# Side-channel attacks

Gianluca Dini
Dept. of Ingegneria dell'Informazione
University of Pisa
Email: gianluca.dini@.unipi.it
Version: 2024-04-11

1

Side-channel attack

## INTRODUCTION

Apr-24 Side-channel attacks 2

2

## Cryptoanalysis

- **Cryptanalysis** is the *art* and *science* of analyzing information systems to study the hidden aspects of the systems
- **Mathematical analysis** of cryptographic algorithms
- **Side Channel Attacks**





## What is a side channel?

- A side channel is based on information gained from the physical implementation of a cryptosystem
- No theoretical weaknesses in the algorithm
- No brute force

# Side channel attacks

UNIVERSITÀ DI PISA

- The attacker must have physical access to the device under attack
- The attacker knows the algorithm under attack
  - The only secret is the key
- Two stages
  - 1st stage → Measurements
  - 2nd stage → Analysis of the measurements
    - Statistical analysis
    - Application of cryptanalysis



5

# Types of side channel attacks

UNIVERSITÀ DI PISA

- Fault injection
- Power analysis
- Timing analysis



6

# Why side channels?

- More effective against modern cryptosystem
- Embedded systems change the threat model
  - The adversary may *physically* attack the system
    - E.g.: smart meter, electronic passports, identity cards, driver licenses, point of sales, digital rights management, access control, pay tv, etc etc
  - The adversary may *physically* interfere with the system
  - The adversary has a *scale advantage*
    - Ex. Extracting one key from a single Pay TV smartcard allows to program several new smartcards with the same key → **clones**





Side channel attacks

# PHYSICAL ATTACKS

## CRT and RSA optimization [→]

UNIVERSITÀ DI PISA

- Chinese Remainder Theorem (CRT) allows us to compute RSA (decryption, signing) more efficiently





## CRT and RSA optimization [→]

UNIVERSITÀ DI PISA

- Problem: efficiently compute $y = x^d \pmod n$
- Solution
  1. Compute $x_p = x \bmod p$ and $x_q = x \bmod q$
  2. Compute $y_p = x_p^{\,d \bmod (p-1)} \bmod p$ and $y_q = x_q^{\,d \bmod (q-1)} \bmod q$
  3. Compute $y = a_p y_p q + a_q y_q p$ where $a_p$ and $a_q$ are properly (pre-)computed coefficients

## CRT and RSA optimization [→]

- Performance advantage
  - Computation of $y_p$ and $y_q$ is the most demanding
  - On average mod exp requires #MUL+#SQ = 1.5t
  - In the case of CRT
    - 2 exponentiations on t/2 bits => 2 × (1.5 t/2) = 1.5t
    - Each squaring/multiplication involves t/2-bit operands → multiplication/squaring takes $O(t^2/4)$
  - The total speedup obtained through CRT is a factor of 4.





## A fault-injection attack against CRT-based RSA (→)

- Attack intuition
  - By injecting a fault, the adversary can factorize n
- The attack
  - Cause an *hw fault* while computing $y_p$ which produces $y'_p$
  - Thus $y' = a_p y'_p q + a_q y_q p$ (step 3)
  - It follows that $y – y' = a_p(m'_p – m_p)q$
  - Thus, $\gcd(y – y', n) = q$ *which can be efficiently computed with the Euclide's algorithm (!)*

# A fault-injection attack against CRT-based RSA

- Practical considerations
  - causing hw fault requires tampering with computing circuitry
  - countermeasures: checking results (10% *slow down*)
    - Still subject to *double-fault attack*.





Side channel attacks

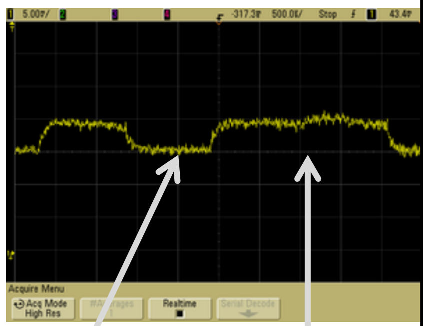
# POWER ANALYSIS

# Power Analysis [→]

UNIVERSITÀ DI PISA

- Power analysis is a side channel attack in which the attacker studies the power consumption of a cryptographic hardware device
  - smart card, tamper-resistant "black box", or integrated circuit
- The attack is non-invasive
- Simple power analysis (SPA) involves *visual examination* of graphs of the current used by a device over time.
  - Variations in power consumption occur as the device performs different operations.

POWER ANALYSIS OF RSA



Key bit = 0
No multiplication

Key bit = 1
multiplication

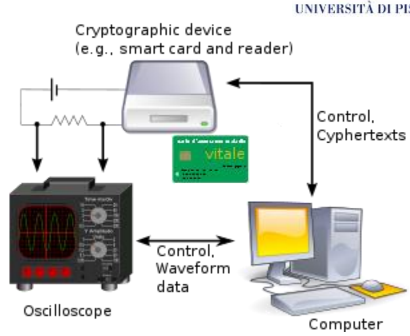
Apr-24 Side-channel attacks 16

16

# Power Analysis [→]

UNIVERSITÀ DI PISA

- Differential power analysis (DPA) involves *statistically* analyzing power consumption measurements from a cryptosystem.
  - DPA attacks have signal processing and error correction properties which can extract secrets from measurements which contain too much noise to be analyzed using simple power analysis.




Kocher, Paul, Joshua Jaffe, and Benjamin Jun. "Differential power analysis." *Annual International Cryptology Conference*. Springer, Berlin, Heidelberg, 1999.

Apr-24 Side-channel attacks 17

17

Side channel attacks

# TIMING ATTACKS





# Timing attack

UNIVERSITÀ DI PISA

- A timing attack is a side channel attack in which the attacker attempts to compromise a cryptosystem by analyzing the time taken to execute cryptographic algorithms
  - Exploit execution time that depends on inputs (e.g., key!)
  - Require *precise* measurement of time
  - Application dependent
  - E.g., square-and-multiply for exp mod n
    - time depends on number of "1" in the key
    - Statistical analysis of timings with same key and different inputs

## Timing Attack against HMAC

- Example(*): George Keyczar crypto library (Python, Java) [simplified]

```
def Verify(key, msg, tag):
    return HMAC(key, msg) == tag
```

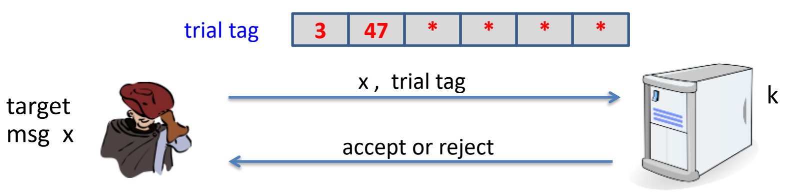
- The problem
  - Operator '==' is implemented as a byte-by-byte comparison
  - Comparator returns false when first inequality found
  - This provides a timing side-channel

(*) N.Lawson. "Side-Channel Attacks on Cryptographic Software," *IEEE Security & Privacy*, 2009

Apr-24 Side-channel attacks 20

20

## Timing attack




**Timing attack:** To compute tag for target message x

**Step 0:** The adversary queries server with random tag. The time of this query is considered the reference time (w.h.p.)

**Step 1:** Loop over all possible first byte values and query server, until verification takes a little longer

**Step 2:** Repeat for all subsequent tag bytes until valid tag found

Apr-24 Side-channel attacks 22

22

# Defense #1

Make string comparator always take same time (Python) :

```
return false if  tag  has wrong length
result = 0
for x, y in zip( HMAC(key,msg) , tag):
    result |= ord(x) ^ ord(y)
return result == 0
```

*Can be difficult to ensure due to optimizing compiler*

Apr-24 Side-channel attacks 23

23

# Defense #2

Make string comparator always take same time (Python) :

```
def Verify(key, msg, tag):
    mac = HMAC(key, msg)
    return HMAC(key, mac) == HMAC(key, tag)
```

*Attacker doesn't know values being compared!*

Apr-24 Side-channel attacks 24

24