PV181 Laboratory of security and applied cryptography

Seminar 9:
Crypto-libraries protected against hardware attacks

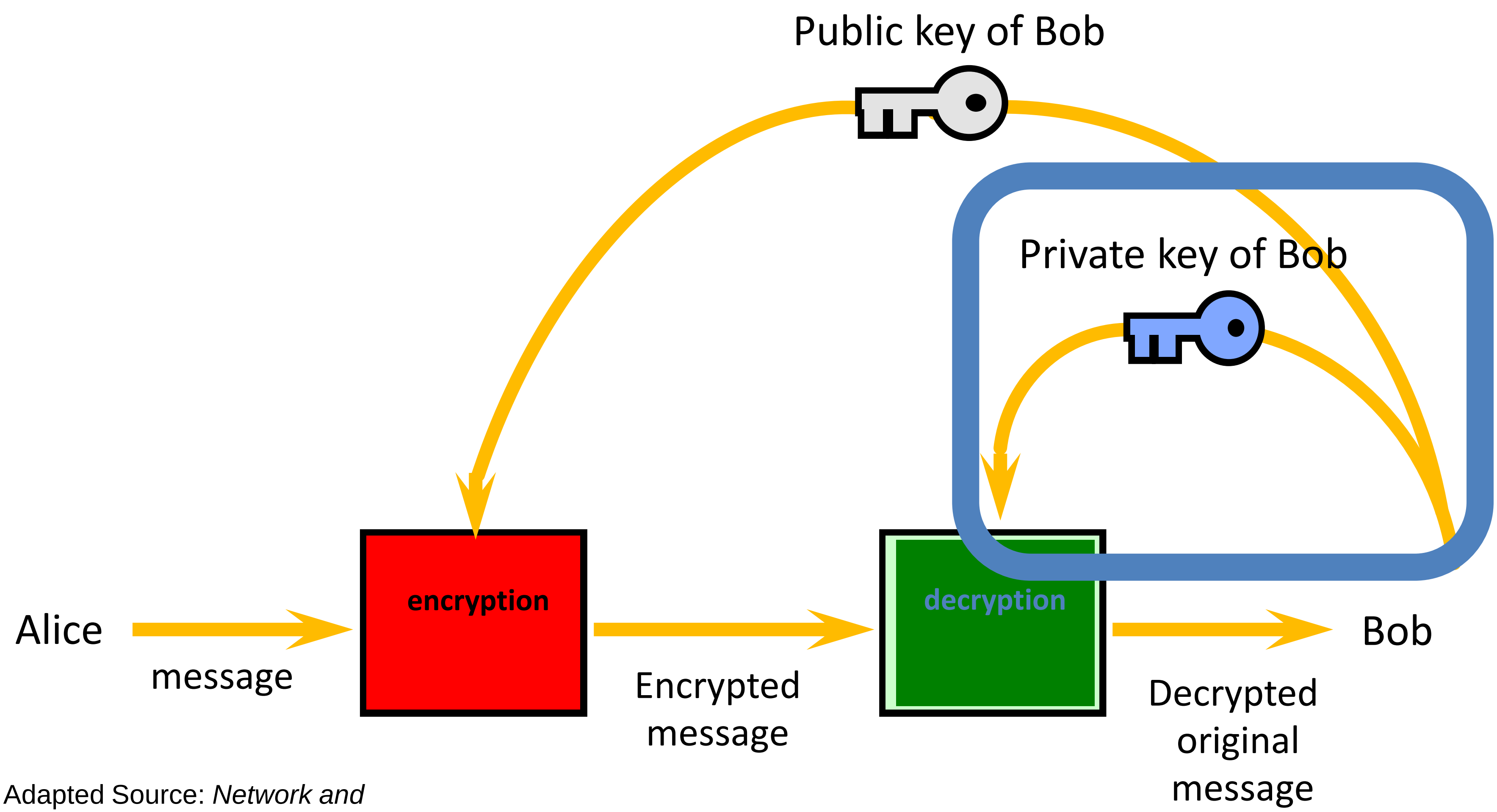
Łukasz Chmielewski
chmiel@fi.muni.cz

CRoCS
Centre for Research on Cryptography and Security

Outline

- Recall + goal of this seminar
  - Digital signatures
  - RSA vs. ECC
- Side Channel + Fault Injection speed run
- Secured X25519 library: sca25519
  - Demo Exercise
- Python Exercise
  - Securing RSA execution
- No Assignment this week ☺

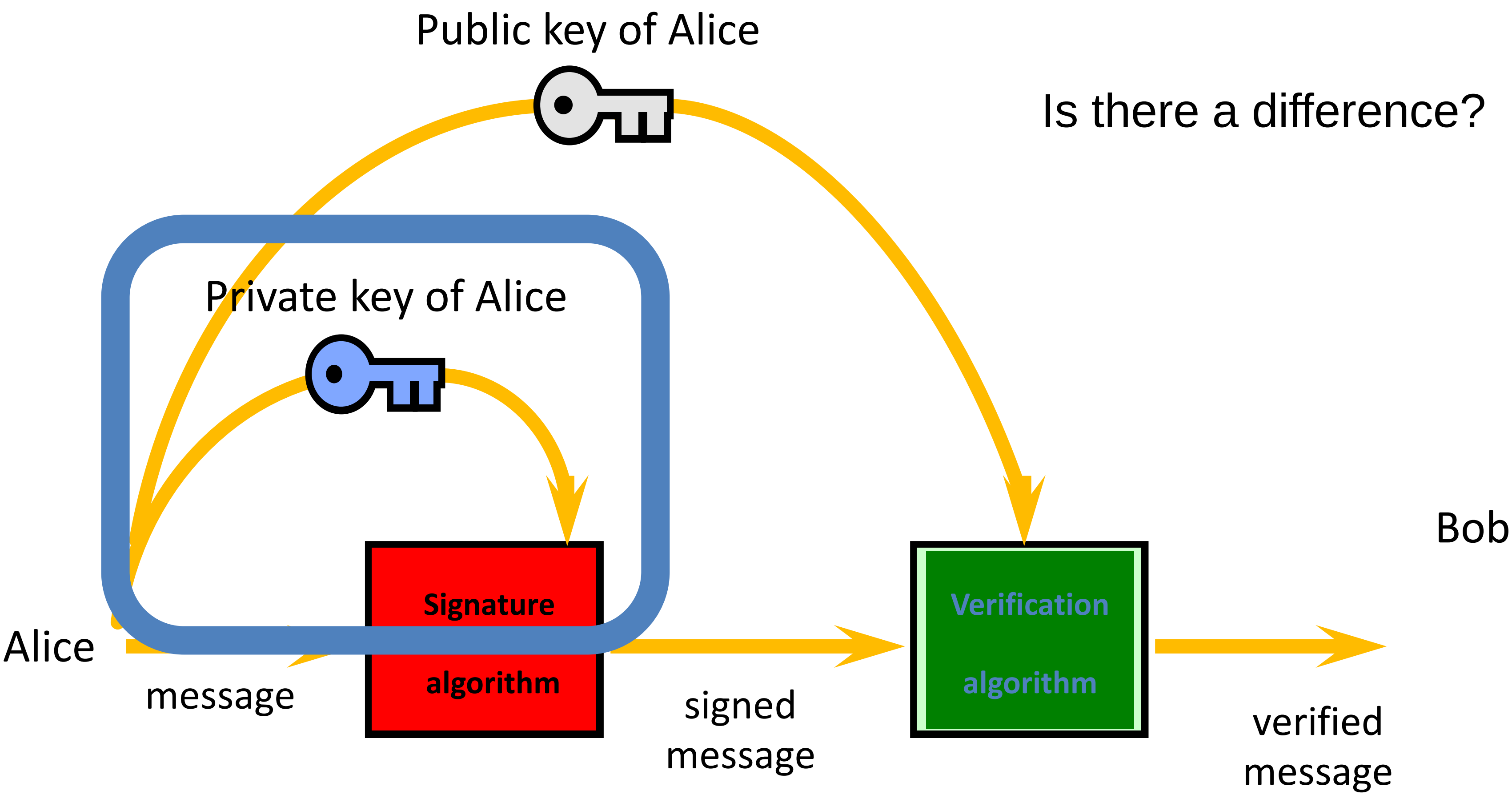
Recall: Asymmetric cryptosystem



Adapted Source: *Network and Internetwork Security* (Stallings)

Recall: Digital signature scheme

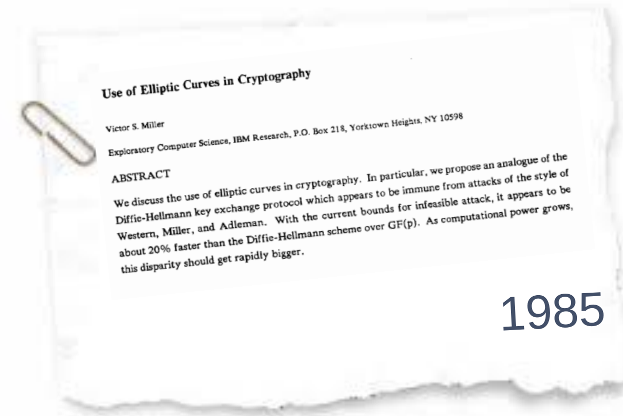
Is there a difference?



Source: *Network and Internetwork Security* (Stallings)

Recall: RSA vs. ECC

- exponentiation ≈ scalar multiplication
- multiplication ≈ points addition
- squaring ≈ point doubling



Use of Elliptic Curves in Cryptography

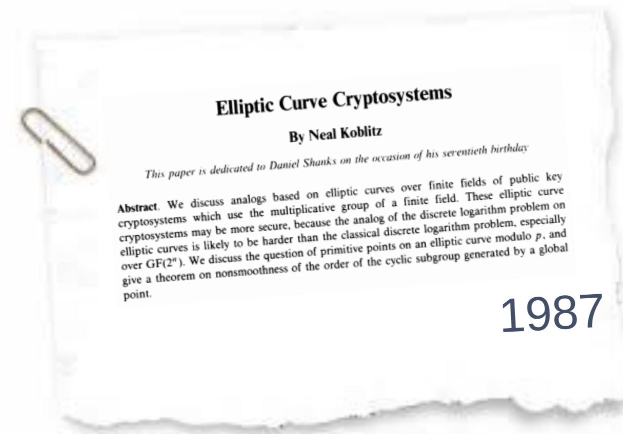
Victor S. Miller

Exploratory Computer Science, IBM Research, P.O. Box 218, Yorktown Heights, NY 10598

ABSTRACT

We discuss the use of elliptic curves in cryptography. In particular, we propose an analogue of the Diffie-Hellmann key exchange protocol which appears to be immune from attacks of the style of Western, Miller, and Adleman. With the current bounds for infeasible attack, it appears to be about 20% faster than the Diffie-Hellmann scheme over GF(p). As computational power grows, this disparity should get rapidly bigger.

1985



Elliptic Curve Cryptosystems

By Neal Koblitz

This paper is dedicated to Daniel Shanks on the occasion of his seventieth birthday

Abstract. We discuss analogs based on elliptic curves over finite fields of public key cryptosystems which use the multiplicative group of a finite field. These elliptic curve cryptosystems may be more secure, because the analog of the discrete logarithm problem on elliptic curves is likely to be harder than the classical discrete logarithm problem, especially over $GF(2^n)$. We discuss the question of primitive points on an elliptic curve modulo p, and give a theorem on nonsmoothness of the order of the cyclic subgroup generated by a global point.

1987

Why is hardware security important?

Card / Money Theft

Identity Theft

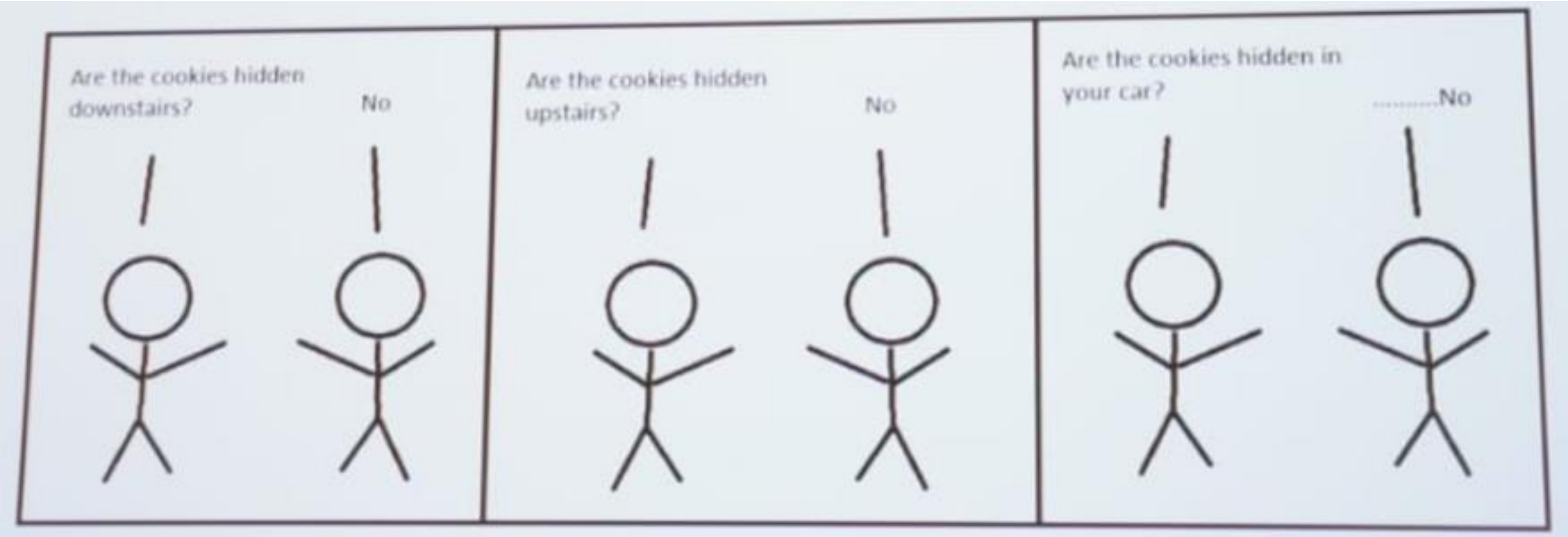
- **Premium**

Phone / Money Theft

Impersonation

Side-Channel Analysis

Cookies Example



Passive vs Active Side Channels

Passive: analyze device behavior

Active: change device behavior

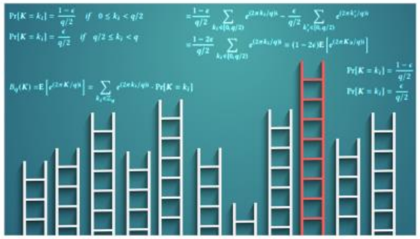
Recent Practical Attacks

November 13, 2019

May 28, 2020

LadderLeak: Side-channel security flaws exploited to break ECDSA cryptography

Existing attacks refined to attack elliptic curve algorithm



SCA Titan: January 7, 2021

October 3, 2019

Researchers Discover ECDSA Key Recovery Method

December 12, 2019

PLUNDERVAULT —

Intel's SGX coughs up crypto keys when scientists tweak CPU voltage

Install fixes when they become available. Until then, don't sweat it.

DAN GOODIN - 12/10/2019, 11:41 PM

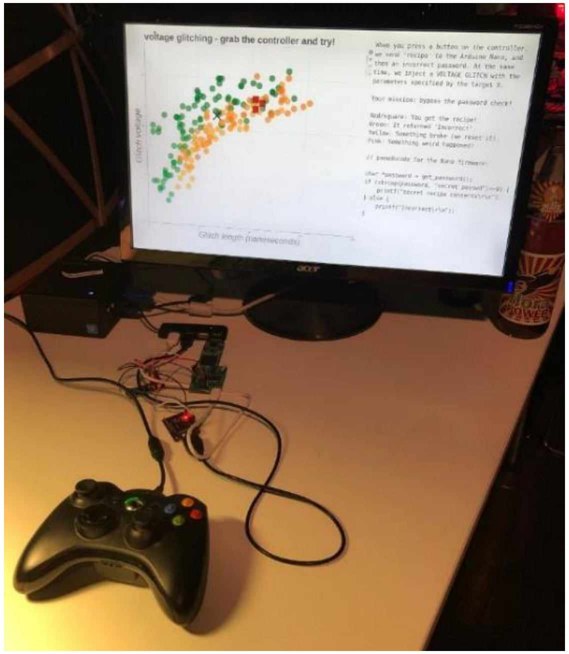
Side Channels

- Time
- Power
- Electro Magnetic Emanations
- Light
- Sound
- Temperature
- …

What can be attacked & why?

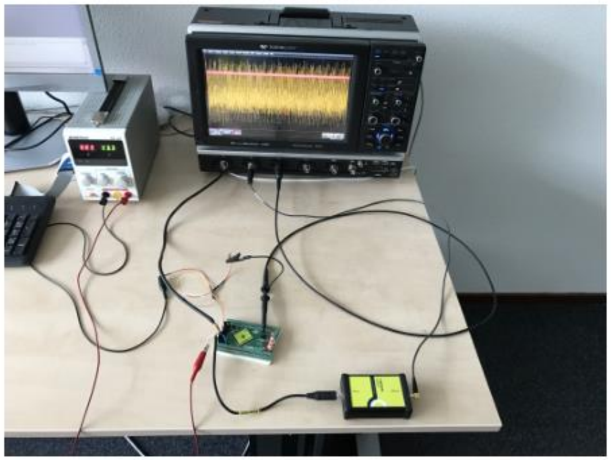
- Type of device?
- What kind of primitive?
- How much control do you have?
- What can you access?
- What would be the attacker’s goal?
- What is your goal?
- Where is the money?
- …

Practical Setup Spectrum

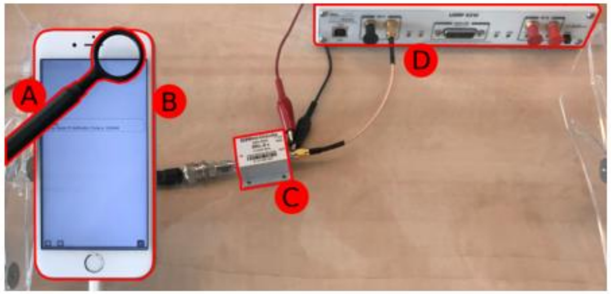


Some Other Practical Setups

DPA setup with ARM CortexM4

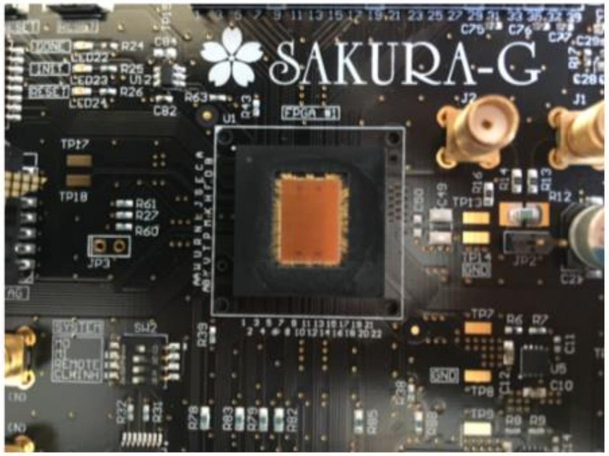


Tempest

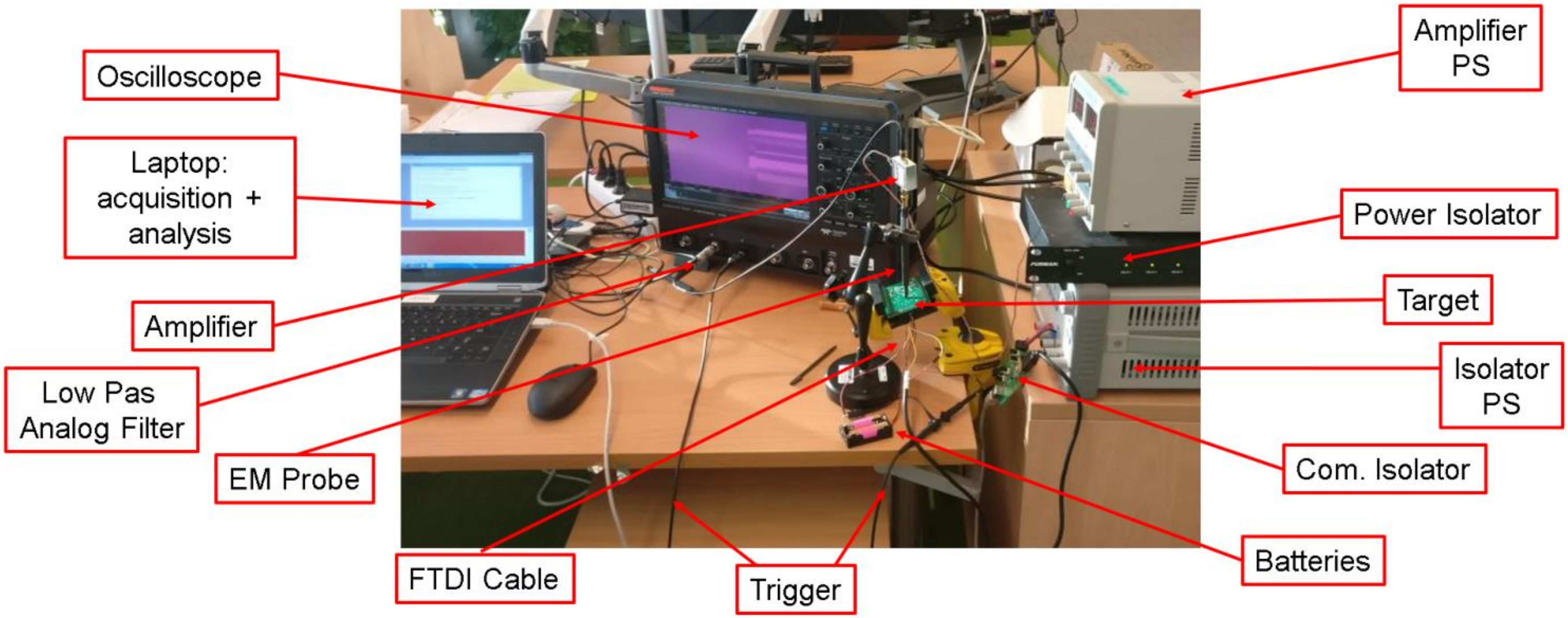


FA setup

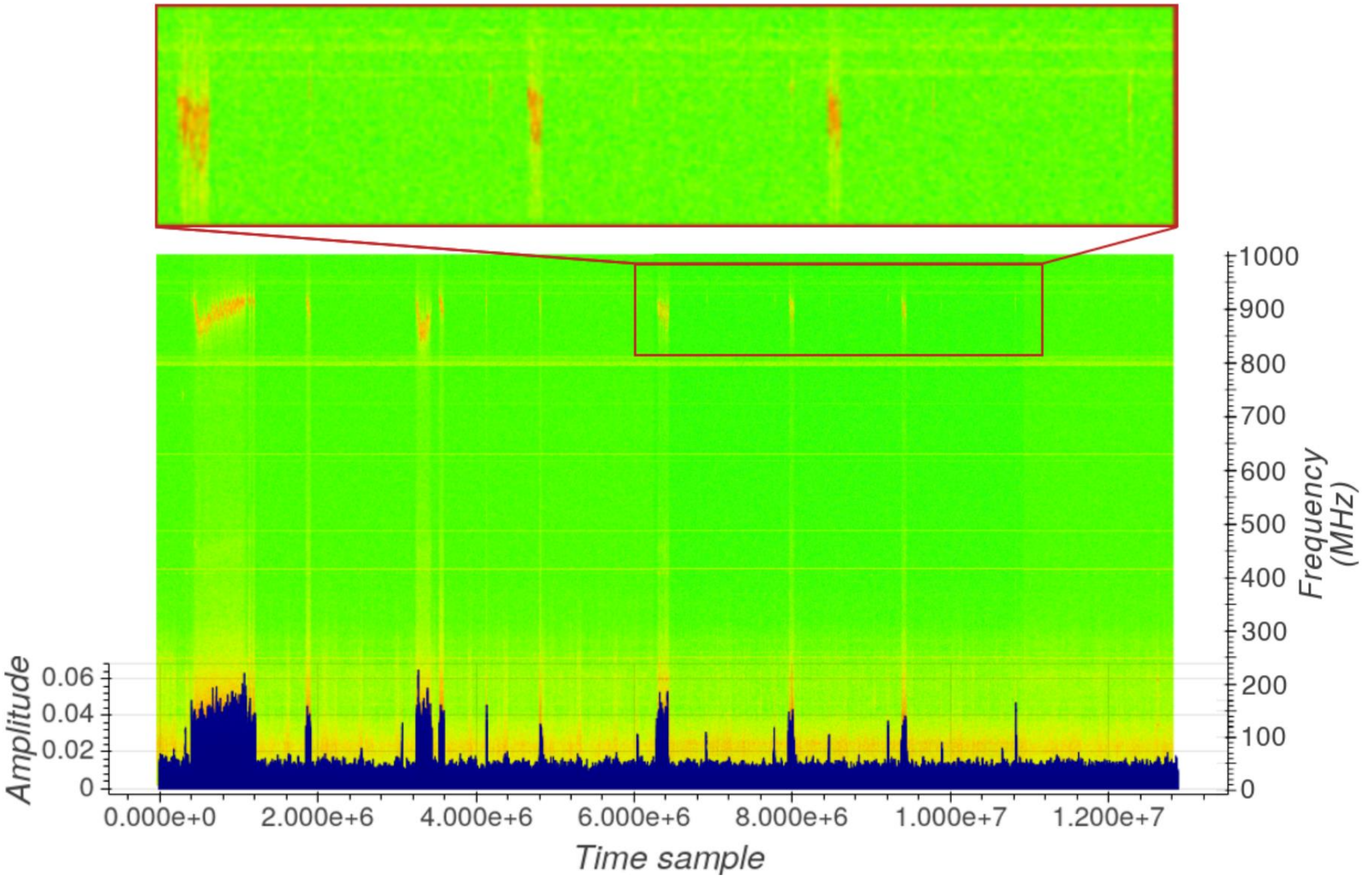
FPGA board for SCA



Actual (overcomplicated?) setup

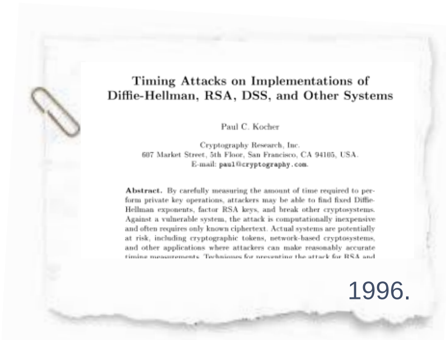


Example Side Channel Attack:
GPU running NN



Simple Power Analysis (SPA) on RSA

```
ModExp(c) {
    A = 1
    for ( i = n-1; i≥0; i– –)
        A = A² mod N          S
        if (d_i = =1)
            A = A*c mod N     M
        end if
    end for
    Return A = c^d mod N
}
```



Timing Attacks on Implementations of
Diffie-Hellman, RSA, DSS, and Other Systems

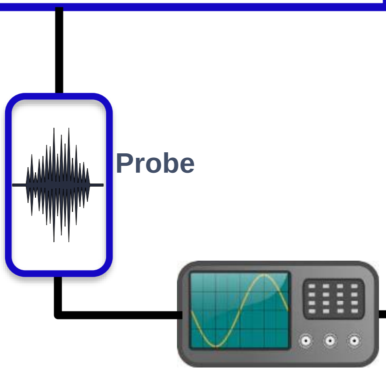
Paul C. Kocher

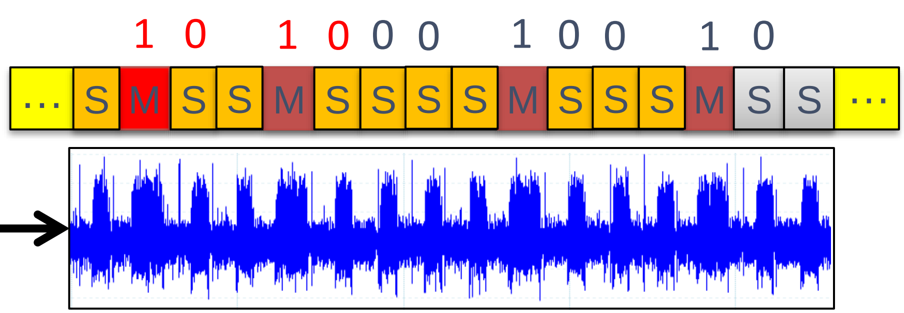
Cryptography Research, Inc.
607 Market Street, 5th Floor, San Francisco, CA 94105, USA.
E-mail: paul@cryptography.com.

Abstract. By carefully measuring the amount of time required to perform private key operations, attackers may be able to find fixed Diffie-Hellman exponents, factor RSA keys, and break other cryptosystems. Against a vulnerable system, the attack is computationally inexpensive and often requires only known ciphertext. Actual systems are potentially at risk, including cryptographic tokens, network-based cryptosystems, and other applications where attackers can make reasonably accurate

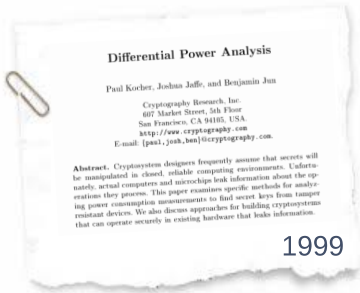
1996.

"By carefully measuring the *amount of time* required to perform private key operations, attackers may be able to find […] RSA keys."





Differential (Correlation) Power Analysis

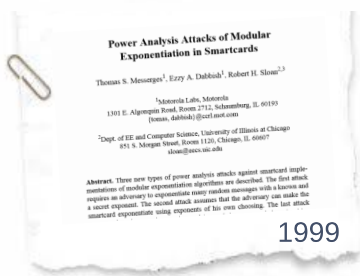

Differential Power Analysis

Paul Kocher, Joshua Jaffe, and Benjamin Jun

Cryptography Research, Inc.
607 Market Street, 5th Floor
San Francisco, CA 94105, USA.
http://www.cryptography.com
E-mail: {paul,josh,ben}@cryptography.com.

Abstract. Cryptosystem designers frequently assume that secrets will
be manipulated in closed, reliable computing environments. Unfortu-
nately, actual computers and microchips leak information about the op-
erations they process. This paper examines specific methods for analyz-
ing power consumption measurements to find secret keys from tamper
resistant devices. We also discuss approaches for building cryptosystems
that can operate securely in existing hardware that leaks information.

1999


Power Analysis Attacks of Modular Exponentiation in Smartcards

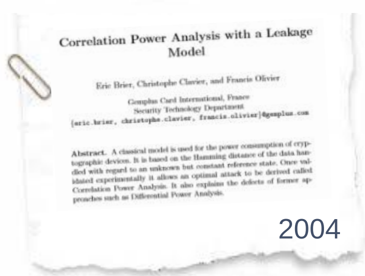
Thomas S. Messerges[1], Ezzy A. Dabbish[1], Robert H. Sloan[2,3]

[1]Motorola Labs, Motorola
1301 E. Algonquin Road, Room 2712, Schaumburg, IL 60193
{tomas, dabbish}@ccrl.mot.com

[2]Dept. of EE and Computer Science, University of Illinois at Chicago
851 S. Morgan Street, Room 1120, Chicago, IL 60607
sloan@eecs.uic.edu

Abstract. Three new types of power analysis attacks against smartcard imple-
mentations of modular exponentiation algorithms are described. The first attack
requires an adversary to exponentiate many random messages with a known and
a secret exponent. The second attack assumes that the adversary can make the
smartcard exponentiate using exponents of his own choosing. The last attack

1999

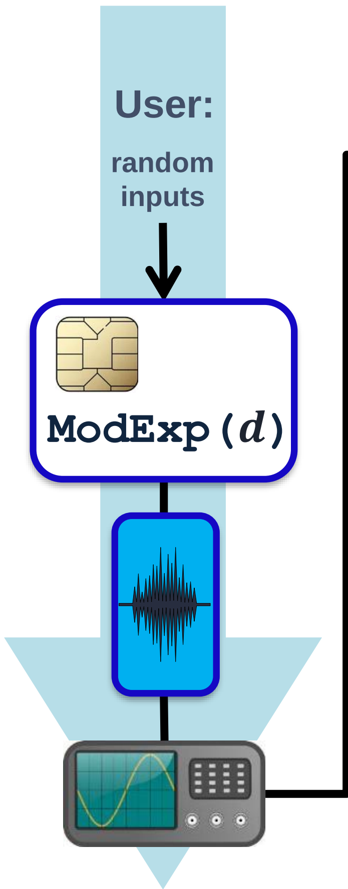

Correlation Power Analysis with a Leakage Model

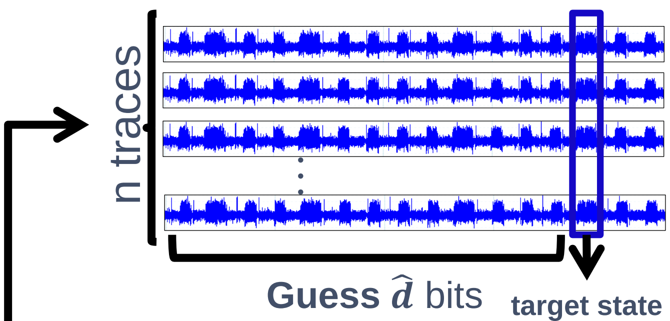
Eric Brier, Christophe Clavier, and Francis Olivier

Gemplus Card International, France
Security Technology Department
{eric.brier, christophe.clavier, francis.olivier}@gemplus.com

Abstract. A classical model is used for the power consumption of cryp-
tographic devices. It is based on the Hamming distance of the data han-
dled with regard to an unknown but constant reference state. Once val-
idated experimentally it allows an optimal attack to be derived called
Correlation Power Analysis. It also explains the defects of former ap-
proaches such as Differential Power Analysis.

2004





f_i = **Selection Function**(random inputs, $\hat{d}$, target state)

- HW of a register
- HD between current and previous register state
- ID model (value of a register)

$$f_i = \begin{cases} 0 & if\ HW \leq 16 \\ 1 & if\ HW > 16 \end{cases} \qquad f_i = HW(reg\_state)$$

DPA = Difference of Means

CPA = Pearson correlation

Goals of Fault Injection

- The goal is to change a critical value or to change the flow of a program.
- Faults can be injected in several ways:
  - Power glitches can disturb the power supply to the processor, resulting in wrong values read from memory.
  - Optical glitches with laser can force any elementary circuit to switch, enabling the attacker to achieve a very specific change of data values or behavior.
  - Clock manipulation by introducing a few very short clock cycles which may lead to the device misinterpreting a value read from memory.
  - Cutting the power to the processor while performing important computations, hoping to either prevent the system from taking measures against a detected attack or get the system into a vulnerable state when the power is back.
- Differential Fault Analysis (DFA)

Fault Injection Example:
the "unlooper" device

Question 0:
Software for PIN code verification

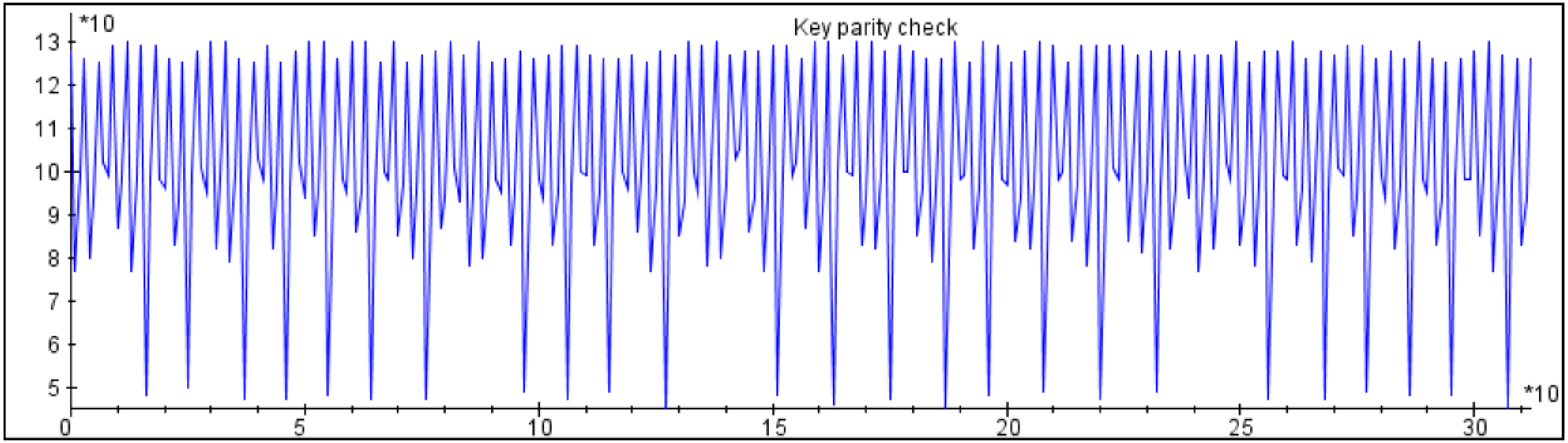
```
Input: 4-digit PIN code
Output: PIN verified or rejected
Process CheckPIN (pin[4])
int pin_ok=0;
if (pin[0]==5)
   if (pin[1]==9)
      if (pin[2]==0)
         if (pin[3]==2)
            pin_ok=1;
         end
      end
   end
end
return pin_ok;
EndProcess
```

- What is the problem here?
- What are the execution times of the process for PIN inputs?
  - [0,1,2,3], [5,3,0,2], [5,9,0,0]
- The execution time increases as we get closer to
  - [5,9,0,2]

Task 0 – parity check for DES key

```
public static boolean checkParity ( byte[]key, int offset) {
     for (int i = 0; i < DES_KEY_LEN; i++) { // for all key bytes
             byte keyByte = key[i + offset];
             int count = 0;
             while (keyByte != 0) { // loop till no '1' bits left
                    if ((keyByte & 0x01) != 0) {
                         count++; // increment for every '1' bit
                    }
                    keyByte >>>= 1; // shift right
             }
             if ((count & 1) == 0) { // not odd
                    return false; // parity not adjusted
             }
     }
     return true; // all bytes were odd
}
```

Task 0 – parity check for DES key cont'd



Tell me what is the key ☺

Question 1:
faster and more secure modexp - Montgomery ladder

```
x0=x; x1=x^2
for j=k-2 to 0 {
  if dj=0
    x1=x0*x1; x0=x0^2
  else
    x0=x0*x1; x1=x1^2
  x1=x1 mod N
  x0=x0 mod N
}
return x0
```

$x_0=x;\ x_1=x^2$
for $j=k-2$ to 0 {
  if $d_j=0$
    $x_1=x_0*x_1;\ x_0=x_0^2$
  else
    $x_0=x_0*x_1;\ x_1=x_1^2$
  $x_1=x_1 \bmod N$
  $x_0=x_0 \bmod N$
}
return x_0

Both branches with the same number and type of operations (unlike square and multiply on previous slide)

Is it constant-time & secure? Why?

Question 2:
even more secure modexp

```
x0=x; x1=x^2
for j=k-2 to 0 {
  b=dj
  x(1-b)=x0*x1; xb=xb^2
  x1=x1 mod N
  x0=x0 mod N
}
return x0
```

$x_0 = x;\ x_1 = x^2$
for $j = k-2$ to 0 **{**
  $b = d_j$
  $x_{(1-b)} = x_0 * x_1;\ x_b = x_b^2$
  $x_1 = x_1 \bmod N$
  $x_0 = x_0 \bmod N$
}
return x_0

Memory access often is not constant time!
Especially in the presence of caches.

Is it constant-time & secure? Why?

Question 3:
even more secure modexp

```
x_0=x; x_1=x^2
for j=k-2 to 0 {
  b=d_j
  x_(1-b)=x_0*x_1; x_b=x_b^2
  x_1=x_1 mod N
  x_0=x_0 mod N
}
return x_0
```

Memory access often is not constant time!
Especially in the presence of caches.

Is it constant-time & secure? Why?

Question 4:
even more more secure modexp

```
x0=x; x1=x^2; sw = 0
for j=k-2 to 0 {
  b=dj
  cswap(x0,x1,b⊕sw)
  sw = sw⊕di
  x1=x0*x1; x0=x0^2
  x1=x1 mod N
  x0=x0 mod N
}
return x0
```

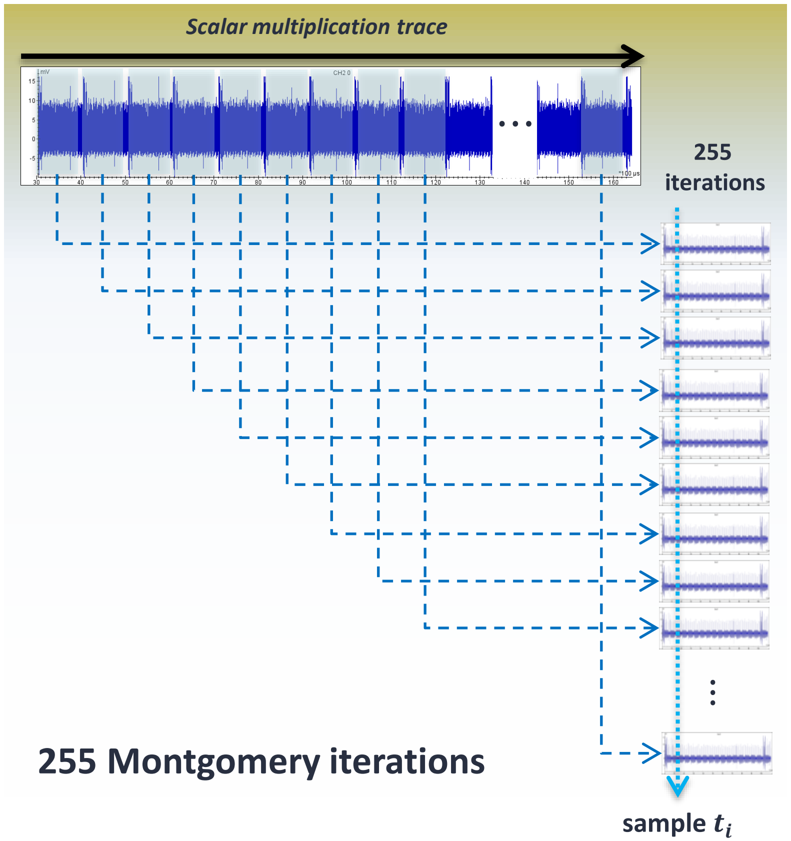
Constant-time? Depends on the cswap… but it can be ☺
Other-side channels? Depends 😐

Is it constant-time & secure? Why?

Question 5:
Arithmetic Cswap – constant-time?

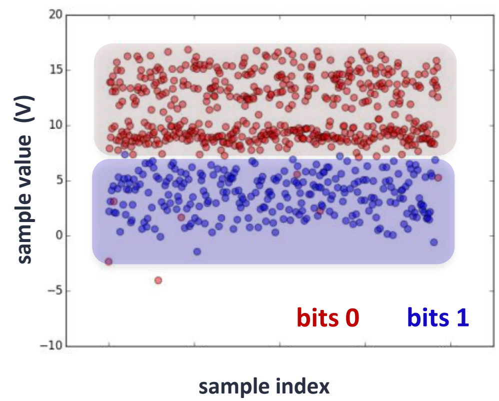
```
void fe25519_cswap(fe25519* in1, fe25519* in2, int condition)
{
    int32 mask = condition;
    uint32 ctr;
    mask = -mask;
    for (ctr = 0; ctr < 8; ctr++)
    {
        uint32 val1 = in1->as_uint32[ctr];
        uint32 val2 = in2->as_uint32[ctr];
        uint32 temp = val1;
        val1 ^= mask & (val2 ^ val1);
        val2 ^= mask & (val2 ^ temp);
        in1->as_uint32[ctr] = val1;
        in2->as_uint32[ctr] = val2;
    }
}
```

Question 5:
Arithmetic Cswap – secure against other side-channels?



255 Montgomery iterations

Apply clustering (e.g. **k-means**), Template Attack, Deep Learning to the set of 255 samples:



255 Samples

Message and exponent blinding

$$c = m^d \bmod N$$

1. $m_r = m.r^{-e} \bmod N$ → message blinding
2. $d_r = d + r * \varphi(n)$ → exponent blinding
3. $c_r = m_r{}^{d_r} \bmod n$ → blinded exponentiation
4. $c = c_r * r \bmod n$ → message “unblinding”

The sequence of operations (S, M) is related to the exponent bits.

However:

- If ***d*** is random: the sequence of exponent bits changes for every RSA execution
- If ***m*** is random: Intermediate data is random (masked) → hardly predicted!

DPA is based on the prediction of intermediate data.

Thesis: *Any side-channel attack requiring **multiple traces** are repelled by message **and** exponent blinding countermeasures.*

For ECC there are corresponding countermeasures: coordinate blinding, scalar blinding, blinded scalar multiplications, and no unblinding ☺

SCA&FI-protected Elliptic Curve library

- A protected library for ECDH
  - key exchange & session key establishment
  - It will be published in TCHES2023 volume 1 and
    - presented at Ches 2023 in Prague
- Download the library from github
- Useful links:
  - https://eprint.iacr.org/2021/1003
  - https://github.com/sca-secure-library-sca25519/sca25519
- Taking care of ECDSA:
  - https://eprint.iacr.org/2022/1254
  - I will add it to the repository later on.

Seminar Tasks

- Task 1 – analyze the code of the ephemeral implementation with respect to Questions 1 to 5.
  - How is protected?
  - Work in pairs and discuss your thoughts.
- Task 2 - compare implementations – what is the difference?
  - Hint: you can have a look at the paper and the repo too.
- Task 3 – how different implementations are measuring efficiency?
- Task 4 – do you see any fault injection countermeasures?

Seminar Tasks Cont'd

- Let's do the efficiency DEMO.
- (Optional) Tasks 5 – try to perform various measurements of the efficiency of one (chosen by you) implementation.
  - We have only two boards so people can do it in small groups and change.
- Task 6: protect the RSA implementation with exponent blinding! – see the RSA.py
- Super-optional Task 7: protect the implementation with message blinding! – see the RSA.py

No Assignment

- ☺