_i}{n} = \bar{X}$$

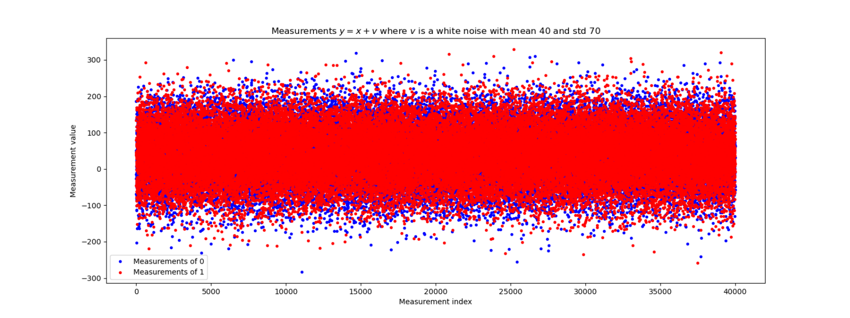
- What does this mean in practice?

Law of large numbers: white noise



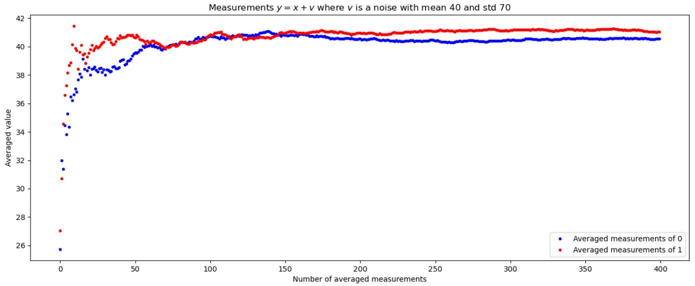
- Consider a situation when we measure a process: $y = x + \nu$, where x is 0 or 1 and ν is a white noise with mean $\mu = 40$ and std $\sigma = 70$
- Use python to emulate measurements

Law of large numbers: white noise



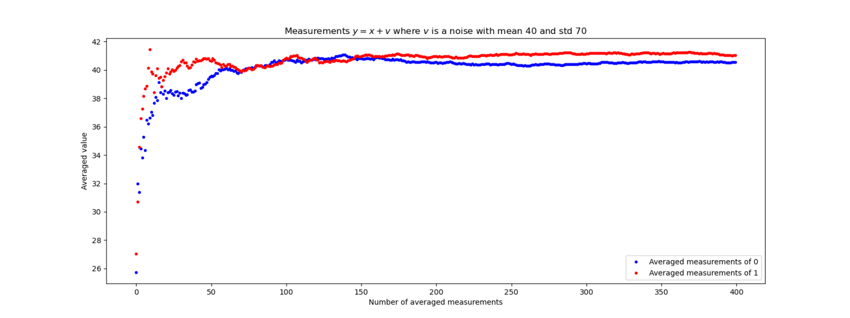
- Consider a situation when we measure a process: $y = x + \nu$, where x is 0 or 1 and ν is a white noise with mean $\mu = 40$ and std $\sigma = 70$
- Use python to emulate measurements
- By looking at raw measurements, 0 and 1 can not be distinguished

Law of large numbers: white noise



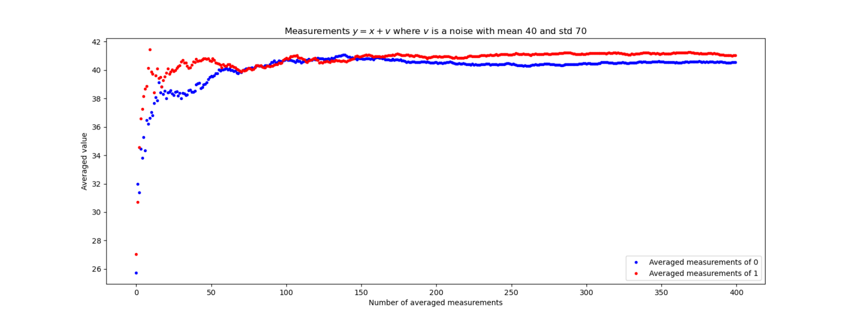
- Here comes the Law of Large Numbers

Law of large numbers: white noise



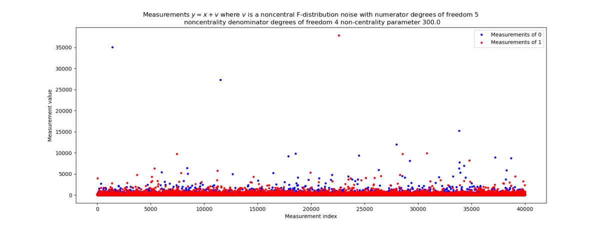
- Here comes the Law of Large Numbers
- For $x = 0$ the curve converges to 40
- For $x = 1$ the curve converges to 41

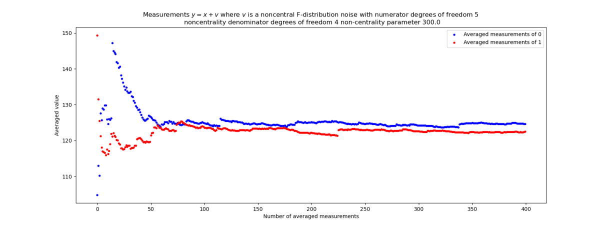
Law of large numbers: white noise



- Here comes the Law of Large Numbers
- For $x = 0$ the curve converges to 40
- For $x = 1$ the curve converges to 41
- Now we see a clear difference

Law of large numbers: more complex noise



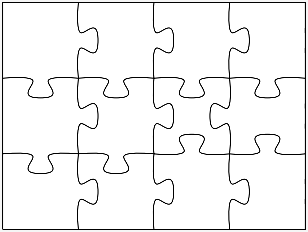


- Measurements with the noncentral F-distribution noise

Law of large numbers: conclusion

- LLN shows that averaging a "sufficiently" big number of samples will converge noise to constant
- In certain cases, we can directly average measurements for side-channel attacks
- Nevertheless, by averaging, we can lose important information about random numbers

Puzzle 3: Power models



Power models

- We cannot model the exact current drain (power consumption):
  - A digital circuit is too complex, and the number of switching transistors is huge
  - Current drain will depend on technology, DC voltage, temperature and other parameters

Power models

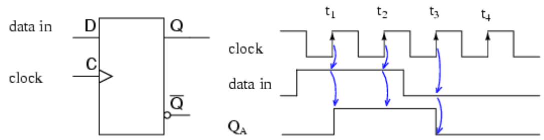
- We cannot model the exact current drain (power consumption):
  - A digital circuit is too complex, and the number of switching transistors is huge
  - Current drain will depend on technology, DC voltage, temperature and other parameters
- Side-channel attacks don't try to have the exact current drain value (I would like to see if someone does it)

Power models

- We cannot model the exact current drain (power consumption):
  - A digital circuit is too complex, and the number of switching transistors is huge
  - Current drain will depend on technology, DC voltage, temperature and other parameters
- Side-channel attacks don't try to have the exact current drain value (I would like to see if someone does it)
- Instead, side-channel attacks work with power consumption models

Registers

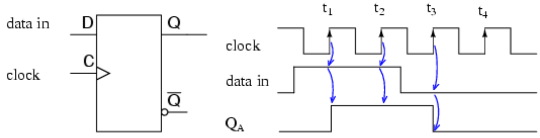
- Digital logic contains multiple registers (e.g., CPU registers)
- Registers are made of flip-flops



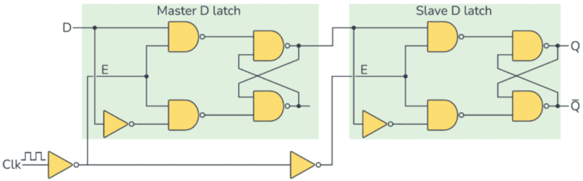
Data present at clock time is transfered from **D** to **Q**.

Registers

- Digital logic contains multiple registers (e.g., CPU registers)
- Registers are made of flip-flops

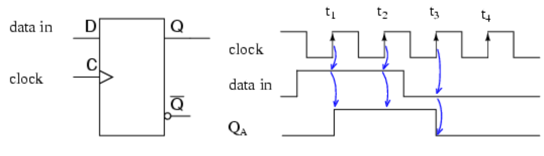


Data present at clock time is transfered from **D** to **Q**.

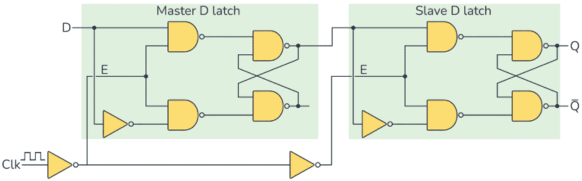


Registers

- Digital logic contains multiple registers (e.g., CPU registers)
- Registers are made of flip-flops

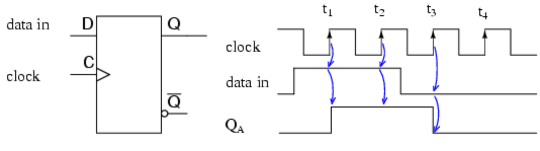


Data present at clock time is transfered from D to Q.



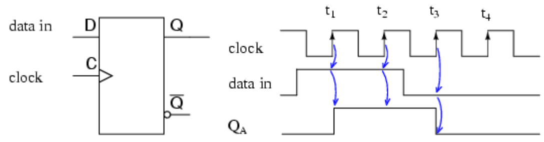
- D flip-flop is made of combinational logic, but the value is updated only on clock rising edge
- D flip-flop elements drain current

Registers



Data present at clock time is transfered from D to Q.

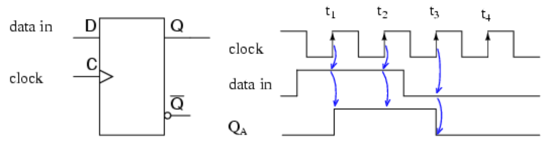
Registers



Data present at clock time is transfered from D to Q.

- Flip-flops consume current during transitions $0 \rightarrow 1$ and $1 \rightarrow 0$

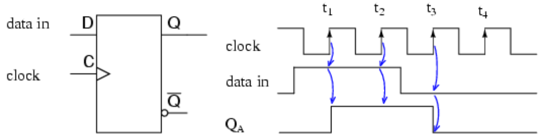
Registers



Data present at clock time is transfered from D to Q.

- Flip-flops consume current during transitions $0 \rightarrow 1$ and $1 \rightarrow 0$
- As seen with the inverter, consumption during the transitions $0 \rightarrow 1$ and $1 \rightarrow 0$ differ

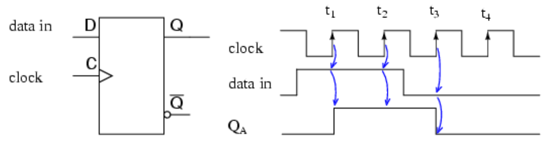
Registers



Data present at clock time is transfered from D to Q.

- Flip-flops consume current during transitions $0 \to 1$ and $1 \to 0$
- As seen with the inverter, consumption during the transitions $0 \to 1$ and $1 \to 0$ differ
- Denote consumptions: $i(0 \to 1) = \alpha$ and $i(1 \to 0) = \beta$

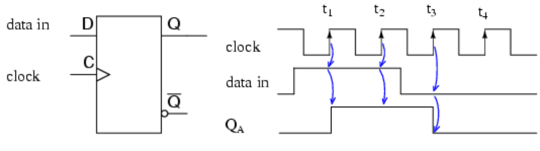
Registers



Data present at clock time is transfered from D to Q.

- Flip-flops consume current during transitions $0 \to 1$ and $1 \to 0$
- As seen with the inverter, consumption during the transitions $0 \to 1$ and $1 \to 0$ differ
- Denote consumptions: $i(0 \to 1) = \alpha$ and $i(1 \to 0) = \beta$
- α and β depend on a microcontroller (technology, layout), power supply, and many other parameters

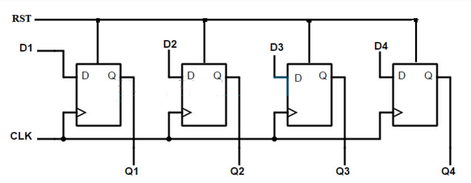
Registers



Data present at clock time is transfered from D to Q.

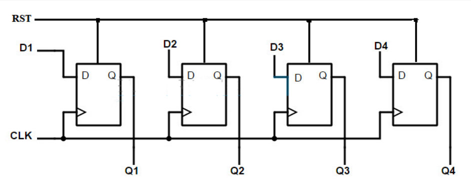
- Flip-flops consume current during transitions $0 \to 1$ and $1 \to 0$
- As seen with the inverter, consumption during the transitions $0 \to 1$ and $1 \to 0$ differ
- Denote consumptions: $i(0 \to 1) = \alpha$ and $i(1 \to 0) = \beta$
- α and β depend on a microcontroller (technology, layout), power supply, and many other parameters
- Assume that $\beta = \alpha + \delta$

Registers



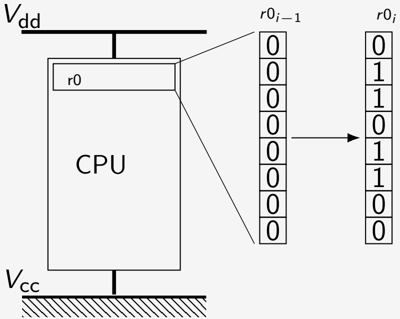
- Flip-flops are assembled into registers (CPU registers, special registers, etc.)

Registers



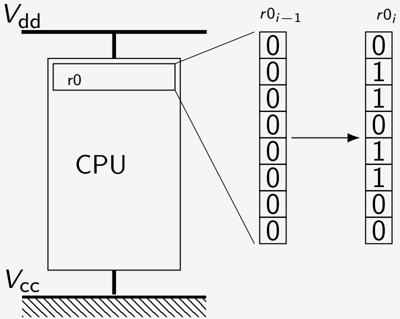
- Flip-flops are assembled into registers (CPU registers, special registers, etc.)
- Register's flip-flops have the same layout and components, so they consume equally

Power consumption of a register change



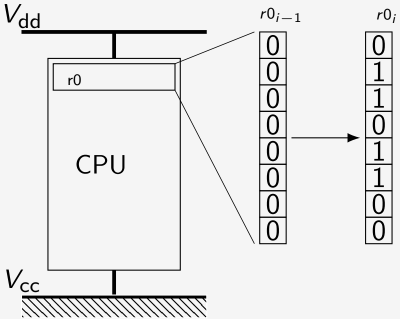
- When register switches the current is drained (power is consumed)

Power consumption of a register change



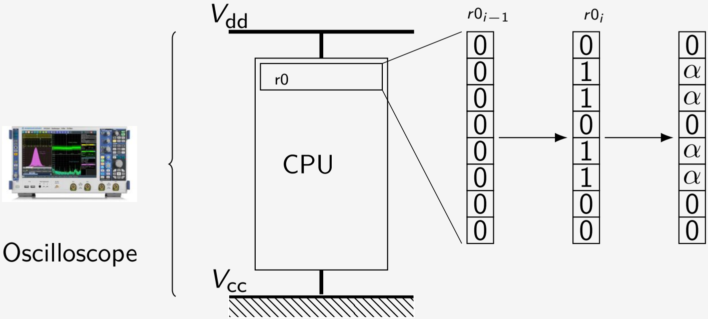
- When register switches the current is drained (power is consumed)
- The exact current consumption is not possible to model

Power consumption of a register change



- When register switches the current is drained (power is consumed)
- The exact current consumption is not possible to model
- Instead, side-channel analysis studies the differences between various operations (kind of true)

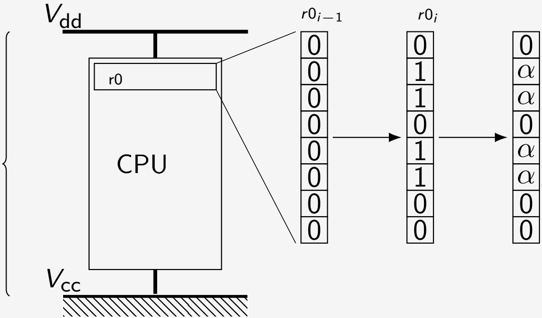
Power consumption of a register change



- Current drain of a transition $0x00 \rightarrow 0x6C$ shall be proportional to the number of switched bits: $i(0x00 \rightarrow 0x6C) \approx 4 \cdot i(0 \rightarrow 1) = 4 \cdot \alpha$

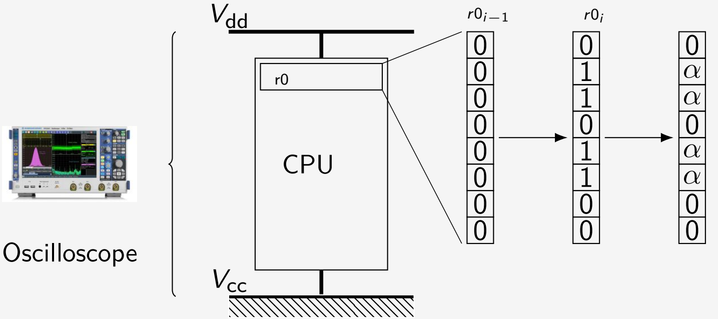
Power consumption of a register change

Oscilloscope



- Current drain of a transition $0x00 \rightarrow 0x6C$ shall be proportional to the number of switched bits: $i(0x00 \rightarrow 0x6C) \approx 4 \cdot i(0 \rightarrow 1) = 4 \cdot \alpha$
- In practice measurements of the transition will include noise σ: $i(0x00 \rightarrow 0x6C) = 4 \cdot \alpha + \sigma$

Power consumption of a register change



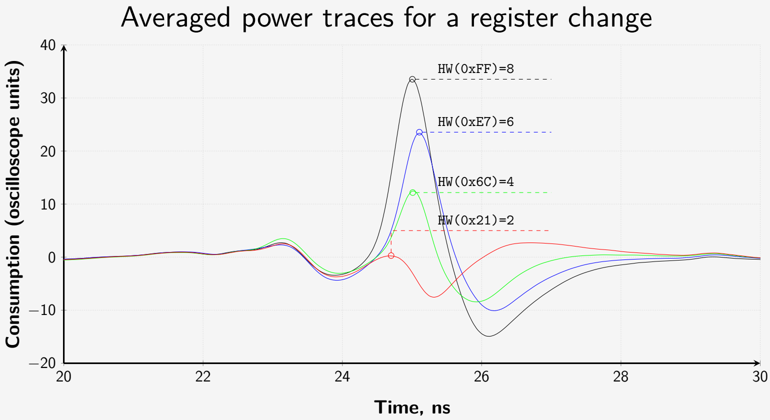
- Current drain of a transition $0x00 \rightarrow 0x6C$ shall be proportional to the number of switched bits: $i(0x00 \rightarrow 0x6C) \approx 4 \cdot i(0 \rightarrow 1) = 4 \cdot \alpha$
- In practice measurements of the transition will include noise σ: $i(0x00 \rightarrow 0x6C) = 4 \cdot \alpha + \sigma$
- The most significant noise is coming from other microcontroller blocks (video processor, modem, DMA) and environment (cross talks, EM signals around, etc.)

- Hamming distance denotes the number of different bits between the previous and the new register values: $\mathrm{HD}(r_{i-1}, r_i)$

- Hamming distance denotes the number of different bits between the previous and the new register values: $\mathrm{HD}(r_{i-1}, r_i)$
- $i(0x00 \rightarrow 0x6C) \approx \mathrm{HD}(0x00, 0x6C) \cdot i(0 \rightarrow 1) + \sigma = 4 \cdot \alpha + \sigma$

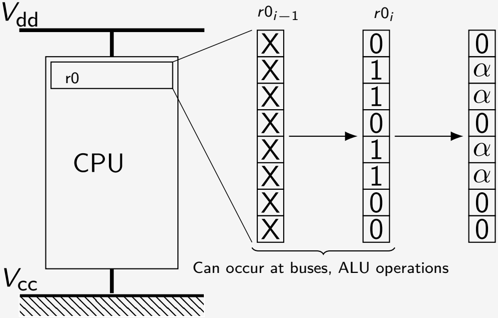
- Hamming distance denotes the number of different bits between the previous and the new register values: $\mathrm{HD}(r_{i-1}, r_i)$
- $i(0x00 \to 0x6C) \approx \mathrm{HD}(0x00, 0x6C) \cdot i(0 \to 1) + \sigma = 4 \cdot \alpha + \sigma$
- If the previous register value is equal to $0x00$ then we better use Hamming weight name (the number of bits set to 1 in the new register: $\mathrm{HW}(r_i) = \#1$)

- Hamming distance denotes the number of different bits between the previous and the new register values: $\mathrm{HD}(r_{i-1}, r_i)$
- $i(0x00 \rightarrow 0x6C) \approx \mathrm{HD}(0x00, 0x6C) \cdot i(0 \rightarrow 1) + \sigma = 4 \cdot \alpha + \sigma$
- If the previous register value is equal to $0x00$ then we better use Hamming weight name (the number of bits set to 1 in the new register: $\mathrm{HW}(r_i) = \#1$)
- Hamming weight model is practically confirmed by many side-channel attacks



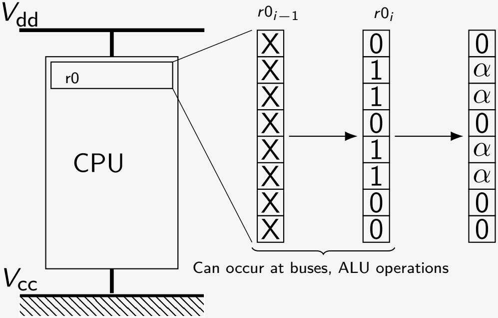
- $i(0x00 \rightarrow 0x21) = \text{HW}(0x21) \cdot \alpha + \sigma = 2 \cdot \alpha + \sigma$
- $i(0x00 \rightarrow 0x6C) = \text{HW}(0x6C) \cdot \alpha + \sigma = 4 \cdot \alpha + \sigma$
- $i(0x00 \rightarrow 0xE7) = \text{HW}(0xE7) \cdot \alpha + \sigma = 6 \cdot \alpha + \sigma$
- $i(0x00 \rightarrow 0xFF) = \text{HW}(0xFF) \cdot \alpha + \sigma = 8 \cdot \alpha + \sigma$

Power consumption of a register change



- Although previous register value can be different from $0x00$, this change $0x00 \rightarrow 0x6C$ can still be observed

Power consumption of a register change



- Although previous register value can be different from $0x00$, this change $0x00 \rightarrow 0x6C$ can still be observed
- The reason is that the register value is transmitted over buses (or written to other registers of ALU or other IPs)

Questions?

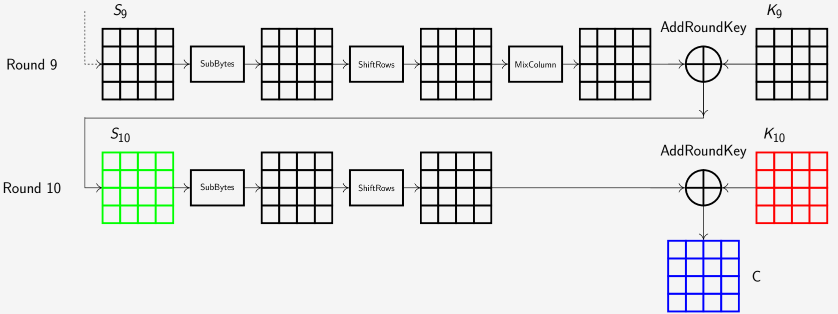
Questions?

- Consider 8-bit register. How many different values Hamming weight model can have: $\mathrm{HW}(r_i) = 0, 1, 2...?$

Questions?

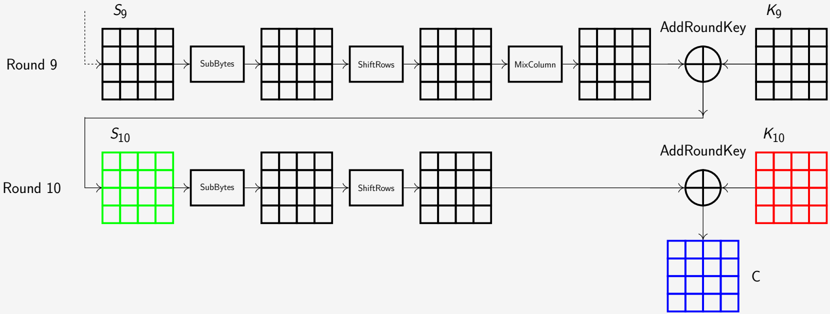
- Consider 8-bit register. How many different values Hamming weight model can have: $\mathrm{HW}(r_i) = 0, 1, 2...?$
- The same situation for 128-bit register (hardware implementation of AES): $\mathrm{HW}(r_i) = 0, 1, 2...?$

AES-128 Assignment 2



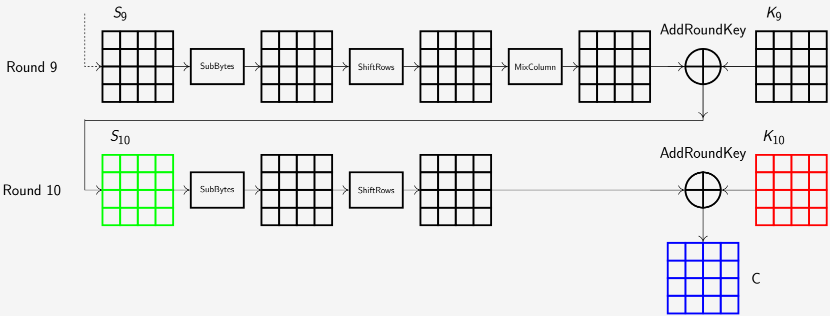
- This time a developer left a debug option that prints out Hamming weights of bytes of the 10th round state $HW(S_{10})$ and a ciphertext C. What can be wrong with that?

AES-128 Assignment 2



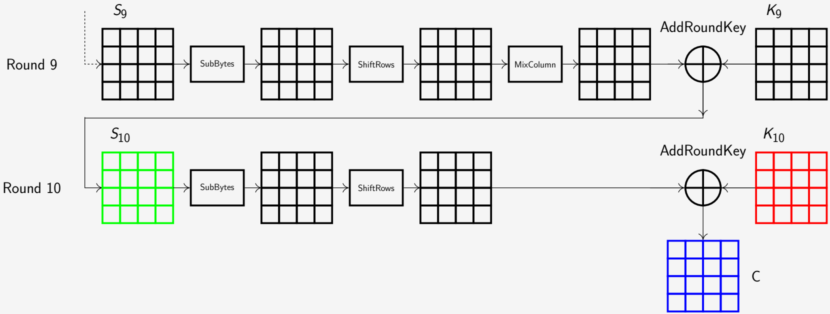
- This time a developer left a debug option that prints out Hamming weights of bytes of the 10th round state $HW(S_{10})$ and a ciphertext C. What can be wrong with that?
- Find K_{10} and then get the master key using the provided code

AES-128 Assignment 2



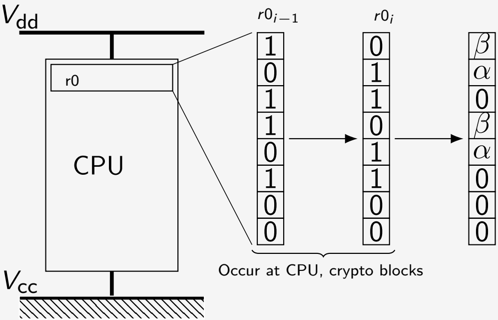
- This time a developer left a debug option that prints out Hamming weights of bytes of the 10th round state $HW(S_{10})$ and a ciphertext C. What can be wrong with that?
- Find K_{10} and then get the master key using the provided code
- $C = ShiftRows(SubBytes[S_{10}]) \oplus K_{10}$

AES-128 Assignment 2



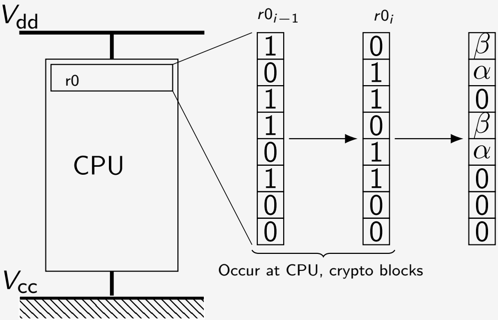
- This time a developer left a debug option that prints out Hamming weights of bytes of the 10th round state $HW(S_{10})$ and a ciphertext C. What can be wrong with that?
- Find K_{10} and then get the master key using the provided code
- $C = ShiftRows(SubBytes[S_{10}]) \oplus K_{10}$
- $HW(S_{10}) = HW(InvSubBytes[InvShiftRows(K_{10} \oplus C)])$

Power consumption of a register change



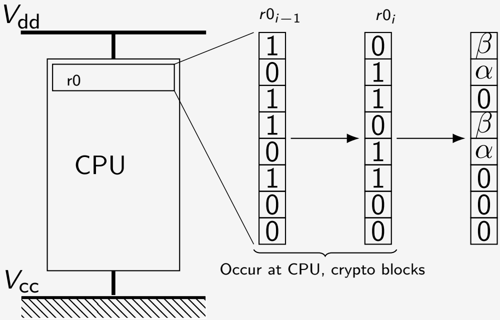
- If a previous register value is different from 0x00, the current drain can be still modelled

Power consumption of a register change



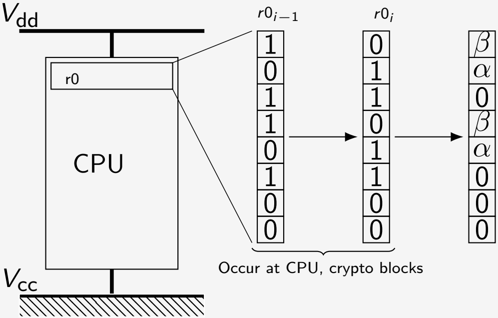
- If a previous register value is different from 0x00, the current drain can be still modelled
- Assume current drains: $i(0 \rightarrow 1) = \alpha$, and $i(1 \rightarrow 0) = \beta$
- The current is never balanced: $\alpha \neq \beta, \beta = \alpha + \delta$

Power consumption of a register change



- $i(0xD4 \to 0x6C) = 2 \cdot \alpha + 2 \cdot \beta + \sigma$

Power consumption of a register change



- $i(0xD4 \rightarrow 0x6C) = 2 \cdot \alpha + 2 \cdot \beta + \sigma$
- Side-channel attack people were lasy... so they assume that $|\alpha| >> |\delta|$ and $|\beta| >> |\delta|$, so they treat $|\delta|$ as colored noise
- $i(0xD4 \rightarrow 0x6C) = 2 \cdot \alpha + 2 \cdot \beta + \sigma = 2 \cdot \alpha + 2 \cdot (\alpha + \delta) + \sigma = 4 \cdot \alpha + (2 \cdot \delta + \sigma) = 4 \cdot \alpha + \sigma' = \mathrm{HD}(0xD4, 0x6C) \cdot \alpha + \sigma'$

- If a previous r_{i-1} and a new r_i register values are known, side-channel attacks use Hamming Distance model
- $i(r_{i-1} \rightarrow r_i) = \mathrm{HD}(r_{i-1}, r_i) \cdot \alpha + \sigma$

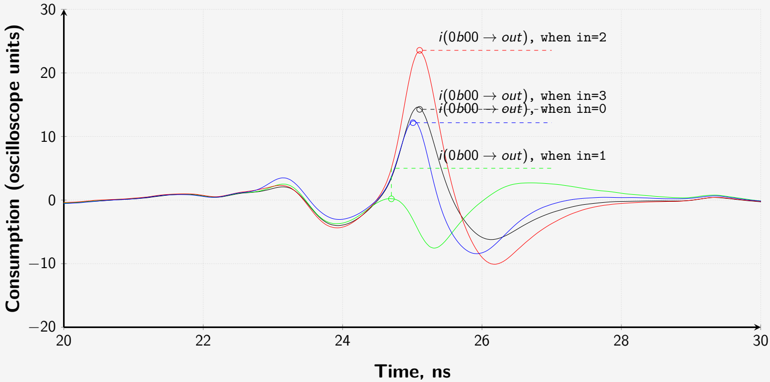
- If a previous r_{i-1} and a new r_i register values are known, side-channel attacks use Hamming Distance model
- $i(r_{i-1} \to r_i) = \mathrm{HD}(r_{i-1}, r_i) \cdot \alpha + \sigma$
- Yes, sometime people try to characterize δ and use more precise models with weights:
- $\alpha \neq \beta, \beta = 1.1 \cdot \alpha$

- If a previous r_{i-1} and a new r_i register values are known, side-channel attacks use Hamming Distance model
- $i(r_{i-1} \rightarrow r_i) = \mathrm{HD}(r_{i-1}, r_i) \cdot \alpha + \sigma$
- Yes, sometime people try to characterize δ and use more precise models with weights:
- $\alpha \neq \beta, \beta = 1.1 \cdot \alpha$
- In that case our previous Hamming distance become equal:
- $i(0xD4 \rightarrow 0x6C) = 2 \cdot \alpha + 2 \cdot \beta + \sigma = 2 \cdot \alpha + 2 \cdot 1.1 \cdot \alpha + \sigma = 4.2 \cdot \alpha + \sigma$

- If a previous r_{i-1} and a new r_i register values are known, side-channel attacks use Hamming Distance model
- $i(r_{i-1} \rightarrow r_i) = \mathrm{HD}(r_{i-1}, r_i) \cdot \alpha + \sigma$
- Yes, sometime people try to characterize δ and use more precise models with weights:
- $\alpha \neq \beta, \beta = 1.1 \cdot \alpha$
- In that case our previous Hamming distance become equal:
- $i(0xD4 \rightarrow 0x6C) = 2 \cdot \alpha + 2 \cdot \beta + \sigma = 2 \cdot \alpha + 2 \cdot 1.1 \cdot \alpha + \sigma = 4.2 \cdot \alpha + \sigma$
- Characterisation of the coefficient θ: $\beta = \theta \cdot \alpha$ takes time, plus it might depend on the current operation

- If a previous r_{i-1} and a new r_i register values are known, side-channel attacks use Hamming Distance model
- $i(r_{i-1} \rightarrow r_i) = \mathrm{HD}(r_{i-1}, r_i) \cdot \alpha + \sigma$
- Yes, sometime people try to characterize δ and use more precise models with weights:
- $\alpha \neq \beta, \beta = 1.1 \cdot \alpha$
- In that case our previous Hamming distance become equal:
- $i(0xD4 \rightarrow 0x6C) = 2 \cdot \alpha + 2 \cdot \beta + \sigma = 2 \cdot \alpha + 2 \cdot 1.1 \cdot \alpha + \sigma = 4.2 \cdot \alpha + \sigma$
- Characterisation of the coefficient θ: $\beta = \theta \cdot \alpha$ takes time, plus it might depend on the current operation
- Nevertheless, if you get the key with the Hamming distance model you don't need to care of deeper characterisation

Pen and pensil excersise

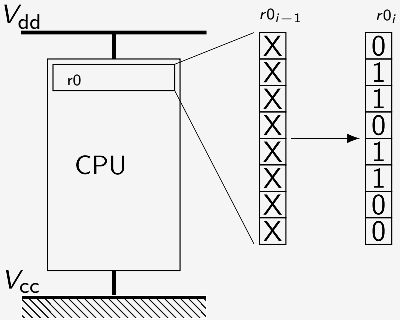


- The CPU computes $in \oplus key = out$ on 2-bit registers *in* and *out*
- Input values are known to an attacker (0,1,2,3)
- Output values are not known to an attacker; but for each *in* value he measured power consumption of the *out* register transition $0b00 \to out$
- You need to find the *key* (by hands)

Hamming distance key points

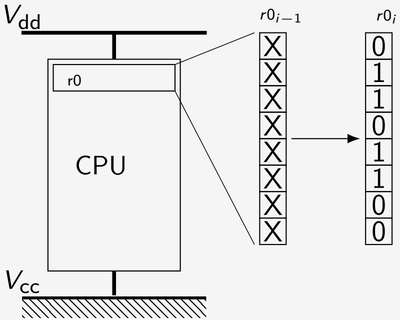
- By looking into power consumption Hamming distance model can be distinguished: $i(r_{i-1} \rightarrow r_i) = \mathrm{HD}(r_{i-1}, r_i) \cdot \alpha + \sigma$
- The "pen and pensil" attack is kind of side-channel attack
- Real side-channel attacks use slightly more advanced statistics than visual observations

Power consumption of a register change



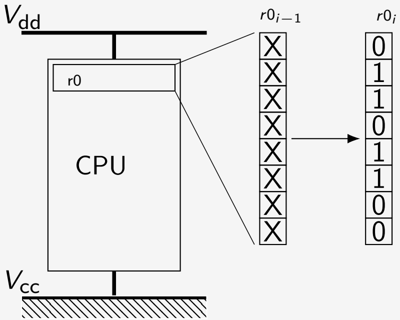
- What if you don't know the previous register value?

Power consumption of a register change



- What if you don't know the previous register value?
- The only approach is to use Hamming weight model: $i(0x00 \rightarrow r_i) = \mathrm{HW}(r_i) \cdot \alpha + \sigma$

Power consumption of a register change



- What if you don't know the previous register value?
- The only approach is to use Hamming weight model: $i(0x00 \rightarrow r_i) = \mathrm{HW}(r_i) \cdot \alpha + \sigma$
- Hamming weight model also works when the previous register value r_{i-1} is random

- Why Hamming weight model works even if the previous register value r_{i-1} is random

- Why Hamming weight model works even if the previous register value r_{i-1} is random
- Consider a situation of 1 single flip-flop (one bit)

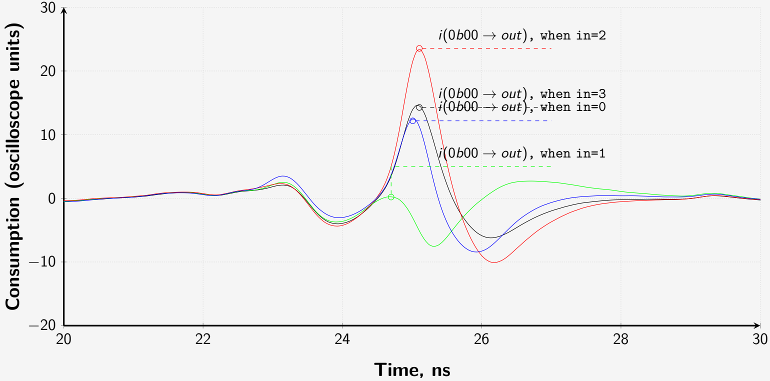
- Why Hamming weight model works even if the previous register value r_{i-1} is random
- Consider a situation of 1 single flip-flop (one bit)
- Many measurements of the operations $i(X \to 0)$ and $i(X \to 1)$ are performed

- Why Hamming weight model works even if the previous register value r_{i-1} is random
- Consider a situation of 1 single flip-flop (one bit)
- Many measurements of the operations $i(X \to 0)$ and $i(X \to 1)$ are performed
- An attacker only knowns that previous flip-flop value X is random (however he knows the new register value)

- Let us average the power measurements for $i(X \to 0)$ and $i(X \to 1)$

$$\sum_0^n \frac{i(X \to 0)}{n} = \sum_0^{n/2} \frac{i(0 \to 0)}{n/2} + \sum_0^{n/2} \frac{i(1 \to 0)}{n/2} + \sum_0^n \frac{\sigma}{n} = 0 + \alpha + const$$

$$\sum_0^n \frac{i(X \to 1)}{n} = \sum_0^{n/2} \frac{i(0 \to 1)}{n/2} + \sum_0^{n/2} \frac{i(1 \to 1)}{n/2} + \sum_0^n \frac{\sigma}{n} =$$
$$= \beta + 0 + const = \alpha + \delta + 0 + const$$



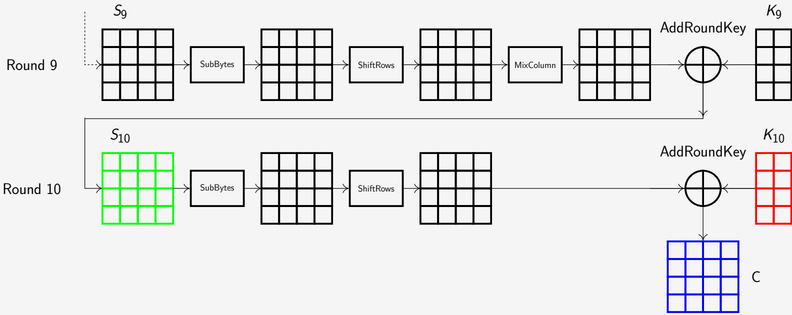
- Remember this excersise?

- So far we have considered three pieces of the puzzle:
  - CMOS power consumption to understand the nature of leakage
  - Law of large numbers to deal with noise
  - Power consumption models
- This is normal if you did not catch any of the topics

- So far we have considered three pieces of the puzzle:
  - CMOS power consumption to understand the nature of leakage
  - Law of large numbers to deal with noise
  - Power consumption models
- This is normal if you did not catch any of the topics
- We will consider them by excersises

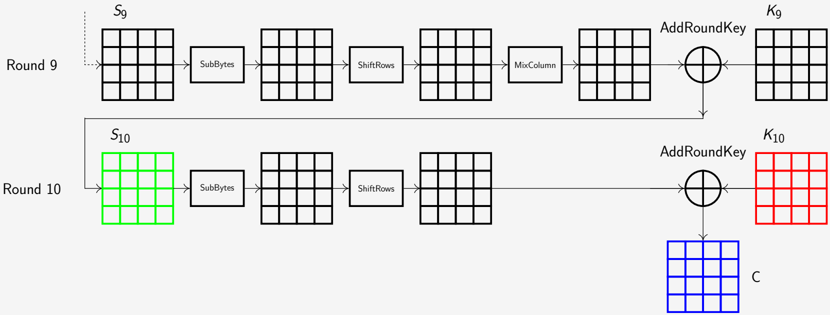
- So far we have considered three pieces of the puzzle:
  - CMOS power consumption to understand the nature of leakage
  - Law of large numbers to deal with noise
  - Power consumption models
- This is normal if you did not catch any of the topics
- We will consider them by excersises
- All the excersises will be based on AES examples

AES-128 Assignment 3



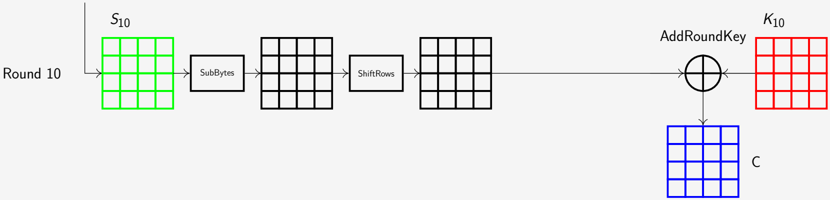
- This time a developer left a debug option that prints out Hamming weights of bytes of the 10th round state $HW(S_{10})$ somewhere among other useless information and a ciphertext C. What can be wrong with that?

AES-128 Assignment 3



- This time a developer left a debug option that prints out Hamming weights of bytes of the 10th round state $HW(S_{10})$ somewhere among other useless information and a ciphertext C. What can be wrong with that?
- Find K_{10} and then get the master key using the provided code

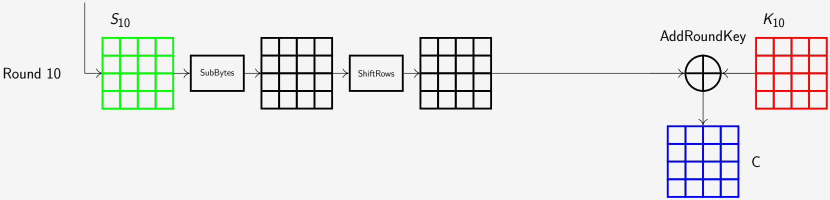
AES-128 Assignment 3 Recap



C^1	18	···	72
C^2	2f	···	89
C^i	···	···	···
C^n	5d	···	b7

	Random data				Real leakage			Random data	
L_t^1	7	3	8	4	5	···	8	6	2
L_t^2	5	1	6	2	3	···	7	4	0
L_t^i	···	···	···	···	···	···	···	···	···
L_t^n	1	6	2	7	3	···	5	0	5

AES-128 Assignment 3 Solution scheme



C^1	18	···	72
C^2	2f	···	89
C^i	···	···	···
C^n	5d	···	b7

	Random data				Real leakage			Random data	
L_t^1	7	3	8	4	5	···	8	6	2
L_t^2	5	1	6	2	3	···	7	4	0
L_t^i	···	···	···	···	···	···	···	···	···
L_t^n	1	6	2	7	3	···	5	0	5

Create power consumption model for one Sbox operation

$K_0 = 0$	$K_j = j$	$K_{FF} = 255$
$HW(\mathrm{Sb}^{-1}[K_0 \oplus 18]) = 5$	···	$HW(\mathrm{Sb}^{-1}[K_{FF} \oplus 18]) = 2$
$HW(\mathrm{Sb}^{-1}[K_0 \oplus 2f]) = 3$	···	$HW(\mathrm{Sb}^{-1}[K_{FF} \oplus 2f]) = 0$
···	···	···
$HW(\mathrm{Sb}^{-1}[K_0 \oplus 5d]) = 3$	···	$HW(\mathrm{Sb}^{-1}[K_{FF} \oplus 5d]) = 5$

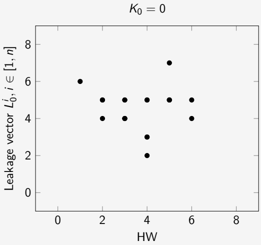
AES-128 Assignment 3 Solution scheme with visualisation

C^1	**18**	···	72
C^2	**2f**	···	89
C^i	···	···	···
C^n	**5d**	···	b7

	Random data				Real leakage			Random data	
L_t^1	7	3	8	4	5	···	8	6	2
L_t^2	5	1	6	2	3	···	7	4	0
L_t^i	···	···	···	···	···	···	···	···	···
L_t^n	1	6	2	7	3	···	5	0	5

Create power consumption model for one Sbox operation

$K_0 = 0$	$K_j = j$	$K_{FF} = 255$
$HW(\mathrm{Sb}^{-1}[K_0 \oplus 18]) = 5$	···	$HW(\mathrm{Sb}^{-1}[K_{FF} \oplus 18]) = 2$
$HW(\mathrm{Sb}^{-1}[K_0 \oplus 2f]) = 3$	···	$HW(\mathrm{Sb}^{-1}[K_{FF} \oplus 2f]) = 0$
···	···	···
$HW(\mathrm{Sb}^{-1}[K_0 \oplus 5d]) = 3$	···	$HW(\mathrm{Sb}^{-1}[K_{FF} \oplus 5d]) = 5$



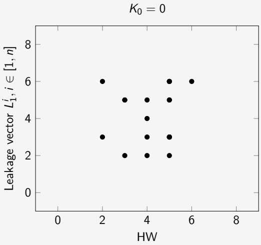
AES-128 Assignment 3 Solution scheme with visualisation

C^1	18	···	72
C^2	2f	···	89
C^i	···	···	···
C^n	5d	···	b7

	Random data				Real leakage			Random data	
L_t^1	7	3	8	4	5	···	8	6	2
L_t^2	5	1	6	2	3	···	7	4	0
L_t^i	···	···	···	···	···	···	···	···	···
L_t^n	1	6	2	7	3	···	5	0	5

Create power consumption model for one Sbox operation

$K_0 = 0$	$K_j = j$	$K_{FF} = 255$
$HW(\mathrm{Sb}^{-1}[K_0 \oplus 18]) = 5$	···	$HW(\mathrm{Sb}^{-1}[K_{FF} \oplus 18]) = 2$
$HW(\mathrm{Sb}^{-1}[K_0 \oplus 2f]) = 3$	···	$HW(\mathrm{Sb}^{-1}[K_{FF} \oplus 2f]) = 0$
···	···	···
$HW(\mathrm{Sb}^{-1}[K_0 \oplus 5d]) = 3$	···	$HW(\mathrm{Sb}^{-1}[K_{FF} \oplus 5d]) = 5$



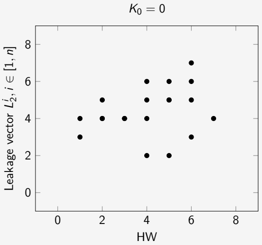
AES-128 Assignment 3 Solution scheme with visualisation

C^1	18	···	72
C^2	2f	···	89
C^i	···	···	···
C^n	5d	···	b7

	Random data				Real leakage			Random data	
L_t^1	7	3	8	4	5	···	8	6	2
L_t^2	5	1	6	2	3	···	7	4	0
L_t^i	···	···	···	···	···	···	···	···	···
L_t^n	1	6	2	7	3	···	5	0	5

Create power consumption model for one Sbox operation

$K_0 = 0$	$K_j = j$	$K_{FF} = 255$
$HW(\mathrm{Sb}^{-1}[K_0 \oplus 18]) = 5$	···	$HW(\mathrm{Sb}^{-1}[K_{FF} \oplus 18]) = 2$
$HW(\mathrm{Sb}^{-1}[K_0 \oplus 2f]) = 3$	···	$HW(\mathrm{Sb}^{-1}[K_{FF} \oplus 2f]) = 0$
···	···	···
$HW(\mathrm{Sb}^{-1}[K_0 \oplus 5d]) = 3$	···	$HW(\mathrm{Sb}^{-1}[K_{FF} \oplus 5d]) = 5$



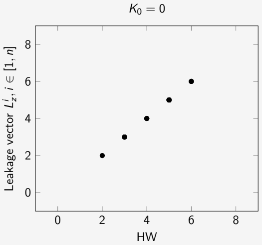
AES-128 Assignment 3 Solution scheme with visualisation

C^1	18	...	72
C^2	2f	...	89
C^i	...	...	...
C^n	5d	...	b7

	Random data				Real leakage			Random data	
L_t^1	7	3	8	4	5	...	8	6	2
L_t^2	5	1	6	2	3	...	7	4	0
L_t^i	...	...	...	...	...	...	...	...	...
L_t^n	1	6	2	7	3	...	5	0	5

Create power consumption model for one Sbox operation

$K_0 = 0$	$K_j = j$	$K_{FF} = 255$
$HW(\text{Sb}^{-1}[K_0 \oplus 18]) = 5$	...	$HW(\text{Sb}^{-1}[K_{FF} \oplus 18]) = 2$
$HW(\text{Sb}^{-1}[K_0 \oplus 2\text{f}]) = 3$	...	$HW(\text{Sb}^{-1}[K_{FF} \oplus 2\text{f}]) = 0$
...	...	...
$HW(\text{Sb}^{-1}[K_0 \oplus 5\text{d}]) = 3$	...	$HW(\text{Sb}^{-1}[K_{FF} \oplus 5\text{d}]) = 5$



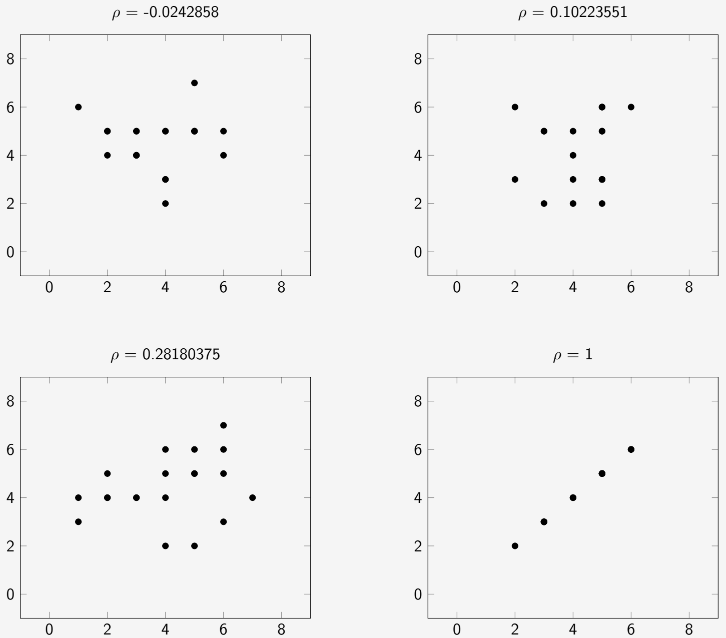
Dependency between two vectors

x	$f(x) \rightarrow y$	y
142	$f(142) \rightarrow 78$	78
147	$f(147) \rightarrow 25$	25
248	$f(248) \rightarrow 42$	42
132	$f(132) \rightarrow 126$	126
91	$f(91) \rightarrow 30$	30
50	$f(50) \rightarrow 100$	100
...		...
160	$f(160) \rightarrow 62$	62

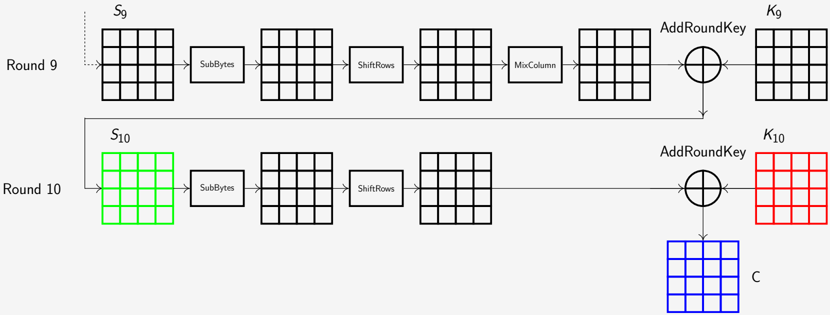
- A dependency between two vectors can be mathematically computed
- There are more than 40 different ways to compute the dependency
- We will focus on Pearson Correlation Coefficient

$$\rho = \frac{\sum_{i=0}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=0}^{n}(x_i - \bar{x}^2)}\sqrt{\sum_{i=0}^{n}(y_i - \bar{y}^2)}}$$

- Pearson Correlation Coefficient shows only if the two vectors have linear dependency
- PCC takes values from -1 to 1

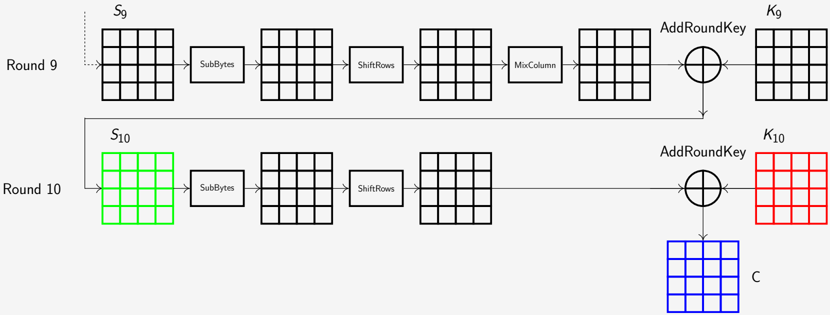

ρ = -0.0242858
ρ = 0.10223551
ρ = 0.28180375
$\rho = 1$
0 2 4 6 8
0 2 4 6 8

AES-128 Assignment 4



- This time a developer left a debug option that prints out Hamming weights of bytes of the 10th round state $HW(S_{10})$ among other useless information and a ciphertext C. All the information (except the ciphertext) has noise. What can be wrong with that?

AES-128 Assignment 4



- This time a developer left a debug option that prints out Hamming weights of bytes of the 10th round state $HW(S_{10})$ among other useless information and a ciphertext C. All the information (except the ciphertext) has noise. What can be wrong with that?
- Find K_{10} and then get the master key using the provided code

AES-128 Assignment 4 Recap

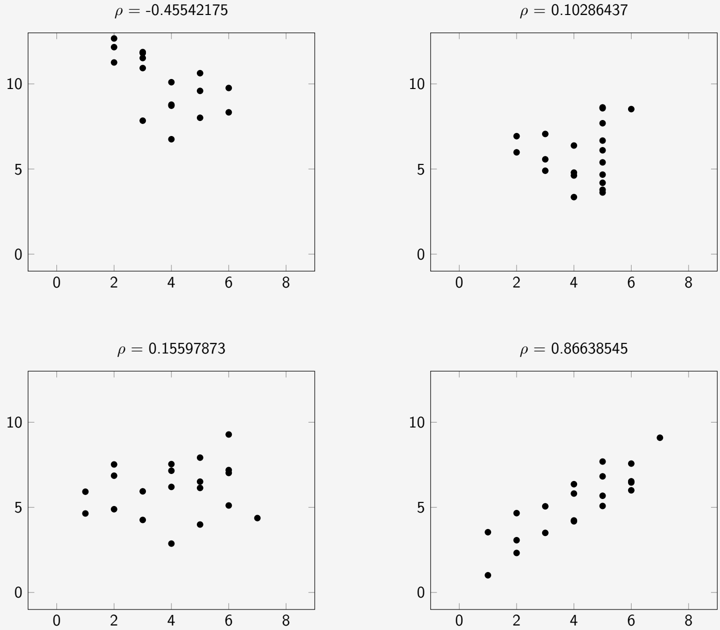
C^1	18	···	72
C^2	2f	···	89
C^i	···	···	···
C^n	5d	···	b7

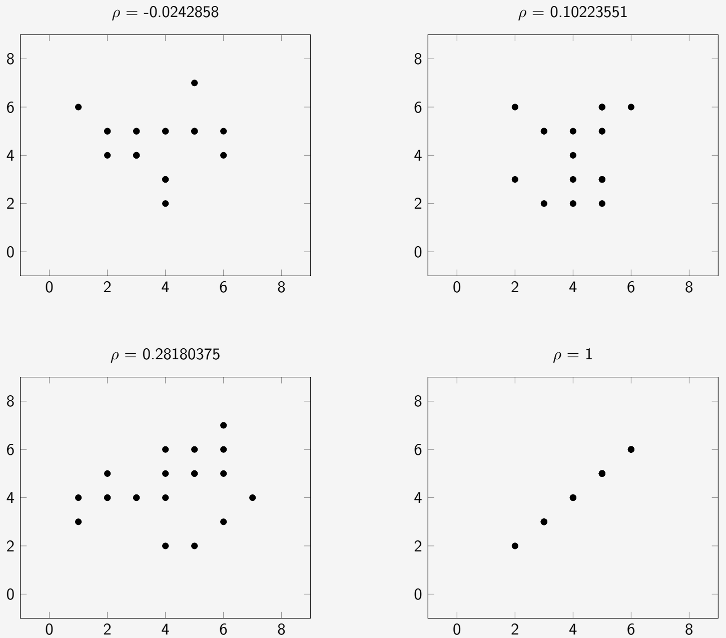
	Random data				Leakage + noise			Random data	
L_t^1	7.8	4.4	8	4.6	5.5	···	8.4	7	3.6
L_t^2	6.1	2.7	6.3	2.9	3.8	···	7.7	5.3	1.9
L_t^i	···	···	···	···	···	···	···	···	···
L_t^n	2.7	6.3	2.9	8.5	4.4	···	6.3	1.9	5.5

AES-128 Assignment 3 Recap

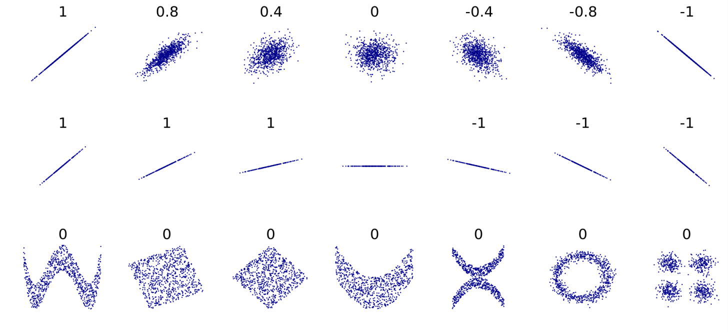
C^1	18	⋯	72
C^2	2f	⋯	89
C^i	⋯	⋯	⋯
C^n	5d	⋯	b7

	Random data				Real leakage			Random data	
L_t^1	7	3	8	4	5	⋯	8	6	2
L_t^2	5	1	6	2	3	⋯	7	4	0
L_t^i	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯
L_t^n	1	6	2	7	3	⋯	5	0	5


ρ = -0.45542175
ρ = 0.10286437
ρ = 0.15597873
ρ = 0.86638545
0
5
10
0
2
4
6
8


ρ = -0.0242858
ρ = 0.10223551
ρ = 0.28180375
$\rho = 1$
0
2
4
6
8

Pearson correlation coefficient

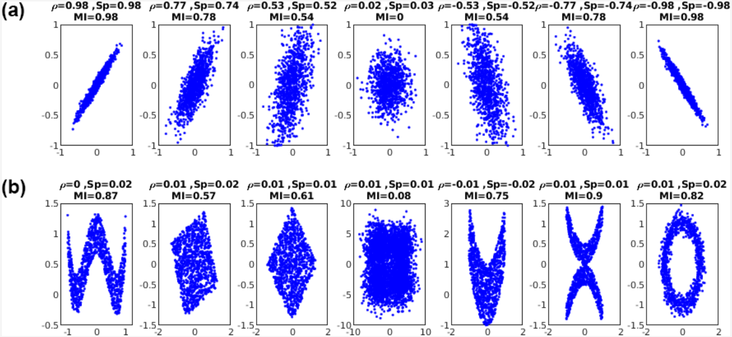


Python code for:

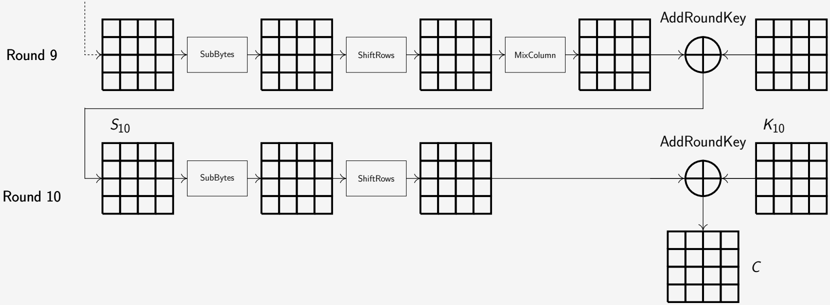
- Pearson Correlation Coefficient
- Spearman Correlation Coefficient
- Mutual Information

```
import numpy
from scipy import stats
from sklearn.metrics import adjusted_mutual_info_score

arr1 = numpy.random.randint(0,20, (5000))
arr2 = numpy.random.randint(0,20, (5000))

pcc = numpy.corrcoef(arr1, arr2)[0,1]
sp = scipy.stats.spearmanr(arr1, arr2)[0]
mi = adjusted_mutual_info_score(arr1,arr2)

print('PCC SP MI',pcc, sp, mi)
```

PCC, SP, MI


(a)
ρ=0.98 ,Sp=0.98 MI=0.98
ρ=0.77 ,Sp=0.74 MI=0.78
ρ=0.53 ,Sp=0.52 MI=0.54
ρ=0.02 ,Sp=0.03 MI=0
ρ=-0.53 ,Sp=-0.52 MI=0.54
ρ=-0.77 ,Sp=-0.74 MI=0.78
ρ=-0.98 ,Sp=-0.98 MI=0.98
(b)
ρ=0 ,Sp=0.02 MI=0.87
ρ=0.01 ,Sp=0.02 MI=0.57
ρ=0.01 ,Sp=0.01 MI=0.61
ρ=0.01 ,Sp=0.01 MI=0.08
ρ=-0.01 ,Sp=-0.02 MI=0.75
ρ=0.01 ,Sp=0.01 MI=0.9
ρ=0.01 ,Sp=0.02 MI=0.82

AES-128 Assignment 5



- This time we have real power measurements taken during AES-128 software implementation on ESP32
- Find the master key

Thank you!

Roman Korkikian, Nicolas Oberli

Side-channels and Fault Attacks
February 23rd, 2023 - June 29th, 2023

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch