# System Security HS2023

Srdjan Čapkun

# Lecturers

Prof. Srdjan Čapkun
https://syssec.ethz.ch/

Prof. Shweta Shinde
https://sectrs.ethz.ch/

# What is system security?

Introductory lectures and courses focus on simplified models

Alice, Bob, Eve

**Real systems are more complicated than that.**

Think of:

multi tenancy (e.g., smartphone),

Computation outsourcing (e.g., cloud),

Interaction with the physical environment,

User interfaces …

# Security is “everywhere”

- Hardware and software
- OS, apps, UI
- design and implementation
- digital and physical
- ...

# Topics covered by this course

Main themes:

1. **Basic principles of secure system design**
2. **Common ways to attack systems**

# Topics covered by this course

Selection of topics:

- Topic 1. Side Channels and Tempest
- Topic 2. Architectural Support for Security
- Topic 3. Trusted Execution Environments

- Topic 4. Attacks and vulnerabilities
- Topic 5. Bug finding
- Topic 6. OS & Virtualization Security

# Exam

- 20% exercises and related reports
- 80% final written exam (in session)

- You can expect one question / problem per Topic.
- Questions might come from lab sessions as well.

# Lectures Logistics

Every Mon 10:15-12:00
HG D 1.2

Slides will be uploaded to Moodle

# Exercise Organisation

# Your TAs

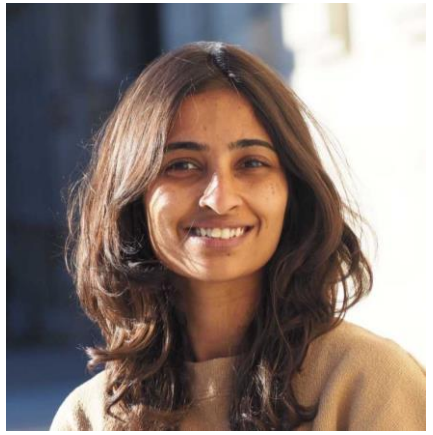
System Security Group

Friederike Groschupp

Dr. Yoshimichi Nakatsuka

Secure & Trustworthy Systems Group

Mark Kuhne



Supraja Sridhara

- Student TAs
  - Riccardo Negri
  - Andrea Lepori

# Contact

- Moodle forums
  - Questions about lecture and exercise content and general administration
  - Separate Forums for graded assignments
- SysSec-Exercise@lists.inf.ethz.ch
  - Questions about your individual situation
  - Please do not send emails to our personal emails

# Exercise Sessions

- Thursday 14:15 – 16:00 in HG D3.2
- Thursday 16:15 – 18:00 in CAB G11

- Both exercise sessions cover the same topics
- Sessions are not recorded
- Most exercises are ungraded and handed out the week before they are discussed in the exercise sessions
- First exercise session this Thursday (28.09.)

# Graded Assignments

- 3 graded assignments that make up 20% of the final grade
- Two weeks to solve, no exercise session in the middle
- No extensions possible – plan ahead

- Submission through Moodle
- Grading is done manually and might take some time

- If you did the course already: You can keep the grades for the graded assignments that *stayed the same*. Write us an email in case you want to do this.

# Graded Assignment Schedule

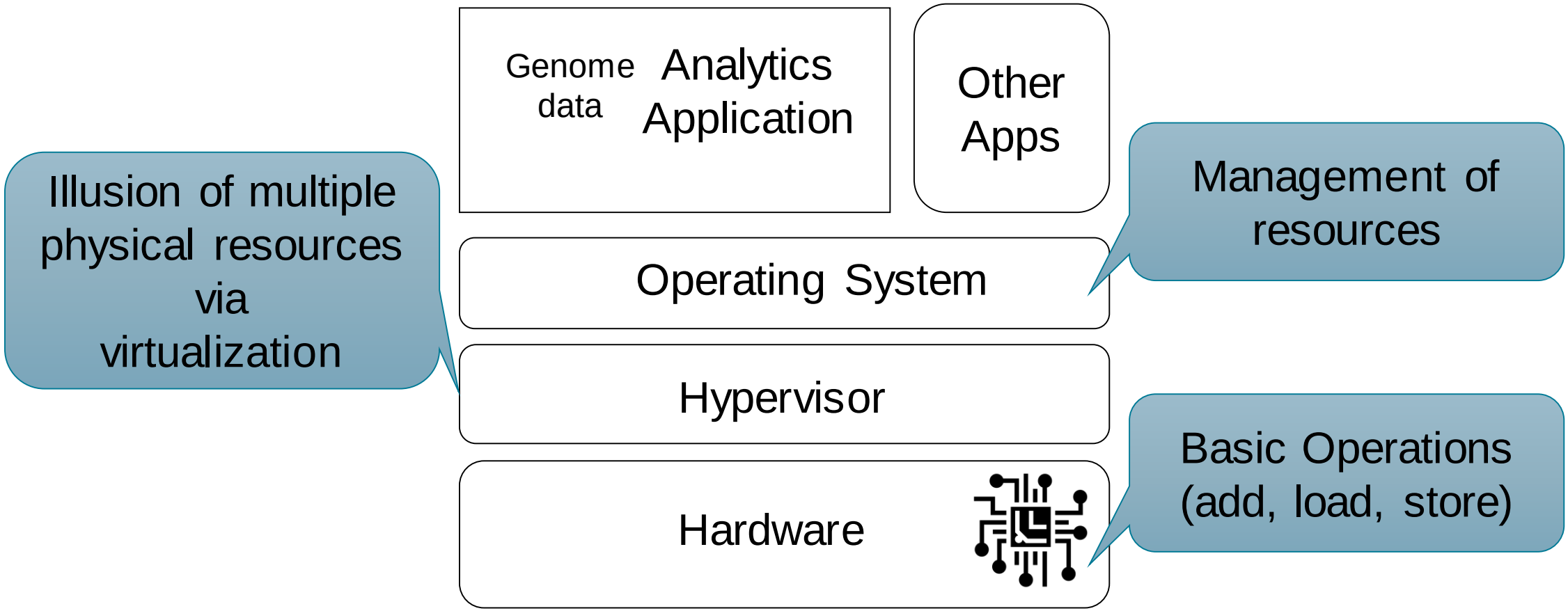
- Assignment 1
  - Publication: 28.09, 14:00
  - Submission: 12.10, 14:00
- Assignment 2
  - Publication: 19.10, 14:00
  - Submission: 02.11, 14:00
- Assignment 3
  - Publication: 23.11, 14:00
  - Submission: 07.12, 14:00

# Overview

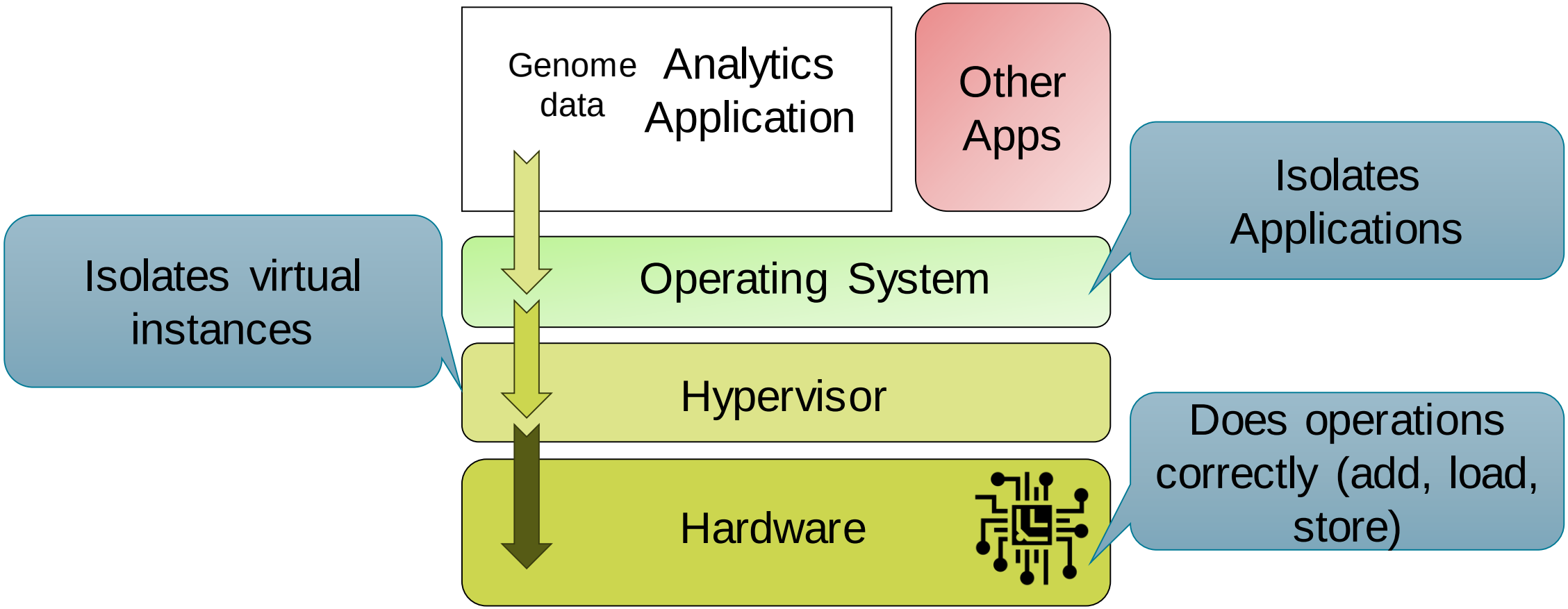
# Course Goals

- Critically assess the security flaws in a given system
- Detect and exploits security bugs
- Design secure applications
- Implement secure solutions while being aware of their limitations

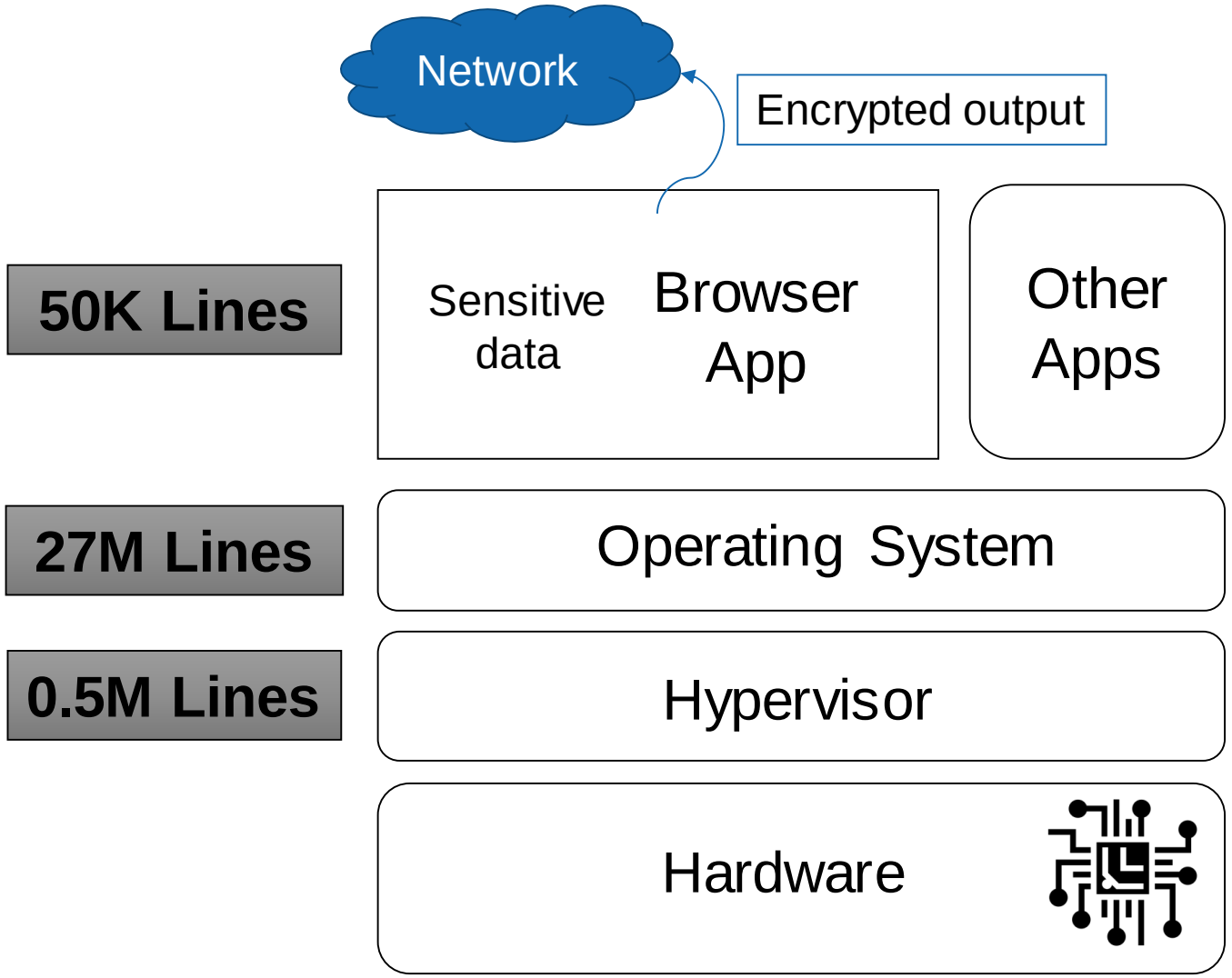
# A Brief History of Computer Software Systems



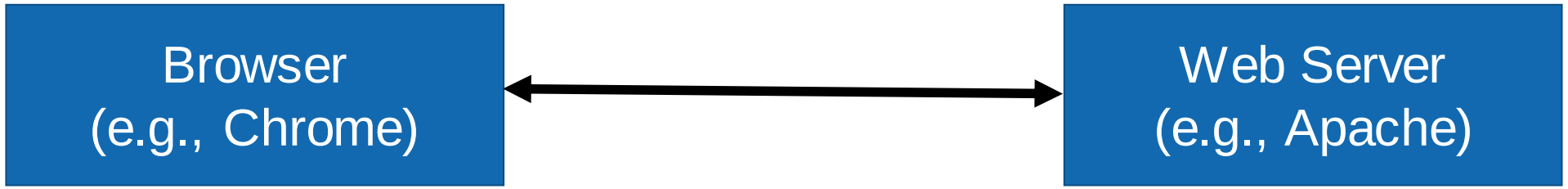
# A Brief History of Computer Software Security



# Making the computing stack secure



# Example Scenario




- What can go wrong?
- Who is the attacker?
  - What is their capability?
- What can we trust?
- What security do we want to achieve?

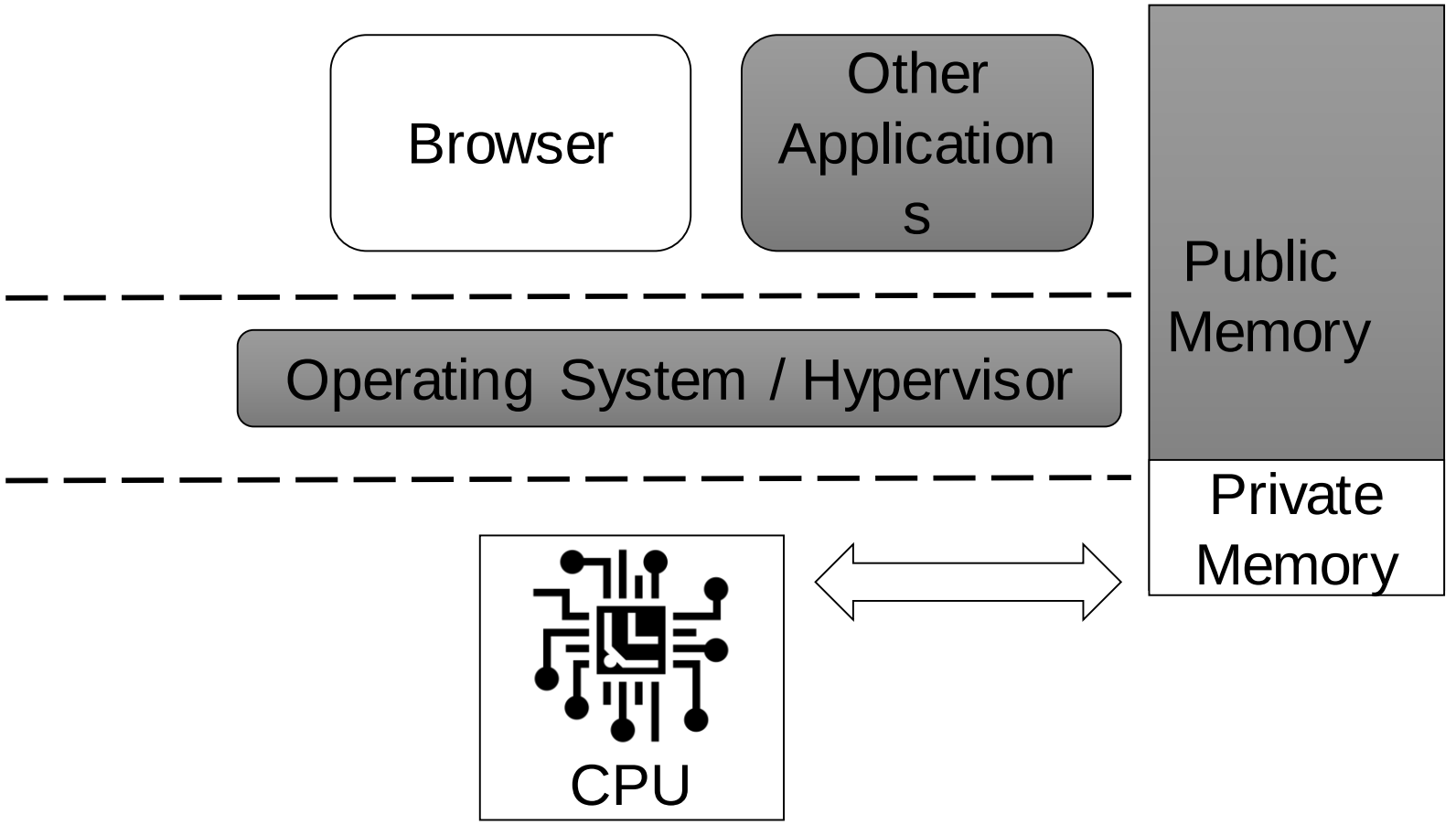
# Side Channels

- In the context of the browser
  - On the phone
  - On the laptop
  - On a shared device
- In the context of the webserver
  - On a dedicated hosting environment
  - On a cloud machine

# Architectural Support for Security

- Privilege levels
- Address space virtualization
- MMU
- Memory encryption

# Trusted Execution Environments

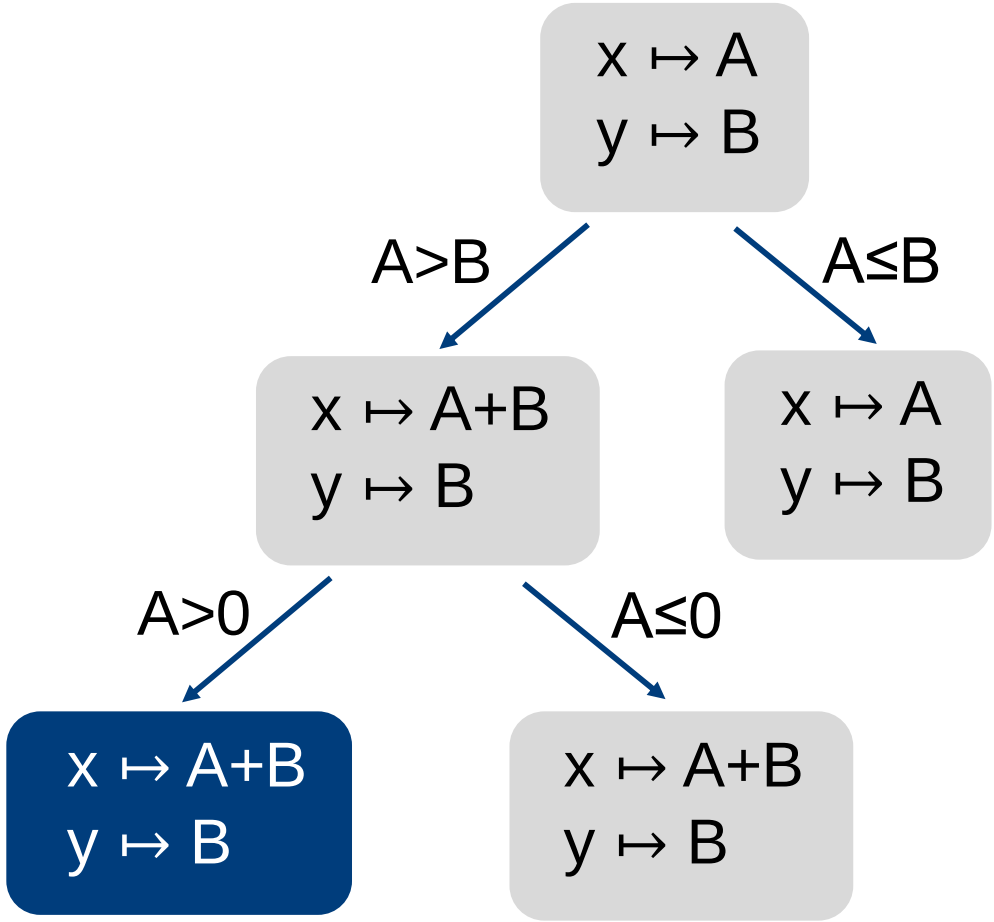


# Attacks and vulnerabilities

```
char buf[BUFSIZE];
gets(buf);
```

# Bug finding

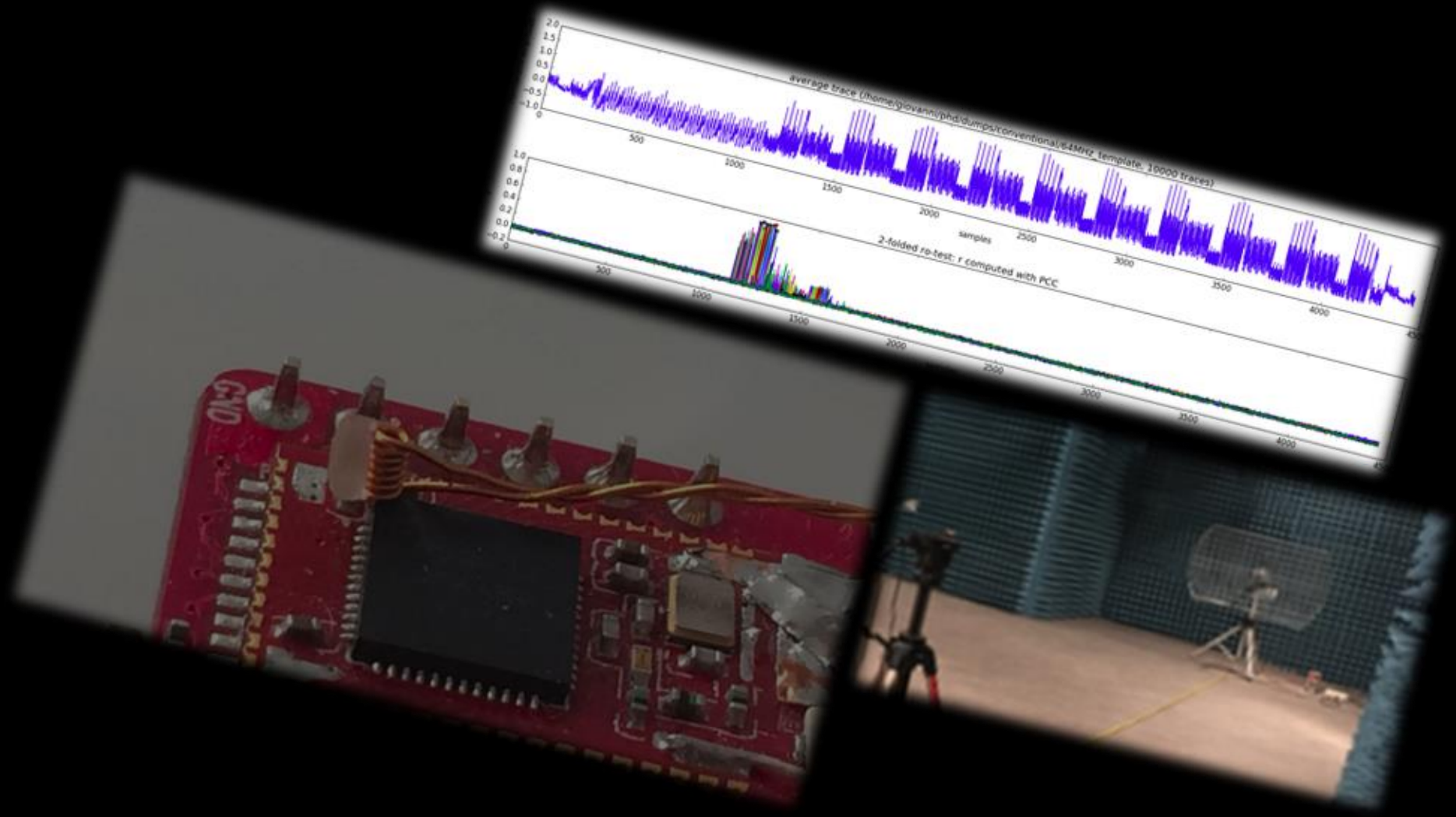
```
1 def f (x, y):
2   if (x>y)
3     x = x+y
4     if (x-y > 0)
5       assert false
6   return (x, y)
```




A and B are symbolic variables

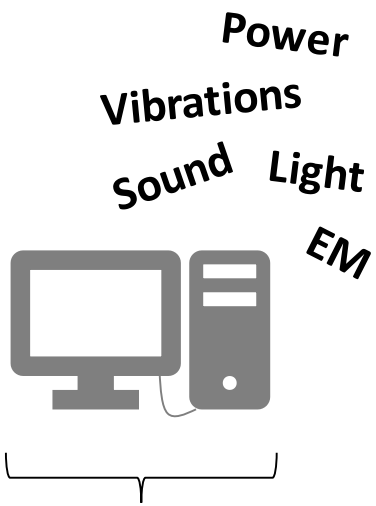
# OS & Virtualization Security

- Syscall Filtering
- Memory Safety
- Nested Virtualization




# 1 – Side Channels & Tempest

# Emission Security



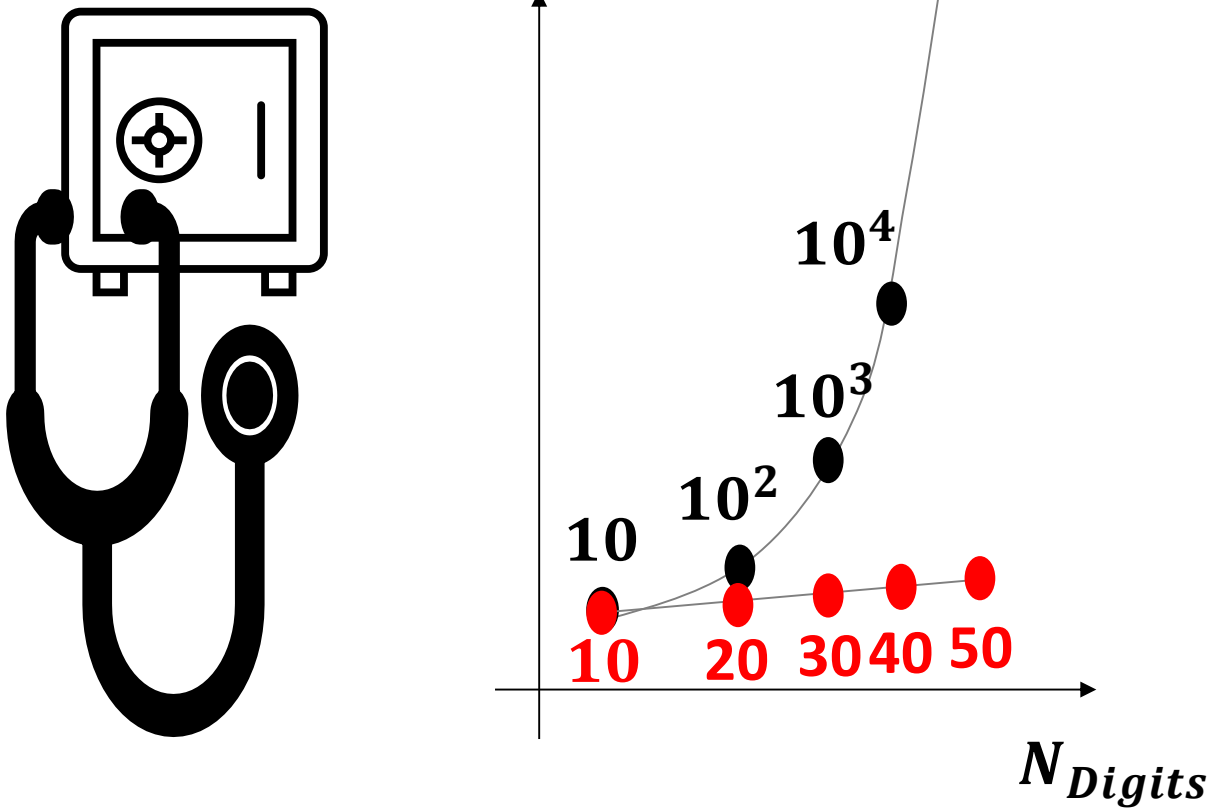

**Compromising Emanations**
Physical signals related to digital activity
Break the assumption of higher-level abstractions
Root cause of many attacks

**Software + Hardware**
Many abstractions, but in the end, built from physical components

R. J. Anderson, “Security Engineering - a Guide to Building Dependable Distributed Systems” (2. Ed.) (Wiley, 2008).

https://youtu.be/Kp9WtBRytxs

# The classic example (Side Channels Analogy)




**In theory…**
Number of combinations exponential with the number of digits

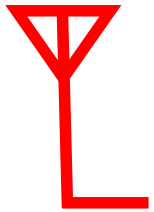
**Physical leakage…**
You hear a "click" noise when one digit is good

**In practice…**
Attack one digit at a time
Complexity is linear!
The combination can be recovered!

**Note:** good analogy for side channels against symmetric encryption (more later)

# Another example (TEMPEST)

**Secure device**



**Attacker**

**In theory…**

Secure computer in secure location

No way to eavesdrop confidential data

E.g., computer in embassy

**Physical leakage…**

The screen/cable has EM leaks

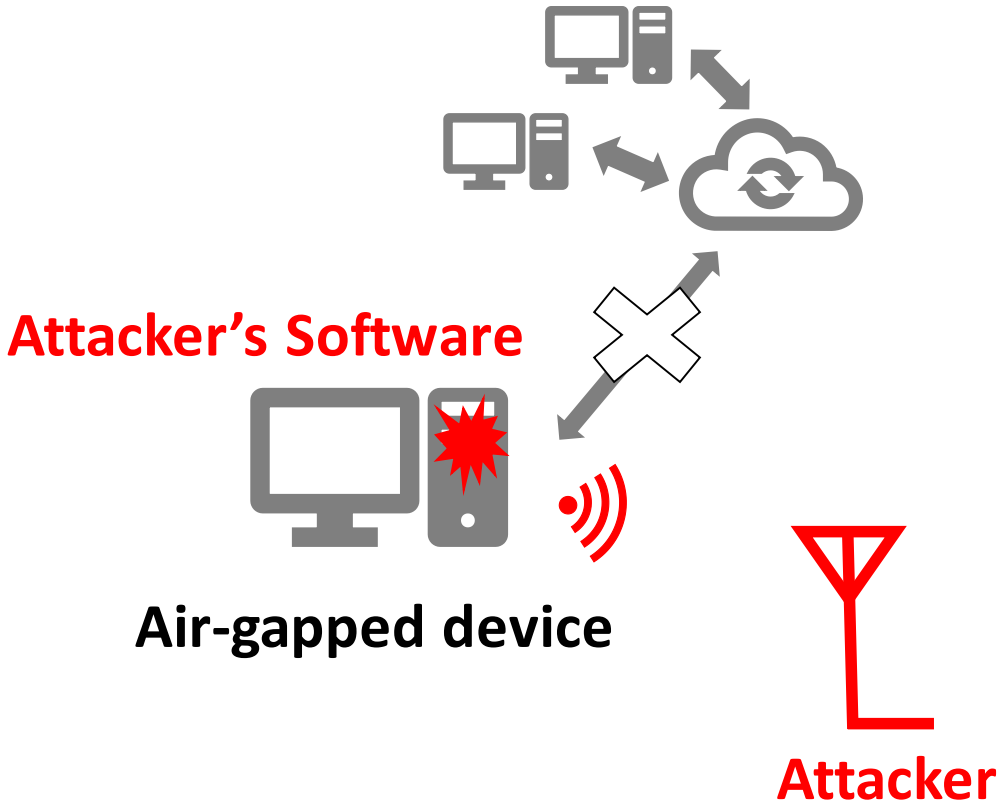
Video image propagates via radio

**In practice…**

Monitor EM leakage from outside

Eavesdropping is possible!

# Yet another example (Soft-TEMPEST)



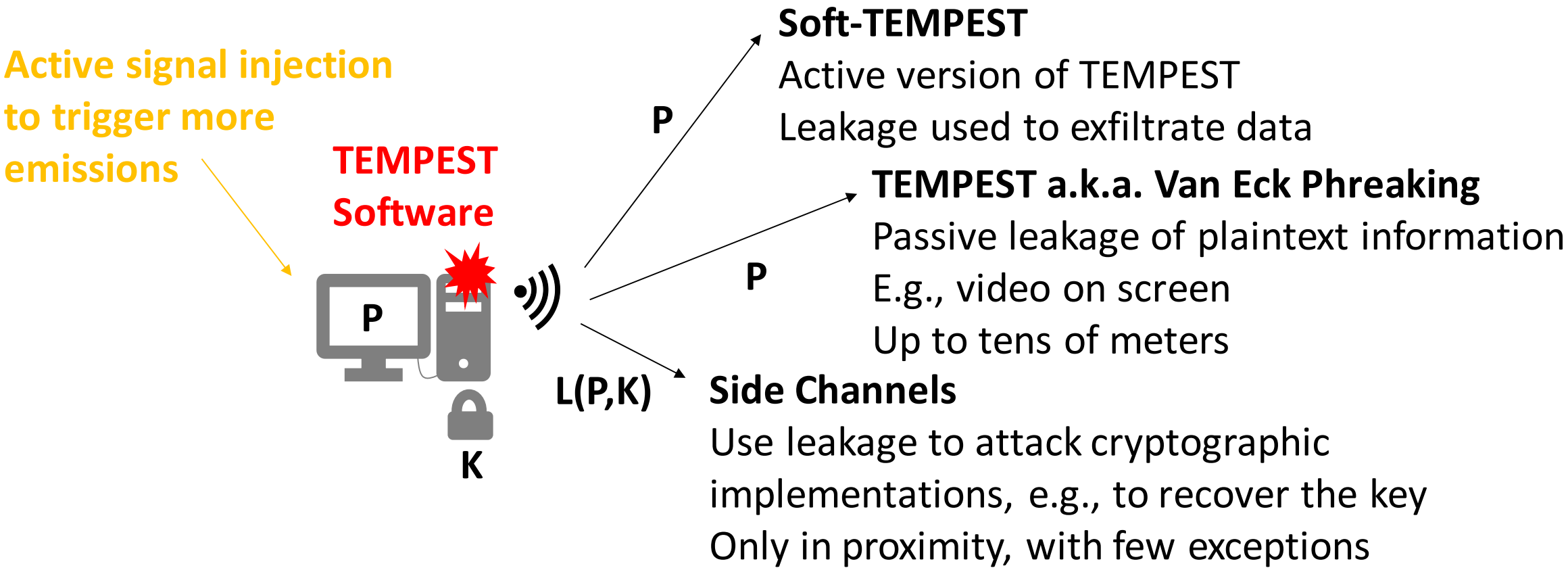

**In theory…**
Fully disconnected
Even an attacker able to execute code cannot exfiltrate data

**Physical leakage…**
Software execution triggers and modulates EM radiation

**In practice…**
Exfiltrate data via EM radiation
Communication is possible!

Let’s try to categorize all the main attacks before explaining them in detail

# An informal classification of the main attacks





"TEMPEST: A Signal Problem" (NSA, 1972).

W. van Eck, "Electromagnetic Radiation from Video Display Units: An Eavesdropping Risk?," Comput. Secur. 4, no. 4 (1985).

M. G. Kuhn and R. J. Anderson, "Soft Tempest: Hidden Data Transmission Using Electromagnetic Emanations," in Information Hiding (1998).

D. Agrawal et al., "The EM Side-Channel(s)," in CHES 2002.

C. Ramsay and J. Lohuis, TEMPEST Attacks against AES, 2017.

G. Camurati et. al., "Screaming Chanels: When Electromagnetic Side Channels Meet Radio Tranceivers", in CCS 2018)

A. T. Markettos, "Active Electromagnetic Attacks on Secure Hardware" (PhD Thesis, University of Cambridge, UK, 2011).

# An informal classification of the main attacks

Type of physical leakage
- Execution time
- Power
- Magnetic and Electromagnetic
- Optical
- Thermal
- Acoustic and vibrational
- Reflection of injected signals
- …

Type of attack
- Passively recover plaintext
- Actively exfiltrate data
- Attack cryptographic implementations

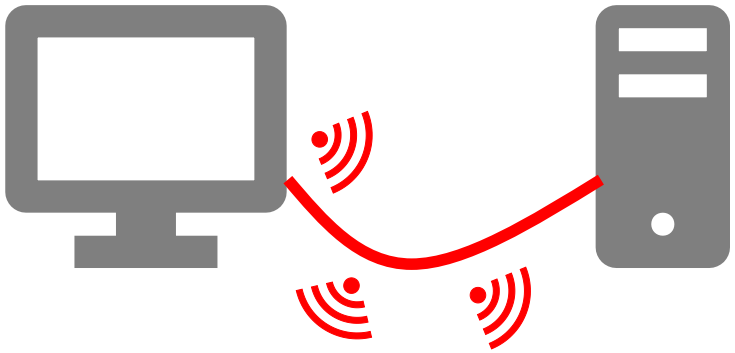
**Side note:** benign applications too!
- Monitoring
- Intrusion detection
- Malware detection
- Reverse engineering
- …

# TEMPEST

# TEMPEST a.k.a. Van Eck Phreaking

Codename in declassified NSA documents about countermeasures

One of the first results in public literature (followed by a very large literature)

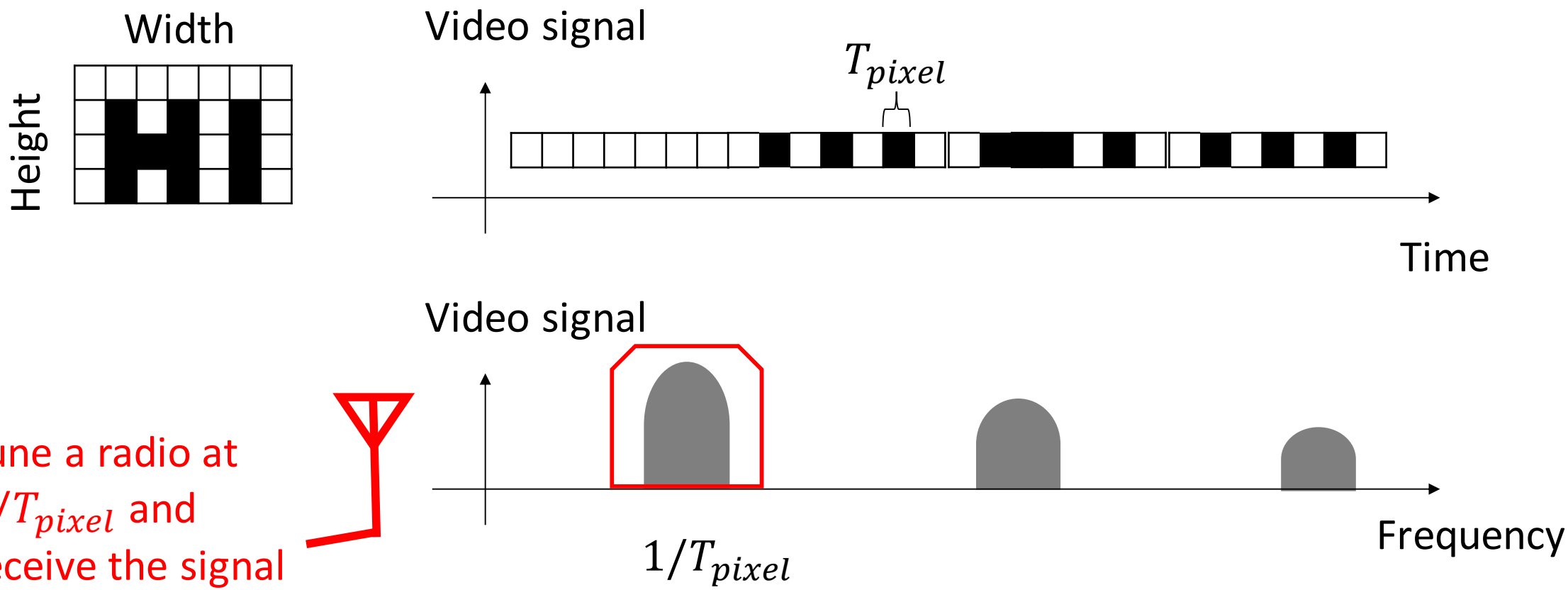


Video signal

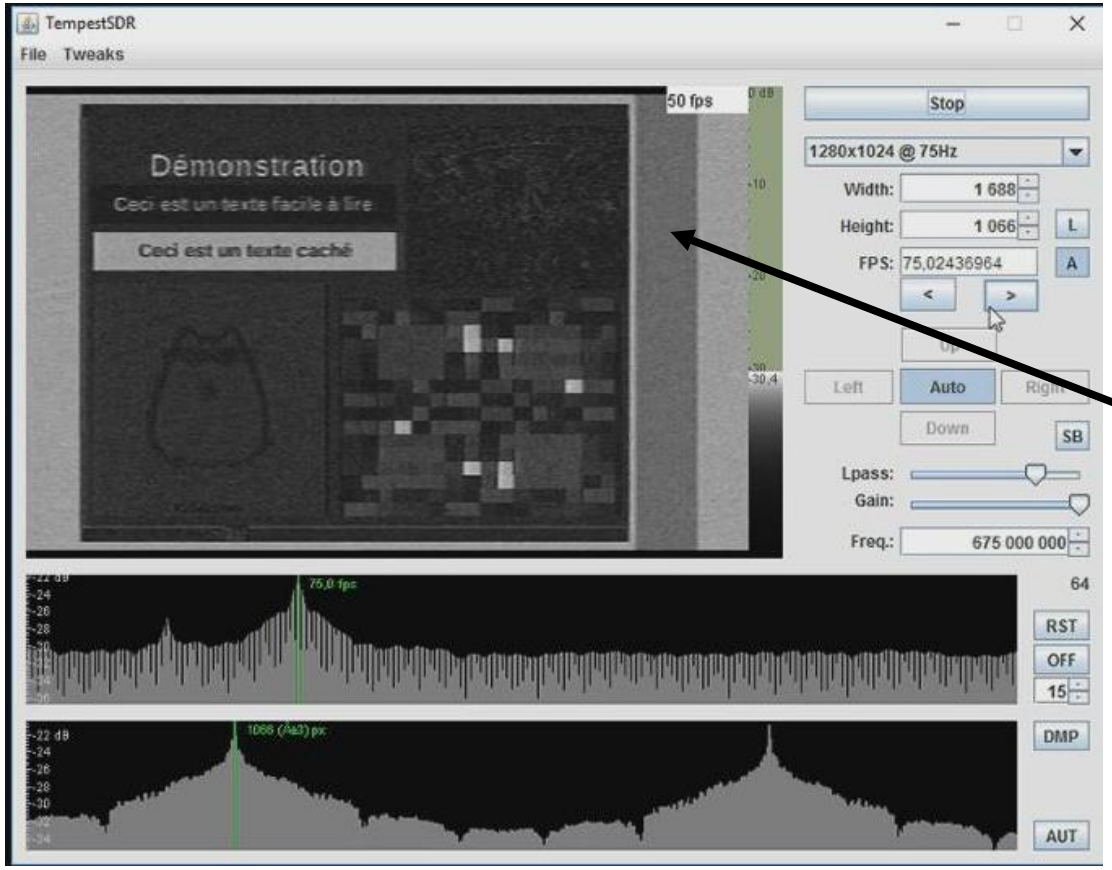
- Signal in wire/connector/etc. not well shielded
- Current in wires generates EM waves
- Modulated with the pixel values

"TEMPEST: A Signal Problem" (NSA, 1972).
W. van Eck, "Electromagnetic Radiation from Video Display Units: An Eavesdropping Risk?," Comput. Secur. 4, no. 4 (1985).
Raphael C.-W. Phan, "Review of Security Engineering: A Guide to Building Dependable Distributed Systems, 2nd Edition by Ross J. Anderson," Cryptologia 33, no. 1 (2009): 102–3.

# A bit more in detail (very simplified)



# TEMPEST in action





Screenshots from https://static.sstic.org/videos2018/SSTIC_2018-06-13_P05.mp4
Nice demo at minute 3.41 (In French but you can watch the video only)

# Many more ingenious ways…
# Look at reflections with a telescope!

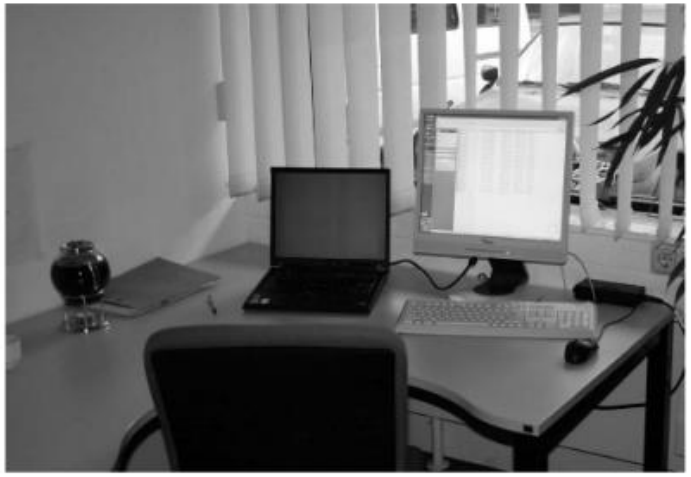


Figure 2. The basic setting: The monitor faces away from the window in an attempt to hide the screen's content.


Figure 1. Image taken with a macro lens from short distance; the distance between the eye and the monitor was reduced for demonstration. Readability is essentially limited by the camera resolution.


Figure 12. Reflections in a 0.5l plastic Coca-Cola bottle, taken from a distance of 5m. Because of the irregular surface, only parts of the text are readable.

Michael Backes, Markus Dürmuth, and Dominique Unruh. Compromising Reflections - or - How to Read LCD Monitors Around the Corner. In Proceedings of the IEEE Symposium on Security and Privacy (SSP '08), Oakland, CA, May 2008.

# Many more ingenious ways…
# Audio from the vibrations of a lightbulb

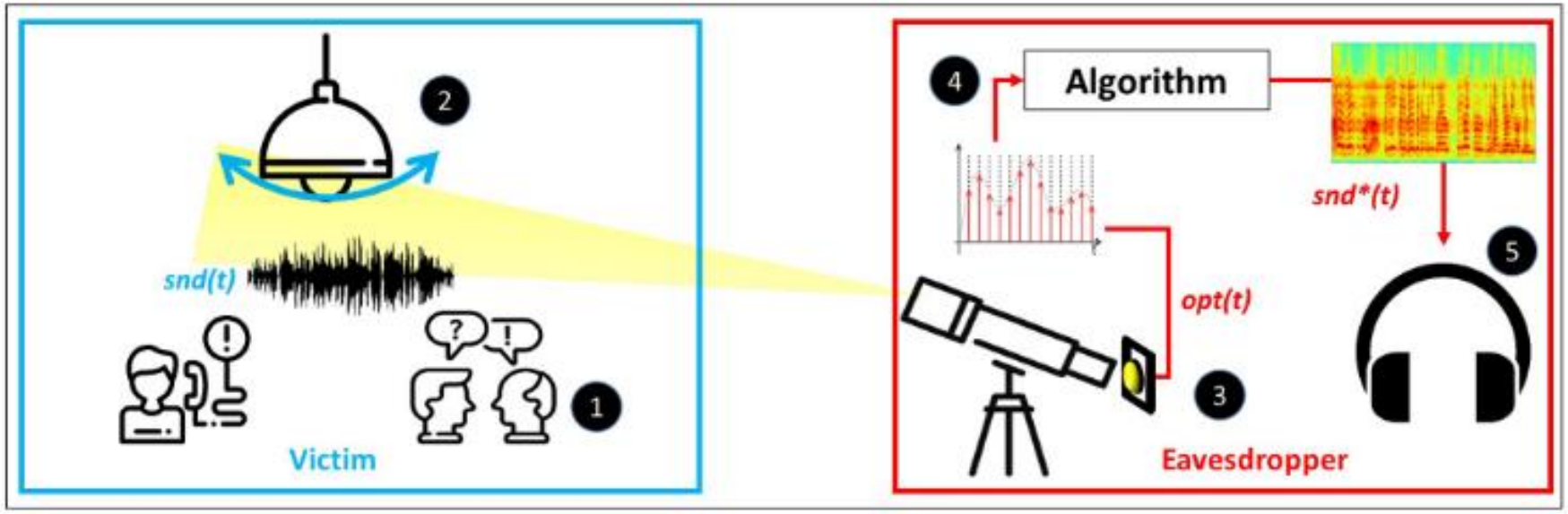



Fig. 2. Lamphone's threat model: The sound $snd(t)$ from the victim's room (1) creates fluctuations on the surface of the hanging bulb (the diaphragm) (2). The eavesdropper directs an electro-optical sensor (the transducer) at the hanging bulb via a telescope (3). The optical signal $opt(t)$ is sampled from the electro-optical sensor via an ADC (4) and processed, using Algorithm 1, to a recovered acoustic signal $snd^*(t)$ (5).

Ben Nassi et al., "Lamphone: Real-Time Passive Sound Recovery from Light Bulb Vibrations," IACR Cryptol. EPrint Arch., 2020

# Many more ingenious ways…
# Audio from the LEDs of a speaker

Figure 9: Experimental setup: the telescope and the four devices used in the experiments. A PDA100A2 electro-optical sensor is mounted on the telescope. The electro-optical sensor outputs voltage which is sampled via an ADC (NI-9234) and processed in LabVIEW.

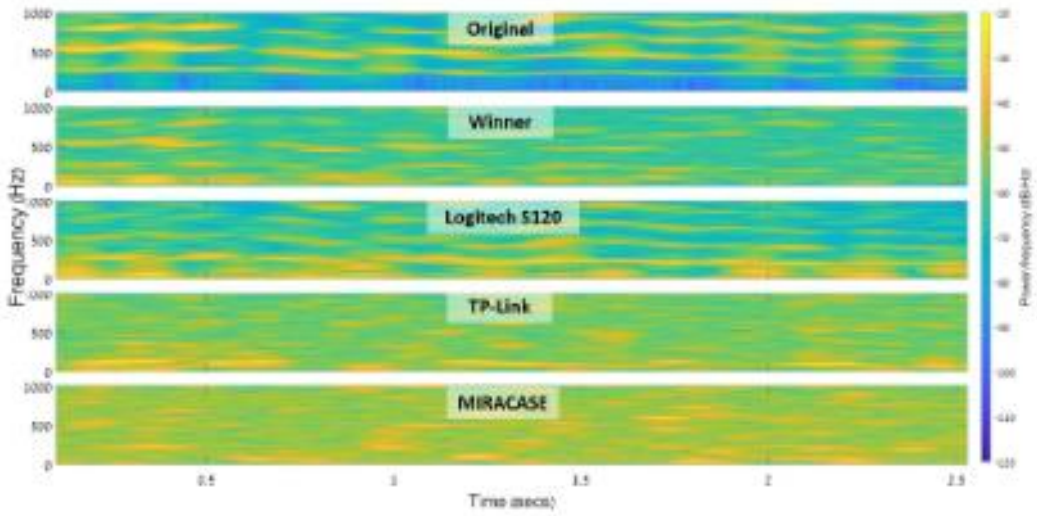



Figure 10: mabw0 sa1:"She had your dark suit in greasy wash water all year" recovered from various devices. The remaining spectrograms from the experiments listed in Table 2 can be seen in Figs. 16-20 in the Appendix.

Ben Nassi et al., "Glowworm Attack: Optical TEMPEST Sound Recovery via a Device's Power Indicator LED," ACM CCS 2021.

# Soft-TEMPEST

# Soft-TEMPEST


Video signal

- Signal in wire/connector/etc. not well shielded
- Current in wires generates EM waves
- Modulated with the pixel values
- Use to transmit data
- Or to add noise on the TEMPEST leakage
- Possible with many other sources of leakage (e.g. memory access)

M. G. Kuhn and R. J. Anderson, "Soft Tempest: Hidden Data Transmission Using Electromagnetic Emanations," in Information Hiding (1998).

# “TEMPEST for Elise”

https://www.youtube.com/watch?v=Zn_qki-084I

http://www.erikyyy.de/tempest/

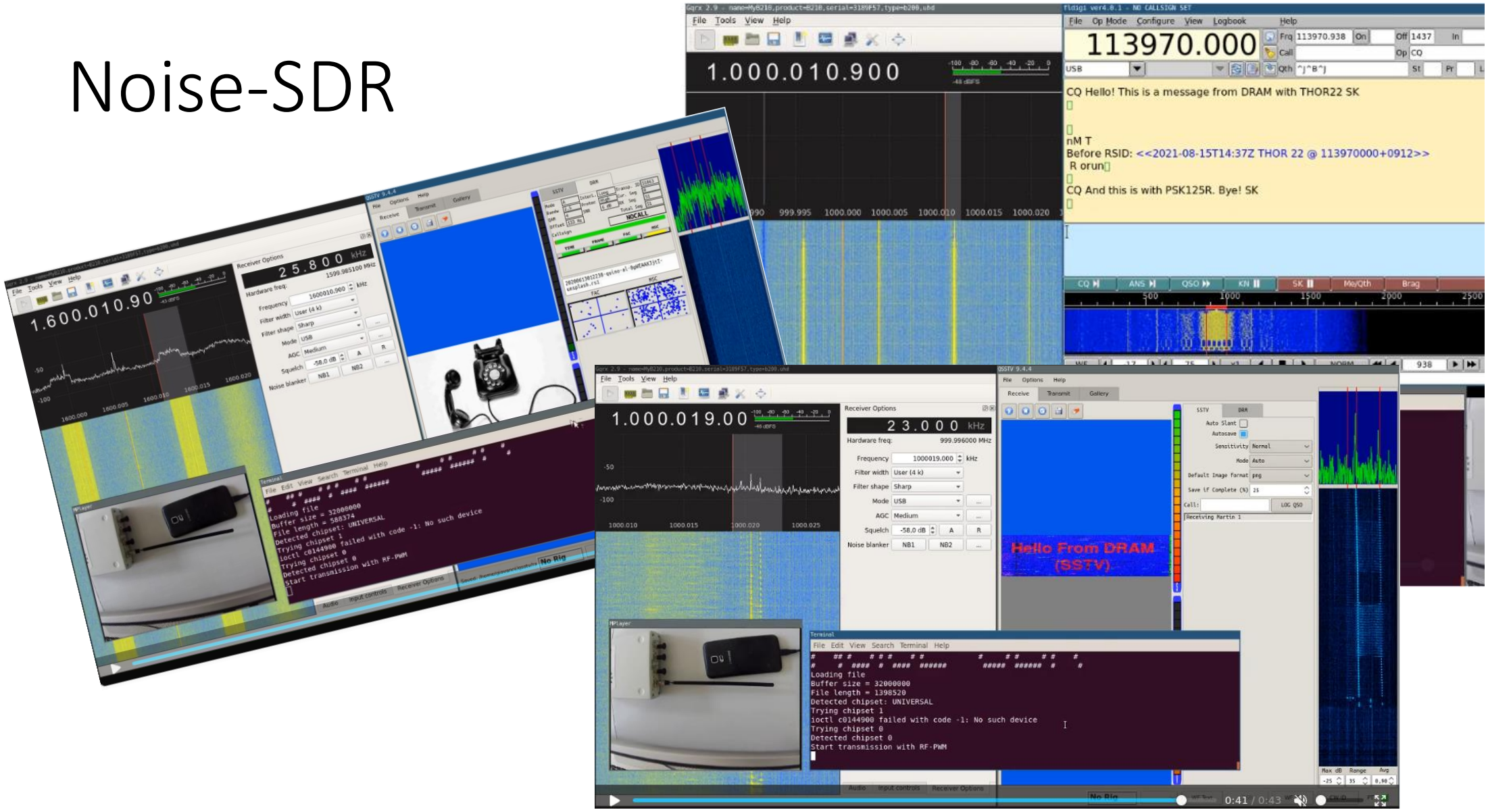
# And much more…

TABLE I
COMPARISON OF SOFTWARE-CONTROLLED ELECTROMAGNETIC AND MAGNETIC LEAKAGE

| Name | Type | Physical layer modulation | Protocol | Applications |
|---|---|---|---|---|
| ***Noise-SDR* (this paper)** | EM | Arbitrary (RF-PWM) | Arbitrary analog or digital protocols | Advanced software-defined radio transmissions |
| Soft-TEMPEST [1], [2] | EM | AM, FSK | Custom | Exfiltration (display to AM radio) |
| AirHopper [3], [4] | EM | FSK (A-FSK, DTMF) | Custom (raw or packet) | Exfiltration (computer screen to smartphone) |
| USBee [5] | EM | FSK (B-FSK) | Custom | Exfiltration (USB bus to SDR) |
| GSMem [6] | EM | OOK (B-ASK) | Custom | Exfiltration (computer to mobile phone) |
| BitJabber [7] | EM | OOK, FSK (M-FSK) | Custom | Exfiltration (computer to SDR) |
| EMLora [8] | EM | Approximated CSS | Custom Lora-like | Exfiltration (computer to SDR) |
| AIR-FI [9] | EM | OOK | Custom | Exfiltration (computer to SDR or WiFi cards that expose physical layer radio measurements) |
| MAGNETO [10] | M | OOK, FSK (B-FSK) | Custom | Exfiltration (computer to smartphone) |
| ODINI [11] | M | OOK (ASK, OOK-OFDM using multiple cores), FSK | Custom (including FEC) | Exfiltration (computer to magnetic bug) |
| Matyunin et al. [12] | M | OOK, FSK ('period based') | Custom | Exfiltration (laptop to smartphone) |

G. Camurati and A. Francillon, "Noise-SDR: Arbitrary Modulation of Electromagnetic Noise from Unprivileged Software and Its Impact on Emission Security" to appear at IEEE S&P 2022.

# Noise-SDR




G. Camurati and A. Francillon, "Noise-SDR: Arbitrary Modulation of Electromagnetic Noise from Unprivileged Software and Its Impact on Emission Security" to appear at IEEE S&P 2022.

# Some References


"TEMPEST: A Signal Problem" (NSA, 1972).
W. van Eck, "Electromagnetic Radiation from Video Display Units: An Eavesdropping Risk?," Comput. Secur. 4, no. 4 (1985).

M. G. Kuhn and R. J. Anderson, "Soft Tempest: Hidden Data Transmission Using Electromagnetic Emanations," in Information Hiding (1998).

Michael Backes, Markus Dürmuth, and Dominique Unruh. Compromising Reflections - or - How to Read LCD Monitors Around the Corner. In Proceedings of the IEEE Symposium on Security and Privacy (SSP '08), Oakland, CA, May 2008.

Ben Nassi et al., "Lamphone: Real-Time Passive Sound Recovery from Light Bulb Vibrations," IACR Cryptol. EPrint Arch., 2020

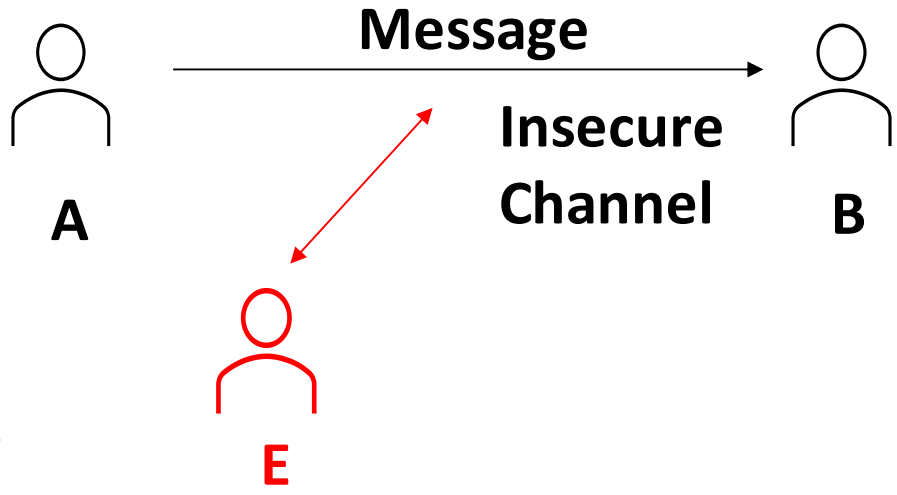
G. Camurati and A. Francillon, "Noise-SDR: Arbitrary Modulation of Electromagnetic Noise from Unprivileged Software and Its Impact on Emission Security" to appear at IEEE S&P 2022.


http://www.erikyyy.de/tempest/
https://github.com/fulldecent/system-bus-radio
https://github.com/martinmarinov/TempestSDR
https://github.com/git-artes/gr-tempest
https://github.com/eurecom-s3/noise-sdr

Try it yourself

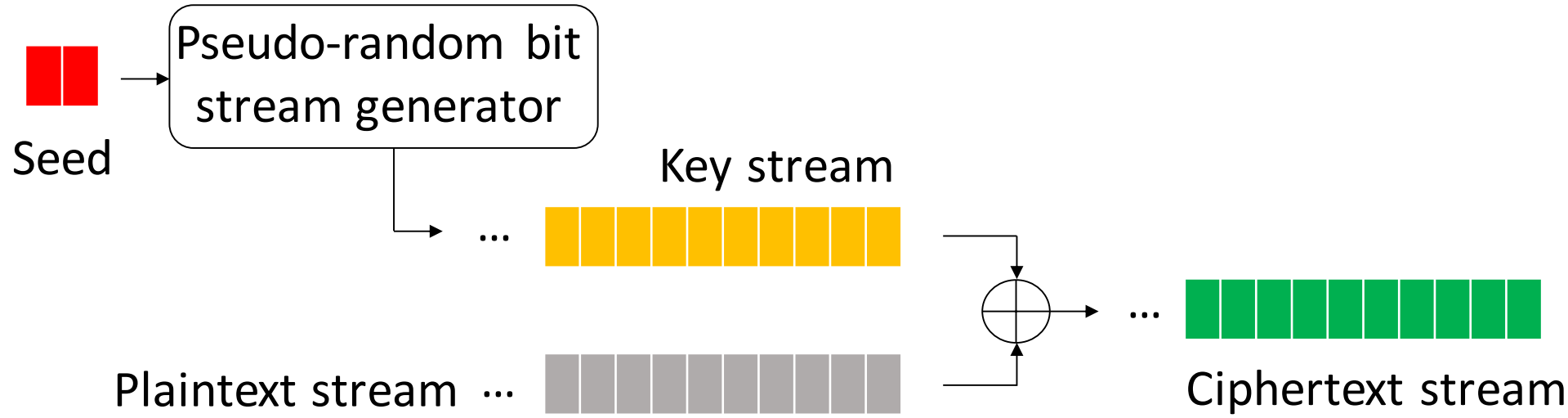
# Side Channels

# Background on crypto…




| | Confidentiality | Authentication | |
|---|---|---|---|
| **Symmetric Crypto** | 1-Symmetric key encryption | 2-Symmetric key auth. | |
| **Asymmetric Crypto** | 3-Public key encryption | 4-Digital signatures | |
| | | | |

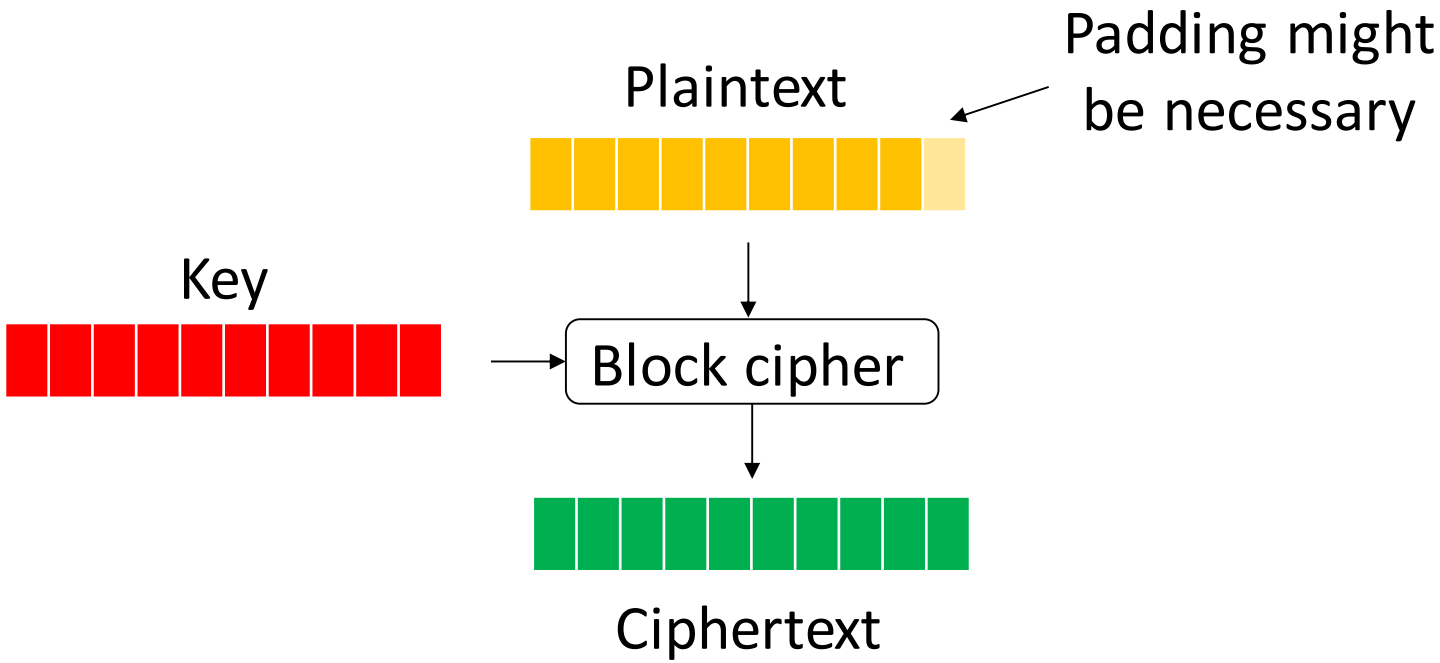
# 1 – Symmetric crypto for confidentiality




**Stream cipher**

- Process a message bit by bit (byte by byte)
- E.g., RC4

# 1 – Symmetric crypto for confidentiality

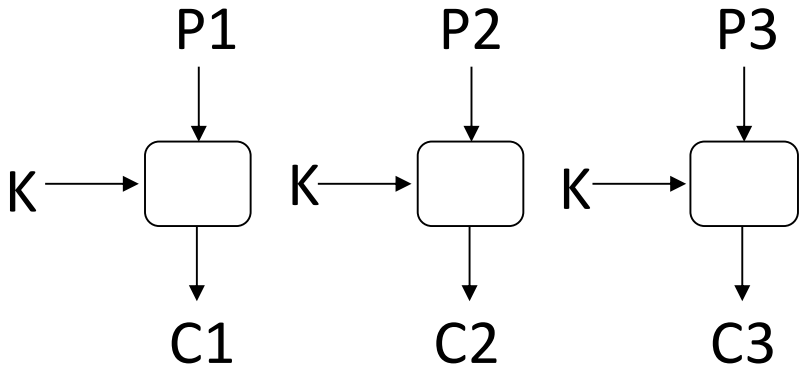



**Block cipher**

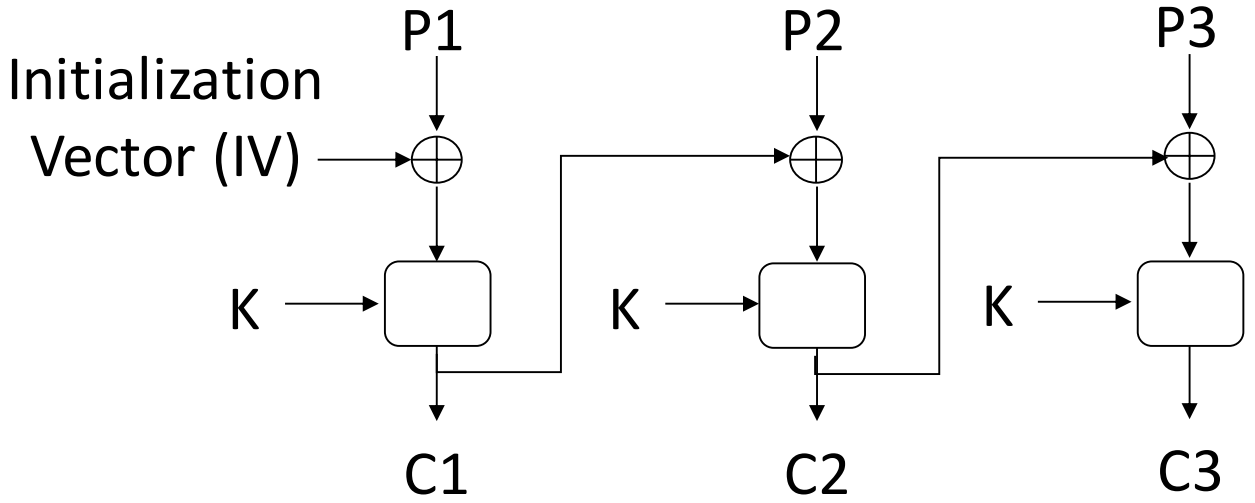
- Process a message block by block
- E.g., DES, AES

Many modes to "concatenate" blocks

# 1 – Symmetric crypto for confidentiality




E.g., Electronic Codebook (ECB)
(Insecure!)

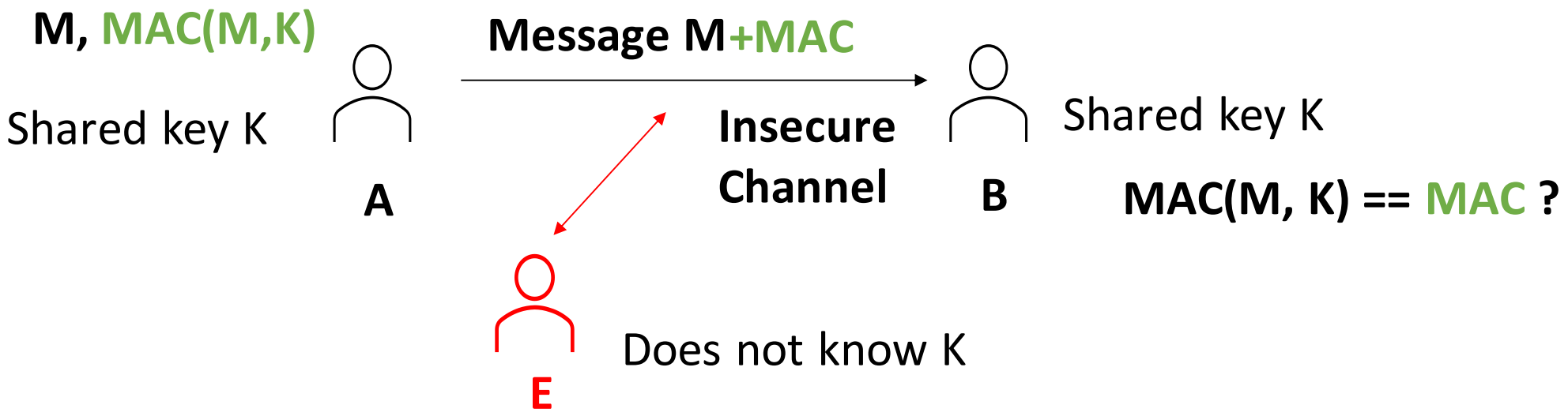



E.g., Cipher Block Chaining (CBC)

**Block cipher**

- Process a message block by block (ECB, CBC, PCBC, CFB, OFB, CTR)
- E.g., DES, AES

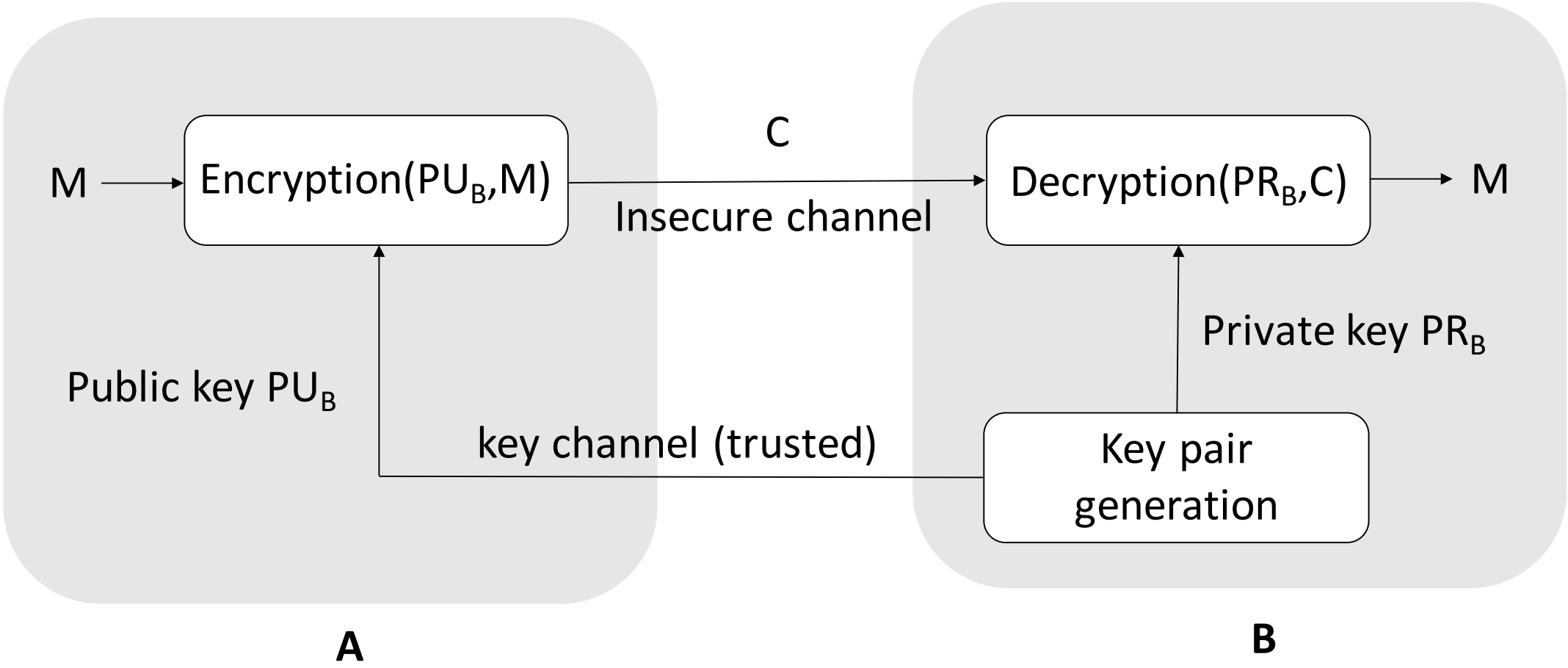
# 2 – Symmetric crypto for authentication



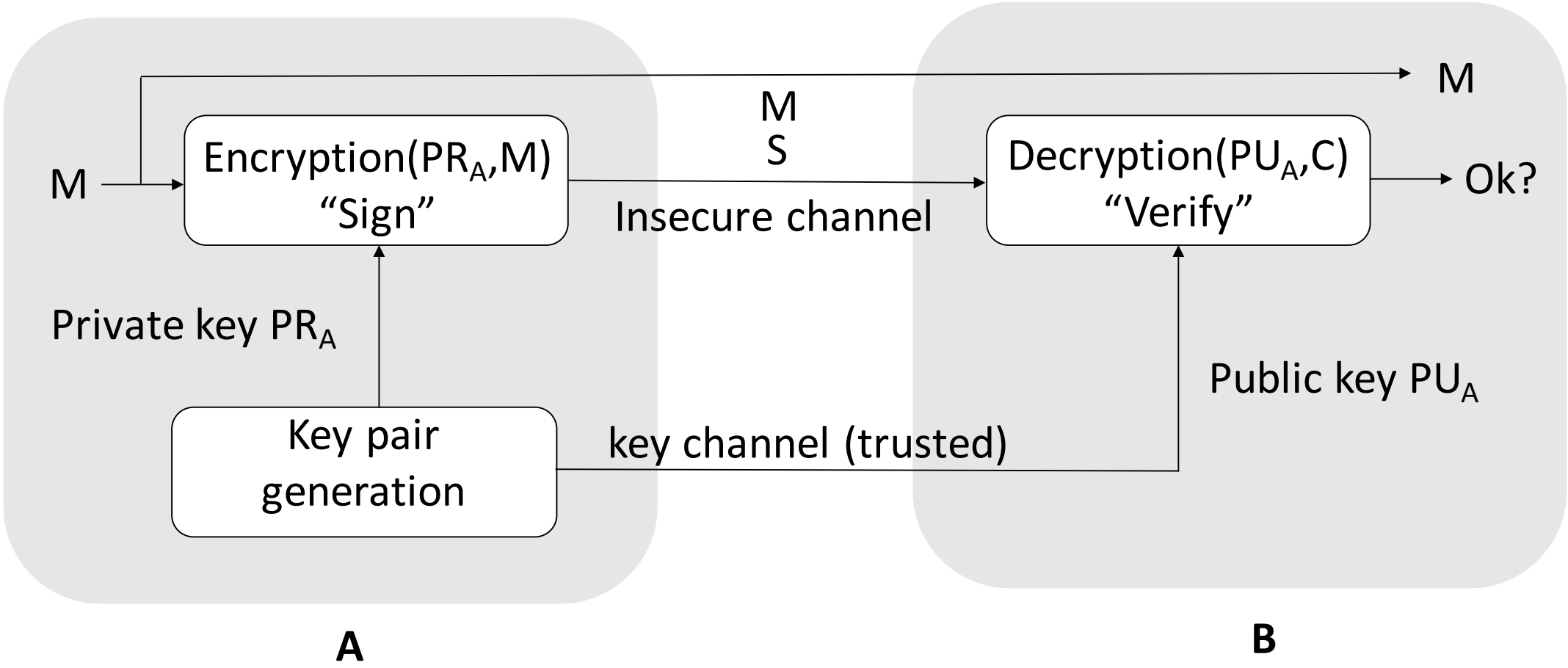

**Message Authentication Code (MAC)**

- MAC = MACFunction(K,M)
- Hard to forge

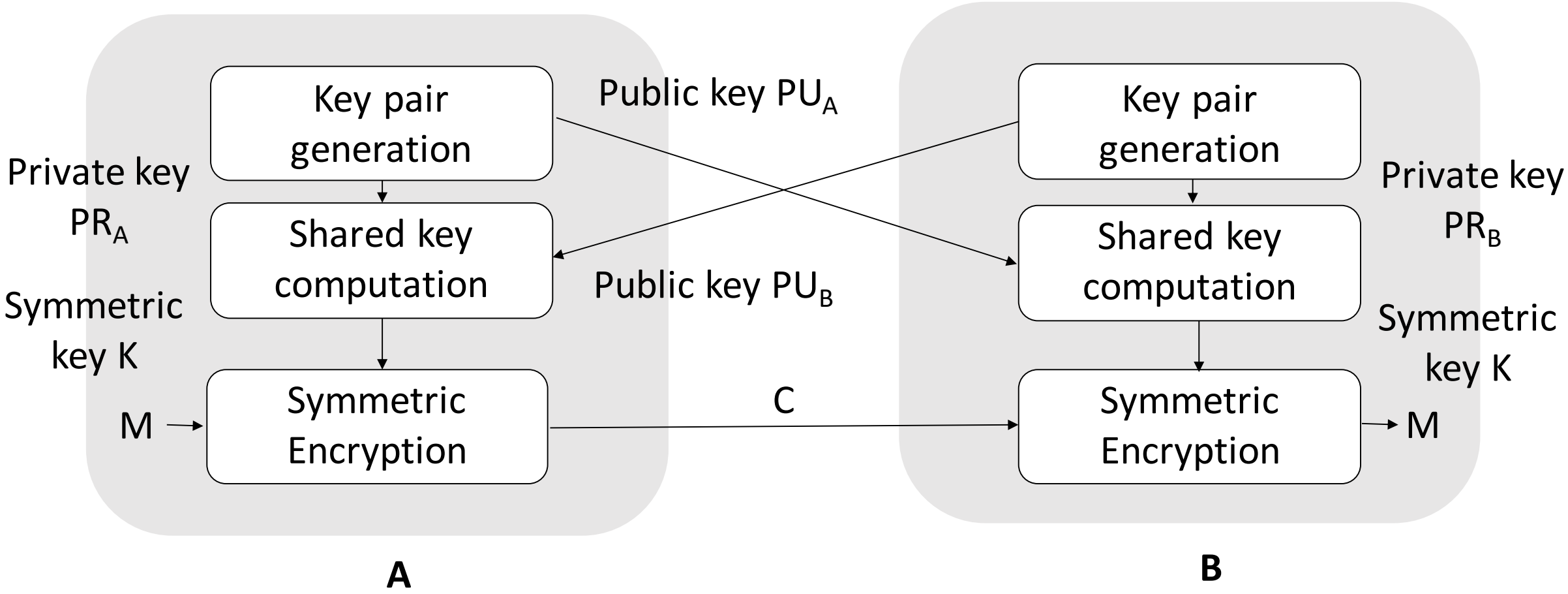
# 3 – Asymmetric crypto for confidentiality



# 4 – Asymmetric crypto for authentication



# Combine the best of the two: asymmetric key exchange + symmetric encryption



# Security of cryptographic algorithms

**Its all about models**

- We **model** system and possible attackers
- Security properties are valid under certain **assumptions**
  - Example:
    - Model: Chosen Plaintext Attack (CPA)
    - Property: Encryption Scheme is CPA-secure

**Problem**

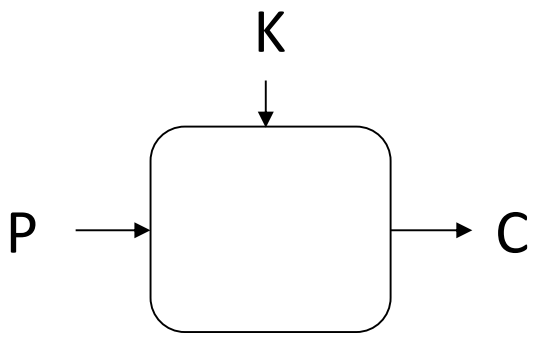
- Until recently models did not consider the implementation…
  - Side channel leakage, fault injection, probing, …
  - The scheme is still secure under the assumptions… but the assumptions are now broken by the implementation issues…

**Note:** Fundamental problem, not mere implementation error

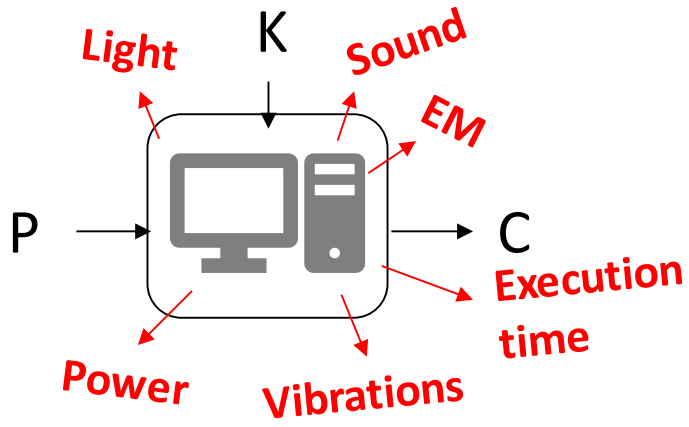
**Solution**

- Model these problems, design countermeasures
- New properties: e.g., secure against side channel attack of type xx

# Security of cryptographic algorithms




Classic cryptanalysis




Side channel analysis

- Understanding of physical principles
- Measurements
- Mathematical modeling

We are ready to get into side channels with concrete examples

# Outline

**Timing**
Measure execution time

- **Classic timing attack against RSA** **(Simple/differential timing analysis)**
- **Remote attacks are possible**
- **Modern example of remote attack on cryptocurrencies**
- Cache side channels and microarchitectural attacks (In a dedicated lesson)

**“Power”**
Measure some physical quantity influenced by execution, e.g., power, EM emissions, …

- **Simple Power Analysis (SPA)**
- **Differential Power Analysis (DPA)**
- **Correlation Power Analysis (CPA)**
- **Examples of other leakage types**

# Disclaimer

**This is a very wide and complex topic**

- Understanding of physical phenomena
- Complex statistical analysis
  - Statistics and probability are hard
  - Intuition is easily fooled
  - Easy to get it totally wrong
- Wide range of goals and techniques

**We will only scratch the surface**

- Only intuitive results
- Trying to give an intuition of more advanced abstraction and formalization
- Simplifications are made to keep things simple (trying to still be rigorous)
- This topic would require a full dedicated course to be covered in depth

# A reminder on “textbook” RSA

**Key generation:**

1. Chose numbers **p**, **q** such that p and q are prime and p != q
2. Compute **n = pq**
3. Compute **ϕ(n) = (p-1)(q-1)**
4. Chose **e** such that e and ϕ(n) are relative prime and 1 < e < ϕ(n)
5. Compute **d** as such that de mod ϕ(n) = 1
6. Public key **PU = {e, n}**
7. Private key **PR = {d, n}**

**Encryption**

- Plaintext **m** < n
- Ciphertext **$C = m^e \bmod n$**

**Decryption**

- Ciphertext **C**
- Plaintext **$m = C^d \bmod n$**

**Signature**

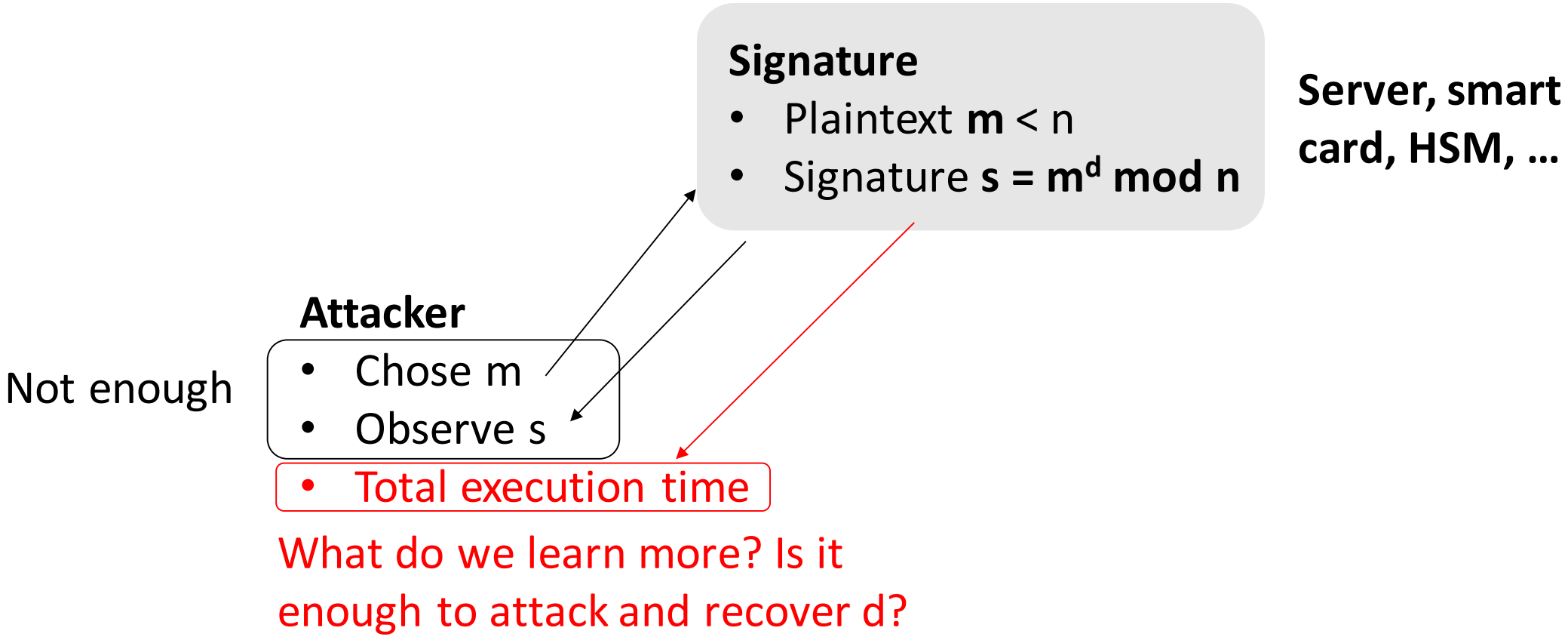
- Plaintext **m** < n
- Signature **$s = m^d \bmod n$**

# Cryptanalysis

**The security of RSA is based on two "hard problems"**

- The RSA problem, i.e., computing the $e^{th}$ root of m modulo n to find m from $C = m^e \bmod n$
- Factoring large numbers into smaller primes

# Cryptanalysis (Chosen/Known Plaintext)
# + Timing side channel analysis



# What can we learn from the execution time? We need to look at a specific implementation

How do we implement exponentiation? -> Square and Multiply

$$s = m^d$$

$$d = d_0 2^{w-1} + d_1 2^{w-2} \ldots + d_{w-1} 2^0$$

$$s = m^{d_0 2^{w-1} + d_1 2^{w-2} \ldots + d_{w-1} 2^0}$$

$$s = m^{d_0 2^{w-1}} m^{d_1 2^{w-2}} \ldots m^{d_{w-1} 2^0}$$

$$s = [[[m^{d_0}]^2 m^{d_1}]^2 \ldots] m^{d_{w-1}}$$

Square

If $d_1$==1 multiply

Square

…

**Note:**

Square and multiply are computed modulo n using Montgomery multiplication/square

# What can we learn from the execution time? We need to look at a specific implementation

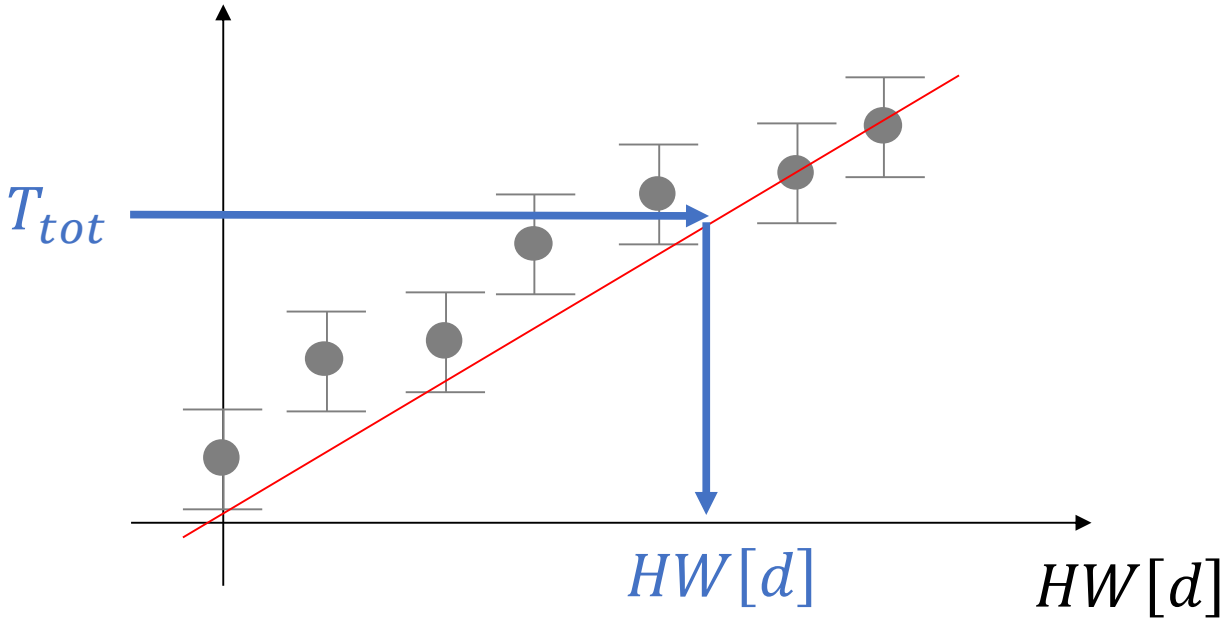
```
squareAndMultiply(m, d, n):
    """ Return s = m^d mod n """
    w = bitlength(d)
    x = m ; Assume d[0] == 1 (MSB)
    for i in 1 to w-1:
        x = x^2 mod n
        if bit i of d == 1:
            x = xm mod n
    return x
```

Problem 1:

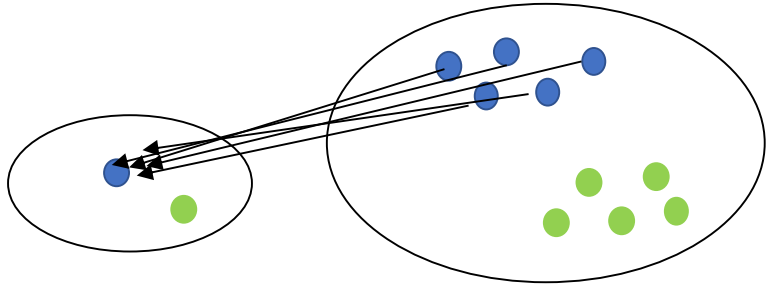
- Key dependent branching
- Execution time depends on the key d
  - If bit i of d is 0 -> faster (0)
  - If bit i of d is 1 -> slower ($T_{modmult}$)

# Key dependent branching still not very useful

$T_{tot} = HW[d]T_{modmult}$




$HammingWeight[d]$ = number of bits at 1




Given the total execution time we could recover the HW of the key…

But a lot of keys have the same HW… (e.g., 100, 010, 001)

**Note:** Still bad for power leakage that can observe bits one by one (more later)

# Let's dig deeper

```
squareAndMultiply(m, d, n):
    """ Return s = m^d mod n """
    w = bitlength(d)
    x = m ; Assume d[0] == 1 (MSB)
    for i in 1 to w-1:
        x = ModSquare(x, x, n)
        if bit i of d == 1:
            x = ModMultiply(m, x, n)
    return x
```

```
Montgomery(m, x, n):
    … some computations …
    if( some condition )
        return u-n # Conditional reduce
    else return u
```

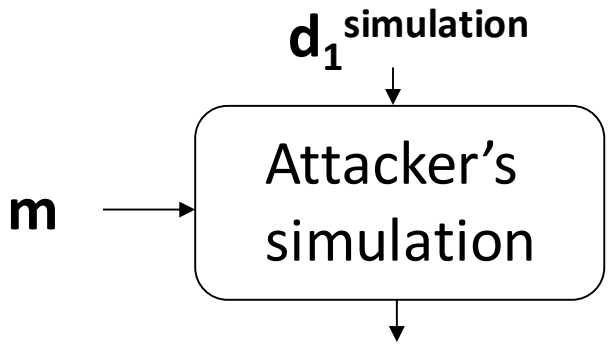
Problem 2:

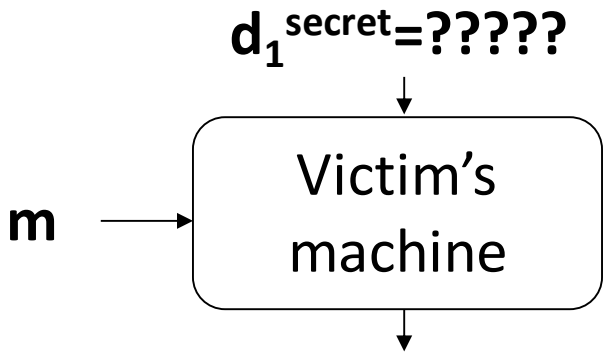
- Montgomery used for modular multiplications because it is more efficient
- Montgomery execution time $T_{mont}$ depends on the plaintext m
  - There is a reduction step done only if necessary

# Let's focus on one iteration and ModSquare

```
                x = m
i = 1           x = ModSquare(x,n)
                if(d1 == 1):
                  x = ModMult(x, m, n)
i = 2           x = ModSquare(x,n)
```

- This ModSquare can be fast (no reduction) or slow (reduction)
- This difference depends on the value of x
- Which at its turn depends on both m and $d_1$
  - We easily know how to compute the condition
  - I.e., we have a "model" of the time leakage
  - I.e., we can predict if "slow" or "fast" given m and $d_1$



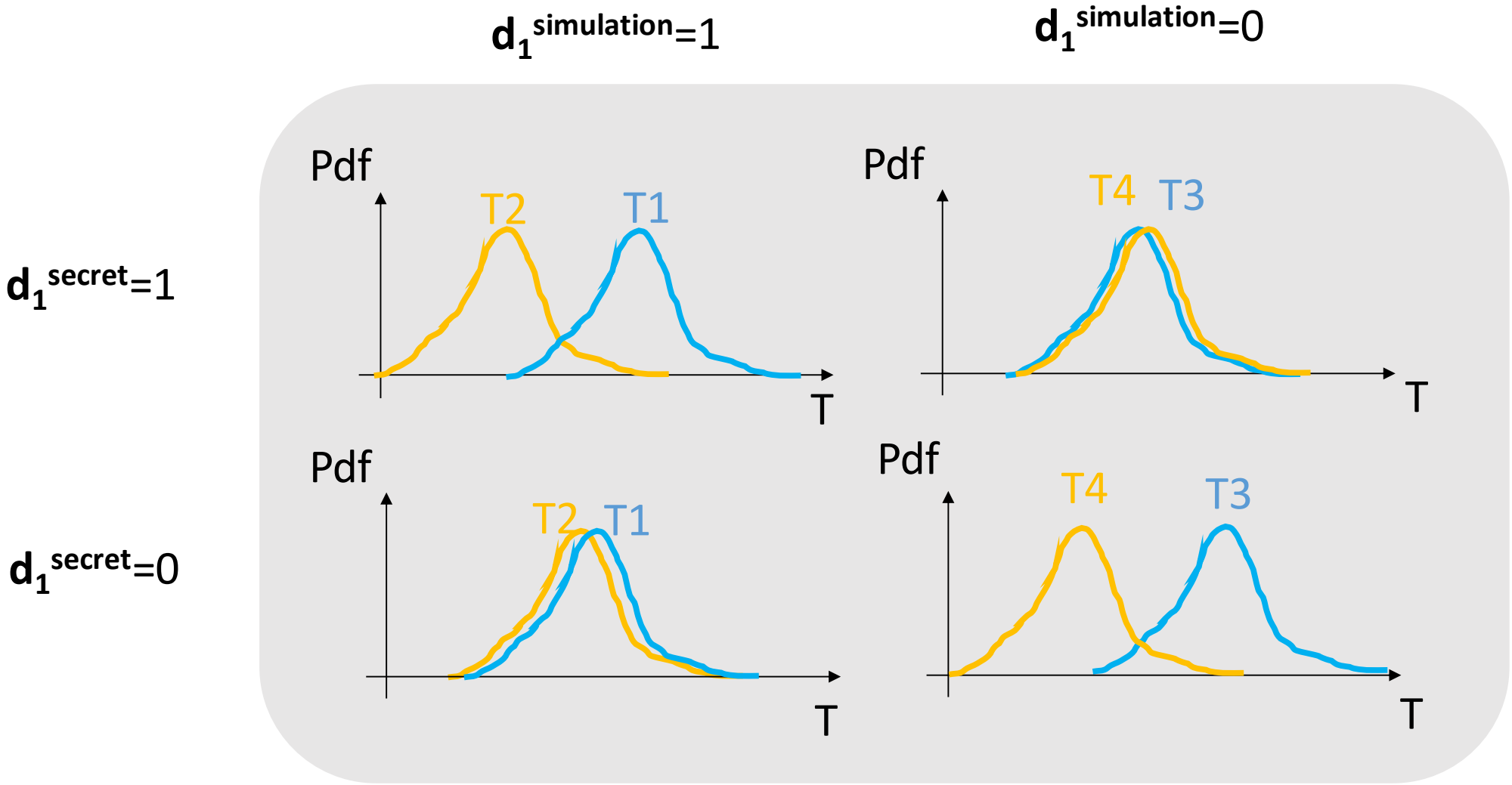

1. Generate 4 sets of random messages
- **M1**: m s.t. simulation is **slow** if $d_1^{simulation}=1$
- **M2**: m s.t. simulation if **fast** if $d_1^{simulation}=1$
- **M3**: m s.t. simulation is **slow** if $d_1^{simulation}=0$
- **M4**: m s.t. simulation is **fast** if $d_1^{simulation}=0$

2. Measure their real total execution time
- **T1**: T(m) for m in M1 (**supposedly slower**)
- **T2**: T(m) for m in M2 (**supposedly faster**)
- **T3:** T(m) for m in M3 (**supposedly slower**)
- **T4:** T(m) for m in M4 (**supposedly faster**)

3. Observation
- If $d_1^{secret}=1$: **T2 is really fast** and **T1 is really slow**, T4 is not faster than T3
- If $d_1^{secret}=0$: **T4 is really fast** and **T3 is really slow**, T2 is not faster than T1

4. Conclusion
- If avg(T1) – avg(T2) > avg(T3) – avg(T4) then $d_1^{secret}=1$ else $d_1^{secret}=0$

# Graphically



# Now that $d_1$ is known, attack the next iteration (and so on and so forth)

```
                x = m
i = 1           x = ModSquare(x,n)
                if(d1 == 1):
                  x = ModMult(x, m, n)
i = 2           x = ModSquare(x,n)
                if(d2 == 1):
                  x = ModMult(x, m, n)
i = 3           x = ModSquare(x,n)
```

- This ModSquare can be fast (no reduction) or slow (reduction)
- This difference depends on the value of x
- Which at its turn depends on both m and $d_1$ and $d_2$

# Caveat

i = w-1

x = ModSquare(x,n)
if($d_{w-1}$ == 1):
  x = ModMult(x, m, n)

return x

- The last condition (on $d_{w-1}$) is not followed by a square…
- We cannot find the last bit of the key
- Only one bit, we can bruteforce

*Actually, the current guess depends on the correctness of the previous guesses

# Summing up: the complete attack algorithm

**Input:** $d_0, \ldots, d_{i-1}$ , $N_{message}$ different known messages $m_j$

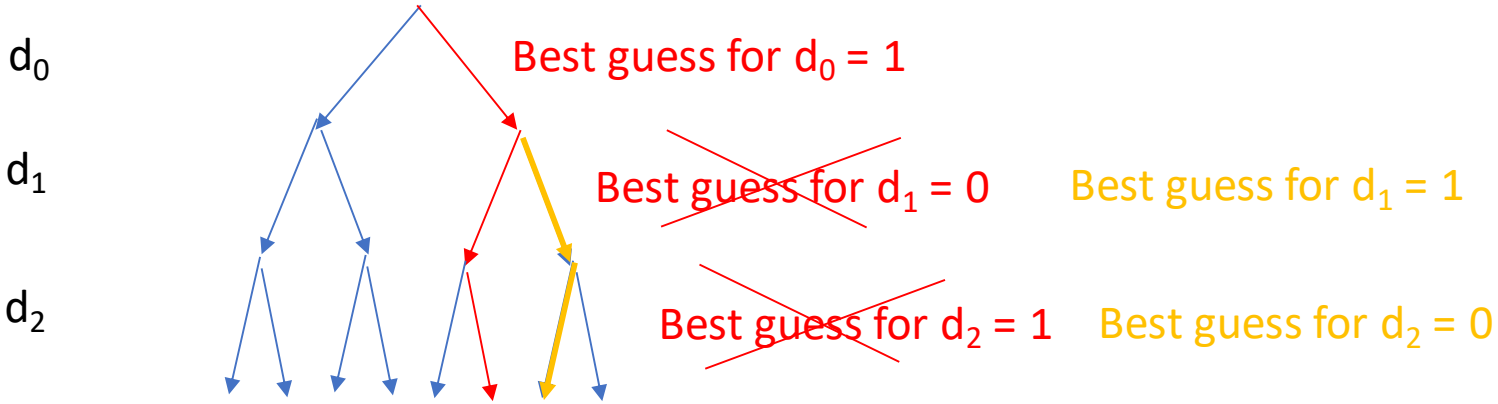
**Output:** best guess of $d_i$

```
M1 = [], M2 = [], M3 = [], M4 = []
for j in range(N_message):
    T_j = measure(RSAvictim(d, m_j)) // d is unknown
    // guess d_i = 1
    Simulate RSA(d_0, ..., d_{i-1},1, m_j) until iteration i
    if (ModSquare at iteration i requires reduction):
        TM2.append(T_j)
    else:
        TM1.append(T_j)
    // guess d_i = 0
    Simulate RSA(d_0, ..., d_{i-1},0, m_j) until iteration i
    if (ModSquare at iteration i requires reduction)
        TM4.append(T_j)
    else:
        TM3.append(T_j)
If avg(TM2) – avg(TM1) > avg(TM4) – avg(TM3) return 1
else return 0
```

**Observations:**

- Divide the problem: focus on one bit at a time, only **two guesses***
- We can **measure** if the victim was fast or slow (very small difference buried in noise)
- Given a **guess**, we can **predict** if the victim was fast or slow using a **model** (e.g., simulation)
- After **many measurements**, we can check **for which guess** the **model best matches the measurement** using a **statistical test**

# Extend and prune + Error correction




If your guesses are correct, your estimate of the total execution time gets better and better each time you guess a new bit

If this is not true, then maybe one of your previous guesses was wrong, invalidating also the next ones…

-> you can try to backtrack and restart

# Exercise

**Given:**

- File with message, signatures and their timings
- Script with the attack presented in the slides
  - Attack the square step
  - No error correction

**Goals:**

- Better understanding of the slides
- Compare attacking the square (the attack of the script) to attacking the multiply (see the paper)

# Some references

Paul C. Kocher, “Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems,” in Advances in Cryptology - CRYPTO ’96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings, ed. Neal Koblitz, vol. 1109, Lecture Notes in Computer Science (Springer, 1996), 104–13, https://doi.org/10.1007/3-540-68697-5_9.

Jean-François Dhem et al., “A Practical Implementation of the Timing Attack,” in Smart Card Research and Applications, ed. Jean-Jacques Quisquater and Bruce Schneier, Lecture Notes in Computer Science (Berlin, Heidelberg: Springer, 2000), 167–82, https://doi.org/10.1007/10721064_15.

We presented one of the attacks explained in this paper

In the paper there are more details, for example, on the fact that prediction i depends on the correctness of predictions 0 to i-1, and an analogy with soft-decoding vs. hard-decoding

Timing measurements generally require local access…

Or are remote timing attacks possible?

# Remote attacks (i.e., over the network)

David Brumley and Dan Boneh, “Remote Timing Attacks Are Practical,” in Proceedings of the 12th Conference on USENIX Security Symposium - Volume 12, SSYM’03 (USA: USENIX Association, 2003), 1.

The classic example (Recommended reading)

Florian Tramer, Dan Boneh, and Kenny Paterson, “Remote Side-Channel Attacks on Anonymous Transactions,” 2020, 2739–56, https://www.usenix.org/conference/usenixsecurity20/presentation/tramer.

A recent example, not only crypto (Recommended reading/video of presentation)

# Countermeasures?

# Countermeasures

**Constant time:**

- Relatively easy for specific cases
  - E.g., modular multiplication without conditional reduction
- Generic protection is hard
  - Identify and eliminate all dependencies of time with plaintext and key
  - Can have performance issues

**What if we artificially add noise?**

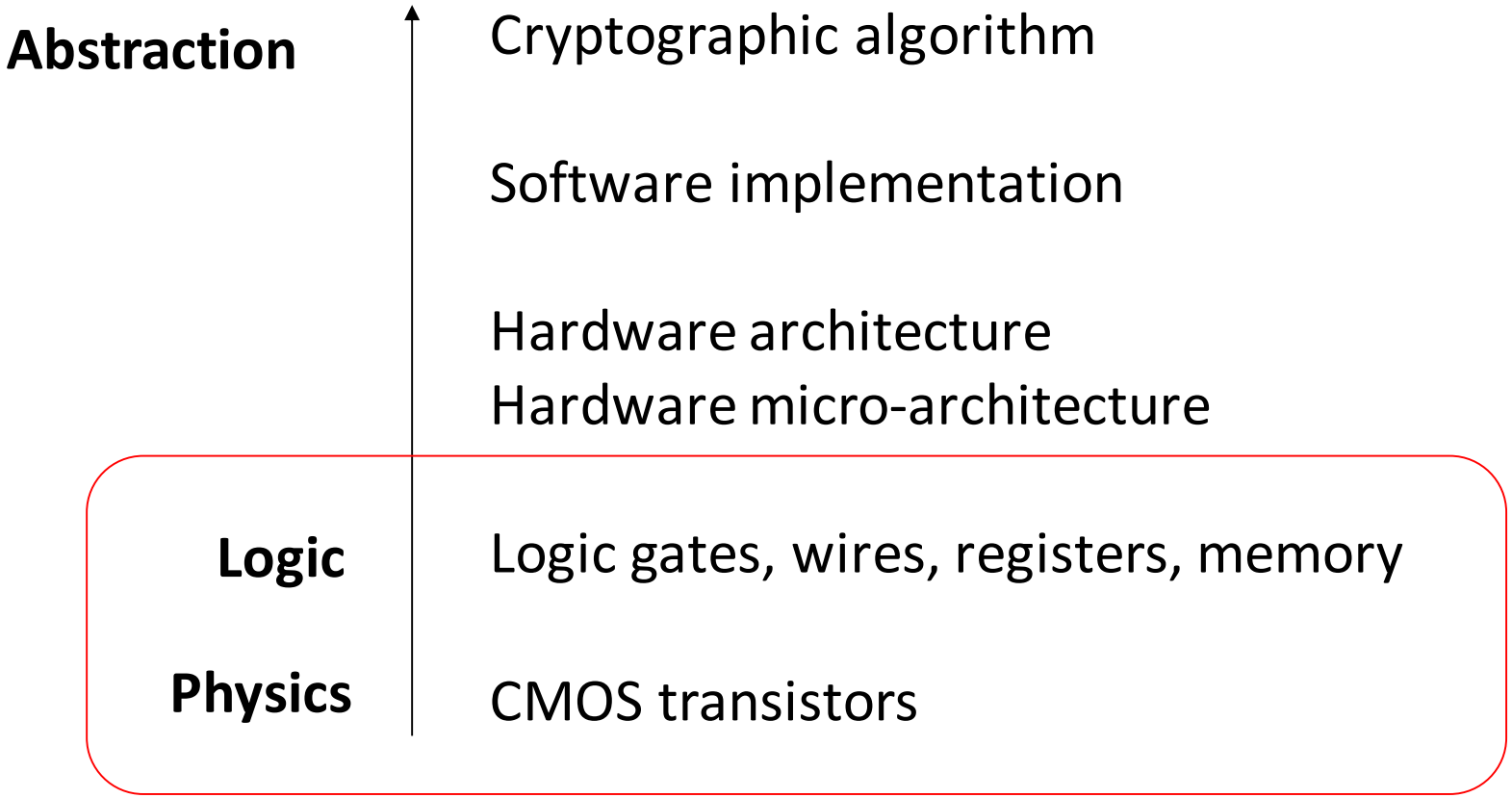
- An attacker just needs more measurements to dig the signal out of noise

**Can we make it impossible for the attacker to guess? -> Masking**

- Mask with a random number X different for each message:
  - $m^d \bmod n \rightarrow [(m \cdot X)^d \bmod n] \cdot [(X^{-1})^d \bmod n] \bmod n$
- Intuitively, given m and $d_i$ the attacker cannot guess ”slow/fast” any more...

# Let’s move to “power” side channels

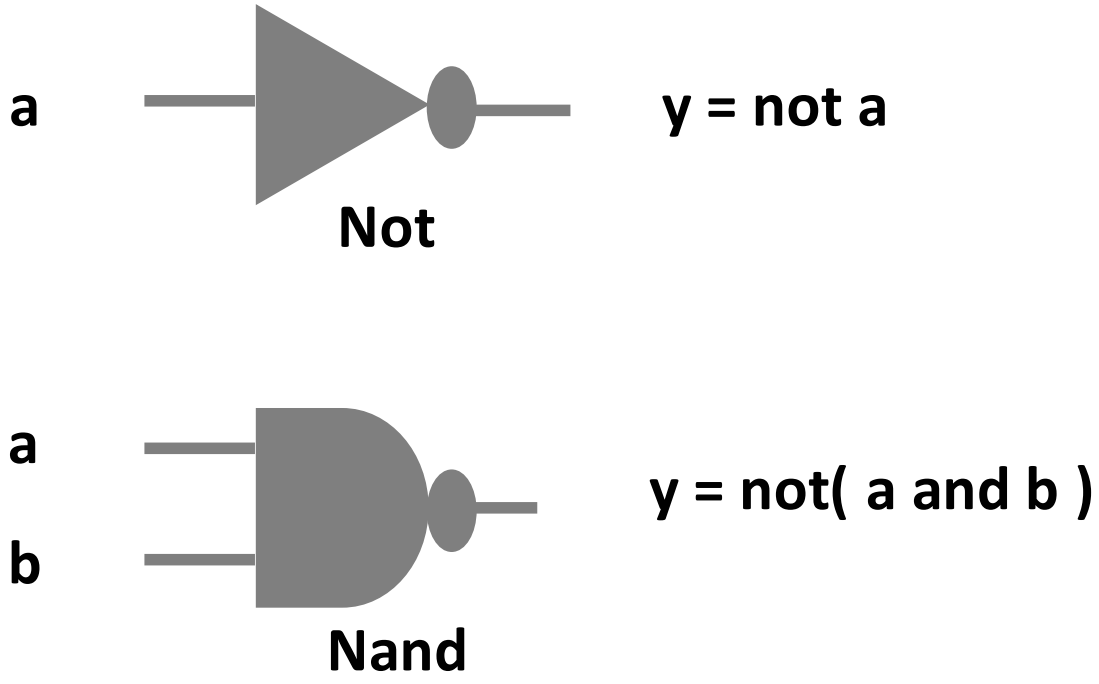
# In the end a cryptographic algorithm runs on hardware, i.e., logical -> physical



# What is a logic gate?

**Logic gate:**

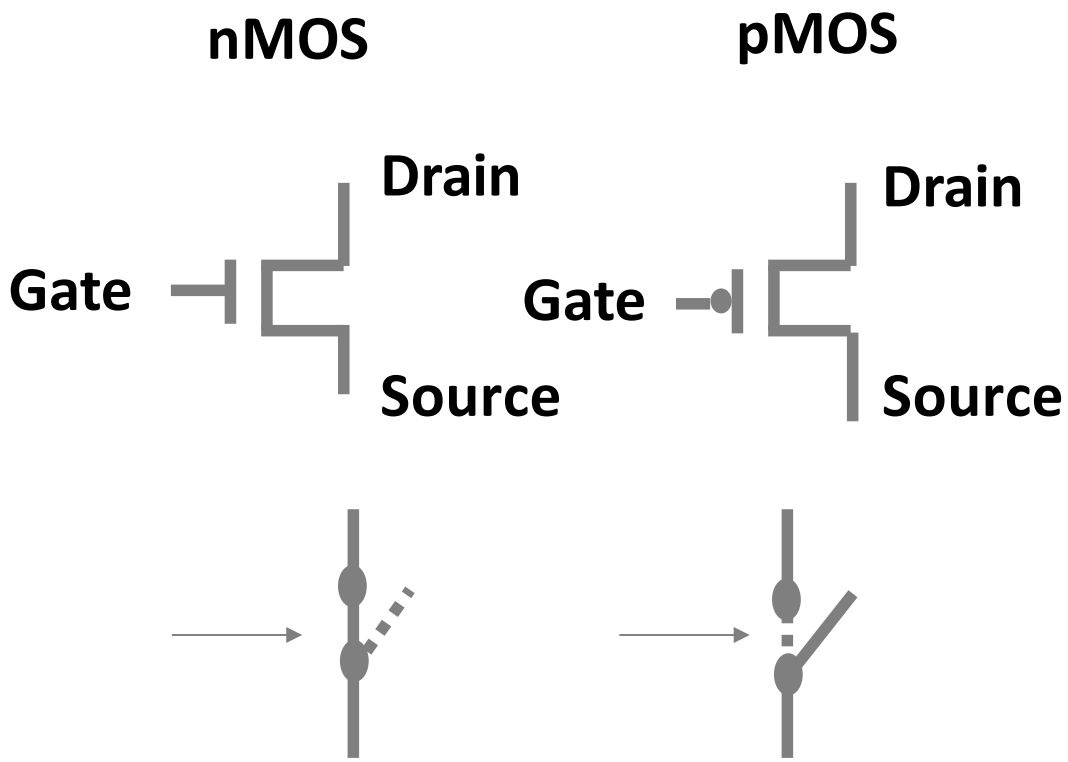
- Electronic component that implements a logic operator
  - E.g., not, and, nand, or, xor
- Stateless (combinatorial)
- Together with memory elements (e.g., registers, RAM) it is used to implement finite state machines



# What is a MOS transistor?

**MOS transistor:**

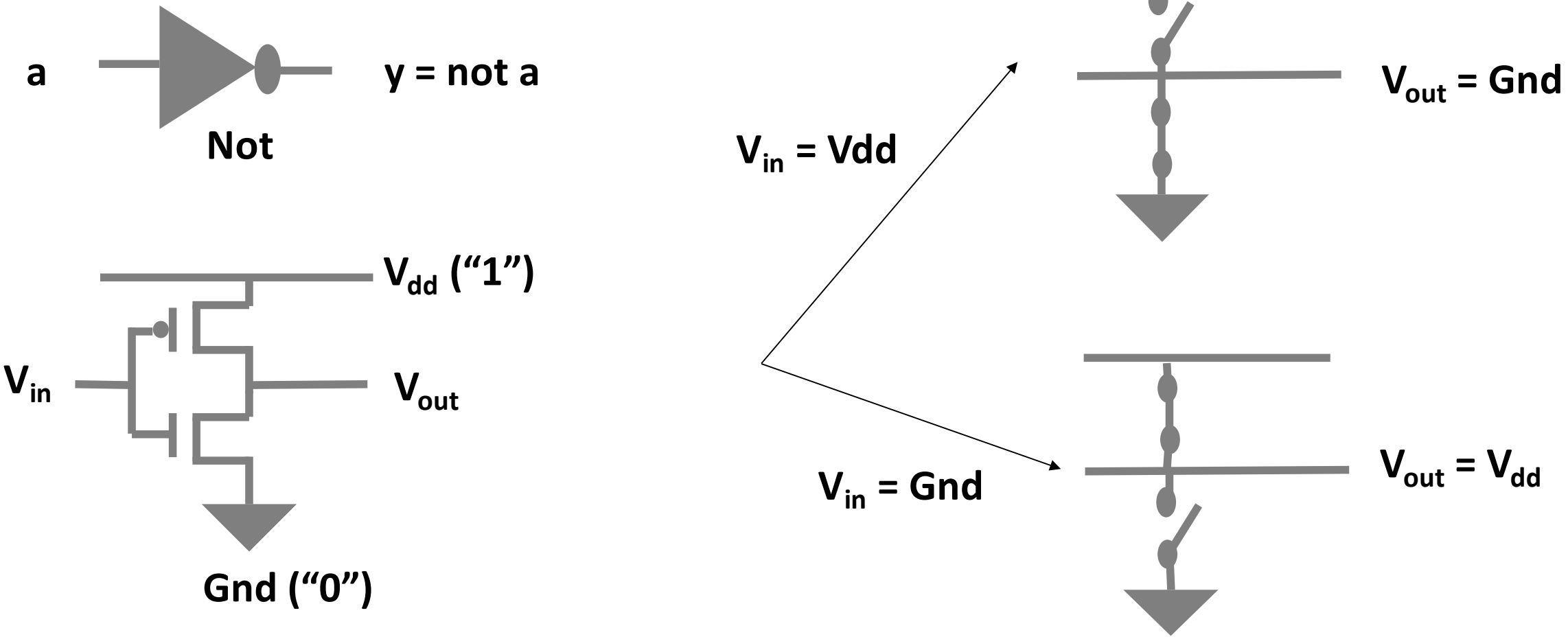
- Transistor
  - Electronic “switch”
  - A very simplified model
- MOSFET
  - Metal Oxide Semiconductor Field Effect Transistor
  - It is a specific type of transistor commonly used in modern digital electronics
  - Two types, n and p
    - Explanation is well beyond the scope of this class
    - They open/close in opposite ways



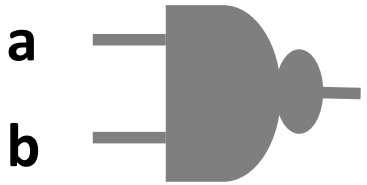

| G | D->S |
|---|---|
| High | On |
| Low | Off |

| G | D->S |
|---|---|
| High | Off |
| Low | On |

# How do you implement logic gates with MOS? Complementary MOS (CMOS)



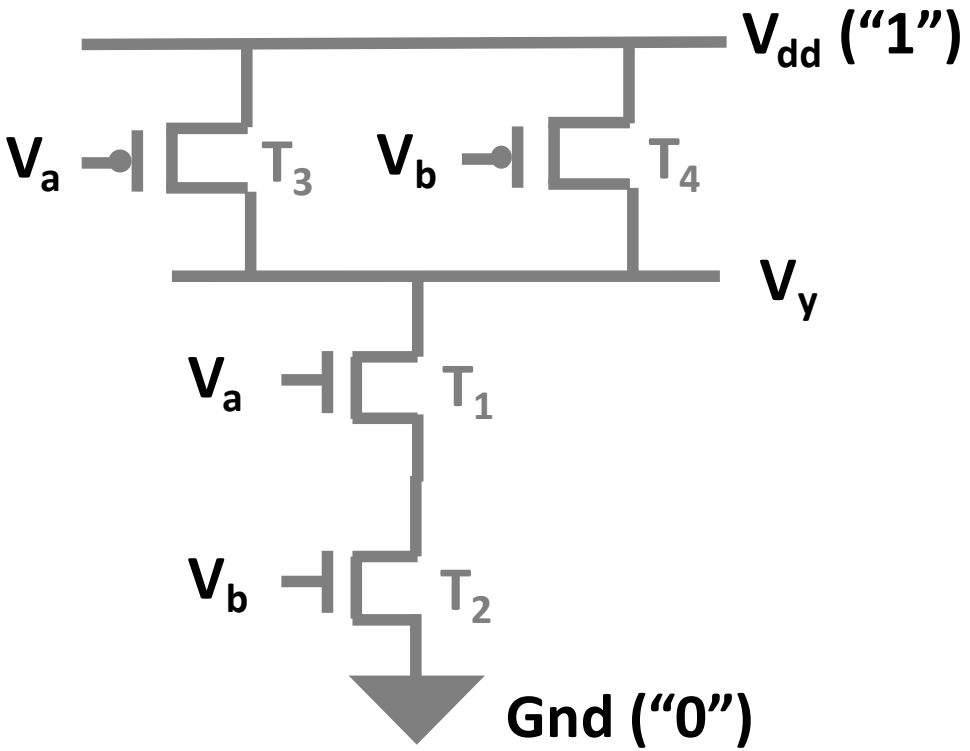
# How do you implement logic gates with MOS? Complementary MOS (CMOS)



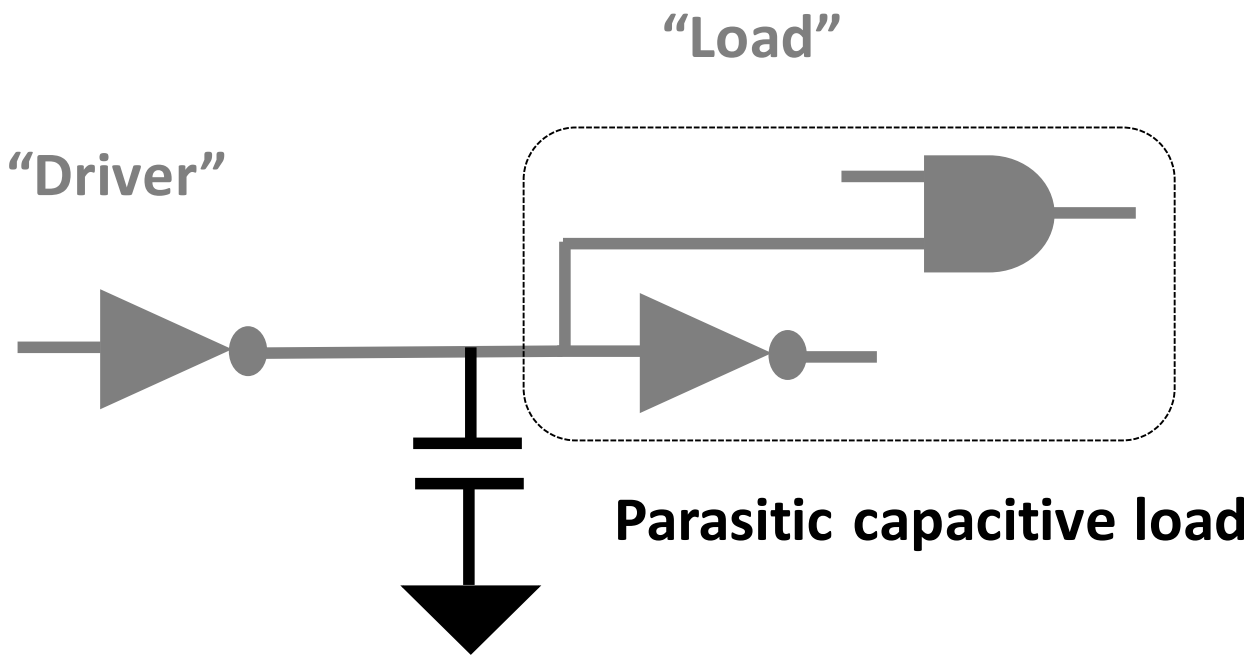

**y = not( a and b )**

**Nand**

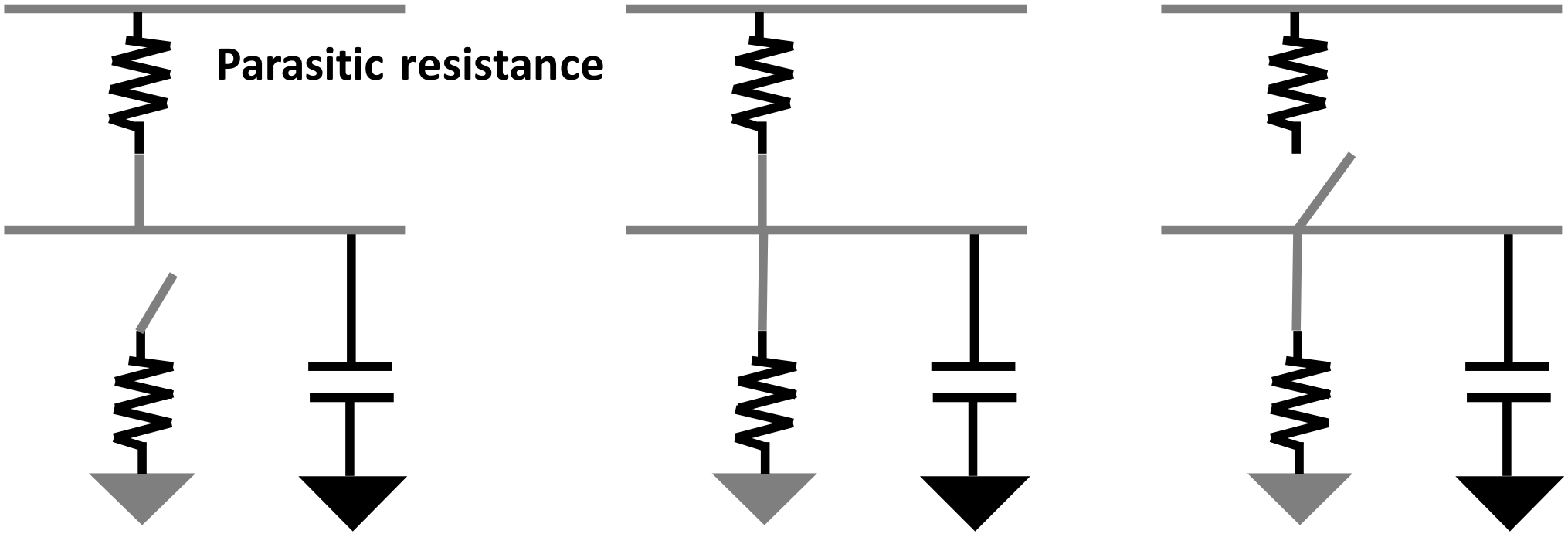
| Input | | Pull down | | Pull up | | Output |
|---|---|---|---|---|---|---|
| $V_a$ | $V_b$ | T1 | T2 | T3 | T4 | $V_y$ |
| “0” | “0” | off | off | on | on | “1” |
| “0” | “1” | off | on | on | off | “1” |
| “1” | “0” | on | off | off | on | “1” |
| “1” | “1” | on | on | off | off | “0” |



# Connecting gates



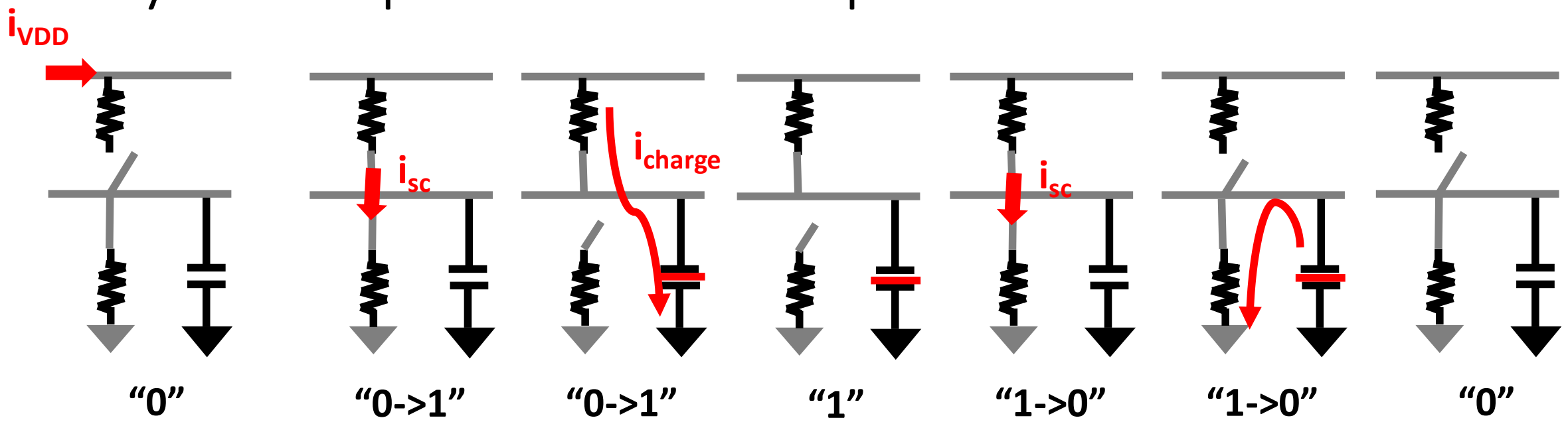
# Switches are not ideal either

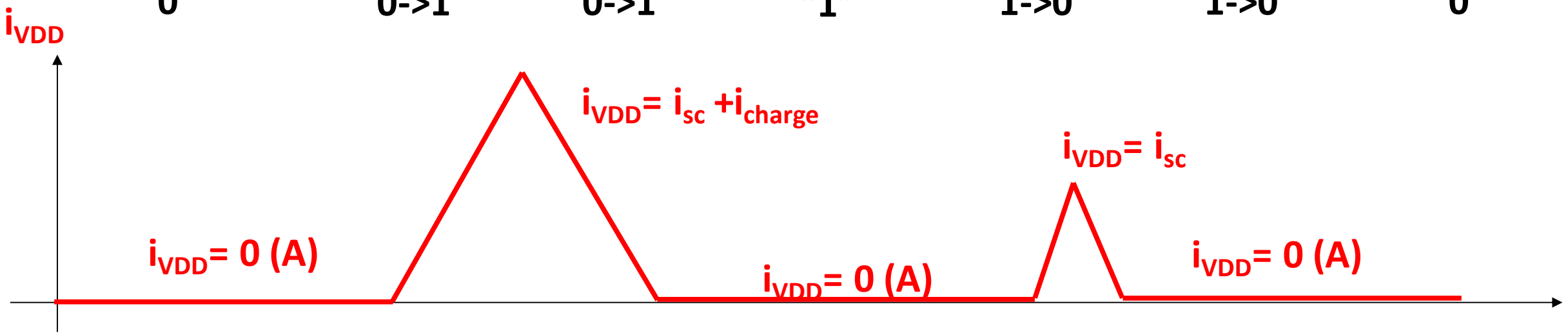



**For a short time during transitions both pull-up and pull-down are on leading to a short circuit**

https://perso.telecom-paristech.fr/pacalet/HWSec/lectures/side_channels/main.pdf

# Dynamic power consumption



# Let's build a model (Hamming Distance)

**Goal:**

- Given the current and past value (state) of a variable Y
- "Predict" the power consumption that led to the current value of Y

**What we know:**

- $I_{"0"} = 0$ No static power (ideally)
- $I_{"1"} = 0$ No static power (ideally)
- $I_{"0->1"} = I_{sc} + I_{charge}$
- $I_{"1->0"} = I_{sc}$

| Past | Current | i |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | $I_{sc}$ |
| 0 | 1 | $I_{sc} + I_{charge}$ |
| 1 | 1 | 0 |

**Model**

- Power proportional to number of bits that transitioned
- i.e., HammingDistance($Y_{current}$, $Y_{previous}$)

# Let's build a model (Hamming Weight)

**Goal:**

- Given the current value (state) of a variable Y
- "Predict" the power consumption that led to the current value of Y

**What we know revisited:**

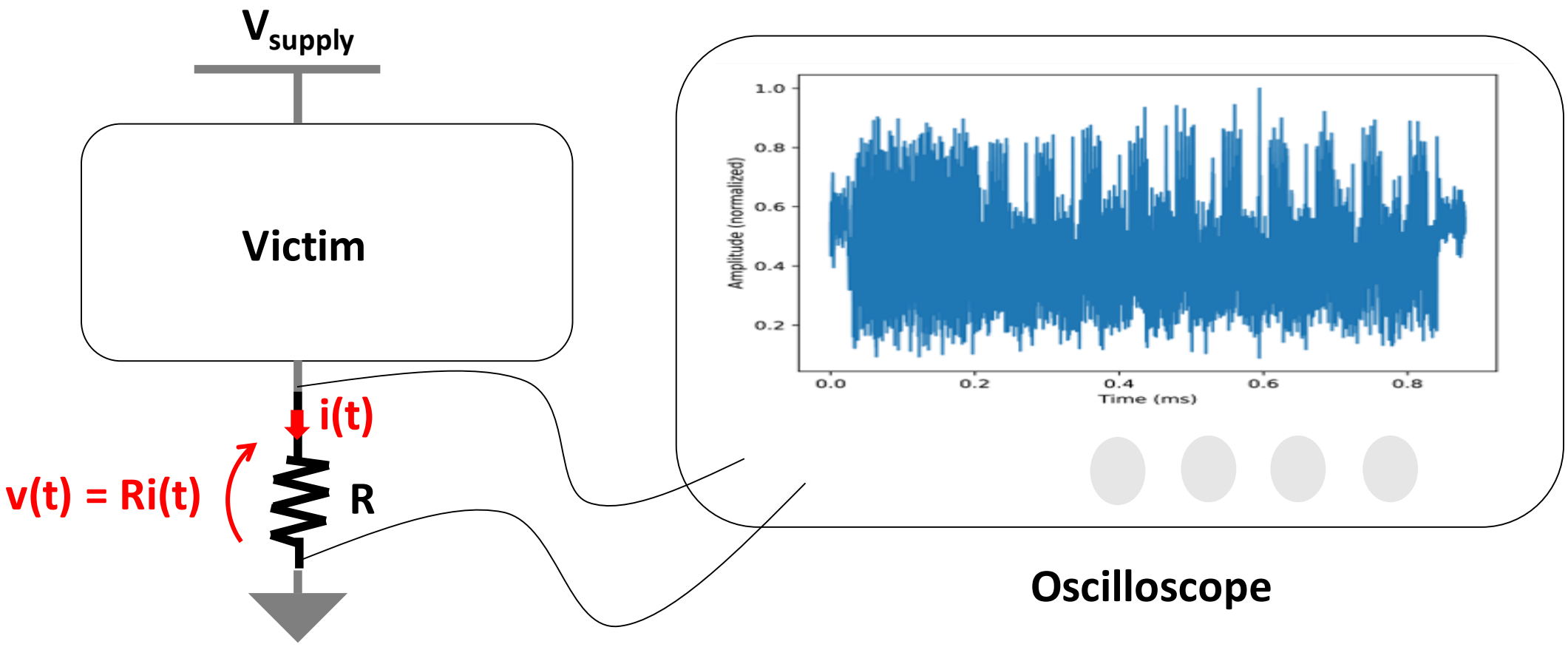
- Static power is ideally 0
- 1's don't consume more than 0's
- But transitions to 1 are in average more expensive than transitions to 0

| Past | Current | i | $i_{avg}$ |
|---|---|---|---|
| 0 | 0 | 0 | $I_{sc}/2$ |
| 1 | 0 | $I_{sc}$ | |
| 0 | 1 | $I_{sc} + I_{charge}$ | $(I_{sc}+I_{charge})/2$ |
| 1 | 1 | 0 | |

**Model**

- Power proportional to number of bits at one
- i.e., HammingWeight($Y_{current}$)

# How to measure? The simplest way



# Lesson learned from this dive in electronics

**Data dependency**

- There are physical phenomena that create a data dependency between logic values and their transitions and the power consumption of the circuit
  - **Note:** we often tend to confuse "power" with "current" because $P = RI^2$
  - **Note:** currents produce EM emissions, leading to EM side channels with similar properties
  - **Note:** we saw a simple example, there are many more parasitic effects and other considerations, but this is a good example to get the intuition

**Measure**

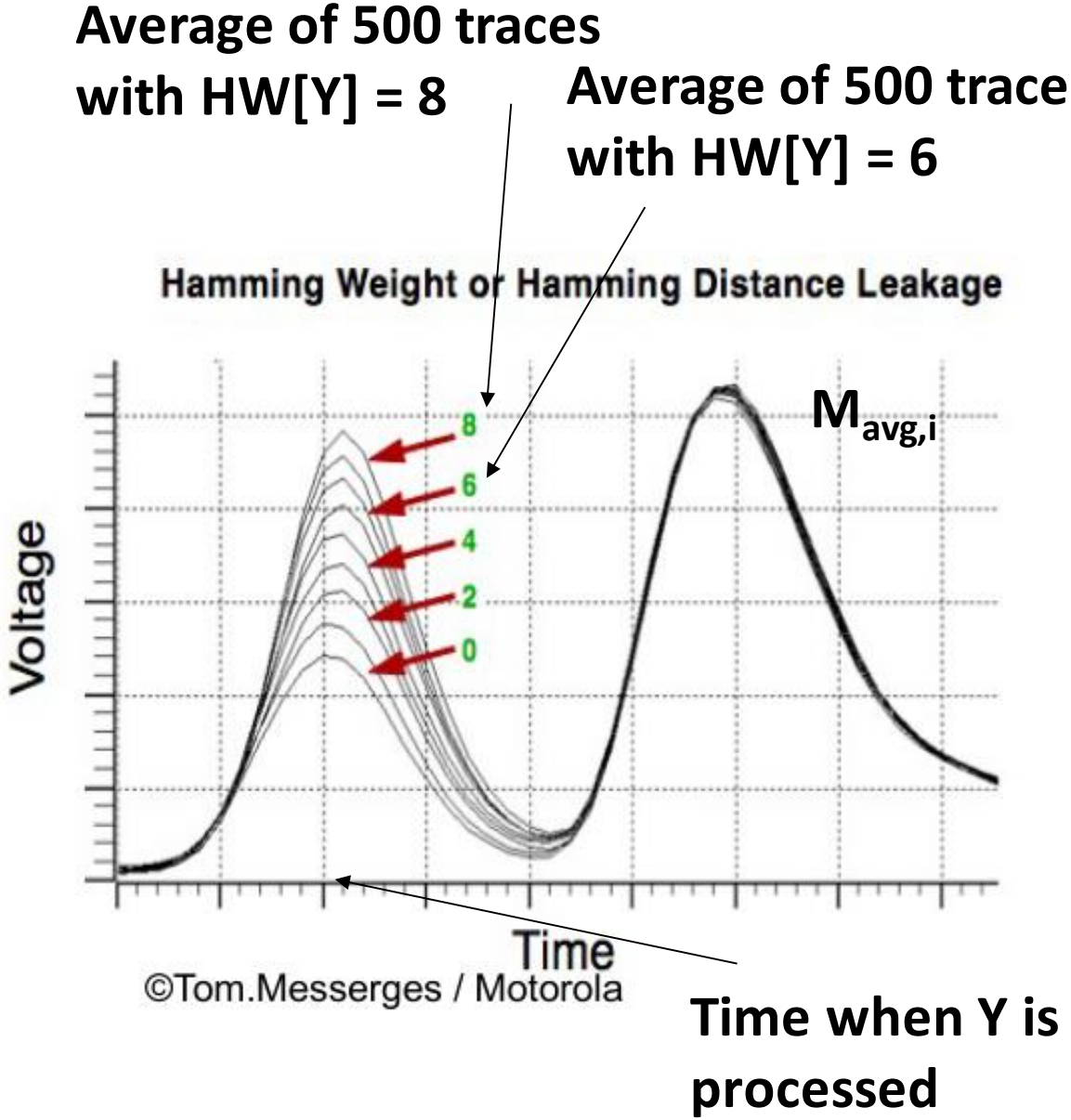
- We can measure the power consumption and observe these phenomena
- Signals are small, many measurements and statistical analysis are often needed

**Model**

- We know how it works: given some logic data manipulated by the software/hardware, we can "predict" the corresponding power consumption

# Example

1. Take a piece of software and focus on an 8-bit variable Y
2. Measure ‘many’ power traces corresponding of a particular Y
3. Align the traces in time
4. Compute the average

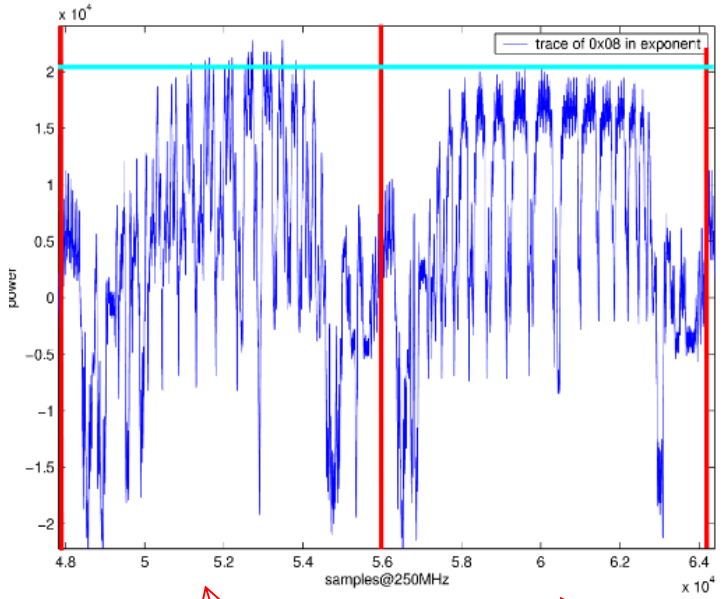


# And now some attacks

# Sometimes we can just look at a single execution: Simple Power Analysis (SPA)

**Recall the squareAndMultiply…**

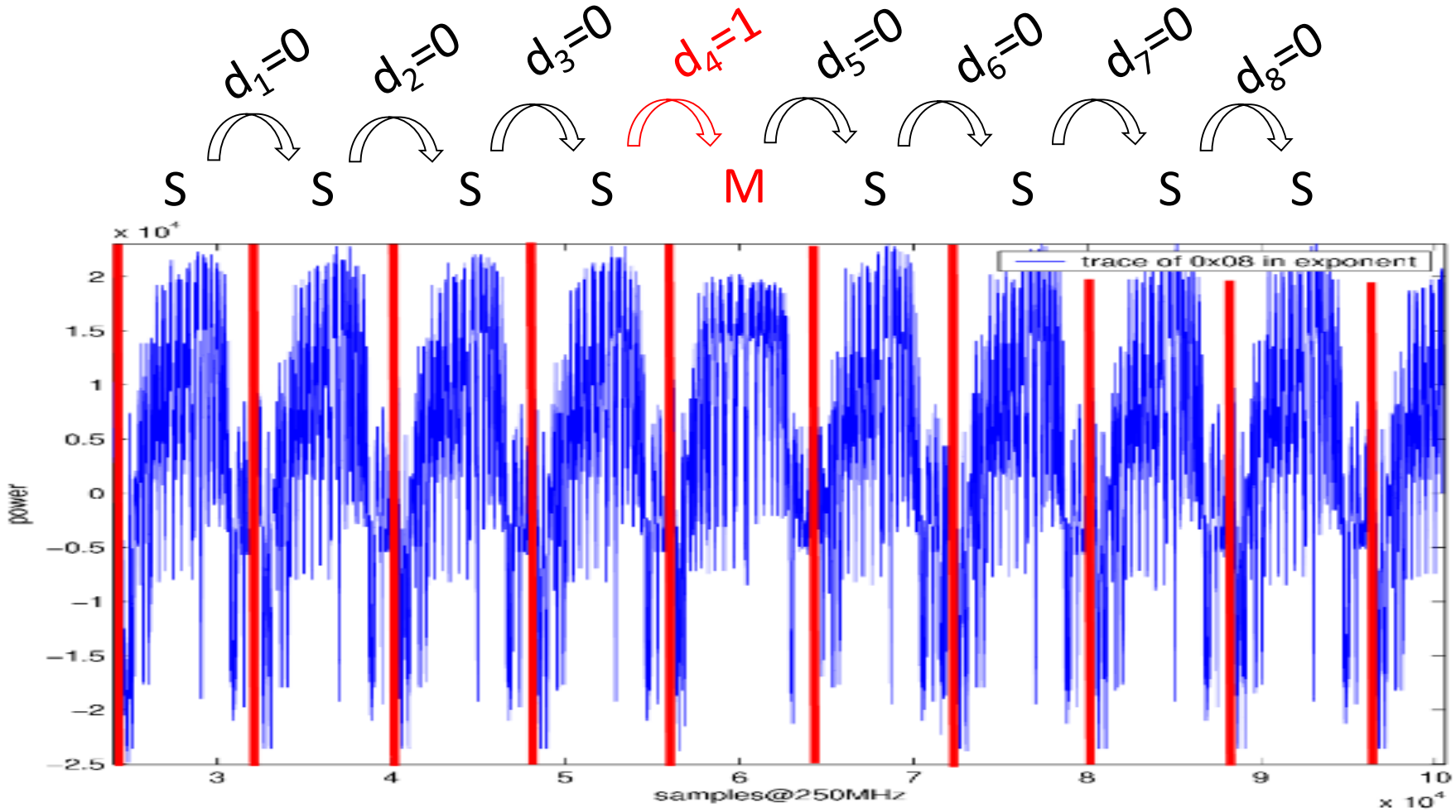
```
squareAndMultiply(m, d, n):
    """ Return s = m^d mod n """
    w = bitlength(d)
    x = m ; Assume d[0] == 1 (MSB)
    for i in 1 to w-1:
        x = x^2 mod n
        if bit i of d == 1:
            x = xm mod n
    return x
```




Multiply executed only if $d_i$ is 1

Square and Multiply look different in the power trace

# Sometimes we can just look at a single execution: Simple Power Analysis (SPA)



# A reminder about AES before attacking it with Differential Power Analysis (DPA)

A symmetric block cipher

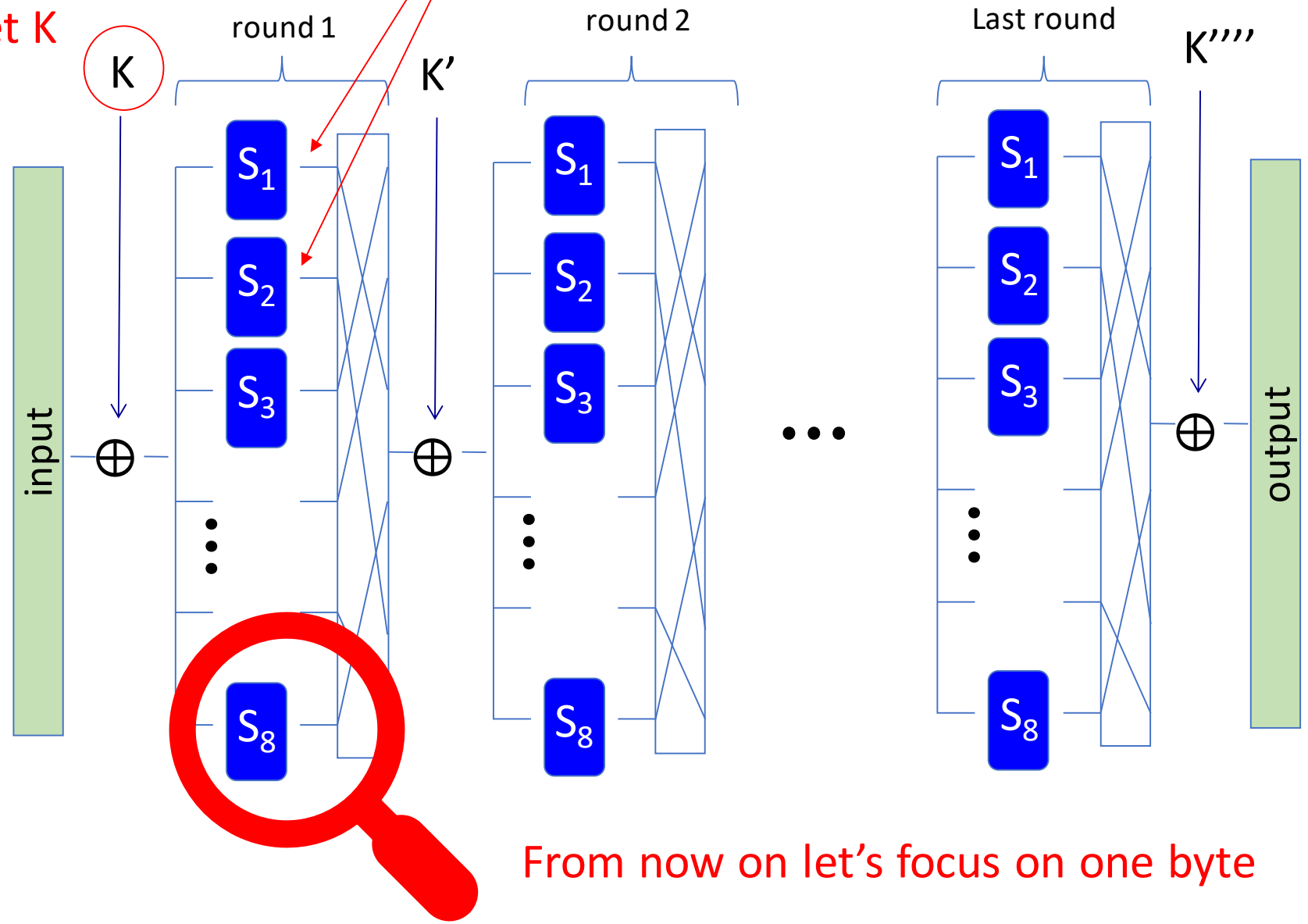
- C = AESEncrypt(P, K)
- P = AESDecrypt(C,K)

“FIPS 197, Advanced Encryption Standard (AES)”.
https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf

A simple implementation
https://github.com/kokke/tiny-AES-c

Goal: find the secret K

Attack one byte at a time at the output of the Sbox of the first round




From now on let's focus on one byte

The S-box used in the **SubBytes()** transformation is presented in hexadecimal form in Fig. 7.

For example, if $s_{1,1} = \{53\}$, then the substitution value would be determined by the intersection of the row with index '5' and the column with index '3' in Fig. 7. This would result in $s'_{1,1}$ having a value of {ed}.

| | | y | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | a | b | c | d | e | f |
| x | 0 | 63 | 7c | 77 | 7b | f2 | 6b | 6f | c5 | 30 | 01 | 67 | 2b | fe | d7 | ab | 76 |
| | 1 | ca | 82 | c9 | 7d | fa | 59 | 47 | f0 | ad | d4 | a2 | af | 9c | a4 | 72 | c0 |
| | 2 | b7 | fd | 93 | 26 | 36 | 3f | f7 | cc | 34 | a5 | e5 | f1 | 71 | d8 | 31 | 15 |
| | 3 | 04 | c7 | 23 | c3 | 18 | 96 | 05 | 9a | 07 | 12 | 80 | e2 | eb | 27 | b2 | 75 |
| | 4 | 09 | 83 | 2c | 1a | 1b | 6e | 5a | a0 | 52 | 3b | d6 | b3 | 29 | e3 | 2f | 84 |
| | 5 | 53 | d1 | 00 | ed | 20 | fc | b1 | 5b | 6a | cb | be | 39 | 4a | 4c | 58 | cf |
| | 6 | d0 | ef | aa | fb | 43 | 4d | 33 | 85 | 45 | f9 | 02 | 7f | 50 | 3c | 9f | a8 |
| | 7 | 51 | a3 | 40 | 8f | 92 | 9d | 38 | f5 | bc | b6 | da | 21 | 10 | ff | f3 | d2 |
| | 8 | cd | 0c | 13 | ec | 5f | 97 | 44 | 17 | c4 | a7 | 7e | 3d | 64 | 5d | 19 | 73 |
| | 9 | 60 | 81 | 4f | dc | 22 | 2a | 90 | 88 | 46 | ee | b8 | 14 | de | 5e | 0b | db |
| | a | e0 | 32 | 3a | 0a | 49 | 06 | 24 | 5c | c2 | d3 | ac | 62 | 91 | 95 | e4 | 79 |
| | b | e7 | c8 | 37 | 6d | 8d | d5 | 4e | a9 | 6c | 56 | f4 | ea | 65 | 7a | ae | 08 |
| | c | ba | 78 | 25 | 2e | 1c | a6 | b4 | c6 | e8 | dd | 74 | 1f | 4b | bd | 8b | 8a |
| | d | 70 | 3e | b5 | 66 | 48 | 03 | f6 | 0e | 61 | 35 | 57 | b9 | 86 | c1 | 1d | 9e |
| | e | e1 | f8 | 98 | 11 | 69 | d9 | 8e | 94 | 9b | 1e | 87 | e9 | ce | 55 | 28 | df |
| | f | 8c | a1 | 89 | 0d | bf | e6 | 42 | 68 | 41 | 99 | 2d | 0f | b0 | 54 | bb | 16 |

**Figure 7. S-box: substitution values for the byte xy (in hexadecimal format).**

"FIPS 197, Advanced Encryption Standard (AES)".
https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf

# Let’s focus on one byte we want to attack

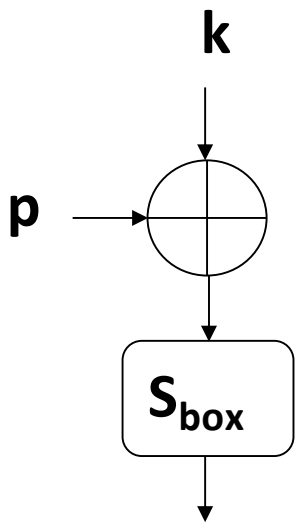
**Plaintext bytes** ($p_0$, $p_2$, …, $p_{15}$)

**Key bytes** ($k_0$, $k_2$, …, $k_{15}$)

```
249  // The SubBytes Function Substitutes the values in the
250  // state matrix with values in an S-box.
251  static void SubBytes(state_t* state)
252  {
253    uint8_t i, j;
254    for (i = 0; i < 4; ++i)
255    {
256      for (j = 0; j < 4; ++j)
257      {
258        (*state)[j][i] = getSBoxValue((*state)[j][i]);
259      }
260    }
261  }
```

Load in a register

First round, SubBytes

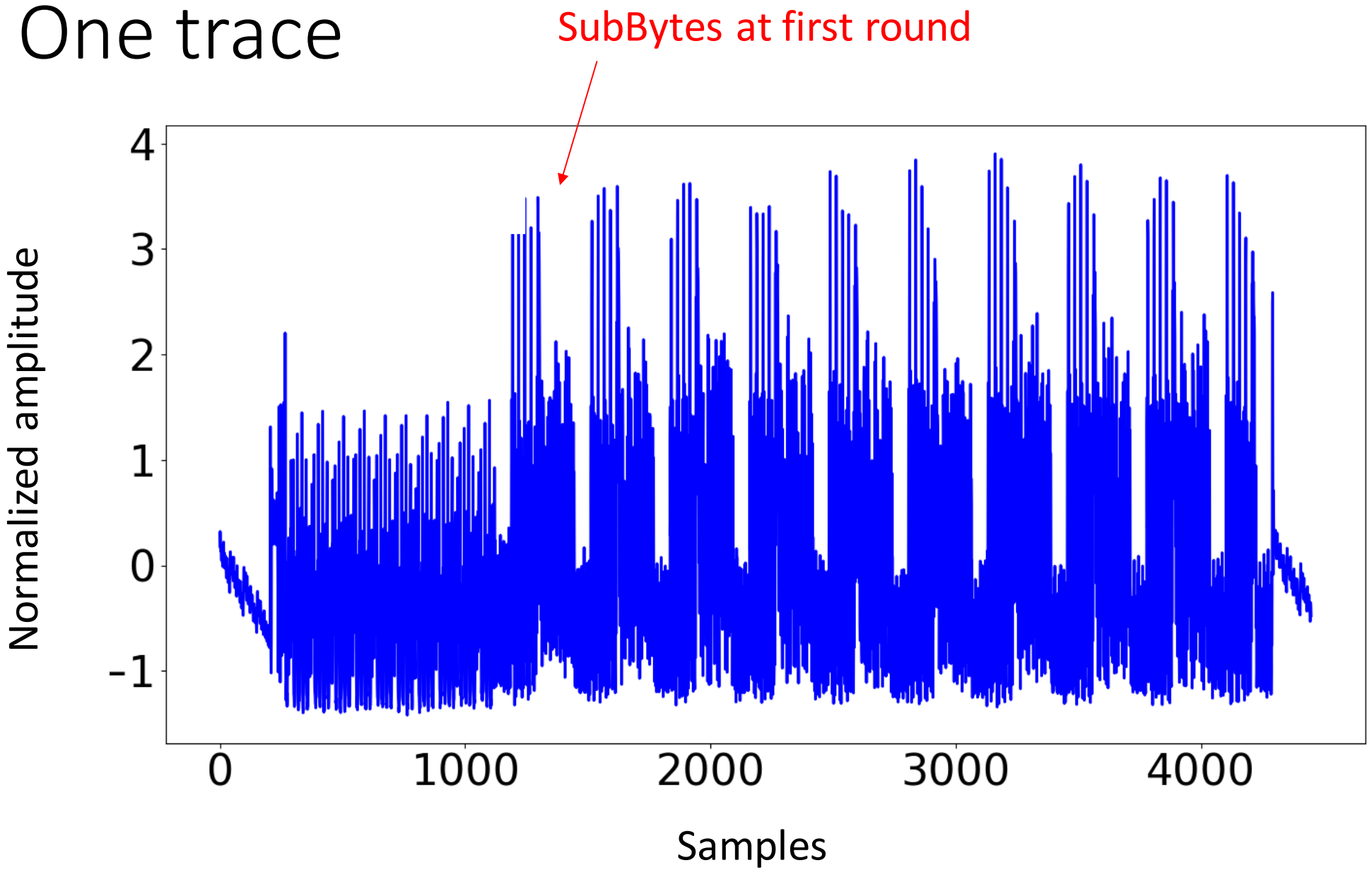



**$y = S_{box}(p \text{ xor } k)$**

Intermediate variable y

For each byte of the plaintext/key

# One trace


SubBytes at first round
Normalized amplitude
4
3
2
1
0
−1
0
1000
2000
3000
4000
Samples

# Find the Point of Interest (POI) (for each byte) (done locally on attacker's system)

**Intermediate variable** y = $S_{box}$(p xor k)

**Leakage measurement** l(y)

**Leakage model** m(y) = HW[y] *(assumption based on physical leakage model that the experiments will validate)*

**Take many measurements**

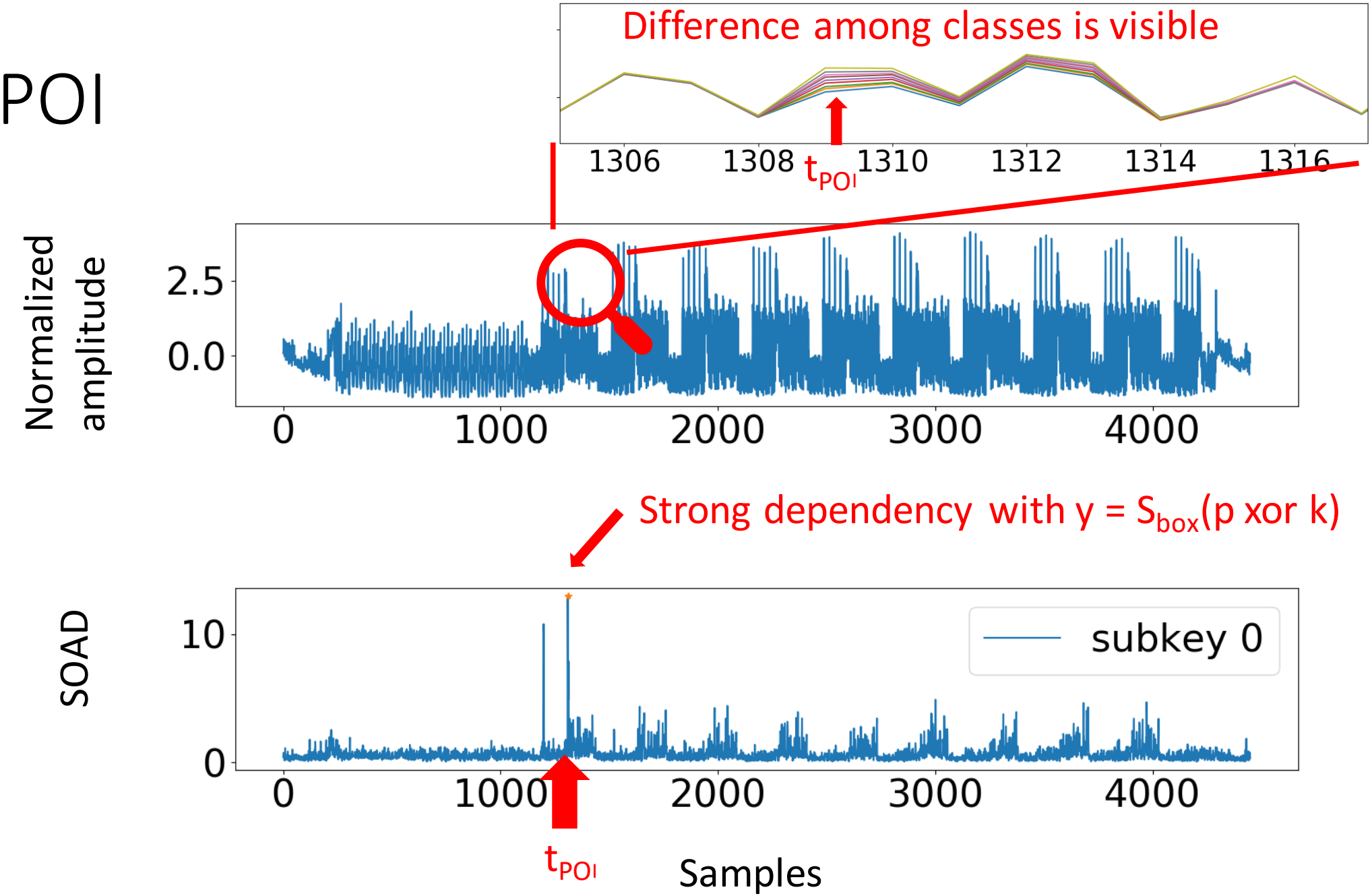
- Variable known key byte k
- Variable known plaintext byte p

**Cluster the traces in 9 classes based on the HW[y] = 0…8**

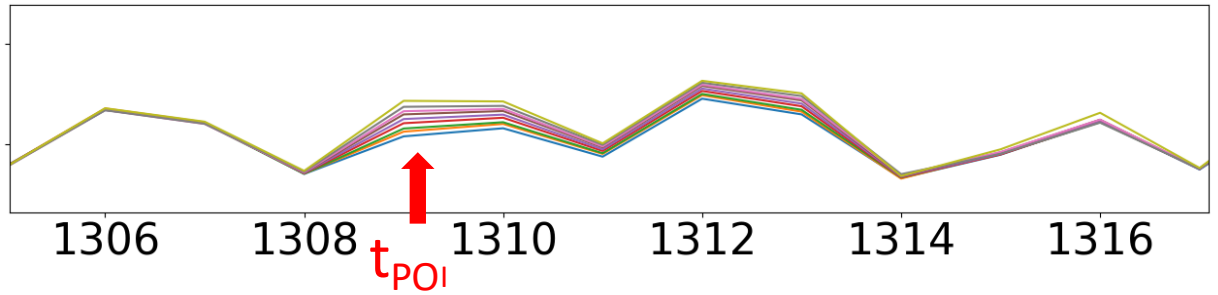
**Compute the Sum Of Absolute Differences**

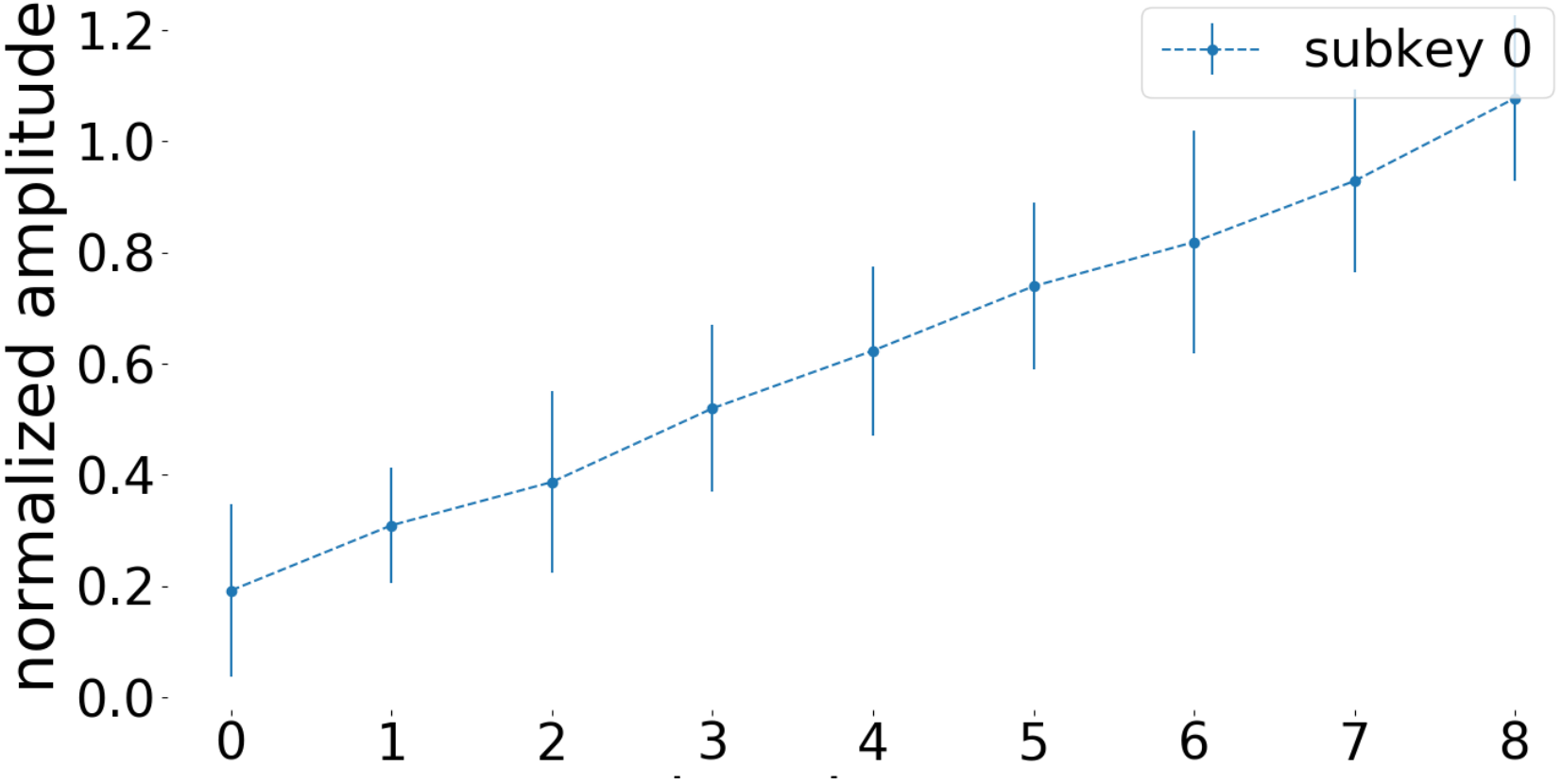
- Compute the average trace for each class
- SOAD = sum of the absolute difference of avg power consumption per each class
  - SOAD is big at time $t_{POI}$ when the measurements show a data dependency with y

# POI


Difference among classes is visible
1306
1308
$t_{POI}$
1310
1312
1314
1316
Normalized amplitude
2.5
0.0
0
1000
2000
3000
4000
Strong dependency with y = $S_{box}$(p xor k)
SOAD
10
0
subkey 0
0
1000
2000
3000
4000
$t_{POI}$
Samples

# Trace($t_{POI}$)




$HW[y] = HW[S_{box}(p \text{ xor } k)]$

# Differential Power Analysis (for each byte)

**Take many measurements L**

- Fixed unknown key byte k
- Variable known plaintext byte p

**for $k_{guess}$ in range 0 to 255:**

**For each measurement l:**

- $y = HW[S_{box}(\mathbf{p} \text{ xor } k_{guess})]$
- If HW[y] <= LowThreshold (e.g., 2)
  - Measure power with **p** as input and put measurement in M1
- else if HW[y] >= HighThreshold (e.g., 6)
  - Measure power with **p** as input and put measurement in M2

$Scores[k_{guess}] = avg\_over\_different\_\mathbf{p}(M2) - avg\_over\_different\_\mathbf{p}(M1)$

**Best guess:** argmax[Scores]

Statistical check of how well the model matches the measurements

This comes from the model

The best score corresponds to the most likely key byte

# Correlation Power Analysis (for each byte)

**Take many measurements L**

- Fixed unknown key byte k
- Variable known plaintext byte p

**for $k_{guess}$ in range 0 to 255:**

$Y = [HW[S_{box}(p \text{ xor } k_{guess})] \text{ for each } p]$

$Scores[k_{guess}] = PearsonCorrelationCoeff(L, Y)$

Replace the previous simple check with the Paerson Correlation Coefficient:

- I.e., check how well the model is linearly correlated with the measured leakage

**Best guess:** argmax[Scores]

# Profiled Correlation Attack (for each byte)

**Take many attack measurements L**

- Fixed unknown key byte k
- Variable known plaintext byte p

**Take many profile measurements**

- Variable known key k
- Variable known plaintext byte p

**for $k_{guess}$ in range 0 to 255:**

Y = [**profile**(Sbox(p xor $k_{guess}$ )) for each p]

Scores[$k_{guess}$] = PearsonCorrelationCoeff(L, Y)

Cluster in 256 classes $M_i$ by value of y=Sbox(p xor $k_{guess}$ )

Compute the average of each class

**profile**(y=i) = avg($M_i$)

Might be better than just assuming HW model

**Best guess:** argmax[Scores]

François Durvaux and François-Xavier Standaert, "From Improved Leakage Detection to the Detection of Points of Interests in Leakage Traces," EUROCRYPT 2016.

# Exercise

**Given:**

- Profiling set, attack set for an unprotected simple AES128
- Script a simple CPA attack

**Goals:**

- Better understanding of the slides
- Try your own improvements
  - Normalize the traces
  - Try to find the POIs with the profile set
  - Run a profiled correlation attack
  - Find other papers describing attacks (e.g., template) and implement them
    - Check the references and search online
    - If you are curious about it, implementing the template attack is instructive and it should take no more than a few hours
  - Etc.

# We just scraped the surface, there is much much much more…

**Leakage identification**

- How to detect data dependency
- T-test, r-test, …

**POIs identification**

- Find the interesting point for attacks
- SOAD, SNR, r-test, …

**Multivariate attacks:**

- Use multiple POIs at the same time

**Other attacks with a profiling step**

- Template attacks

**Higher order attacks**

Complex statistical analysis of multiple measurements

**Etc.**

- ML/DL approaches
- Hardware setup
- Rank enumeration
- Rank estimation
- Using more advanced setups
- Considering static power
- …

# Countermeasures?

# Possible countermeasures

**Problem**

There is a data dependency (of some order) between plaintext, key, and the power

**Possible solutions**

- Add "noise"
  - Desynchronize the traces
  - Inject random noise
- Try to balance the hardware
  - Filters, shielding
  - Make a processor where every instruction/operand consumes the same power
- $N^{th}$ order masking, e.g., $1^{st}$ order
  - Multiply each data with a random variable
  - This algorithmically breaks the dependency making it impossible to guess the intermediate value

Defeated with better signal processing and more measurements

Filtering is not perfect, expensive, can be tampered, etc.

Not easy, expensive

State-of-the-art solution, $(N+1)_{th}$ order attack possible but harder

# Possible countermeasures

**Side note to understand the complexity of the problem**

- Consider 1st order masking scheme
- Mathematical model considers the leakage and prevents 1st order attacks by masking with random values unknown to the attacker
- But…
  - There are still some assumptions
  - Some hardware effects might break the countermeasure

**Example: hidden state**

- The processor's micro-architecture contains some hidden state, e.g., a buffer in the bus to memory, unknown at architectural level
- A value "y xor mask" is (unvoluntarily) loaded in this hidden register that already contains "x xor maks" -> a power leakage revealing the hamming distance between x and y appears

# Some References

https://www.rambus.com/wp-content/uploads/2015/08/DPA.pdf

https://giocamurati.github.io/docs/Thesis_Giovanni_CAMURATI.pdf
Section 2.3 Side Channel Attacks Theory

You can sure find better references and explanations than this one
However, we recommend it because it tries to provide a formal and rigorous explanation while keeping math simple and focusing only on profiled correlation and template attacks

François Durvaux and François-Xavier Standaert, "From Improved Leakage Detection to the Detection of Points of Interests in Leakage Traces," in Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part I, ed. Marc Fischlin and Jean-Sébastien Coron, vol. 9665, Lecture Notes in Computer Science (Springer, 2016), 240–62, https://doi.org/10.1007/978-3-662-49890-3_10.

Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi, "Template Attacks," in Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers, ed. Burton S. Kaliski Jr, Çetin Kaya Koç, and Christof Paar, vol. 2523, Lecture Notes in Computer Science (Springer, 2002), 13–28, https://doi.org/10.1007/3-540-36400-5_3.

More on leakage detection, profiled correlation attacks, and template attacks

# Other similar side channel vectors?

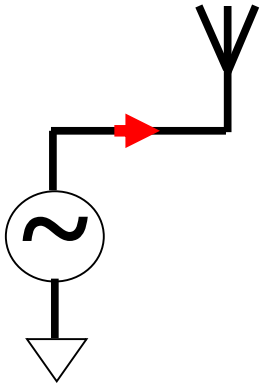
# EM, Sound, …

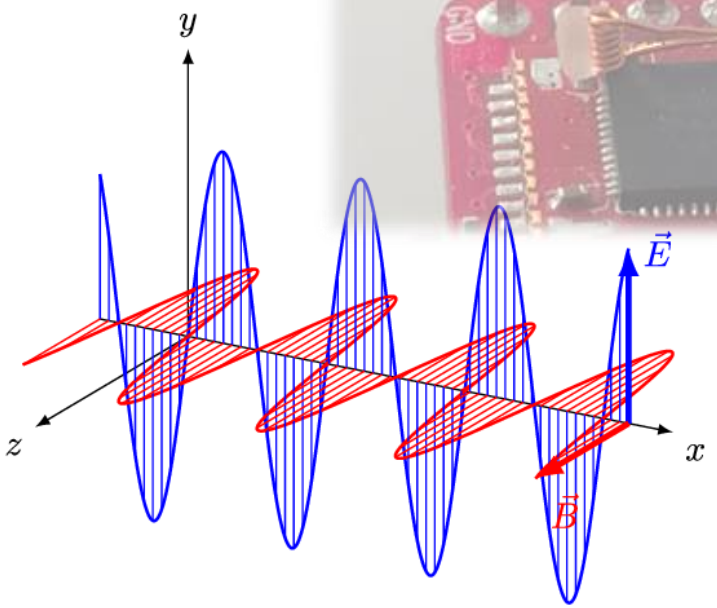
**EM**

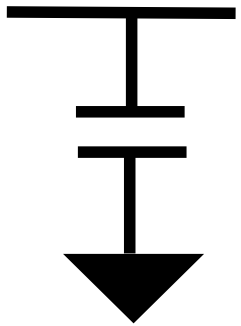
- Currents flowing in cables produce EM signals
- Clock might act as a carrier
- Emissions from localized areas, not all overall power consumption

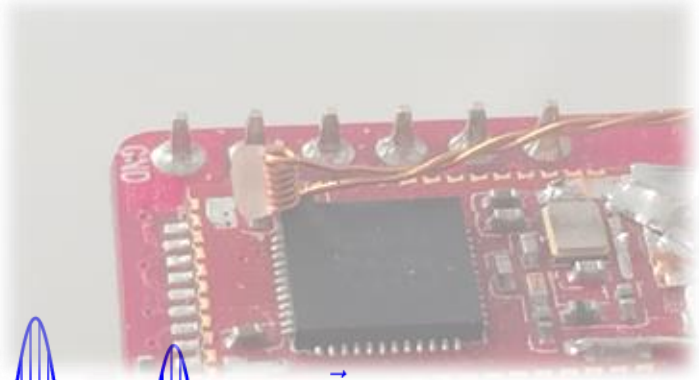
**Sound**

- Currents in certain capacitors make them vibrate and produce sounds







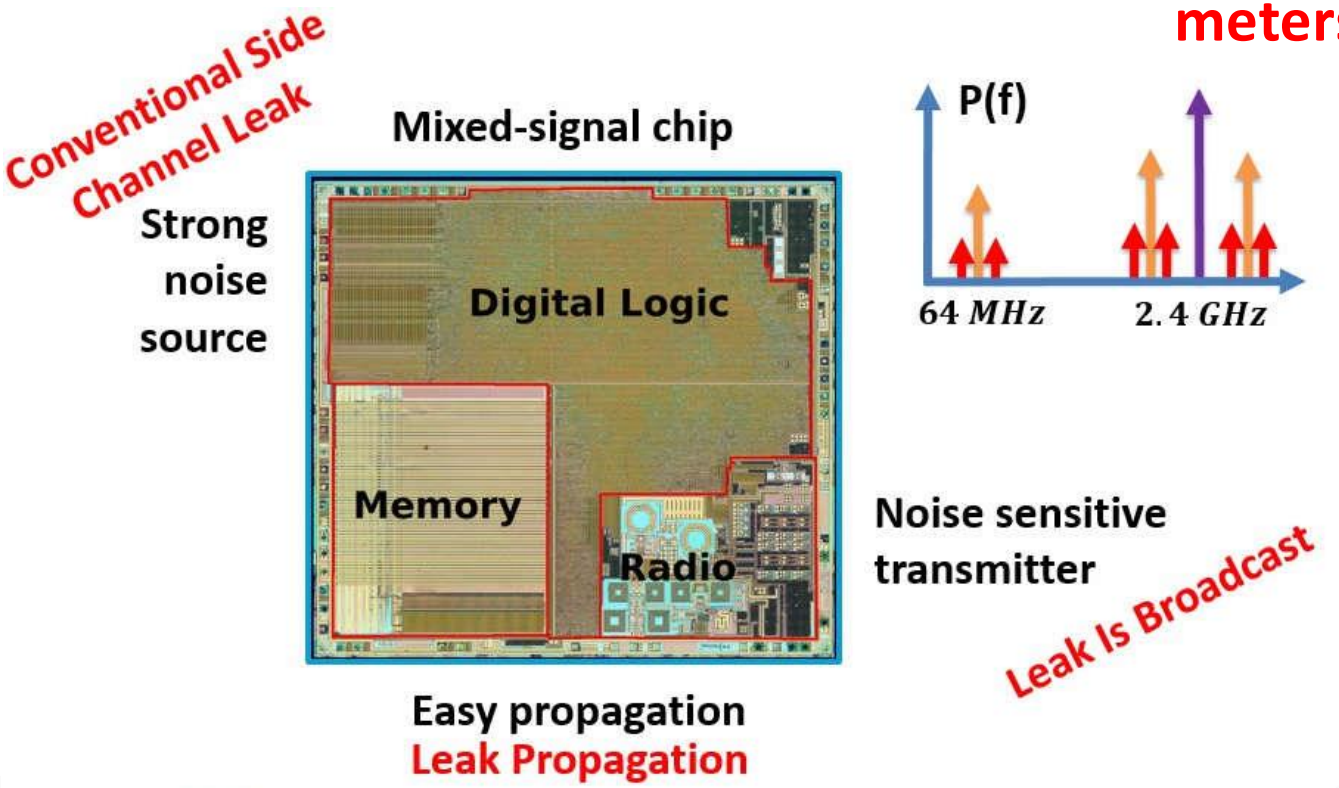
# Large distance / remote power/EM side channel attacks

# Do we always need physical access? No!

**Visible at tens of meters!**

**Screaming Channels**

- Mixed-signal CPU+Radio
- Coupling between the two
- Side channel leakage amplified and broadcast at radio frequency together with data
- **Key recovery attacks possible at 15m**
- AES traces still visible at 60m





G. Camurati, S. Poeplau, M. Muench, T. Hayes, A. Francillon., "Screaming Chanels: When Electromagnetic Side Channels Meet Radio Tranceivers", in CCS 2018

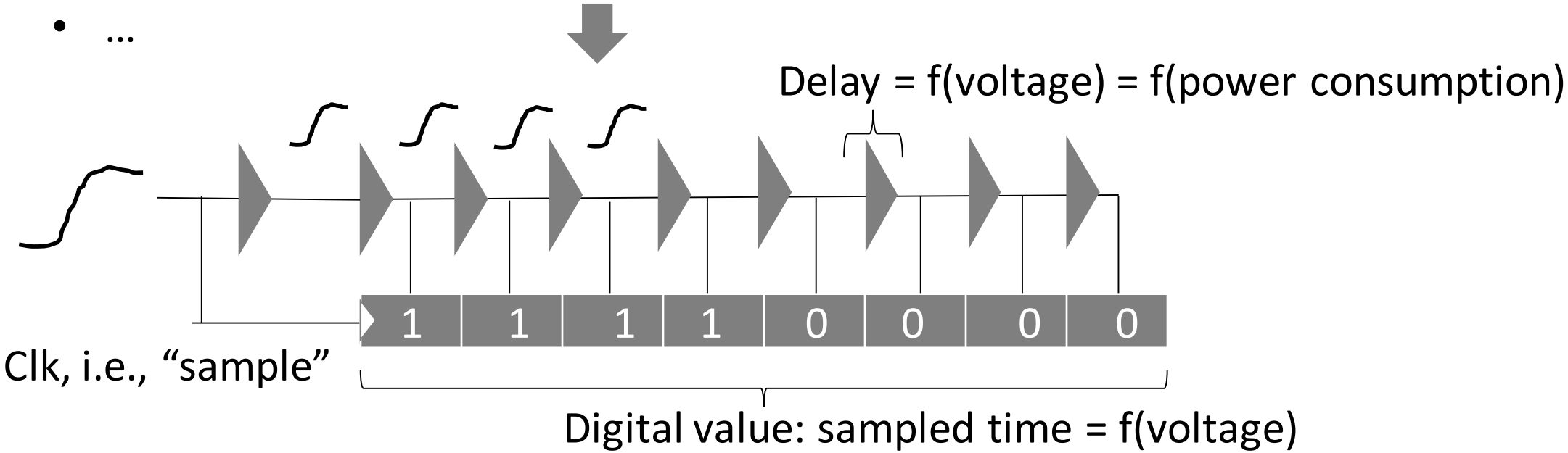
Giovanni Camurati, Aurélien Francillon, and François-Xavier Standaert, "Understanding Screaming Channels: From a Detailed Analysis to Improved Attacks," IACR Transactions on Cryptographic Hardware and Embedded Systems, June 19, 2020, 358–401, https://doi.org/10.13154/tches.v2020.i3.358-401.


Try it yourself https://eurecom-s3.github.io/screaming_channels/

# Do we always need physical access? No!

**Remote side channels**

- On a remote machine there is “some kind of sensor” available
  - E.g., CPU power measurements offered by platform
  - E.g., power consumption from timing delays in delay lines in SoCs
  - E.g., build a special **sensor on an FPGA**, e.g., in the cloud
  - …




Nice overview:
https://cardis2019.fit.cvut.cz/presentations/05_02-Remote%20Side-Channel%20Attacks%20on%20Heterogeneous%20SoC.pdf

# Some References

Dakshi Agrawal et al., “The EM Side-Channel(s),” in Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers, ed. Burton S. Kaliski Jr, Çetin Kaya Koç, and Christof Paar, vol. 2523, Lecture Notes in Computer Science (Springer, 2002), 29–45, https://doi.org/10.1007/3-540-36400-5_4.

Giovanni Camurati, Sebastian Poeplau, Marius Muench, Thomas Hayes, Aurélien Francillon, “Screaming Channels: When Electromagnetic Side Channels Meet Radio Transceivers,” ACM CCS 2018.

Giovanni Camurati, Aurélien Francillon, and François-Xavier Standaert, “Understanding Screaming Channels: From a Detailed Analysis to Improved Attacks,” IACR Transactions on Cryptographic Hardware and Embedded Systems, June 19, 2020, 358–401, https://doi.org/10.13154/tches.v2020.i3.358-401.

Daniel Genkin, Adi Shamir, and Eran Tromer, “RSA Key Extraction via Low-Bandwidth Acoustic Cryptanalysis,” in Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I, ed. Juan A. Garay and Rosario Gennaro, vol. 8616, Lecture Notes in Computer Science (Springer, 2014), 444–61, https://doi.org/10.1007/978-3-662-44371-2_25.

https://cardis2019.fit.cvut.cz/presentations/05_02-Remote%20Side-Channel%20Attacks%20on%20Heterogeneous%20SoC.pdf

# Questions?

# How Do we Fix This?
# *Tamper Resilience*

# Tamper X - classification

- **Tamper resistant** systems take the bank vault approach.
  - This type of system is typified by the outer case design of an automated teller machine (ATM). Thick steel or other robust materials are utilized to slow down the attack by requiring tools and great effort to breach the system.
  - purpose: prevention of break-in

- **Tamper responding** systems use the burglar alarm approach.
  - The defense is the detection of the intrusion, followed by a response to protect the asset. In the case of attended systems, the response may consist of sounding an alarm. **Erasure or destruction of secret data** is sometimes employed to prevent theft in the case of isolated systems which cannot depend on outside response. Tamper responding systems do not depend on robust construction or weight to guard an asset. Therefore, they are good for portable systems or other systems where size and bulk are a disadvantage.
  - purpose: real-time detection of intrusion (and prevention of access to sensitive data)

- **Tamper evident** systems are designed to ensure that if a break-in occurs, **evidence of the break-in is left behind**.
  - This is usually accomplished by chemical or chemical/mechanical means, such as a white paint that 'bleeds' red when cut or scratched, or tape or seals that show evidence of removal. This approach can be very sensitive to even the smallest of penetrations. Frangible (brittle, breakable) covers or seals are other methods available using current technology.
  - purpose: detection of intrusion

- Smartcards – designed to be tamper-resistant – no tamper responsiveness possible
- Cryptoprocessors – much better (internal battery / tamper evident / responsive)

# FIPS 140-2

### Level 1 [ edit ]

Security Level 1 provides the lowest level of security. Basic security requirements are specified for a cryptographic module (e.g., at least one Approved algorithm or Approved security function shall be used). No specific physical security mechanisms are required in a Security Level 1 cryptographic module beyond the basic requirement for production-grade components. An example of a Security Level 1 cryptographic module is a personal computer (PC) encryption board.

### Level 2 [ edit ]

Security Level 2 improves upon the physical security mechanisms of a Security Level 1 cryptographic module by requiring features that show evidence of tampering, including tamper-evident coatings or seals that must be broken to attain physical access to the plaintext cryptographic keys and critical security parameters (CSPs) within the module, or pick-resistant locks on covers or doors to protect against unauthorized physical access.

### Level 3 [ edit ]

In addition to the tamper-evident physical security mechanisms required at Security Level 2, Security Level 3 attempts to prevent the intruder from gaining access to CSPs held within the cryptographic module. Physical security mechanisms required at Security Level 3 are intended to have a high probability of detecting and responding to attempts at physical access, use or modification of the cryptographic module. The physical security mechanisms may include the use of strong enclosures and tamper-detection/response circuitry that zeroes all plaintext CSPs when the removable covers/doors of the cryptographic module are opened.

### Level 4 [ edit ]

Security Level 4 provides the highest level of security. At this security level, the physical security mechanisms provide a complete envelope of protection around the cryptographic module with the intent of detecting and responding to all unauthorized attempts at physical access. Penetration of the cryptographic module enclosure from any direction has a very high probability of being detected, resulting in the immediate deletion of all plaintext CSPs.

Security Level 4 cryptographic modules are useful for operation in physically unprotected environments. Security Level 4 also protects a cryptographic module against a security compromise due to environmental conditions or fluctuations outside of the module's normal operating ranges for voltage and temperature. Intentional excursions beyond the normal operating ranges may be used by an attacker to thwart a cryptographic module's defenses. A cryptographic module is required to either include special environmental protection features designed to detect fluctuations and delete CSPs, or to undergo rigorous environmental failure testing to provide a reasonable assurance that the module will not be affected by fluctuations outside of the normal operating range in a manner that can compromise the security of the module.

# Specialized Devices

- bank cards
- GSM/UMTS SIM cards (store identification key – authentication/charging)
- electronic tickets for mass transport systems
- payTV applications
- access control to buildings
- electronic ID cards, e-passports

Types of devices used

- Smart-cards
- Crypto-processors
- ...

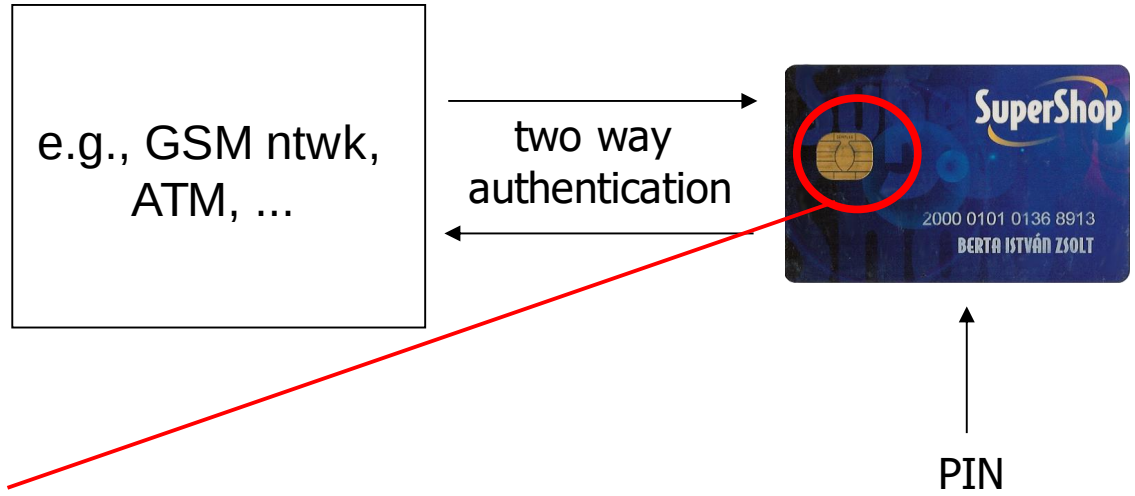
Smart Card


IBM 4758 crypto coprocessor

# Alternative Solutions

# Commodity Solutions

- PIN + card possession enable user authentication
- card holds a key




- GSM/UMTS: operator's trusted piece of hardware on the client's side (shared key/ID + AES + H() + ...).
- ATM: card (key) possession + PIN enable client authentication
- smart cards are intended to protect sensitive data in hostile environments, but …
  - – usually combined with other security measures (e.g., video surveillance, transaction log analysis and blacklisting, …)
  - – when such additional measures are not applied, smart cards become less efficient and fraud prevails (see e.g., payTV systems)
- limitations of smartcards / cryptoprocessors ...

# Reading on (Smart-Card) Hardware Security

- Design Principles for Tamper-Resistant Smartcard Processors http://www.cl.cam.ac.uk/~mgk25/sc99-tamper.pdf
  (M. Kuhn and R. Anderson)

- Cryptographic Processors - a Survey http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-641.pdf
  (M. Kuhn and R. Anderson)

# Hardware Security Module


## Security Features

### Security architecture
- Multilevel military grade security architecture
- Multi-barrier software and hardware architecture with supervision mechanisms

### Encryption/Authentication
- 128/192/256-Bit AES with GCM-, CTR-, ECB-, CBC-, MAC Mode
- Camellia, 3DES (legacy)
- RSA 1024-8192, DSA 512 - 8192
- ECDSA 224-521, GF(P) arbitrary curves, ED25519
- DSA 2048-8192
- Diffie-Hellman 1024, 2048, 4096, ECDH
- SHA-2/SHA-3 (224 - 512), SHA-1, RIPEMED-160, Keccak, HMAC, CMAC, GMAC
- Upgradeable to quantum computer-resistant algorithms

### Key Generation
- Two hardware true random number generators (TRNG)
- NIST SP800-90 compatible random number generator

### Key Management
- Key capacity: up to 30 GB
- Ultra-secure vault for long term keys and certificates
- Up to 120 partitions @ 240 MB secure storage

### Operation
- Number of client connections not restricted
- Unlimited number of backups

### Anti-Tamper Mechanisms
- Several sensors to detect unauthorized access
- Active destruction of key material and sensitive data on tamper
- Transport and multi-year storage tamper protection by digital seal

### Firmware
- Local firmware update on device or optionally on Decanus Remote Control Terminal

### Identity-based authentication
- Multiple security officers (2 out of m)
- Identification based on Smartcard and PIN

## Networking Features

### Software integration
- JCE/JCA Provider
- PKCS#11, OpenSSL
- Microsoft CNG

### Network Management
- IPv4/IPv6
- Monitoring and logging (SNMP V2, syslog)

### Device Management
- Local configuration, remote configuration (Decanus)
- Integrated logging
- Firmware update
- Enhanced diagnostic functions

## Technical Data

### Performance (per second, concurrent)

| | RSA 4096 | ECC 521 | ECC 256 | AES (Mbit) |
|---|---|---|---|---|
| X1000 | 1200 | 1200 | 1200 | >800 |
| X700 | 700 | 320 | 1100 | >700 |
| X400 | 400 | 640 | 1100 | >600 |
| X200 | 200 | 160 | 550 | >500 |

### Power
- Two redundant power supplies, hot pluggable, choice:
  - 100 ... 240 V AC, 50 ... 60 Hz
  - 36 ... 75 V DC
- Power dissipation: 60 W (typ.), 80 W (max.)
- Ultra capacitors for data retention
- Backup lithium battery: Lithium Thionyl Chlorid 0.65g Li, IEC 60086-4, UL 1642, 3.6V

### Interfaces
- 4 Ethernet RJ-45 ports with1 Gbit/s (rear)
- 1 RS-232 management port (front)
- 1 USB management port (front)
- 3 Smart card slots

### Controls
- 3 slots for Securosys Security Smart cards
- 4 LEDs for system and interface status (multicolored)
- 1 liquid crystal display for management information
- Console interface
- Optional Decanus Remote Control Terminal

### Environmental Test Specifications (target)
- EMV/EMC: EN 55022, EN 55024, FCC Part 15 Class B
- Safety: IEC 60950

### Specifications
- Temperature ranges (IEC 60068-2-1 Ad, IEC 60068-2-2 Bd): storage -25...+70 °C; operation 0...+40 °C
- Humidity (IEC 60068-2-78 Cab): 40 °C, 93% RH, non-condensing
- MTBF (RIAC-HDBU-217Plus) at $t_{amb}$=25 °C: 100 000 h
- Dimensions (w×h×d) 440 x 88 x 441 mm (2U 19" EIA standard rack)
- Weight 13.5 kg

### Certification
- FIPS140-2 Level 3 mode
- CC EAL 4+ certified root key storage
- CE, FCC, UL

We strive to continuously improve our offerings and therefore reserve the right to change specifications without notice. Designed and manufactured in Switzerland


EV2.13

https://www.securosys.com/hubfs/2%20WEBSITE%20/2.1%20Products%20/2.1%20Factsheets/181002_Primus-X-Series-Factsheet-E_V2.13.pdf?hsCtaTracking=ad9fb659-5497-4644-9bb2-61e074f4517f%7C0bb25fb5-de95-4e33-9fc2-78be33f12641

# Important Concepts

- Tempest / Soft Tempest / Side Channels
- Tamper ‘X’ Clasiffication
- Timing Side Channnel on RSA (Attacker Model, Reasoning Why it Works, ….)
- Power Analysis of AES (Attacker Model, Reasoning Why it Works, ….)