Advanced Computer Architecture

—

Part III: Hardware Security
Physical Side-Channel Attacks

Paolo Ienne
<paolo.ienne@epfl.ch>

Outline

1. Physical Covert-Channels and Side-Channel Attacks
2. Passive Side-Channel Attacks (Simple and Differential Power Analysis)
3. Active Side-Channel Attacks (Fault Injection)
4. Remote Side-Channel and Fault Attacks
5. Countermeasures to Physical Side-Channel Attacks

1

Physical Covert-Channels and Side-Channel Attacks

Covert- and Side-Channels

Physical Emanation

(power consumption, temperature, electromagnetic waves,…)

Physical

Sender / Victim Process

Receiver / Attacker Process

Measuring Instrument

Microarchitectural

Seen two lectures ago

System State

(…ing, cache, branch predictors, pipelines,…)

Covert- and Side-Channels

- **Most worrying for embedded devices** where physical access is possible
  - Smartcard, etc.
  - IoT is naturally a growing concern

- In the case of **untrusted datacentres**, physical access is to be assumed possible also there (cf. DRAM encryption and integrity problem discussed before)

- And, finally, physical presence is not even always needed
  - FPGAs in SoCs and in datacentres

Invasive vs. Noninvasive

Which device interfaces are used to perform the attack?

- Invasive (and semi-invasive) attacks
  - Typically VLSI devices are unpackaged, layers removed or tampered with, new connections made (using laser cutters, probing stations, focussed ion beam, etc.)
  - Anything is possible…
  - Extremely expensive and thus fairly rare; a serious threat only in extreme cases
- **Noninvasive attacks**
  - **Device attacked only through existing interfaces; usually inexpensive equipment**
  - (Almost) no traces left of the attack
  - Very serious threat

This lecture is essentially about noninvasive attacks

Passive vs. Active

How is the device operating during the attack?

- **Passive attacks**
  - **Device operating as usual**
  - Attacker only observes legitimate input/output and physical emanations
- Active attacks
  - Device, inputs, and environment are manipulated
  - Device operating outside specification and abnormally
  - Attacker learns from this abnormal behavior

This lecture is mostly about passive attacks but there is something about some types of active attacks

Typical Physical Emanations

- Timing
  - One could perhaps see the Evict+Time cache attack as an unconventional timing side-channel attack (not timing of victim but timing of some attacker action)
- **Power consumption**
  - Supply the device through a resistor (1-50 Ω) and measure voltage drop
- Electromagnetic
  - Measure the EM field with a small, often hand-made coil
- Acoustic
  - Capture the noise of a keyboard or of a laptop (usually through capacitors and coils)
- But also temperature, vibration, etc.

Hardware Trojans

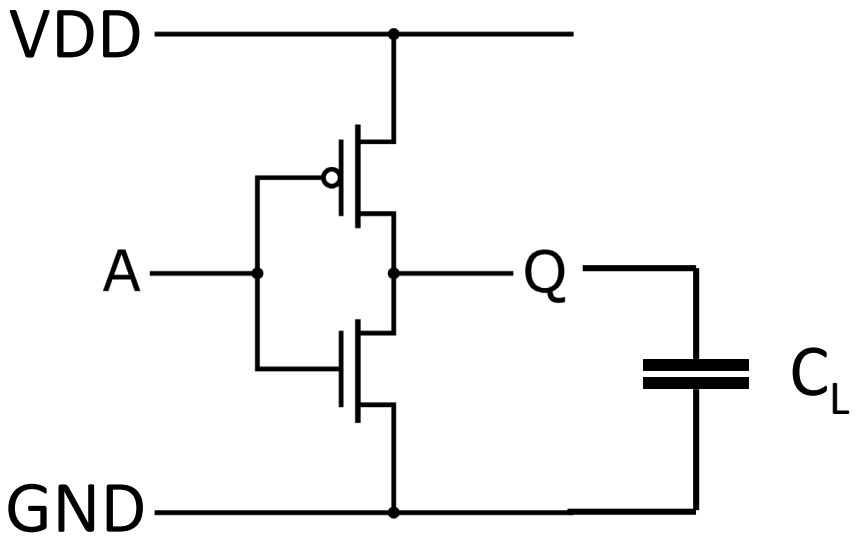
- **Malicious stealthy circuit**, unintended part of an otherwise genuine computer
  - Added through an IP/component purchased from a disreputable source
  - Added by modification of the manufacturing data by a disreputable foundry
  - **Supply-chain attacks**: large-scale global outsourcing in design and manufacturing
- Typically inactive most of the time, until it receives some activation signal and then performs some rogue action
  - **Activation** may be a thinned wire which creates an open after accelerated aging, an FSM brought into an hidden state, a sensor receiving a particular physical signal, etc.
  - **Action** could be a damage to the computer, a critical change in the specification, a corruption of the a data being processed, the leakage of secrets through a covert channel, etc.
- Not further covered in this lecture

2

Passive Side-Channel Attacks

(Simple and Differential Power Analysis)

CMOS Power Consumption Is Data Dependent



- Two causes
  - Mainly, the **parasitic load capacitance C_L** needs to be charged (Q → '1') and discharged (Q → '0')
  - Also, for a very short time, there is a **current path ("short") between VDD and GND** when both transistors are partially conducting

Electromagnetic Attacks

- **Fundamentally similar** to power consumption attacks
- Biot-Savart law

$$\mathbf{B}(\mathbf{r}) = \frac{\mu_0}{4\pi} \int_C \frac{I\, d\boldsymbol{\ell} \times \hat{\mathbf{r}}'}{|\mathbf{r}'|^2}$$

- Ignore everything else: the electromagnetic **field $\mathbf{B}$ depends on the current I** in a wire which in turn depends on the data (see previous slide)

Leakage Models

- To perform an attack, one must **reason about how data** flowing into the circuit **influence the measured emanation**
- One needs mathematical models of leakage but they depend on **technological aspects**
- They can be remarkably approximate (see later) and still allow successful attacks:
  - Hamming Weight HW(x) → how many bits of x are '1'
  - Hamming Distance HD(x_{t+1}, x_t) → how many bits of x changed
- Yet, the more accurate is the model, the more efficient (e.g., shorter time, smaller number of runs) is the attack

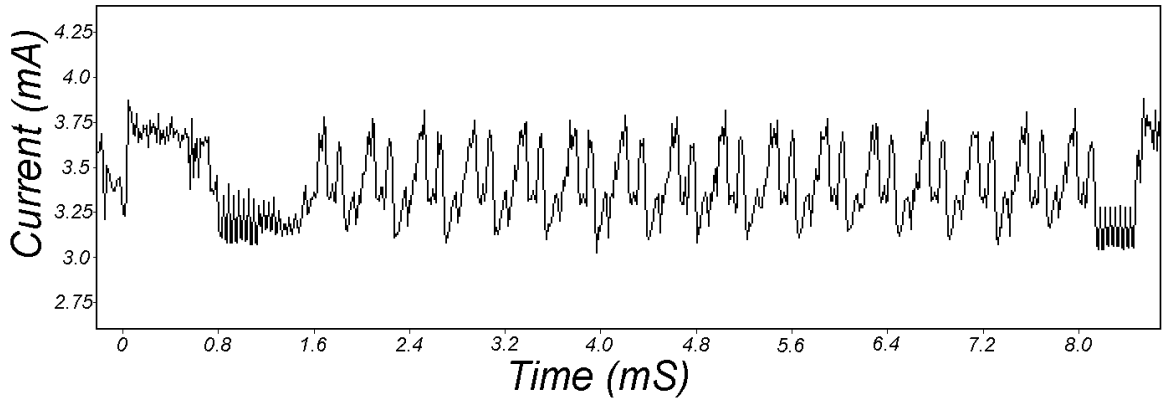
Simple vs. Differential

- Simple
  - Usually **visual inspection of a few power traces** or simple template-based analyses
  - Typically tells something about the operations performed (which may in turn reveal something about the data being processed) but not directly on the data
  - Sort of naïve but effective

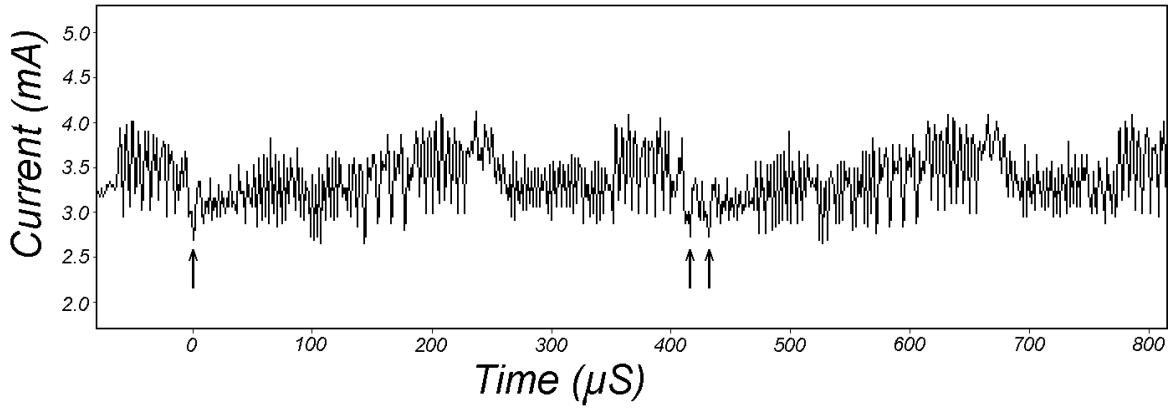
- Differential
  - **Statistical analysis** of large numbers of power traces
  - Tells directly something about the data being processed
  - **Extremely powerful**

Simple Power Attack

- Example of a power trace with a complete DES operation (note the 16 rounds, clearly identifiable)



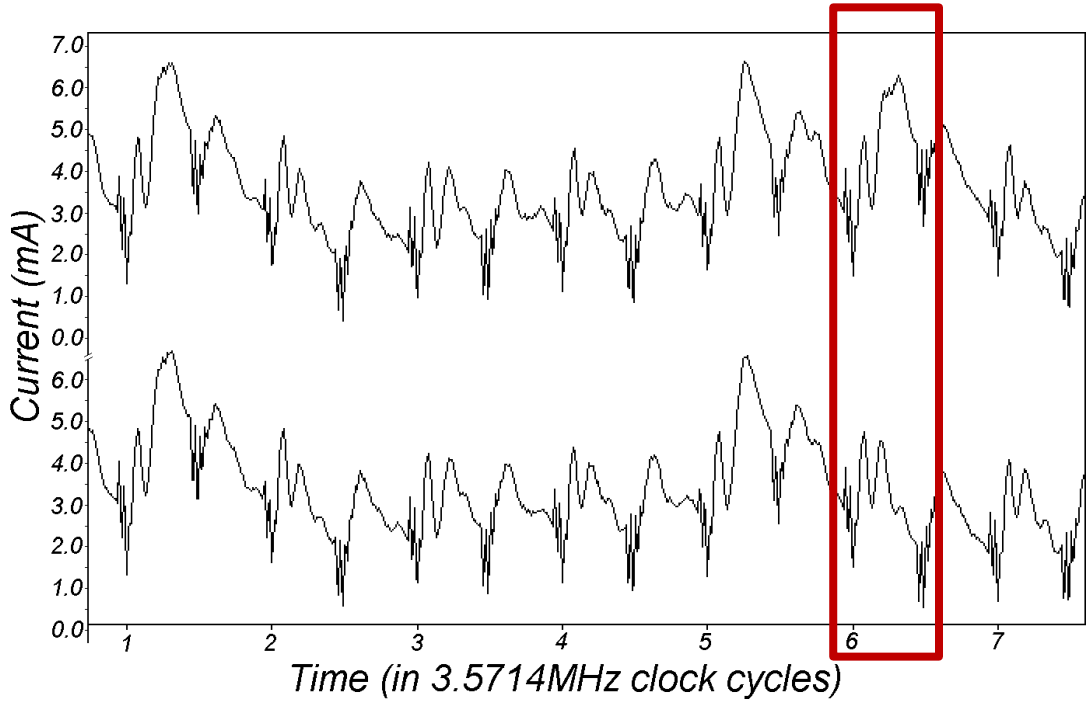
- Zoom in to DES rounds 2 and 3, with some characteristic difference pointed out (one versus two exchanges between registers C and D)



Source: Kocher et al., CRYPTO 1999

Simple Power Analysis

- Further zoom in to the level of clock cycles
- **Difference in control path** (taken vs. not taken branch)
- Clearly, the knowledge of the control path followed may reveal secrets

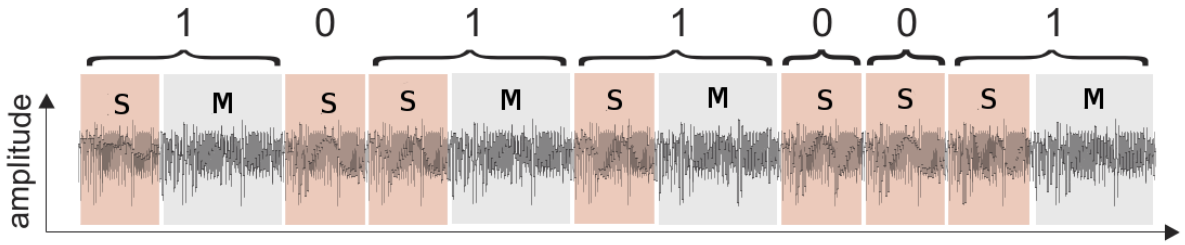


- Note that the difference could even be in the **microcode of the processor**

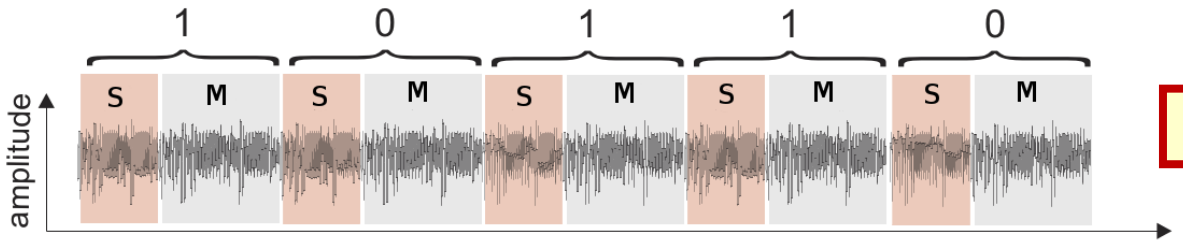
Source: Kocher et al., CRYPTO 1999

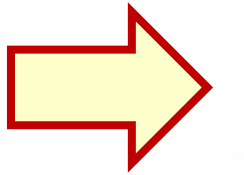
Simple Power Analysis

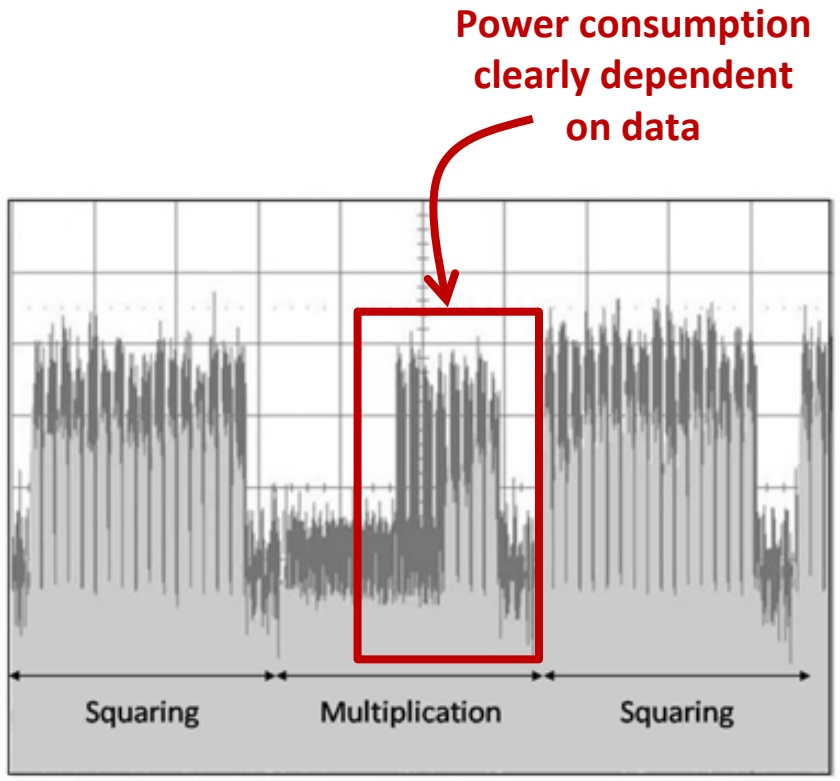
- Different resolutions may help depending on the situation
  - A simple implementation of Montgomery Modular Multiplication easily leaks the data:



  - But even a better one could be attacked:







Source: Courrèges et al., CARDIS 2010

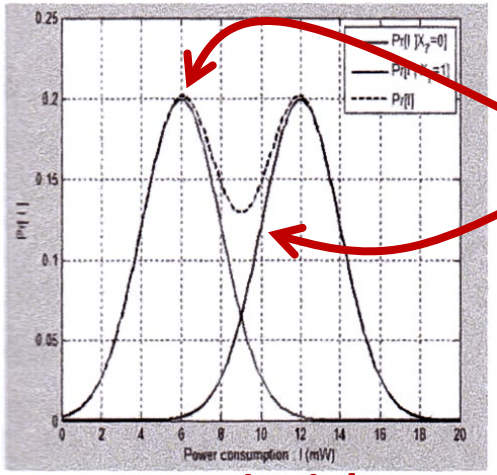
Basic Idea of Differential Power Analysis

- Instead of looking visually for a difference and interpreting ad-hoc what the difference might tell on the secret, find out **statistical correlation between hypotheses on the secret and actual measurement**
  - (Almost) only one (essential) assumption: what is the cryptographic algorithm being run
  - Other assumptions are only mildly influential on the result
- Use **many measurements to cancel noise** (thermal noise, environment noise, data-independent consumption, consumption variations due to nonsecret data, etc.)

1. Record traces with very many different plaintexts
   - It is assumed we can trigger or observe many encryptions with known plaintext, for instance
   - Traces will be very different and **will tell nothing individually** (see SPA; affected by the noise above)
2. Make an hypothesis on the key (or part of it) and **compute some internal signal** (decision or selection function) as a function of the plaintext and of the guess
   - Divide-and-conquer: choose a decision function which depends only on a few bits of the key, so that we have less guesses to make

Measuring Correlation

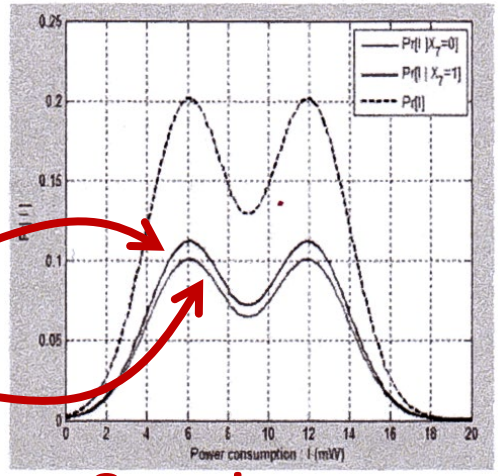
3. Split the measurements into **two groups based on the decision function**
   - If the **guess is wrong**, the decision function computed is uncorrelated with the real internal signal and thus with any point of the power traces → the **average of the two sets will be identical** (for a sufficiently large number of measurements) → the difference approximately null
   - If the **guess is right**, the decision function is equal to a real internal signal and thus correlated to something in the power traces (part of the consumption in some instants) → the **average of the two sets will be different in those instants** → the difference will correspond to the effect of the internal signal and thus nonnull



Guess is right

Two different distributions with different average

Two similar distributions with similar average



Guess is wrong

Source: Bhunia and Tehranipoor, 2019

Decision Function

- A typical decision function for DES could be

$$d = MSB(SBOX_{0..3}(K_{0..5} \oplus P_{0..5}))$$

- Where
  - $K_{0..5}$ is a sub-key that derives directly and reversibly from key K
  - $P_{0..5}$ is obtained from the known plaintext
  - $SBOX_{0..3}()$ is the 4-bit substitution box function
  - MSB() indicates the most significant bit

S-Box or Substitution-Box

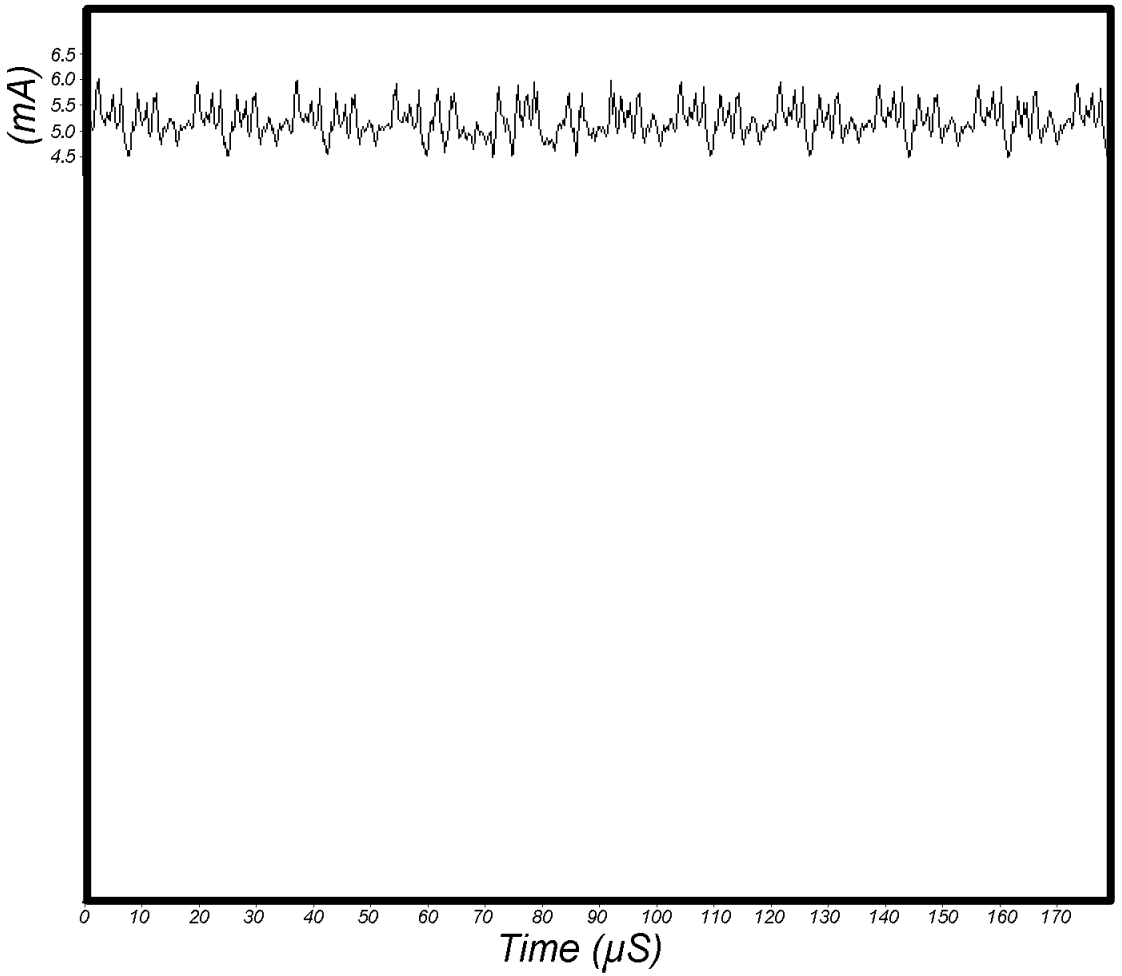
- Basic **nonlinear component** of most cryptographic algorithms, typically used to obscure the relationship between the key and the ciphertext

S_5		Middle 4 bits of input															
		0000	**0001**	**0010**	**0011**	**0100**	**0101**	**0110**	**0111**	**1000**	**1001**	**1010**	**1011**	**1100**	**1101**	**1110**	**1111**
Outer bits	**00**	0010	1100	0100	0001	0111	1010	1011	0110	1000	0101	0011	1111	1101	0000	1110	1001
	01	1110	1011	0010	1100	0100	0111	1101	0001	0101	0000	1111	1010	0011	1001	1000	0110
	10	0100	0010	0001	1011	1010	1101	0111	1000	1111	1001	1100	0101	0110	0011	0000	1110
	11	1011	1000	1100	0111	0001	1110	0010	1101	0110	1111	0000	1001	1010	0100	0101	0011

- Why its output is this a **powerful choice?**
  - Using a bit after the nonlinear substitution box function makes this bit equally dependent on several bits of the key
  - Every **successful attack results in disclosing 6 bits of the key** at once

Differential Traces of DES

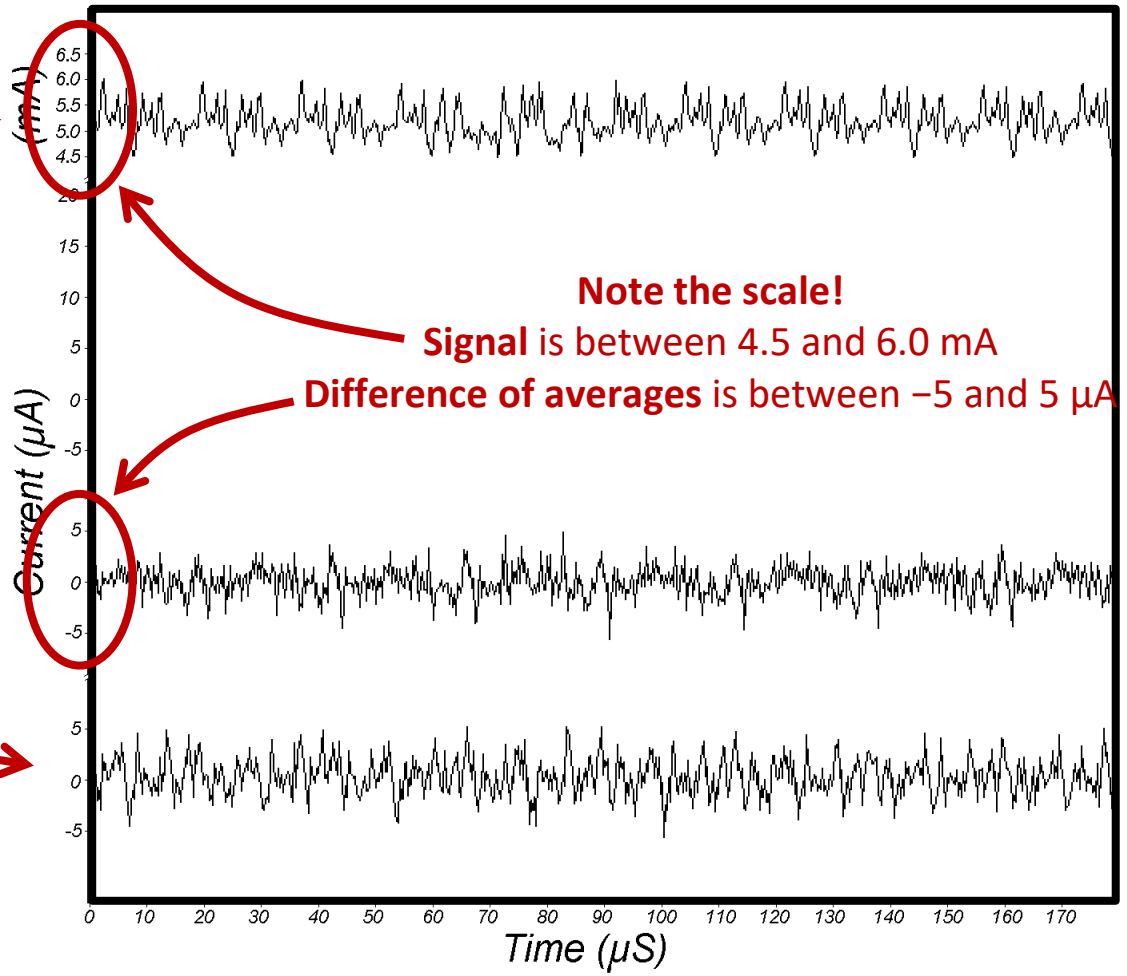
An individual trace
(= SPA) tells nothing



Source: Kocher et al., CRYPTO 1999

Differential Traces of DES

An individual trace
(= SPA) tells nothing



Note the scale!
Signal is between 4.5 and 6.0 mA
Difference of averages is between −5 and 5 µA

Differential traces for wrong key
guesses do not show anything peculiar

Source: Kocher et al., CRYPTO 1999

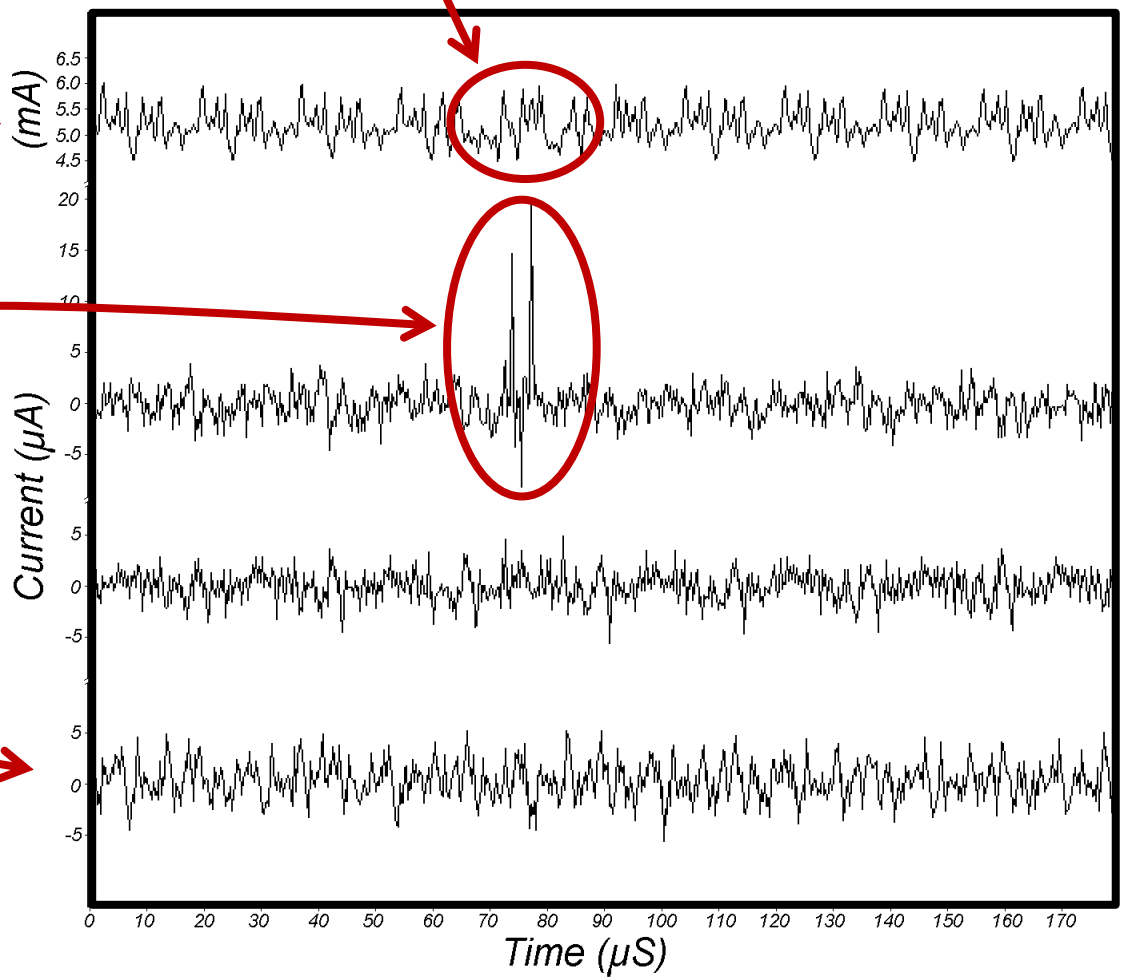
Differential Traces of DES

We do not need to know when and why!
The peaks just indicate a correlation between the effects of our guess and the actual measurement

An individual trace (= SPA) tells nothing

Differential trace for the correct key guess has clear peaks where the traces leak the secret

Differential traces for wrong key guesses do not show anything peculiar



Source: Kocher et al., CRYPTO 1999

Attack Multiple Bits at Once

- Instead of a binary decision function d use a multivalued decision on several bits $D_{0..n}$ as, for instance,

$$D_{0..3} = SBOX_{0..3}(K_{0..5} \oplus P_{0..5})$$

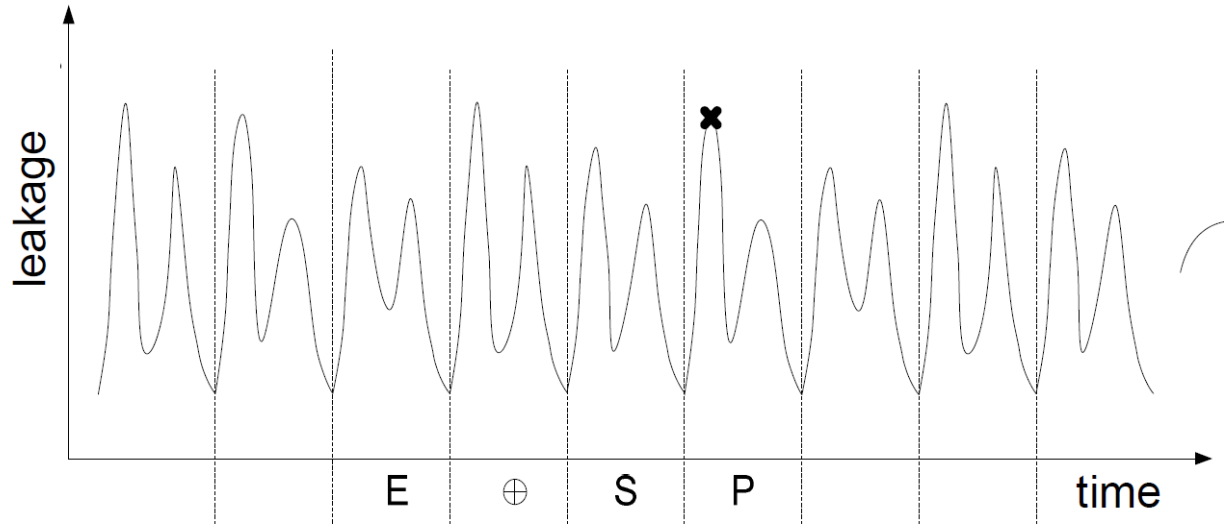
- The idea is that, with the same hypotheses on the key, one can improve the signal to noise ratio by **increasing the number of known bits** influencing power consumption (**more signal**) and **decreasing those unknown** and thus contributing to the noise (**less noise**)
- Moderately effective but not really efficient for DPA; very efficient for a Correlation Power Attack (see later)

Better Leakage Models

- Implicitly we have assumed that there is **correlation between the value** of one signal ('0' or '1') and **the power consumed**
  - d has been used to split traces into buckets
- Maybe this is a bit rough and one could be closer to reality
  - What consumes in CMOS are **transitions**
  - The decision function should instead be the change in a particular signal during the computation
  - $\delta = d \oplus s$, where s was the state of the bit before the transition
  - $\delta = HD(D_{0..n}, S_{0..n})$, where S is the previous state and HD() the Hamming distance
- But what is s or S exactly?
  - If we know something of the implementation (technology), it could be $s = d_{t-1}$ or s = '0' (precharge)
  - Or we can simply assume s = k where k is a constant state
- With more information on the implementation and the technology, one could develop more precise models (but it is **generally unnecessary** for a successful attack)

Selection of Relevant Samples

- Processing a complete trace is very costly
- Use only **relevant information easily extracted with simple power analysis** or some statistical processing
  - E.g., identify where a particular operation takes place and use as a measurement only the maximum power consumption during the execution of the operation



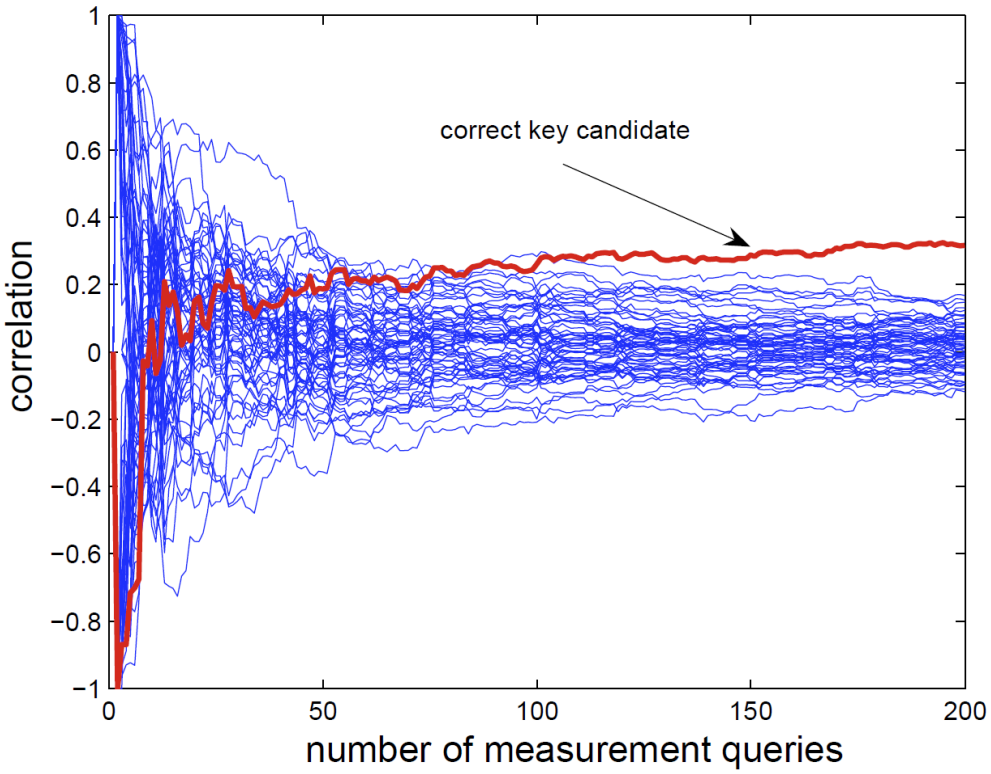
P	R(*L*)
0	1.675
1	1.432
2	1.221
3	1.498
4	1.937

Source: Standaert, 2010

Correlation Power Attacks

- Use a linear correlation factor instead of differences
  - Improves the statistical quality of the decision
- **DPA splits measurements in bins based on the decision function** and checks if the averages of the measurements in the bins differ significantly
- **CPA measures the linear correlation factor** between the **leakage** model (applied to the bits of the decision function) and the **measurements**
  - If correlation factor → 0 when the number of measurements grows, key guess is wrong

Key[0…5]	0	1	**2**	3
corr	-0.09	0.05	**0.32**	-0.11

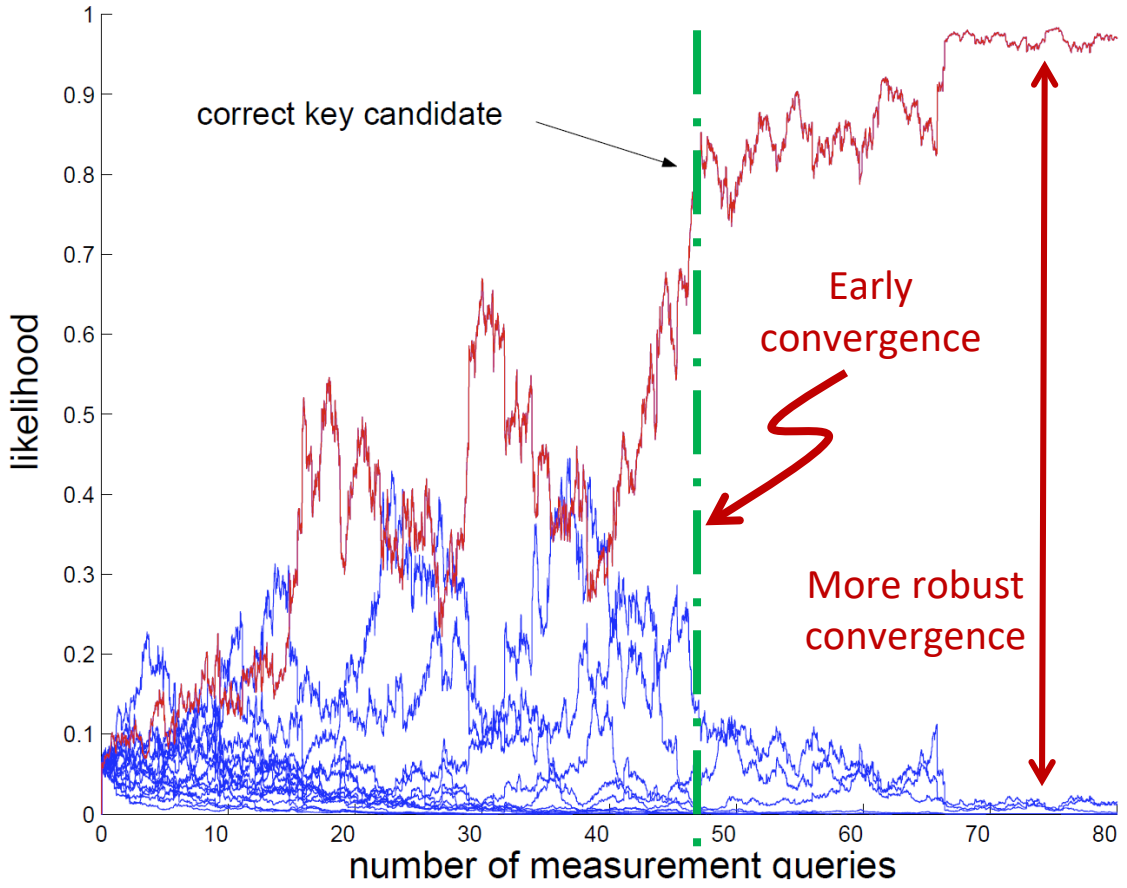


Source: Standaert, 2010

Extremely Powerful Family of Attacks

- **Many improvements possible** on all fronts:
  - Improve measurements and preprocess traces
  - Adaptively select relevant samples
  - Building better statistics, such as higher-order attacks (combining instantaneous power consumption at different times), profiling or template attacks (building knowledge on the device leakage prior to the attack), etc.
  - Better statistical tests
- Overall, **most protections can be circumvented** (at a cost)

Profiled Attack with Bayesian Inference



Source: Standaert, 2010

3

Active Side-Channel Attacks
(Fault Injection)

Fault Injection Attacks

- Semi-invasive attacks which use some mean of **creating a disturbance** on a device to get it **to reveal secrets**
- Somehow attacks based on Rowhammer are fault attacks, but more typical fault injection attacks use physical means to create malfunctions
  - Disturbance to the clock signal or to the power supply
- Usually require a detailed knowledge of the device
  - Which, when, and where to attack?
  - What is the probable effect of the attack?
- Secrets are often revealed by **comparing runs with and without faults**
  - In trivial ways, e.g., by forcing to 0 or 1 a bit of the key and comparing output
  - In very subtle ways, exploiting the structure of the cryptographic algorithms

Fault Injection Methods

- Vary the **supply voltage**
  - Lower the supply voltage → critical path delay exceeds the clock period
  - Generate a voltage spike
- Vary the **clock frequency**
  - Increase the frequency → clock period too short for the critical path
  - Generate a glitch → critical path violated in a specific instance
- **Heat** the device
  - Again, affects the critical path
- Open the VLSI circuit and **shine light on it**
  - Flash → errors all over the place
  - Laser beams → flip individual bits (state, registers)

Fault Injection Countermeasures

- Fairly hard to protect, for creating faults is relatively easy
- Countermeasures not fundamentally different from techniques developed for **fault-tolerant computing** (the only difference is whether the fault is malicious or not)
  - Replication
    - Repeat computation multiple times (temporally or spatially)
    - Then compare (=2) or vote (>=3)
    - Expensive
  - Error detection
    - Parity checks, error correcting-codes, shadow registers
    - Cheaper but may be easier to fool (e.g., with large amount of faults)

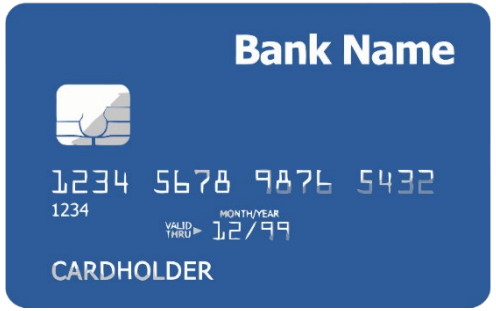
4

Remote Side-Channel and Fault Attacks

Physical Attacks Need Physical Access?

Attacker

Victim

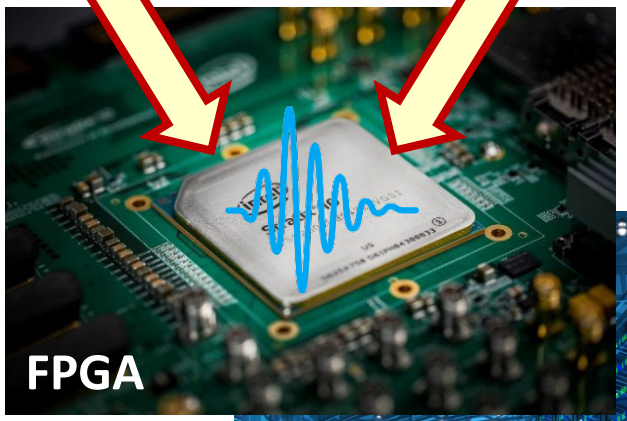


Victim

Accelerator

Attacker

Arbitrary circuitry



How Can an Attacker Measure Power?

- Supply voltage on transistors depends on the current absorbed, because of the finite resistance of the power supply and of the resistance of the power distribution network

$\Delta v(t)$ is proportional to $-i(t)$

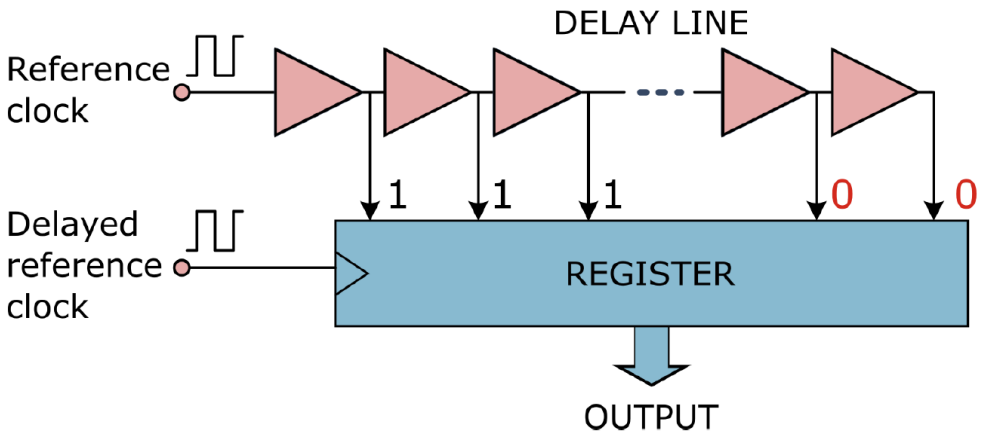
- Combinational delay of gates is approximately inversely proportional to supply voltage

δ is proportional to $1/v(t)$

Attacker needs only to **measure delay**!

Delay-Line Sensors

- Insert a transition in a series of inverters and see, within a fixed time, how far the transition manages to “travel”

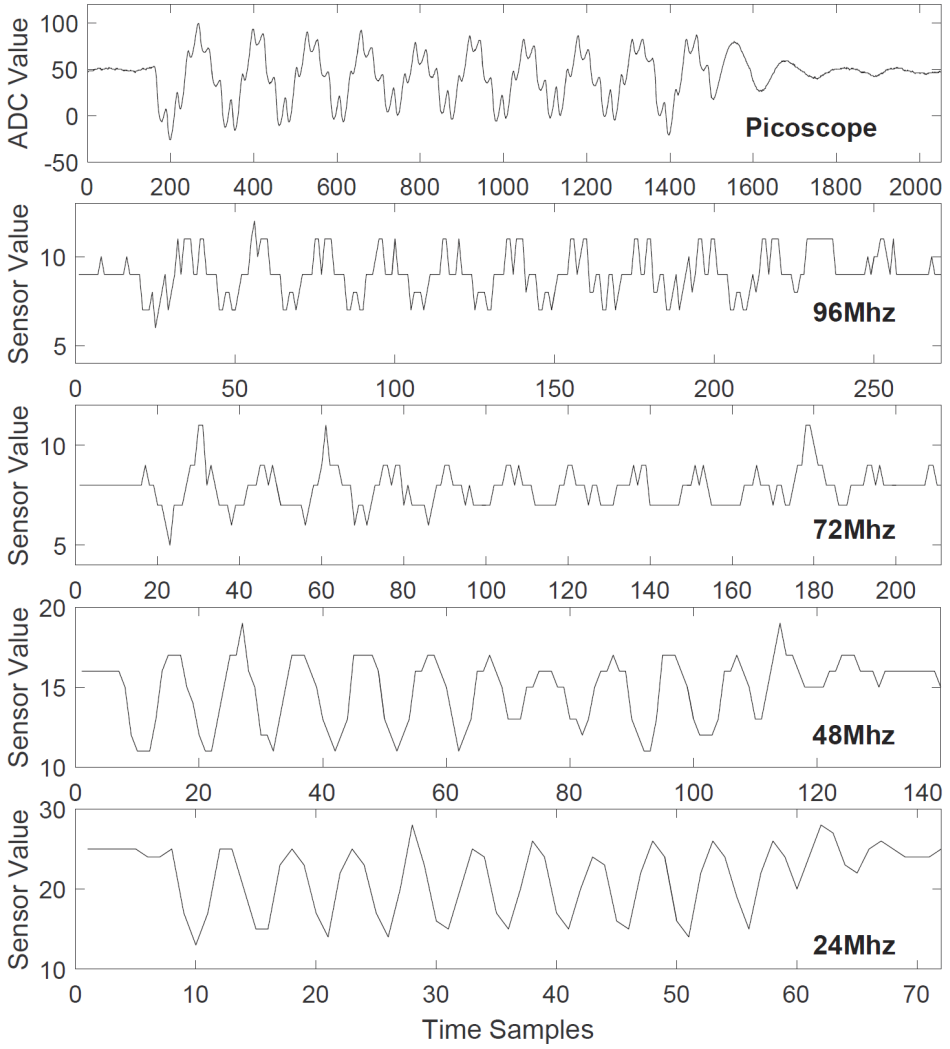


- Use a leading-one counter on OUTPUT: the smaller the result, the larger the instantaneous power consumed
- Plenty of practical issues to solve, but essentially one gets an on-chip digital “oscilloscope”

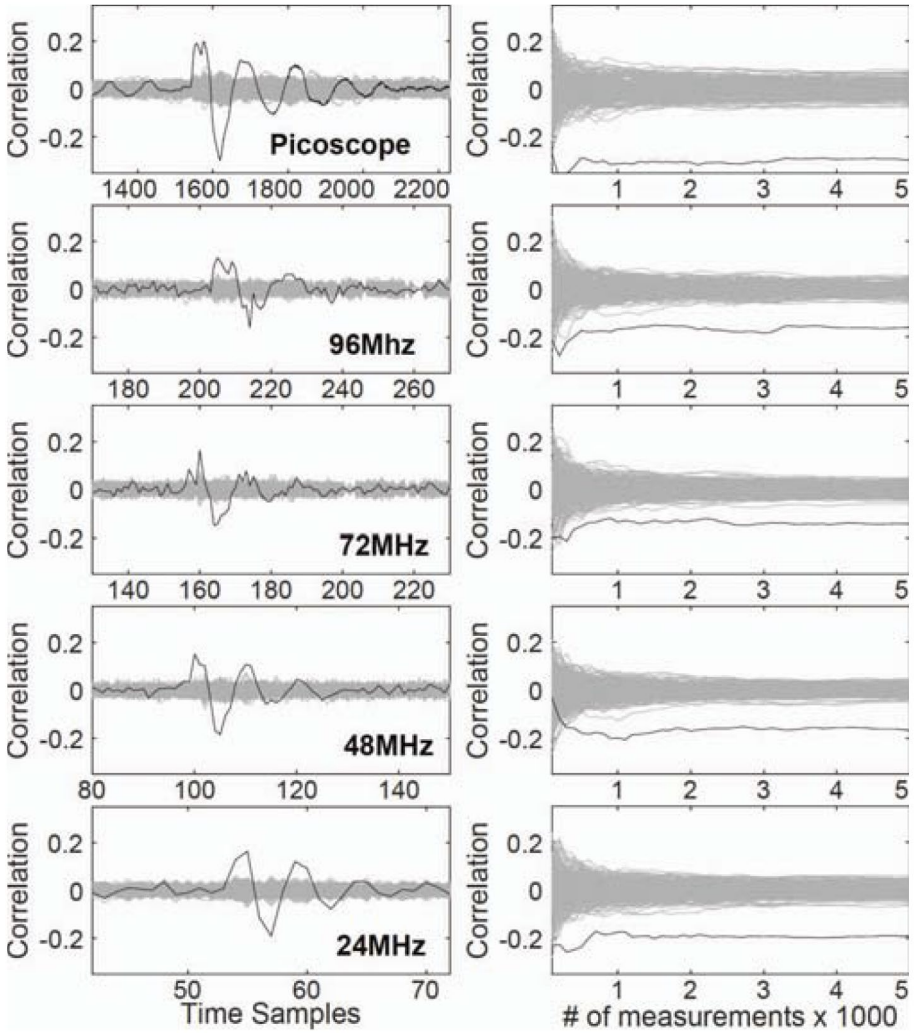
Source: Mirzargar and Stojilović, FPL 2019

Delay-Line Sensors

Measurements



CPA on AES

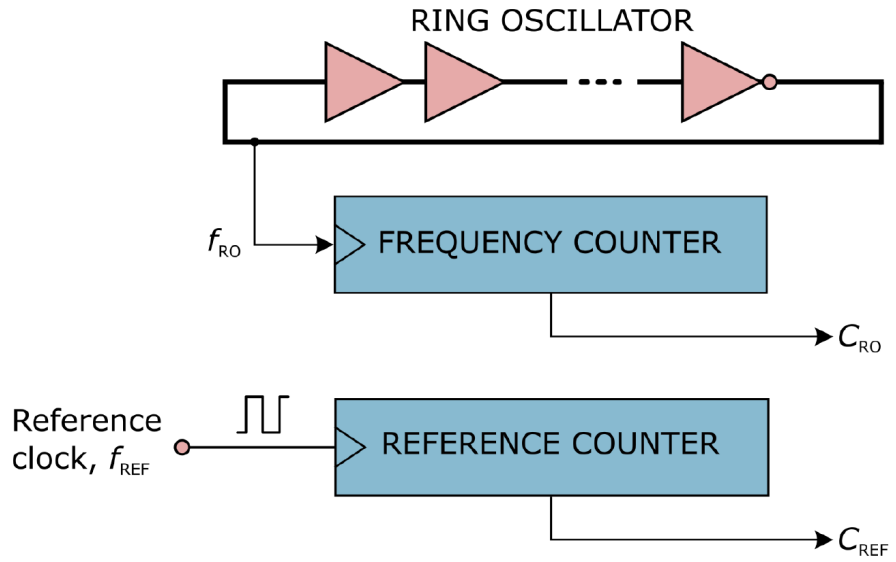


Source: Schellenberg et al., DATE 2018

Ring Oscillator Sensors

- Create a ring oscillator (odd number of inverters in a ring) and measure the frequency by counting transitions and comparing to the transitions in a reference clock

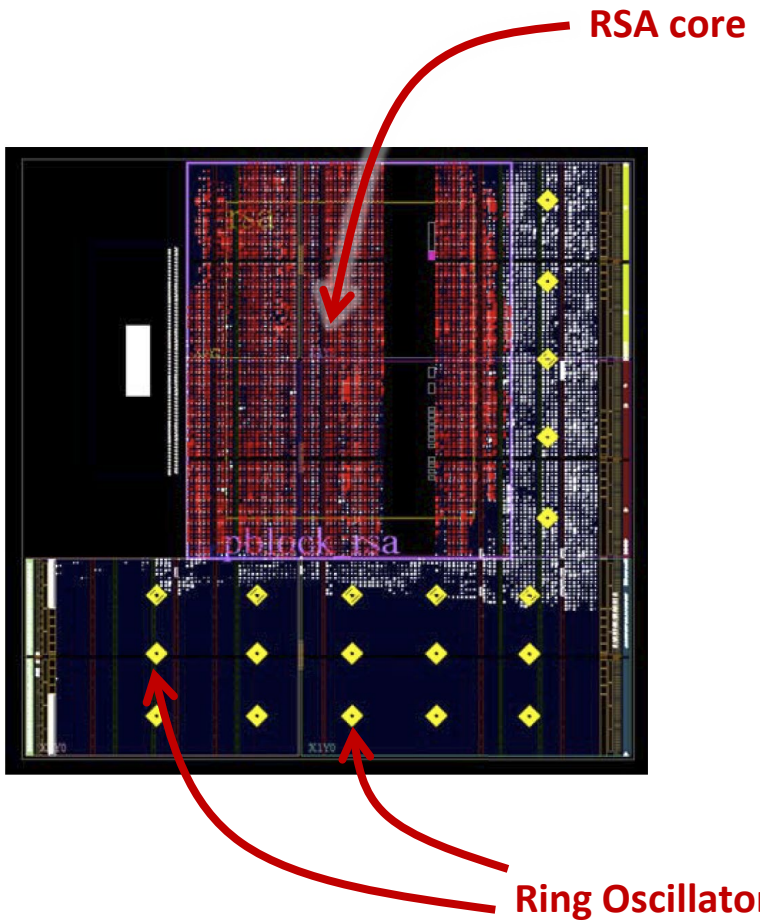
$$C_{RO} / f_{RO} = C_{REF} / f_{REF}$$



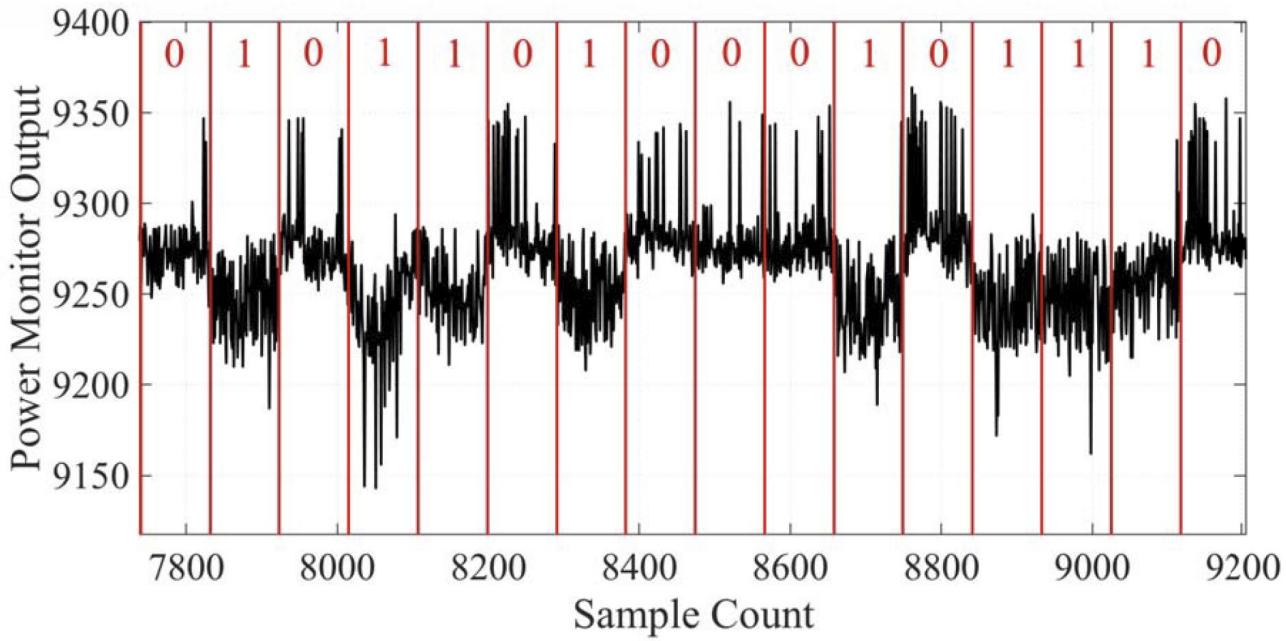
Source: Mirzargar and Stojilović, FPL 2019

- Again, plenty of practical issues to handle properly, but essentially one gets an on-chip digital “oscilloscope”

Ring Oscillator Sensors

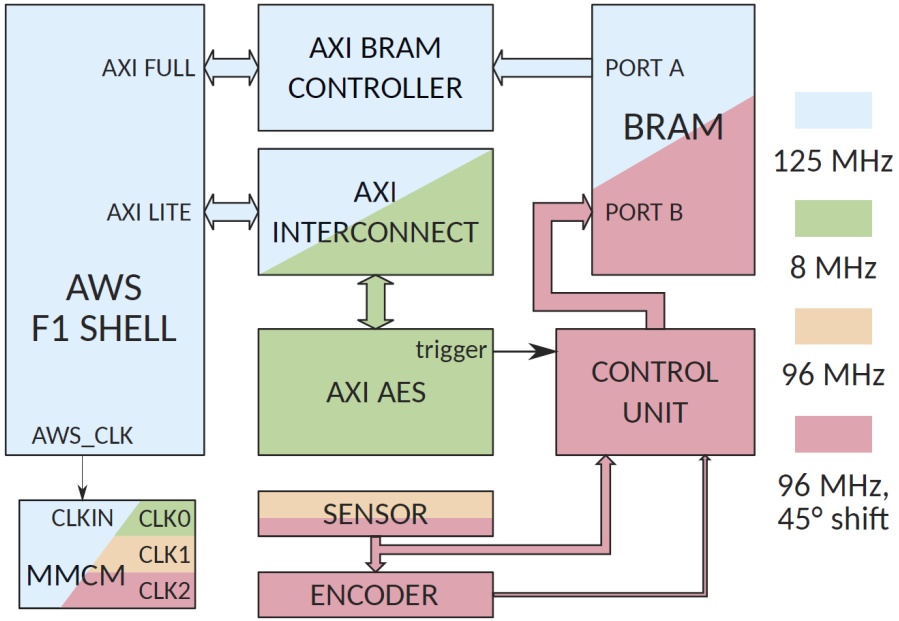


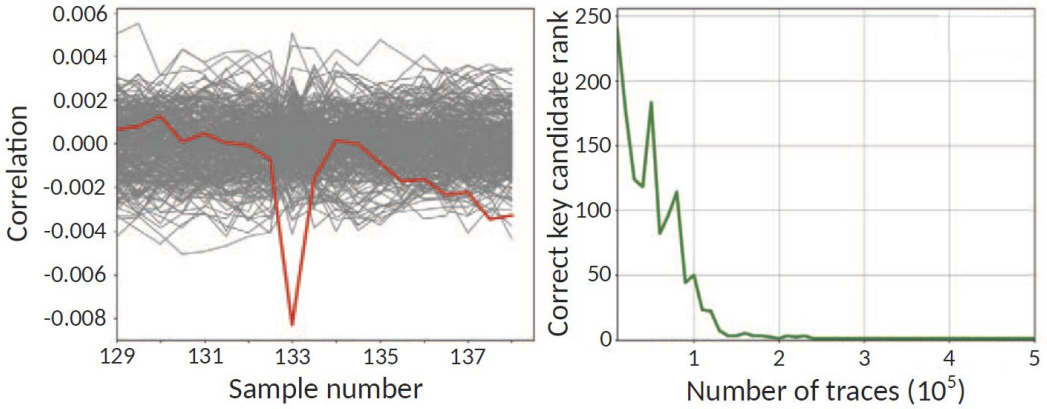
SPA on RSA



Source: Zhao and Suh, IEEE S&P 2018

Attacking an Amazon EC2 F1 Instance





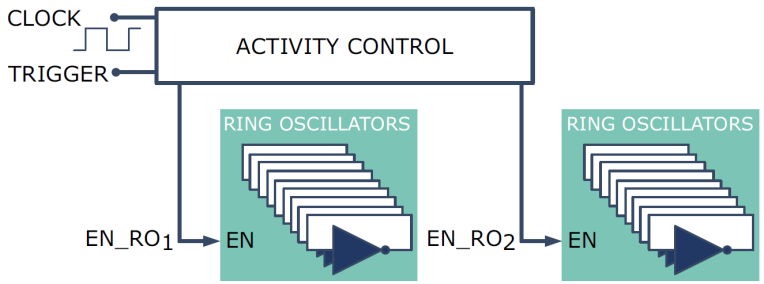
CPA on AES

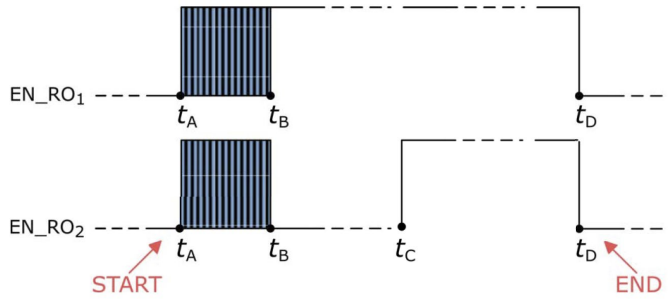
Previous examples are in a controlled environment on commercial test boards, but this is a significant threat also in **real systems** deployed in the cloud and **never reachable physically for the attacker**

Source: Glamočanin et al., DATE 2020

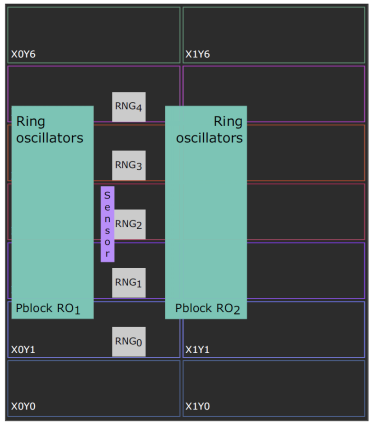
Can One Create Faults Too?

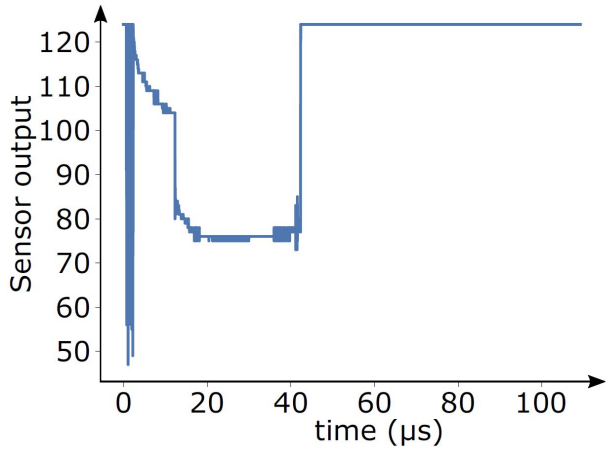
- If drawing current lowers the supply voltage (and thus slows the logic circuits connected), drawing a lot of current may create faults in neighbouring circuits!

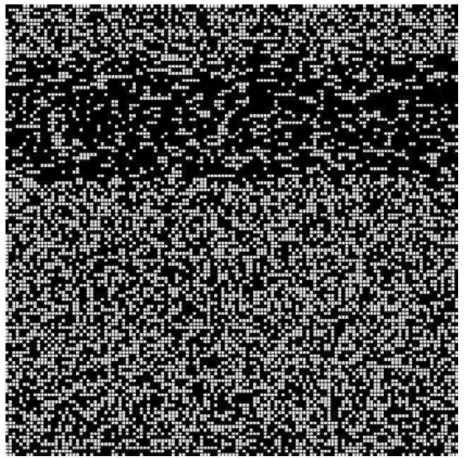




- Force True Random Number Generators to be biased







Source: Mahmoud and Stojilović, DATE 2019

5

Countermeasures to Physical Side-Channel Attacks

Two Major Strategies

- **Physical hiding** ← an engineering approach
  - Perform the same computation an unprotected device would do
  - **Reduce the correlation** between physical emanations (power consumption, electromagnetic field, etc.) and the secret

- **Algorithmic masking** ← a mathematical approach
  - **Randomize the intermediate values** computed in the device
  - Thus, make it harder to correlate physical emanations with secret key hypotheses through intermediate algorithmic values because the latter are now random

Physical Hiding

- Three basic ideas
  - Make the **power** consumption **random**
  - Make the **power** consumption **constant**
  - Increase the noise (**additional random power**)
- They can be applied at **many levels** (but some can only make power consumption random or constant)
  - Software
  - Hardware at architectural level
  - Hardware at logic gate level
  - Hardware at circuit transistor level

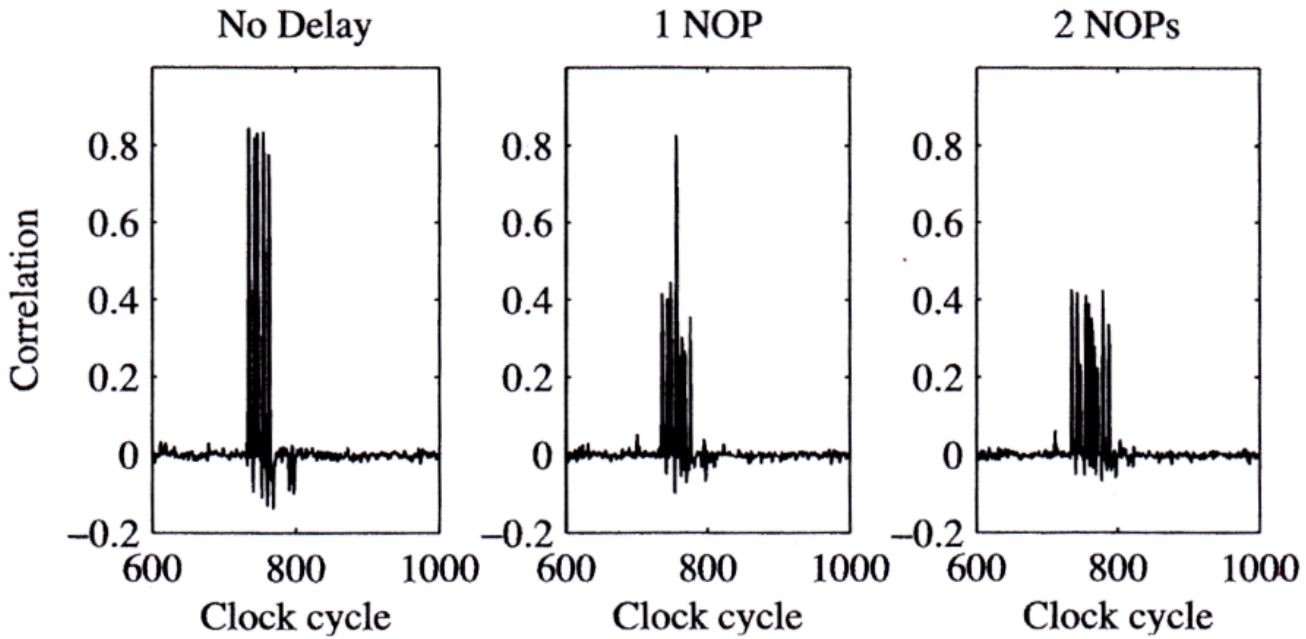
Software Randomization

- Randomize the **instruction sequence**
  - Add random dummy operations in random number during the computation
  - Shuffle the order of operations while preserving semantic equivalence
- Randomly **choose among equivalent implementations** of elementary operations
  - Very much algorithm specific
  - E.g., randomly “undo” optimizations of recursive exponentiation methods for points on an elliptic curve

- Needs good random numbers available
- Mostly useful against SPA or fairly **simple forms of DPA**

Effectiveness of Dummy Operations

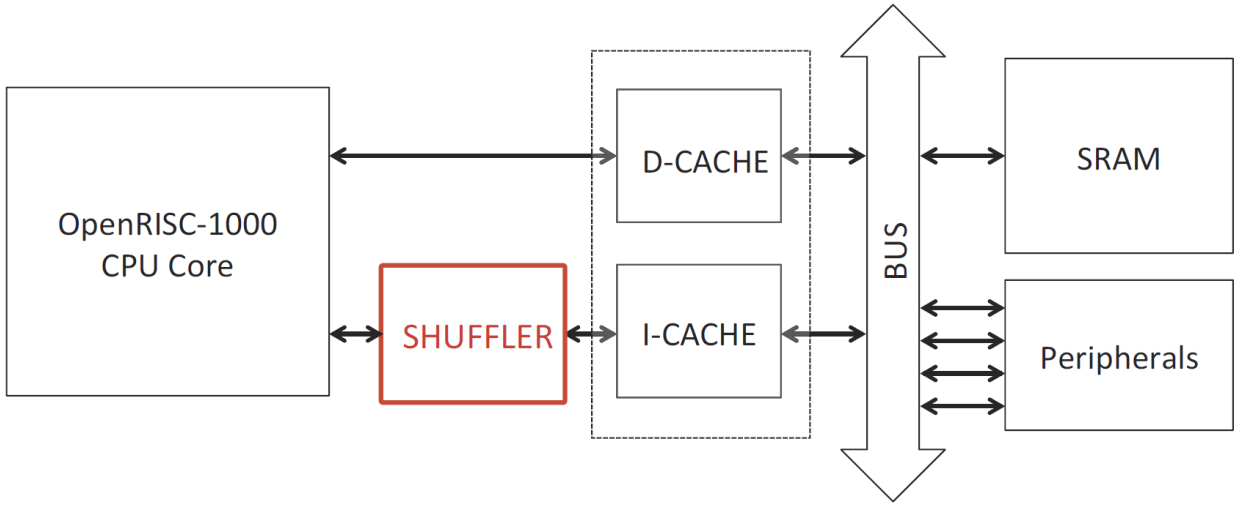
- Correlation **worsens but does not disappear**
- Attacks can be improved to minimize the impact of the countermeasure



Source: Mangard et al., ACM TACO, 2007

Instruction Shuffler

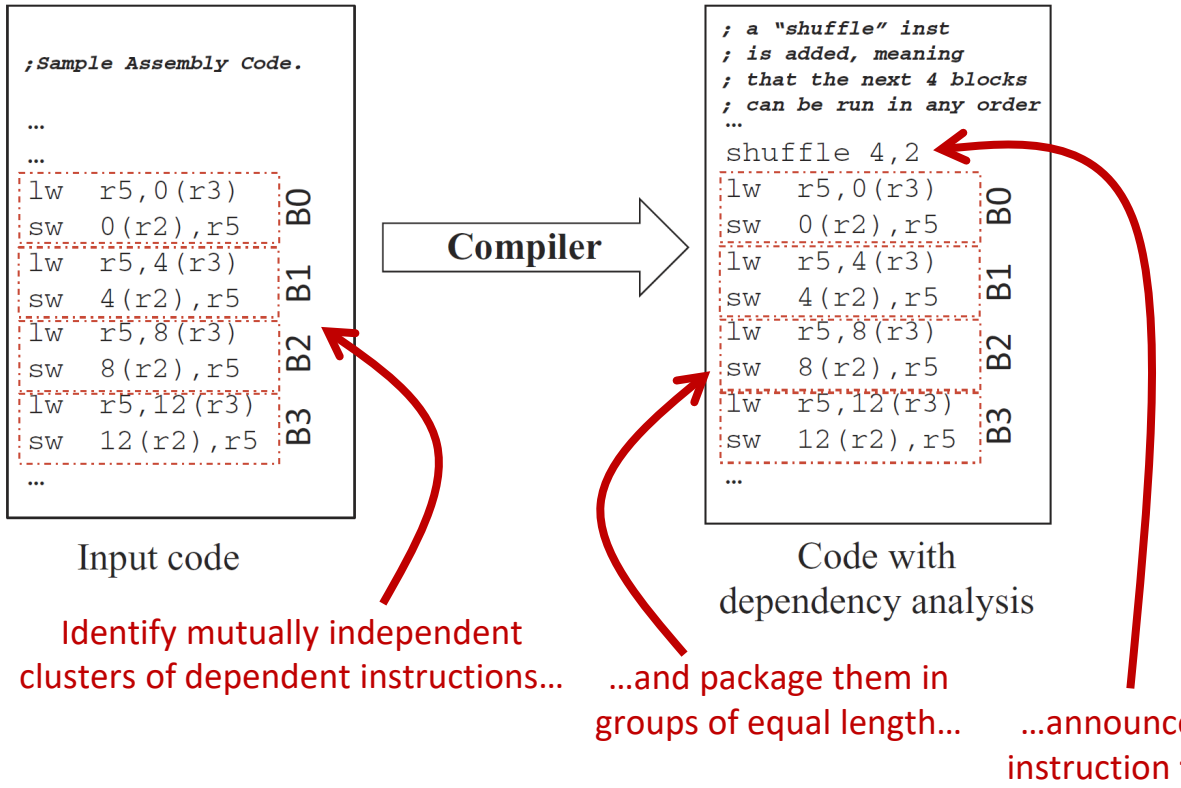
- Prepare the code to know **which instructions can be safely shuffled**
- Have a **shuffler unit supply instructions in random order** to the processor
- **Architecture independent**:
  - The processor has no clue that instructions are being shuffled
  - The shuffler knows nothing of the semantic of the instructions it is shuffling



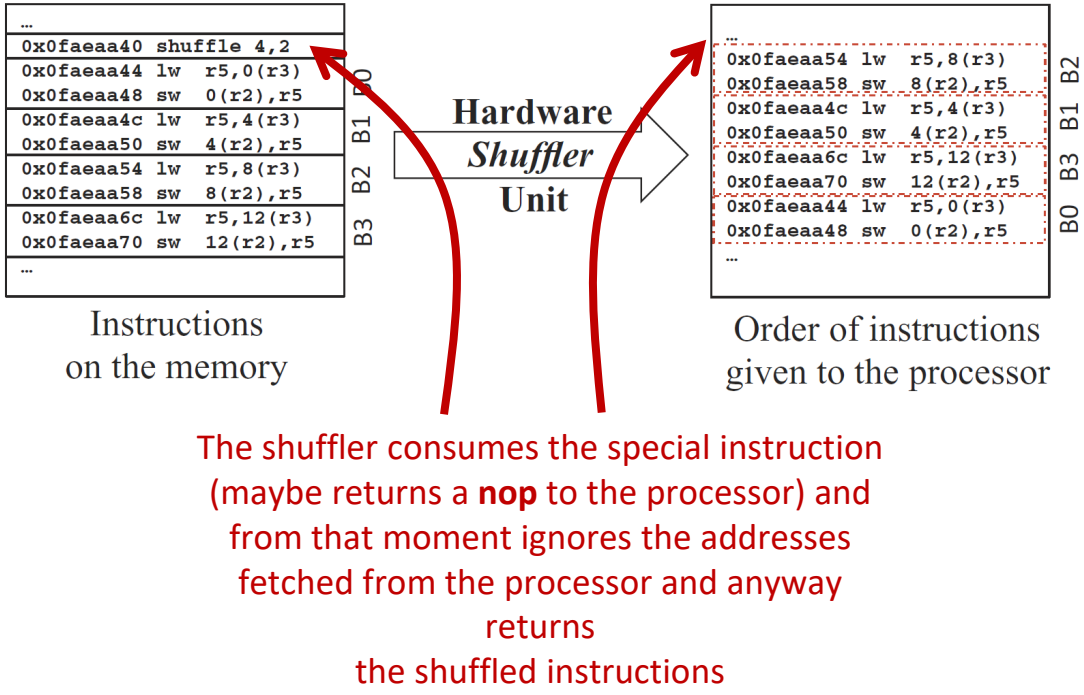
Source: Bayrak et al., ACM TACO, 2012

Static Code Preparation and Dynamic Shuffle

Dependency Analysis
(at compile time)



Order Randomization
(at fetch time, by the hardware shuffler)

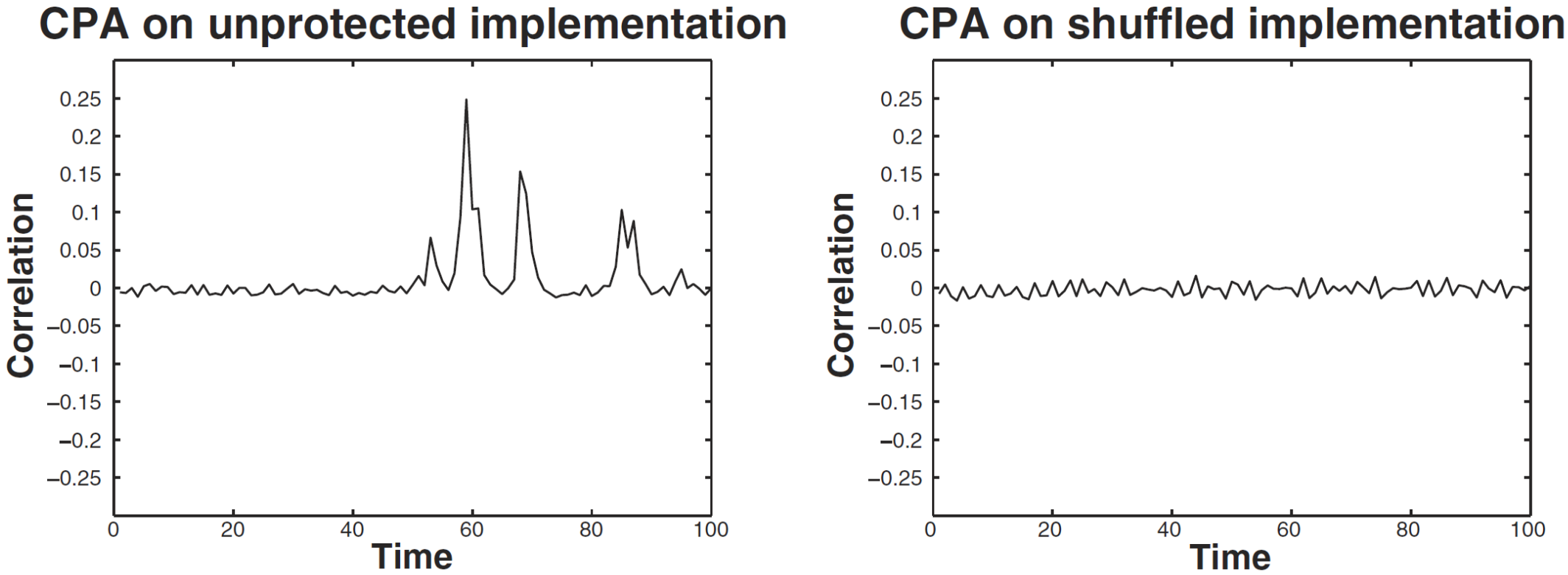


The shuffler consumes the special instruction (maybe returns a **nop** to the processor) and from that moment ignores the addresses fetched from the processor and anyway returns the shuffled instructions

Source: Bayrak et al., ACM TACO, 2012

Shuffling Is an Effective Countermeasure

Attacks on AES-128

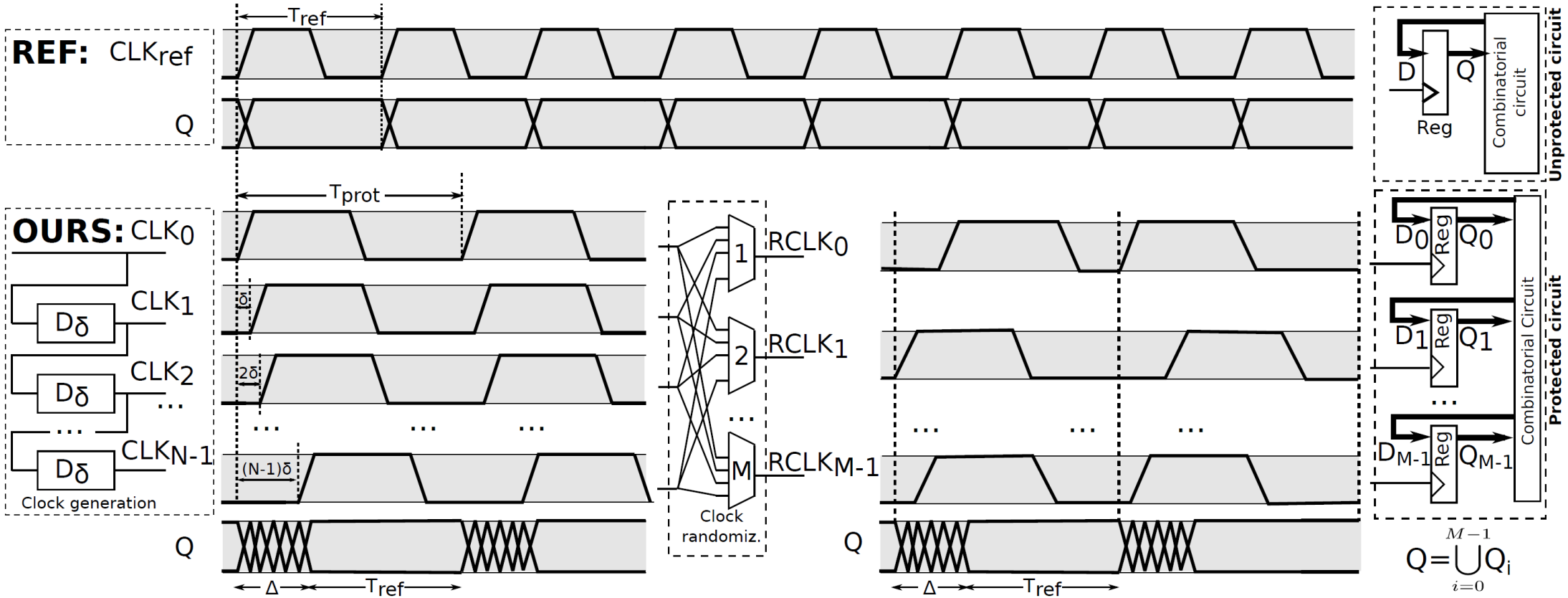


But **preprocessing** of the traces may help…

Source: Bayrak et al., ACM TACO, 2012

Randomizing the Clock(s)

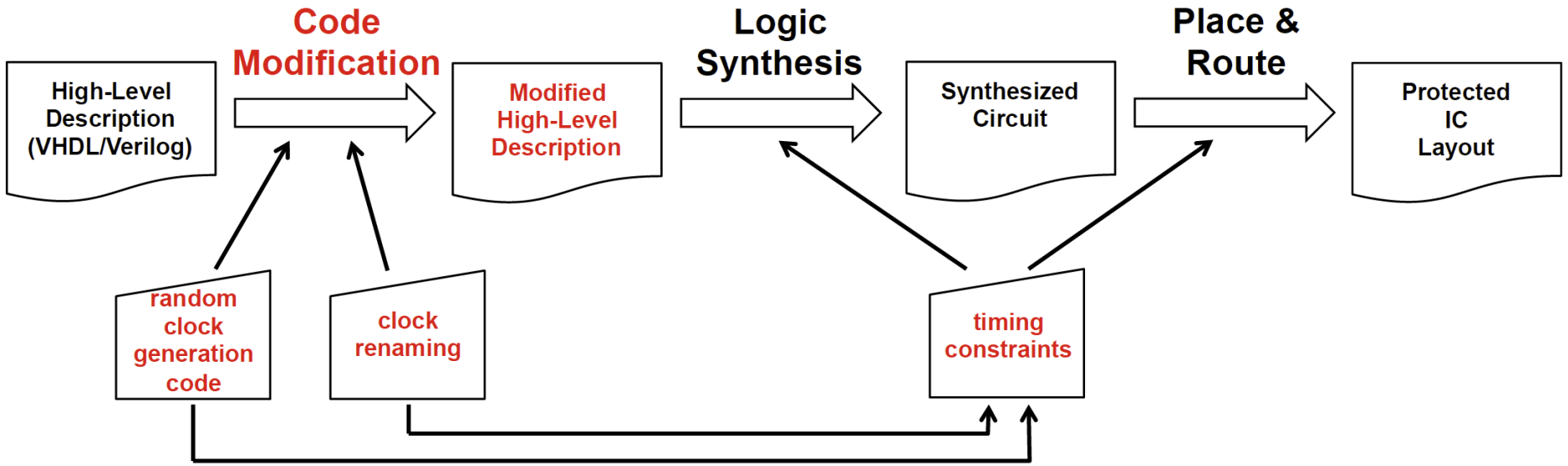
- Apply to the registers of the design **randomly jittered clocks**
- Of course, the critical path must have “space” for the worst jitter (N − 1)δ (i.e., **slower circuit**)



Source: Bayrak et al., DATE 2013

An EDA Tool Friendly Solution

- Jitter exists naturally, so **electronic design tools supports it**
- Of course, jitter introduced in this way is **on a different scale**—but qualitatively it is the same

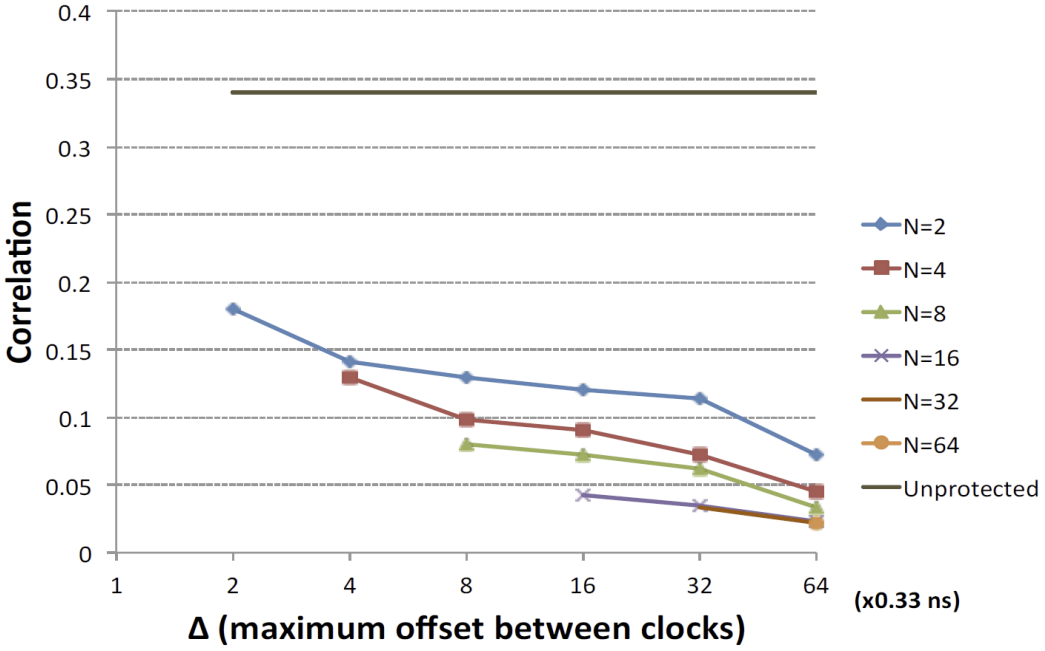


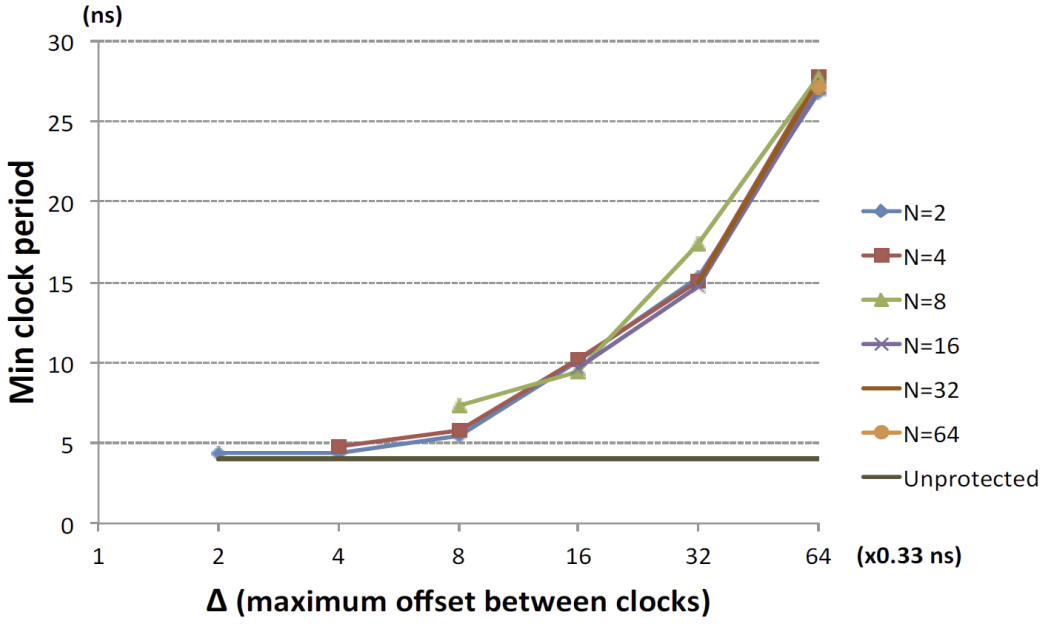
Source: Bayrak et al., DATE 2013

Randomizing Clock is Expensive

- **Impact on timing** is quickly dramatic
- Effective reduction on correlation, but possibly **targeted attacks may break it**, especially for a small N

CPA on AES-128

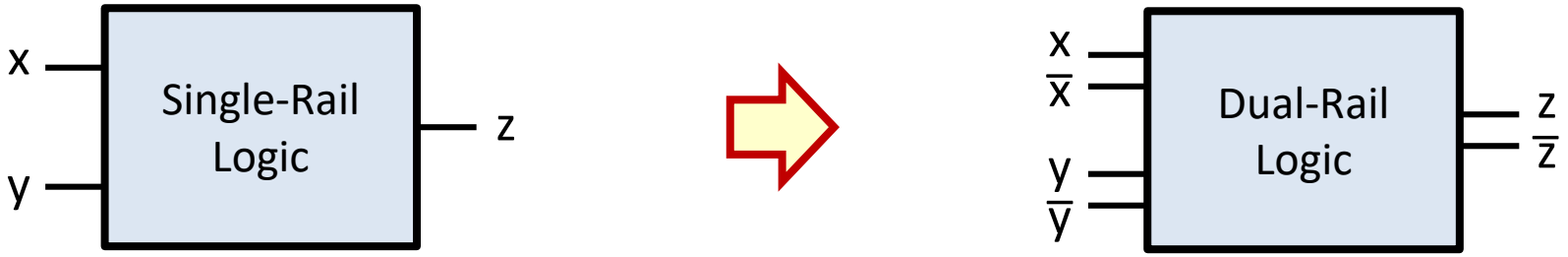




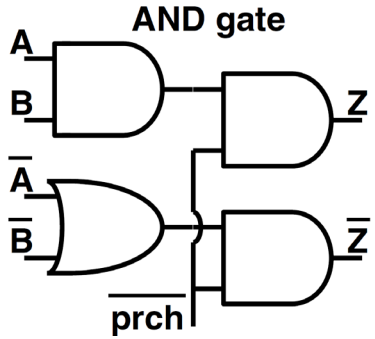
Source: Bayrak et al., DATE 2013

Data-Independent Power Consumption

- Switch from single rail (x) communication to **dual rail** (x and $\overline{x}$)
  - Every transition (0→1 and 1→0) consumes the same



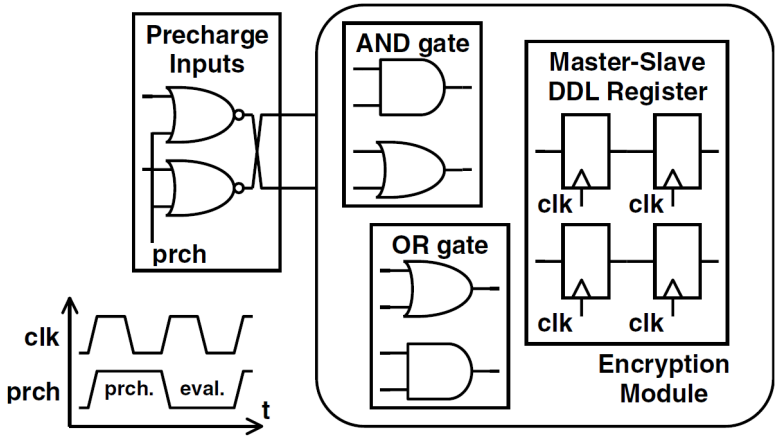
- Add **precharge** to 0 or 1
  - Constant number of switching events, even when no logic transition

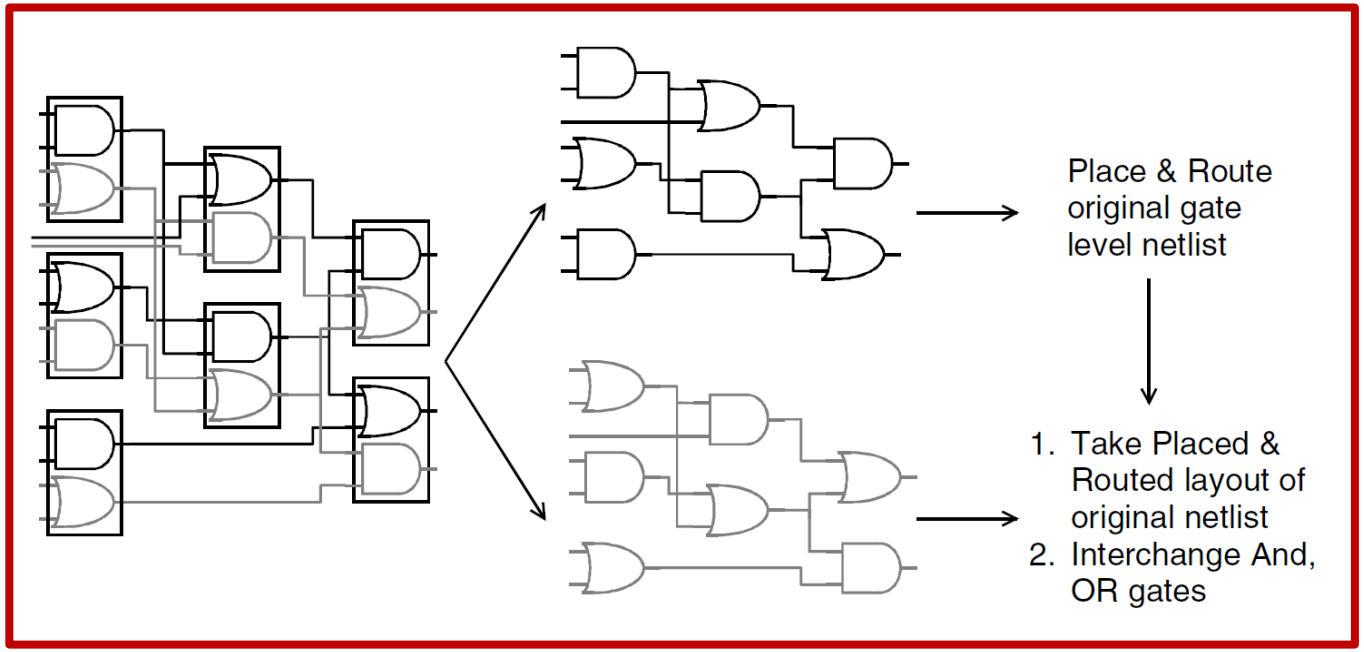


either [0]→0→[0] [0]→1→[0] or [0]→1→[0] [0]→0→[0]

Wave Dynamic Differential Logic

- Create **complementary** circuit with precharge
- Place and route the complementary parts **identically** (so that the parasitic capacitances are as close as possible)
- The price to pay is more **area**, a slower **timing**, and more **energy** consumption

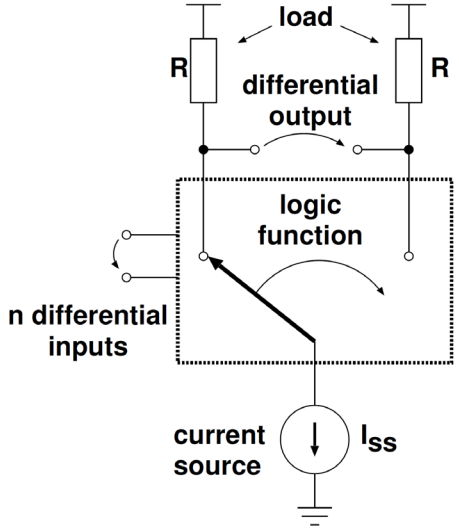


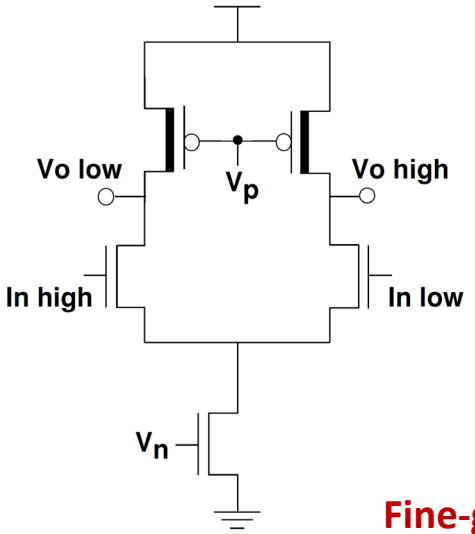


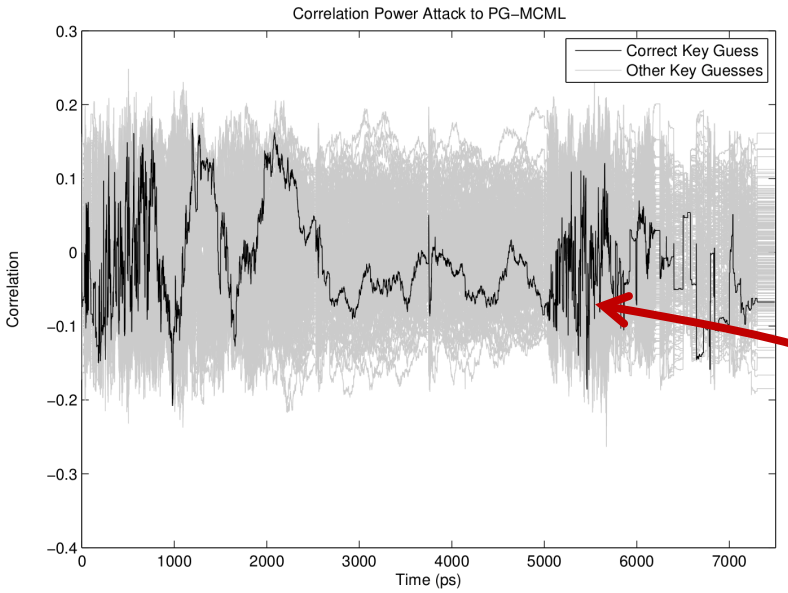
Source: Tiri and Verbauwhede, DATE 2004

Power-Gated MOS Current Mode Logic

- The same idea at **circuit level**
  - Current-mode **differential logic**
- Problem: **static power consumption**
  - Added a power gating transistor
- Create a library of **standard cells** and trick EDA tools to route differential signals together

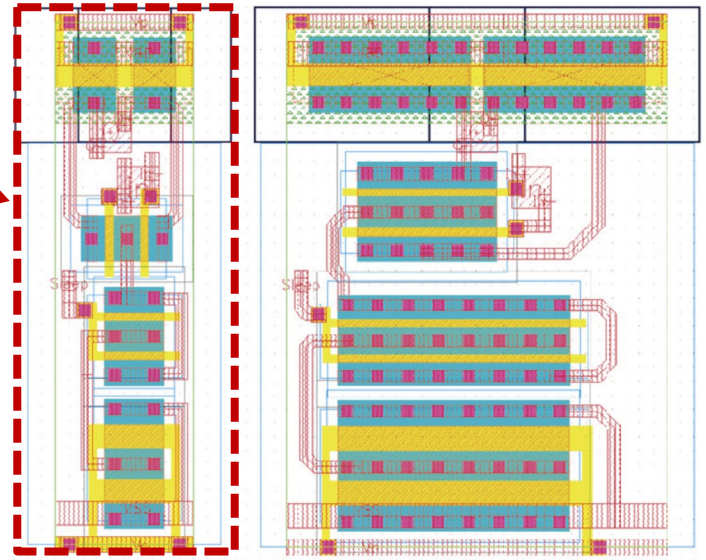




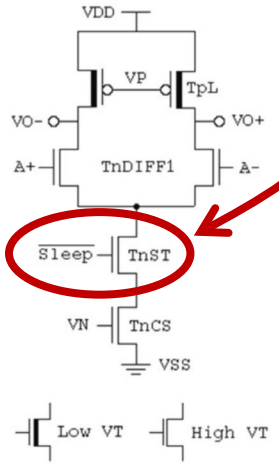


Correct key is indistinguishable

Standard cell



Fine-grain power gating



Source: Cevrero et al., DAC 2011

Two Major Strategies

- **Physical hiding** ← an engineering approach
  - Perform the same computation an unprotected device would do
  - **Reduce the correlation** between physical emanations (power consumption, electromagnetic field, etc.) and the secret

- **Algorithmic masking** ← a mathematical approach
  - **Randomize the intermediate values** computed in the device
  - Thus, make it harder to correlate physical emanations with secret key hypotheses through intermediate algorithmic values because the latter are now random

Masking Idea

- Attacks are based on the correlation between an attacker hypothesis and the corresponding value of an internal signal (responsible for a physical emanation)
- Change the computation of the cryptographic algorithm to **make all internal signals random**
  - The final result of the computation must be the same
  - **Apply a mask to the inputs** $p_i' = f(p, m_i)$ where m_i is a random variable generated internally, *f* an appropriate function, and p may represent the plaintext and/or the key
  - Internal signals are now $s' = g(s, m_0,\ldots, m_n)$ and, due to the random masks $m_0..m_n$, **cannot be computed, for a given hypothesis, by the attacker**
  - **Remove masks from the outputs** $c_j = h(c_j', m_0,\ldots, m_n)$ where *h* is an appropriate function
- The challenge is to keep track of how the masks propagate through the algorithm and to be able to "remove" them from the final result

The Easy Case for Masking

- Suppose that a cryptographic algorithm is **linear** (which of course is not—or at least not completely)
- If it **were** linear:

$$F(x \oplus m, k) = F(x, k) \oplus F(m, k)$$

- One could implement **Boolean masking**
  - Mask the input (e.g., plaintext) with a random mask m: $x' = x \oplus m$
  - Compute F(x', k) which cannot be attacked because x' is unknown (random)
  - Compute F(m, k) which cannot be attacked because m is unknown (random)
  - Produce the output (e.g., ciphertext) F(x, k) as $F(x', k) \oplus F(m, k)$

But the World is Not (Completely) Linear

- Example: AES
  - AddRoundKey: Linear
  - SubBytes: **Not Linear**
  - ShiftRows: Linear
  - MixColumns: Linear
- For the **linear operations**, it is just a question of keeping efficiently track of the masks and **making sure they do not cancel out**
  - Suppose that $a' = a \oplus m$, $b' = b \oplus m$ and the algorithm computes $x = a' \oplus b'$
  - Then $x = a' \oplus b' = a \oplus m \oplus b \oplus m = a \oplus b$ is **not masked** and leaks information
  - One should have used **two masks** $a'' = a \oplus m_1$, $b'' = b \oplus m_2$
  - Then $x = a'' \oplus b'' = a \oplus b \oplus m_1 \oplus m_2$ which is still masked

Nonlinear Operations?

- The **AES S-box is nonlinear**, for instance
- A couple of complex options:
  - Implementing the S-box as a table T(x)
    - Precompute **masked tables** $T_m(x \oplus m) = T(x) \oplus m$
    - **Plenty of tables** need to be computed and stored for all used masks
  - Implementing the S-box mathematically through the multiplicative inverse of a finite-field element
    - **Arithmetic masking** is possible because

$$(a \times m)^{-1} = a^{-1} \times m^{-1}$$

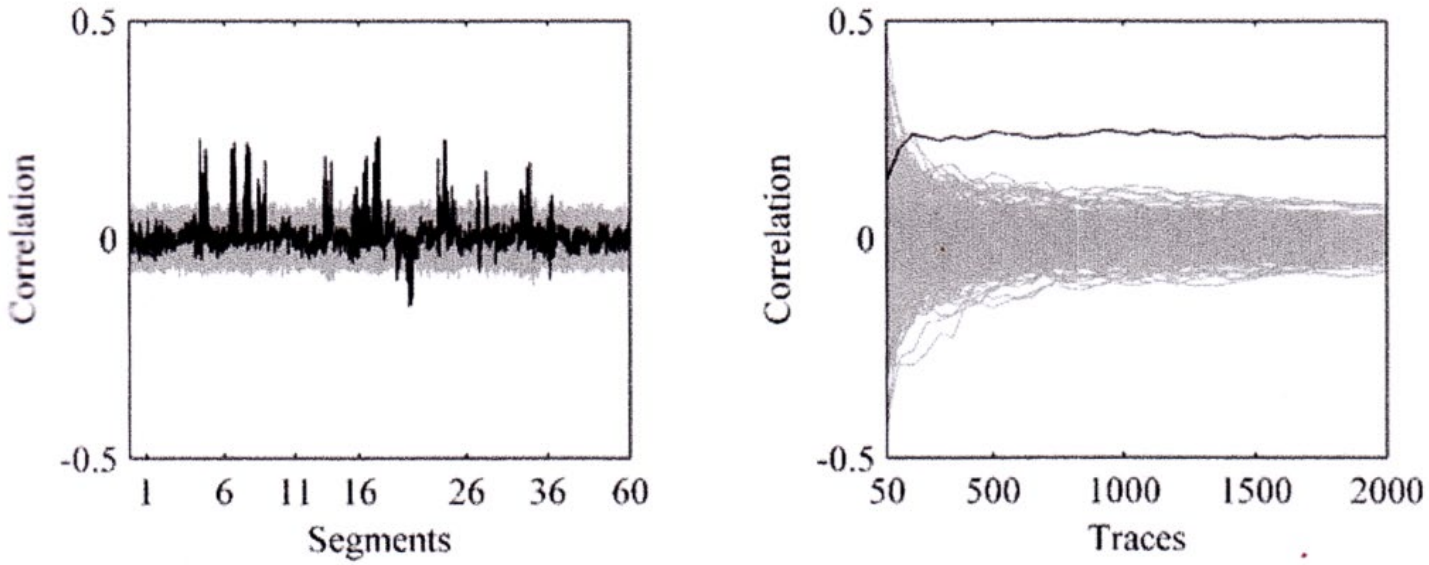
    - How to switch efficiently from Boolean masking to arithmetic masking, etc.

Does Masking Work?

- Perfectly:
  - **Theoretical security** of masking against first-order DPA attacks
- But!
  - One can perform **second-order DPA attacks**
    - Choose two internal values u and v protected as $u_m = u \oplus m$ and $v_m = v \oplus m$
    - Extracting the power samples $p(u_m)$ and $p(v_m)$ of the traces correlated to $HW(u_m)$ and $HW(v_m)$, where HW() is the Hamming weight, is useless for DPA, because we do not know u_m and v_m
    - Instead, we attack an hypothetical internal variable $w_m = u_m \oplus v_m = u \oplus v$ which is unprotected
    - Since w_m does not exist, no point of the trace is directly correlated to $HW(w_m)$
    - Yet, one can show that $|HW(a) - HW(b)|$ correlates well to $HW(a \oplus b)$
    - We use this to manufacture an artificial power sample $p(w_m) = |p(u_m) - p(v_m)|$ from the samples $p(u_m)$ and $p(v_m)$ targeting u_m and v_m
    - Sample $p(w_m)$ can be used to mount an attack on $w_m = u_m \oplus v_m = u \oplus v$ which we can compute

So, Does Masking Work? Only to an Extent…

- Example of a **second-order correlation attack on a masked AES** implementation in software:



- The rest of the "story" is fairly intuitive:
  - One can implement **N^{th}-order masking** where each intermediate value is masked with **N random variables**
  - But **$(N + 1)^{th}$-order DPA attacks** are theoretically effective against **N^{th}-order masking**

Source: Mangard et al., ACM TACO, 2007

Conclusions

- Physical side-channel attacks are probably the **most elusive form of security threat** for cryptographic devices
- There is a **variety of countermeasures** (both from a mathematical and from an engineering perspectives)
- Countermeasures are all **quite expensive** and none removes the possibility of an attack, they **only mitigate the security threats**
- It looks like in practice what works best is the **implementation of many simple countermeasures** at once
- Today, side-channel attacks are a **key threat** to some particular embedded products (e.g., **smartcards**) and not yet of classic computing systems (e.g., datacentres, laptops, smartphones)—but this may change…

References

General

- F.-X. Standaert, *Introduction to Side-Channel Attacks*, in Secure Integrated Circuits and Systems, Springer, 2010
- P. Kocher, J. Jaffe, B. Jun, *Differential Power Analysis*, Crypto, 1999

A Classic Book

- S. Mangard, E. Oswald, Th. Popp, *Power Analysis Attacks*, Springer, 2007