Side-channel attacks

Gianluca Dini
Dept. of Ingegneria dell'Informazione
University of Pisa
Email: gianluca.dini@.unipi.it
Version: 24/04/25

1

Side-channel attack

INTRODUCTION

2

Cryptoanalysis

- **Cryptanalysis** is the *art* and *science* of analyzing information systems to study the hidden aspects of the systems
  we gave hints of math aspects of algorithms
- **Mathematical analysis** of cryptographic algorithms
- **Side Channel Attacks** Another way of attacking

apr-25 Side-channel attacks 3

3

What is a side channel?

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

not searching for algorithm vulnerability, but for implementation vulnerability.
In this course we are mainly interested in protocol vulnerabilities

apr-25 Side-channel attacks 4

4

Side channel attacks

To do this:

- The attacker must have physical access to the device under attack (difficult to perform SCA against server through network) ①
- The attacker knows the algorithm under attack
  - The only secret is the key
- Two stages
  - 1st stage → Measurements
  - 2nd stage → Analysis of the measurements

① Mainly perceived for smartcards (ex. SIMs)

apr-25 Side-channel attacks 5

5

Types of side channel attacks

- Fault injection
  CAUSE A FAULT
- Power analysis
- Timing analysis

3 types, very rough classification

Power consumption and time to complete operation analysis

apr-25 Side-channel attacks 6

6

Why side channels?

UNIVERSITÀ DI PISA

- More effective against modern cryptosystem
- Embedded systems change the threat model
  - The adversary can *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 can *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)

apr-25 Side-channel attacks 7

7

Side channel attacks

FAULT INJECTION: AN EXAMPLE

apr-25 Side-channel attacks 8

8

CRT and RSA optimization [→]

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

CRT and RSA optimization [→]

- Problem: efficiently compute $y = x^d \pmod{n}$, with n a t-bit modulus
- 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

UNIVERSITÀ DI PISA

- 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.

apr-25 Side-channel attacks 12

12

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

UNIVERSITÀ DI PISA

- 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) $a_p y_p q - a_p y'_p q$
  - It follows that $y - y' = a_p(m'_p - m_p)q$ ($a_q y_q p$ is canceled)
  - Thus, $\gcd(y - y', n) = q$ *which can be efficiently computed with the Euclide's algorithm (!)*

apr-25 Side-channel attacks 13

13

A fault-injection attack against CRT-based RSA

- Practical considerations
  - causing hw fault requires tampering with computing circuitry (laser, intense electric field)
  - countermeasures: checking results (10% *slow down*)
    - Still subject to *double-fault attack*.
    - double check (redundant execution, result comparison, error handling):
      - slow down
      - still subject to double-fault attack.

apr-25 Side-channel attacks 14

14

Side channel attacks

POWER ANALYSIS: AN EXAMPLE

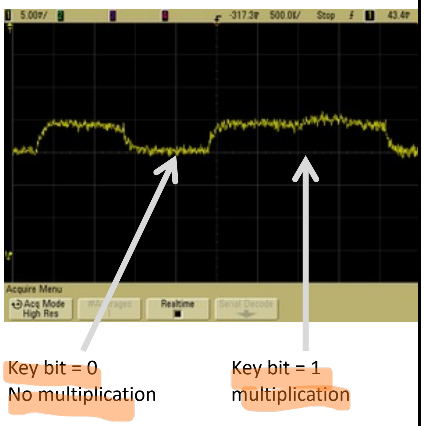
apr-25 Side-channel attacks 15

15

Power Analysis [→]

- 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



apr-25 Side-channel attacks 16

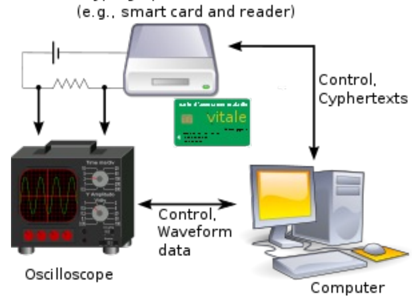
16

Power Analysis [→]

Statistical analysis of Power consumption through different inputs

MORE EFFECTIVE

- 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-25 Side-channel attacks 17

17

Side channel attacks

TIMING ATTACKS: AN EXAMPLE

18

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 You do this in a lab, not on the network
  - 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

19

Timing Attack against HMAC

UNIVERSITÀ DI PISA

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

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

Verification of HMAC

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

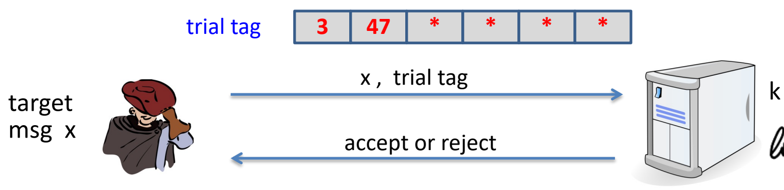
Equality takes more to be verified than inequalities

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

20

Timing attack

UNIVERSITÀ DI PISA



Server behaves like an oracle

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

We start changing the first byte: if first byte is wrong, result time takes the same as step 0. If byte is correct, server moves to second byte and takes more time.

22

Timing attack

- Efficiency/Complexity of the attack
  - Assume sizeof(tag) = 160 bits (20 bytes)
  - Max number of trials for each byte = 256
  - Running time of the attack = 20 × 256 = 5120 (worst case)
  - 5120 << 2^{160}

apr-25 Side-channel attacks 23

23

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-25 Side-channel attacks 24

24

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)
```

↳ Two random values
ATTACK WON'T WORK

If two quantities must be equal, then hashes must be too
So n get the key of the key.

Attacker doesn't know the values being compared!

apr-25 Side-channel attacks 25

25