POLITECNICO DI MILANO

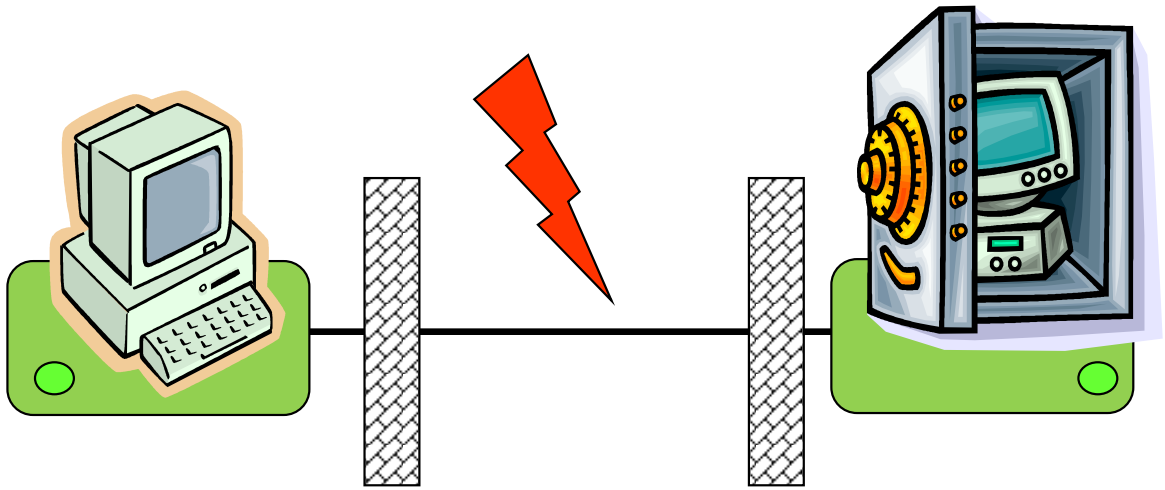
Mostly Harmless: engineering side channel attack safe cryptography

Praticamente resistente: realizzare crittografia protetta da attacchi *side channel*

Gerardo Pelosi

11 September 2018

Traditional threat model



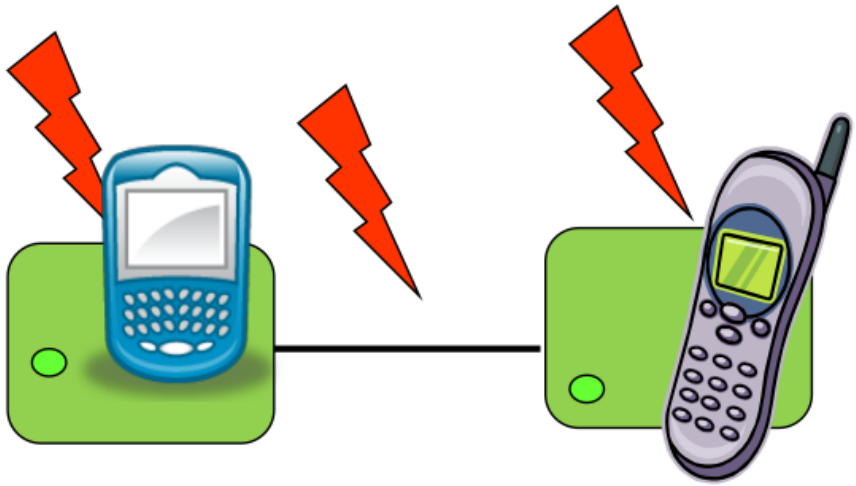
- Attack **on channel** between communicating parties
- Encryption and cryptographic operations in **black** boxes
- Protection by strong mathematic algorithms and protocols
- Computationally secure

Embedded Cryptographic Devices

Identification **Payment** **Communication** **Multimedia**

- The adversary can easily obtain full physical access
- It is possible to (temporarily) seize the device
- The adversary can be a legitimate user
- The adversary may have knowledge of every detail concerning the hw/sw configuration of the device
- The adversary can have a perfectly functional clone of the device (except for the value of the secret key)

Enhanced threat model



- Attack on channel and endpoints
- Encryption and cryptographic operations in gray boxes
- Protection by strong mathematic algorithms and protocols
- Protection by keeping into account the implementation details
- Computationally secure

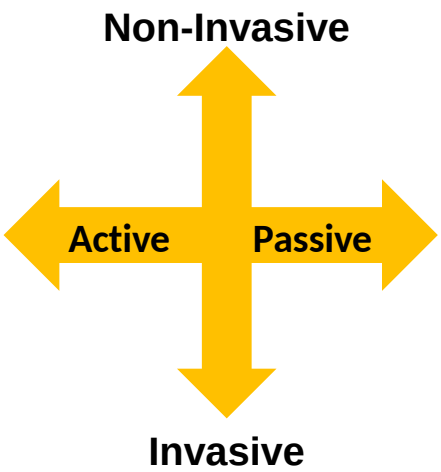
Security of a Cryptosystem

Weakest Link Principle

A chain is only as strong as its weakest link.
The security of a hardware/software component is as secure as any part of the design and deployment process, including persons and technologies.

- The adversary will go for the weakest entry point
  - Disable or go around security mechanisms
  - Guess / spy on passwords (Social engineering)
  - Bribe the security guard
- If you use cryptography, the adversary will try to go around it
  - System designer: thinks of the "right" way to use the system (...to optimize performances and costs)
  - Adversary: does not play by the rules
- Cryptographers has to think like the adversary, anticipate/model (known) attacks, protect crypto-schemes against them

A Taxonomy of Implementation Attacks



- **Passive**: observe and infer
- **Active**: perturbate and conclude

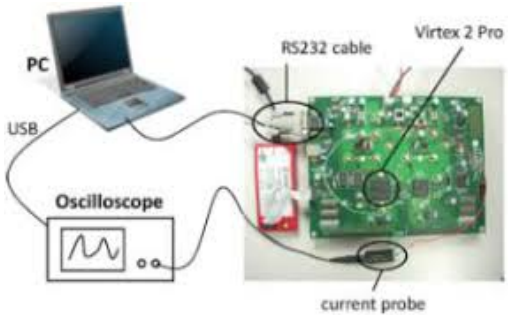
- **Semi-Invasive**: open package, no modif.
- **Invasive**: open package, permanent modifications to the chip

- **Side channels**: passive e usually non-invasive
- **Fault injection**: active attacks, different degrees of invasion
- **Circuit modification**: active and invasive (highly costly threat model)

Passive Side-Channels

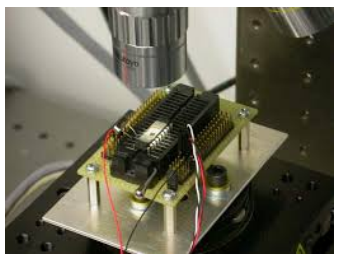
- Passive - Non invasive
  - Timing
    - Overall or partial execution time

  - Power, Electromagnetic (EM) radiation
    - widespread CMOS technology
    - Dynamic power consumption (input data dependent)
    - Electric current (input data dependent) induces an EM field



  - Proven to be a practical threat
    - Sound, Temperature

- Passive - Invasive: optical emission analysis or micro-probing buses/single-cells with very thin needles



The old *Divide et Impera* Principle

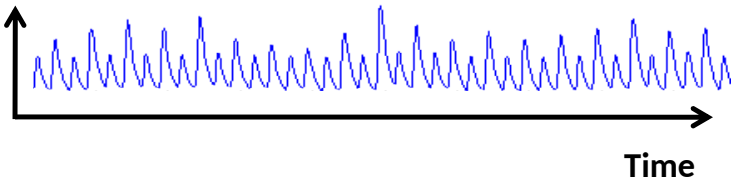
- Breaking into a safe is hard, because one has to solve a single, very hard problem...
- Things are different if it is possible to solve many small problems instead...

- A locksmith can manually manipulate the lock to obtain the combination one number at a time
- He manipulates the safe lock with audible and tactile (through the dial) feedbacks he guesses the internal movements of the lock mechanism

Power Analysis

Measuring Power Consumption

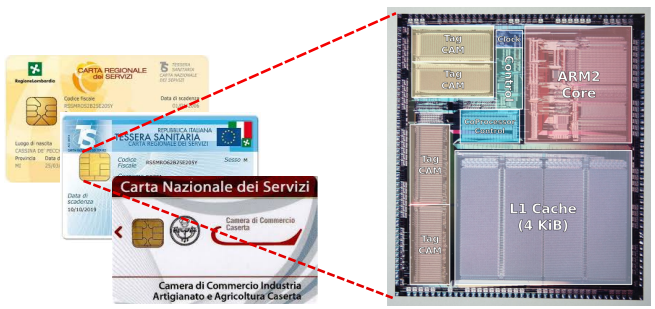
- Do not consider average/peak power consumption
- Record of the instantaneous power over time throughout the execution of the crypto-device

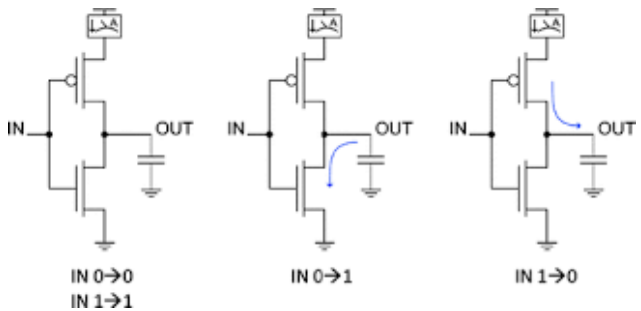


- (Automated) low-cost off-the-shelf measurement setup

Principles

- Constant Supply Voltage – supply current varies
- CMOS is the prominent technology: power consumption depends on input





Power Analysis

- **What can we see looking at a record of the instantaneous power consumption over time (*trace*)?**
- Unintended information leaks from:
  - repetitive patterns: comes from the structure of the cryptographic algorithm and implementation (e.g., loops)
  - Given the control flow of the algorithm/implementation, the trace allows to spot (at least coarsely) what happens when
  - the data flow of the algorithm/implementation leaks from the the amplitude of the recorded signal, allowing to infer on which operand values a repeated operation has been executed
    - different operand values consumes more or less power

Simple Power Analysis (I)

- Visual inspection of a few traces, worst/best case: single shot
- Often exploits direct key dependencies
- Input/output not need to be known, but useful for verification
- Require: expertise, experience, detailed knowledge about target device and implementation

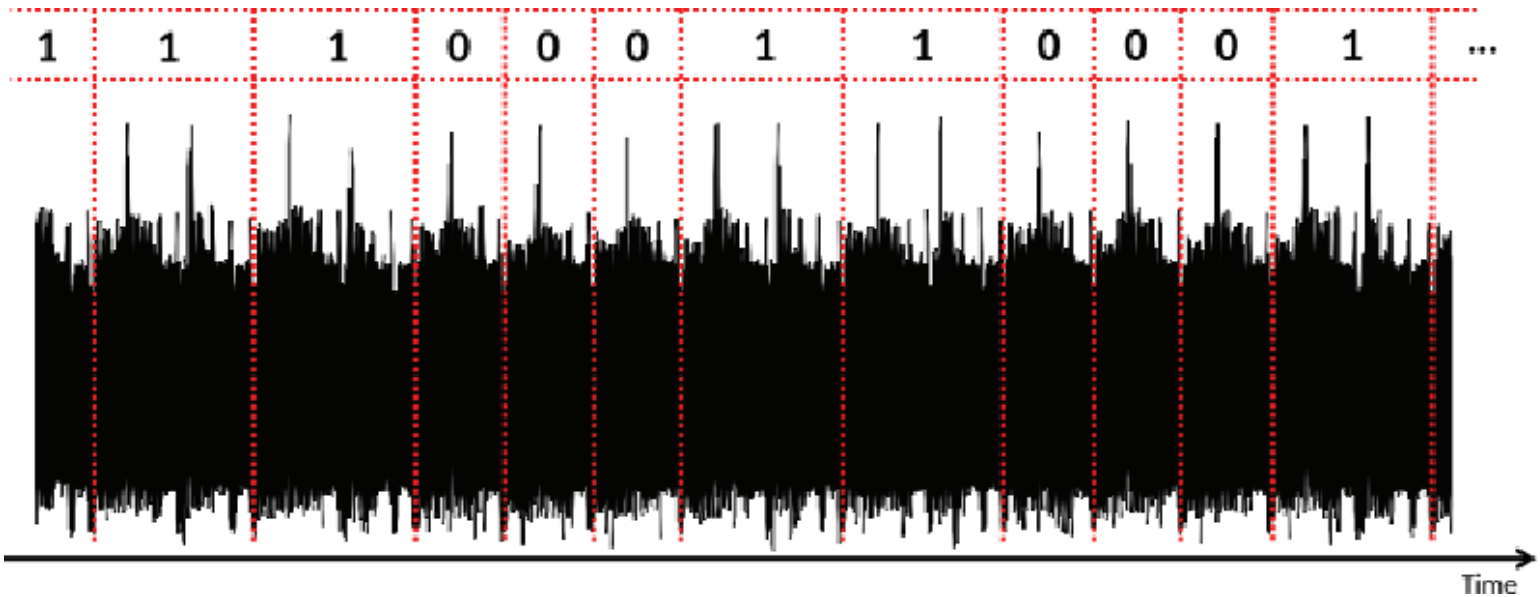
Simple Power Analysis (II)

ECC POINT MULTIPLICATION
(left-to-right binary method)

INPUT: $k=(k_{t-1},\ldots,k_0)_2$, $\underline{P} \in E(F_q)$
OUTPUT: $\underline{Q} = k\underline{P}$

```
Q ← ∞
FOR i=t-1 TO 0
    Q ← 2Q              ; point doubling
    IF k_i == 1
        Q ← Q + P       ; point addition
RETURN Q
```

- Conditional, key-dependent operation
- Different algorithms to compute point addition and point doubling
- Software Implementation on a μC

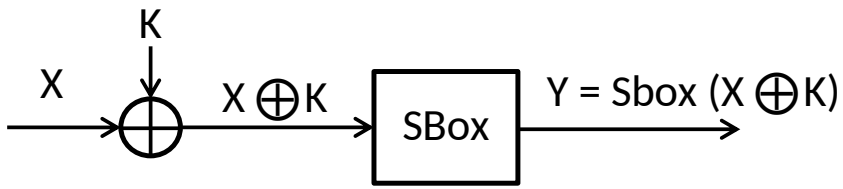


Differential Power Analysis - DPA (I)

- Differential side-channel analysis allows an adversary to confirm or reject a hypothesis about an intermediate state of an implementation
- If this allows to deduce the value of secret data, the analysis becomes an attack
- Three disciplines
  1. Cryptanalysis: target a sensitive intermediate state for which exhaustive key search is easy
  2. Engineering: access to side-channel leakage
  3. Statistics: an "oracle" to verify key hypotheses

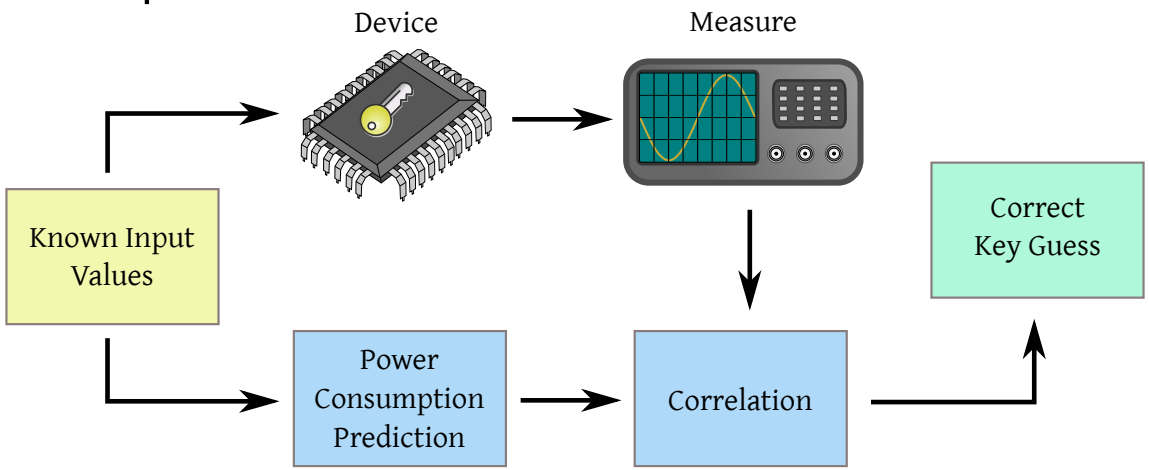
Differential Power Analysis - DPA (II)

- A chip implements an AES-128 encryption function, without power analysis protection ($\texttt{ciphertext} = \texttt{AES} - \texttt{Enc}(\texttt{plaintext}, \texttt{key})$)
- Choose an intermediate computation as "good target" $\mathrm{Y} = \mathrm{f}(\mathrm{X}, \mathrm{K})$, with an unknown key portion K (e.g., 1 byte)



- Repeat *n* times the measurement of the power consumption over the running time of the AES with *n* different plaintexts
- For each value of the key portion K, compute an hypothetical instantaneous power consumption for Y

The Hp showing the highest correlation with the measures points out the correct guess for the the key portion K



Countermeasures against DPAs

HW & SW countermeasures against power analysis tradeoff performances and attack effectiveness. Focusing on SW ones:

masking: invalidates the link between the predicted hypothetical power consumption values and the actual measured ones

- the processing of a sensitive intermediate value is concealed through splitting it in a number of shares (each concealed through a random value) and properly recombining them

hiding: conceals the time at which a sensitive operation is executed on a per-run basis

- execution flow randomization via shuffling the order of some instructions (f.i., changing the order of the accesses to lookup tables)
- insertion of random delays between atomic steps of the algorithm

Countermeasures against DPAs (I)- [PoliMI]

Code Morphing Approach

Key Idea

Dynamically replace code fragments with semantically equivalent ones, differing in the operations employed to perform the computation,
e.g.: $\texttt{r0} \veebar \texttt{r1} \Longleftrightarrow (\texttt{r0} \wedge \neg\, \texttt{r1}) \vee (\neg\, \texttt{r0} \wedge \texttt{r1})$

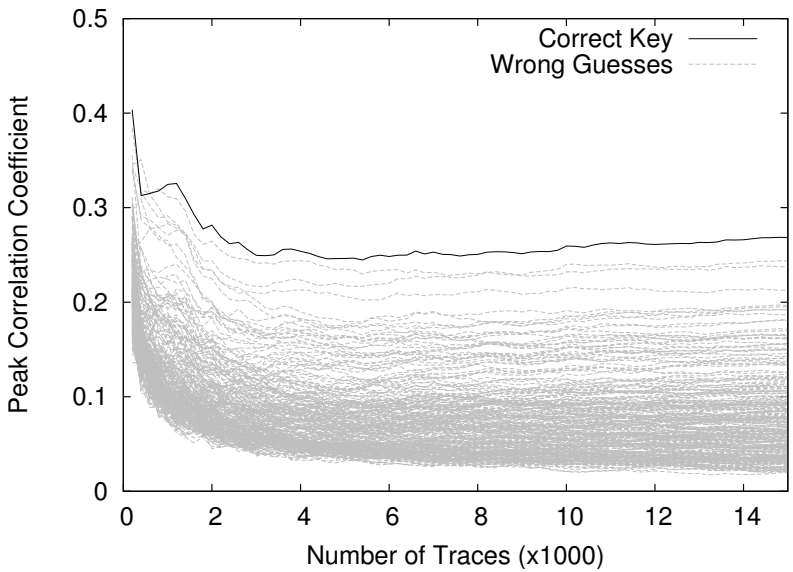
- The cryptographic implementation is combined with a polymorphic engine (hooked at compile-time) which dynamically and automatically transforms the binary code, hindering the design of a consumption model
- The number of possible cipher implementations obtained rewriting picking the fragments independently and randomly is very high
- Rewriting the code often enough prevents the attacker from gathering enough information to infer which operations are running
- masking and hiding techniques naturally fit the approach
- algorithm-agnostic approach fitting any cipher

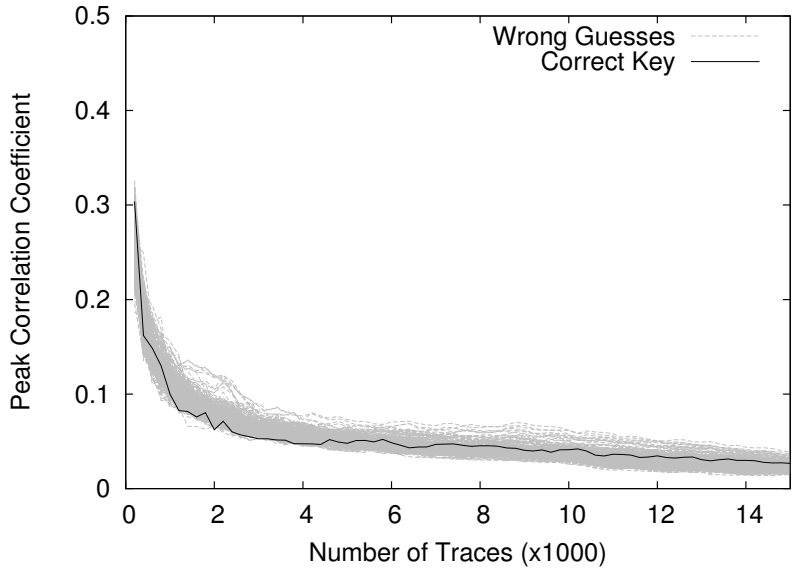
Note: All the code fragment alternatives, and the rewriting strategy are known: no security through obscurity!

Countermeasures against DPAs (II) - [PoliMI]

SPEAr Head200 ARM-926EJS (133MHz) - full fledged 32-bit CPU - no μC

Effectiveness





Efficiency

Morphing Period [no. of runs]	Conf. Intervals Overlap [%]	Normalized Time
100	79.55	×5.00
500	78.98	×1.86
1000	78.94	×1.46
2000	78.76	×1.27

Security-oriented Compiler Pipeline (I) [PoliMI]

Motivation

Ex-post analysis: Resistance against power channel analysis is usually checked through attacking a first prototype

- Obtaining a design time analysis would greatly speedup the process, especially if automated

Countermeasures Application: Power analysis countermeasures are particularly performance sapping and non trivial to implement

- Applying them sparingly, possibly only when needed is fundamental to preserve acceptable performance levels
- Applying them automatically allows a designer to reduce the effort needed to secure an implementation

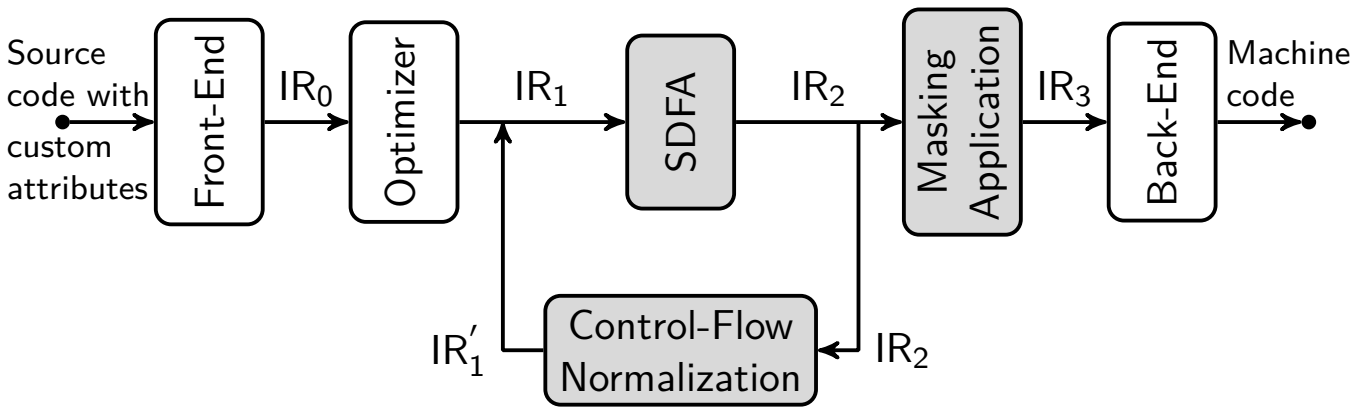
Security-oriented Compiler Pipeline (II) [PoliMI]

Proposed Approach

Security Oriented Dataflow Analysis

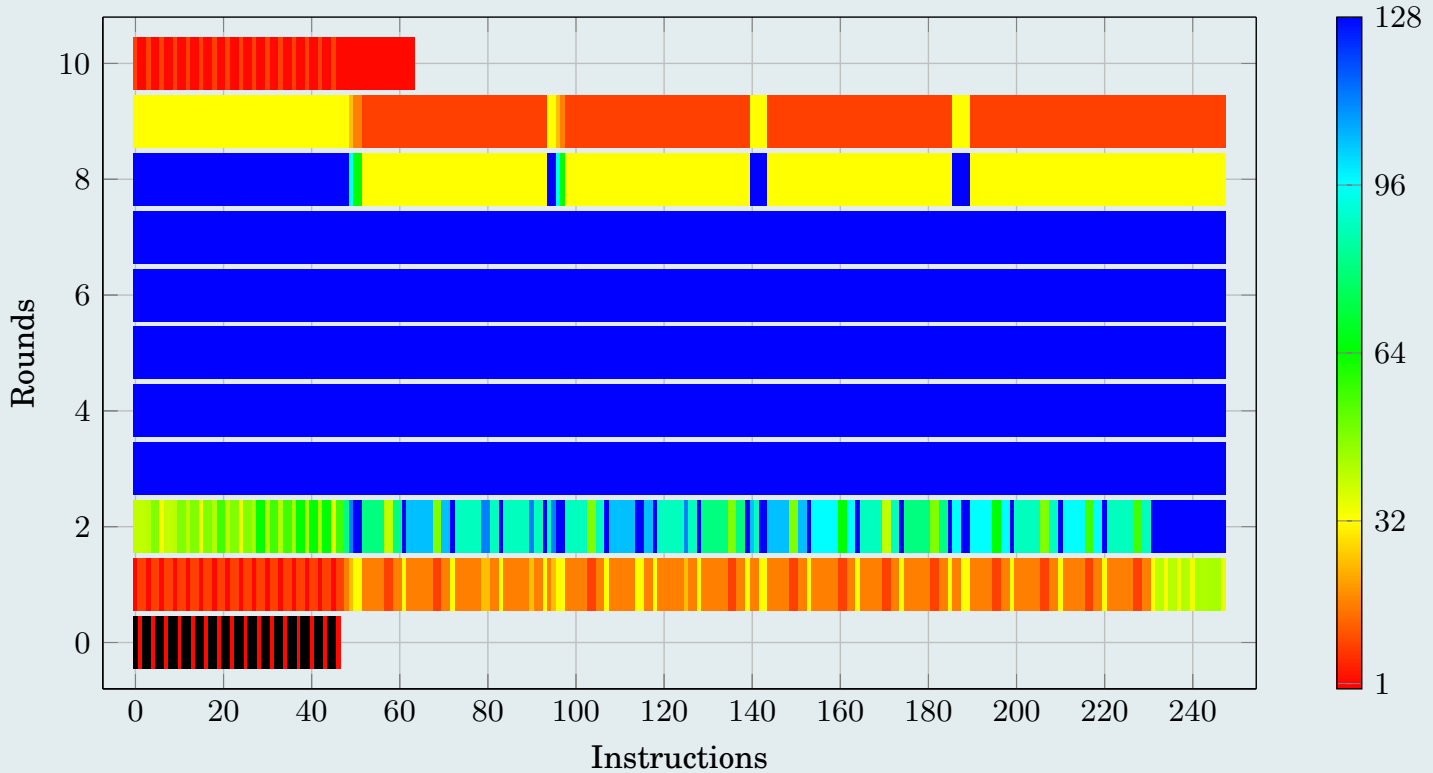
The effective side channel attack surface of a cipher can be determined following how the cipher key is mixed with the inner state

- We apply a specifically designed forward and backward dataflow analysis to the cipher primitive, detecting the amount of key on which each intermediate result depends
- Operations depending on an amount of key bits below a certain threshold are deemed to be in need for protection
- Automated countermeasure application: once vulnerable instructions in a cipher implementation are identified countermeasures can be put in place



Security-oriented Compiler Pipeline (III) [PoliMI]

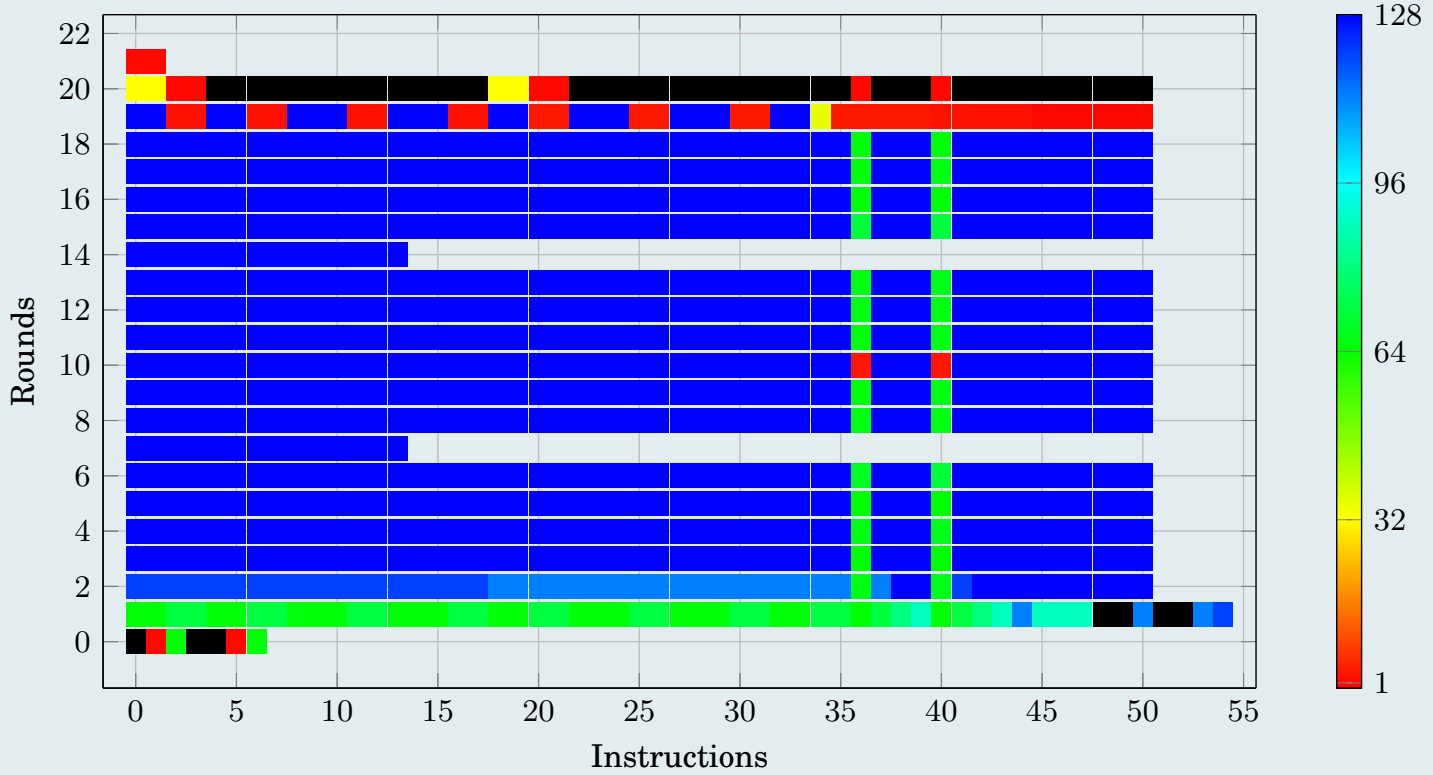
Analysis of AES-128 cipher



- Key dependencies of each intermediate instruction value are evaluated via a Forward-SDFA and a Backward-SDFA

Security-oriented Compiler Pipeline (IV) [PoliMI]

Analysis of CAMELLIA-128 cipher



- The automatic analysis discovered an undetected flaw in the inner rounds of this ISO-standard block cipher

Concluding Remarks

- Security as a design dimension
  - Adding security against implementation attacks consumes resources: Extra area, time, power, product development, ...
- Attacker will go for the easiest entry point
  - If strong crypto-algorithm, try other weaknesses
  - Monitor power consumption, EM radiation, time, ...
  - Inject glitches: clock, voltage, lasers, ...
- Threat of power analysis attacks:
  - Passive and non-invasive, low-cost equipment, ...
  - Arms-race between attacks and countermeasures
- Challenges:
  - evaluate the effectiveness of attacks against superscalar CPUs
  - assess the importance of μarchitectural features in pinpointing the sources of information leakage (e.g., shared internal pipeline buffers maybe responsible for critical information leakage - which cannot be counteracted at ISA-level)
  - counteract machine-learning template-based attacks

Thanks for Your Attention!

Questions ?

Gerardo Pelosi: gerardo.pelosi@polimi.it,
http://home.deib.polimi.it/pelosi