Processor Extensions for Security

Arnaud Tisserand

CNRS, IRISA laboratory, CAIRN research team

École ARCHI
Lille, Nord
June 8–12th 2015

Part I

Introduction

Security: Applications & Aspects

Cryptographic Features

Software vs Hardware Support

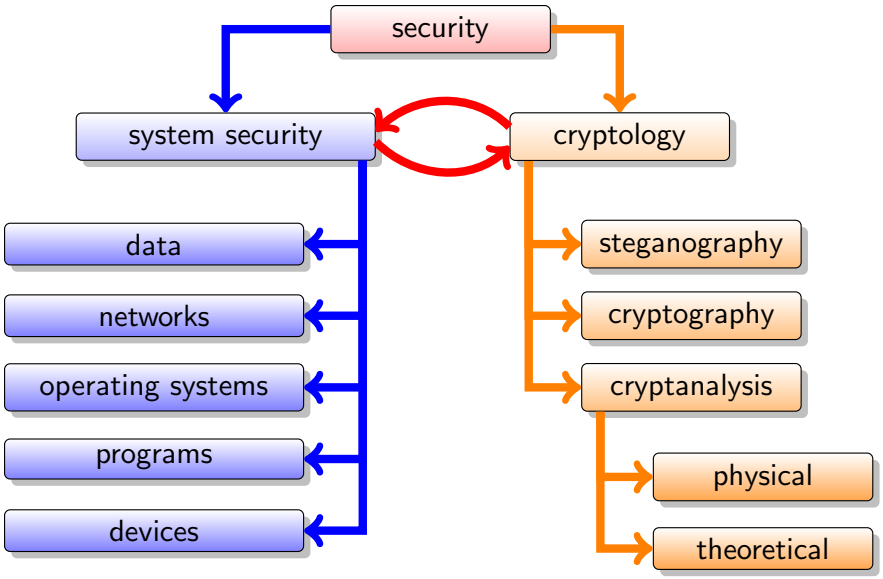
Hardware Acceleration Solutions

Summary

Applications with Security Needs

We need protections against:

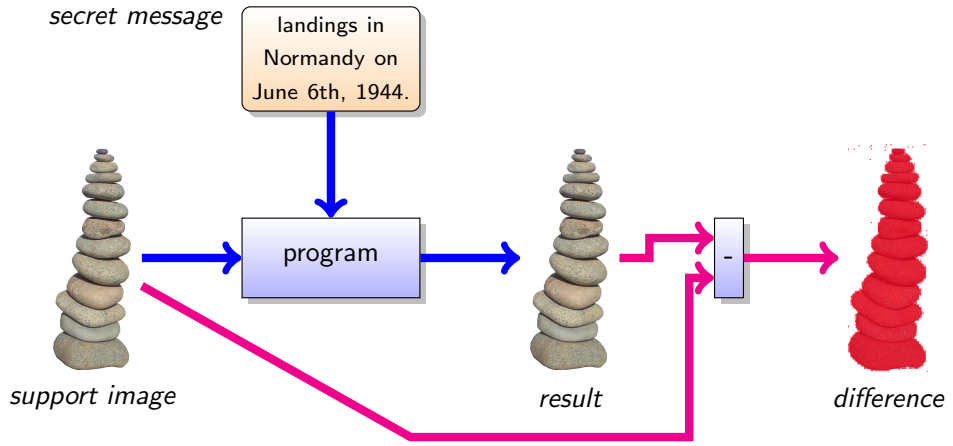
Security Aspects



Steganography

Cryptography: art of secret
Steganography: art of dissimulation

Principle: hide a secret message into another message (support)



Cryptographic Features

Objectives:

- Confidentiality
- Integrity
- Authenticity
- Non-repudiation
- ...

Cryptographic primitives:

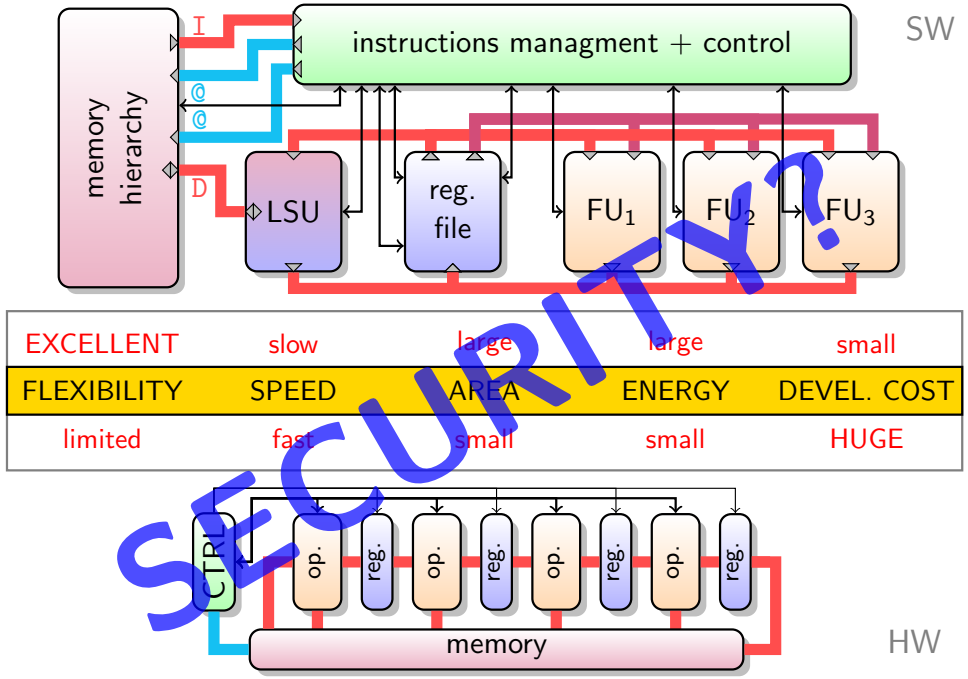
- Encryption
- Digital signature
- Hash function
- Random numbers generation
- ...

Implementation issues:

- Performances: speed, delay, throughput, latency
- Cost: device (memory, size, weight), low power/energy consumption, design
- Security: protection against attacks

Applications: smart cards, computers, Internet, telecommunications, set-top boxes, data storage, RFID tags, WSN, smart grids...

Software vs Hardware Support



Hardware Acceleration Solutions

At cluster/network/system level: (autonomous) dedicated processors

- Digital signal processors (DSPs)
- Network processors
- Multimedia processors
- Cryptographic processors

At computer level: co-processors and accelerators

- Dedicated cards for specific applications: video (GPU), audio, ...
- Cryptographic co-processors

At processor/core level: instruction set extensions

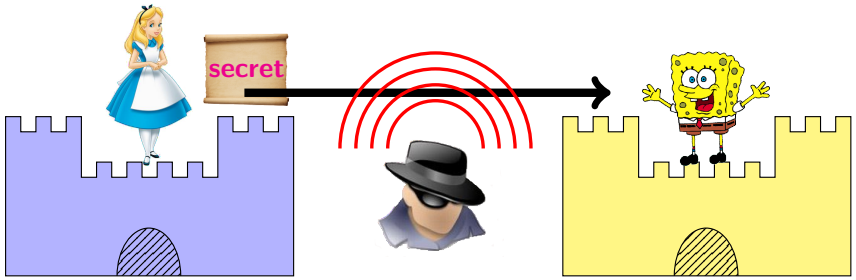
- Vector/matrix computations, SIMD, FMA, small floats, data shuffling, bit manipulation, cache interaction, prefetching
- Multimedia & signal processing applications
- Cryptographic extensions (AES, GF(2^m) multiplication)

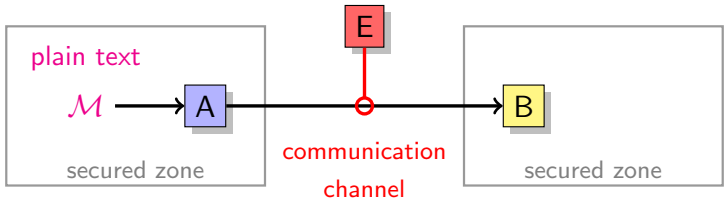
Part II

Security Background

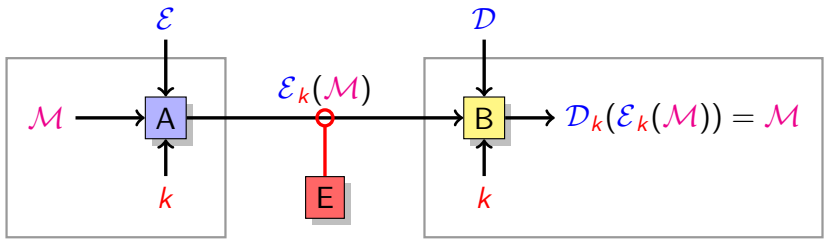
Basic Cyphering

Alice wants to secretly send a message to Bob in such a way Eve (eavesdropper/spy) should have **no** information



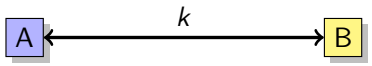
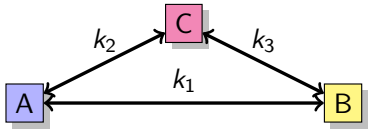
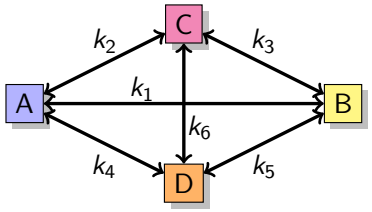


Symmetric / Private-Key Cryptography

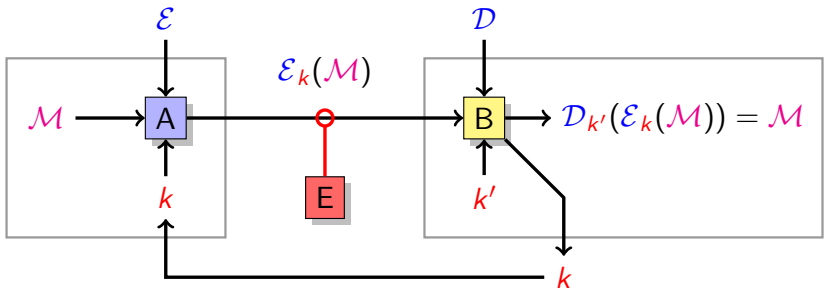


- A: Alice, B: Bob
- $\mathcal{M}$: plain text/message
- $\mathcal{E}$: encryption/ciphering algorithm, $\mathcal{D}$: decryption/deciphering algorithm
- k: secret key to be shared by A and B
- $\mathcal{E}_k(\mathcal{M})$: encrypted text
- $\mathcal{D}_k(\mathcal{E}_k(\mathcal{M}))$: decrypted text
- E: eavesdropper/spy

Symmetric Cryptography Limitation

people		required keys	
n	list	list	number
2	A, B		1
3	A, B, C		3
4	A, B, C, D		6
n	A,...		$\frac{n\times(n-1)}{2}$

Asymmetric / Public-Key Cryptography



- k: B's public key (known to everyone including E)
- $\mathcal{E}_k(\mathcal{M})$: ciphered text
- k': B's private key (must be kept secret)
- $\mathcal{D}_{k'}(\mathcal{E}_k(\mathcal{M}))$: deciphered text

Symmetric or Asymmetric Cryptography?

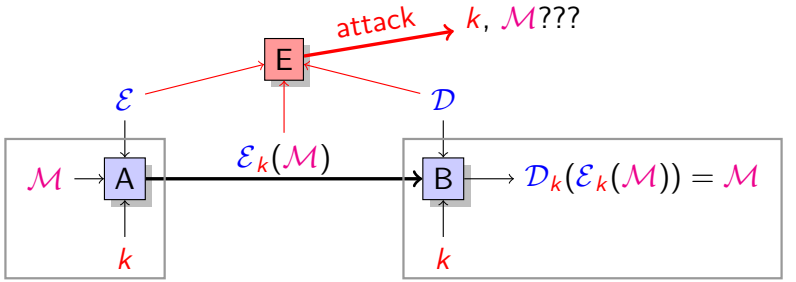
Private-key or symmetric cryptography:

- simple algorithms
  - → fast computation
  - → limited cost (silicon area, energy)
- ☹ requires a key exchange
- ☹ key distribution problem for n persons

Public-key or asymmetric cryptography:

- no key exchange
- only 2 keys per person (1 private, 1 public)
- allows digital signature
- more complex algorithms
  - → slower computation
  - → higher cost

Theoretical Attacks



Notations:

- $\mathcal{M}$ plain text
- $\mathcal{E}$ encryption algorithm
- $\mathcal{D}$ decryption algorithm
- k secret key
- $\mathcal{C} = \mathcal{E}_k(\mathcal{M})$ ciphered text
- ▭ secured zone

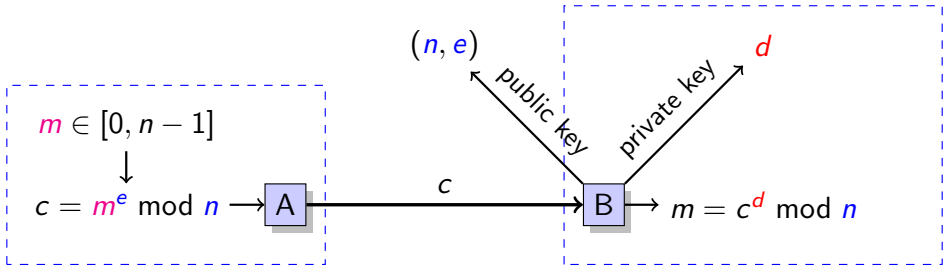
RSA 768 Attack in December 2009

6 months on 80 parallel computers ($\equiv$ 1 500 years for a single computer!)

RSA-768 =
33478071698956898786044169848212690817704794983713768568912431388982883793878002287614711652531743087737814467999489
×
36746043666799590428244633799627952632279158164343087642676032283815739666511279233373417143396810270092798736308917

Source: article
`http://eprint.iacr.org/2010/006.pdf`
Factorization of a 768-bit RSA modulus. Thorsten Kleinjung, Kazumaro Aoki, Jens Franke, Arjen K. Lenstra, Emmanuel Thome, Joppe W. Bos, Pierrick Gaudry, Alexander Kruppa, Peter L. Montgomery, Dag Arne Osvik, Herman te Riele, Andrey Timofeev, and Paul Zimmermann

RSA Ciphering (Rivest, Shamir, Adleman 1977)



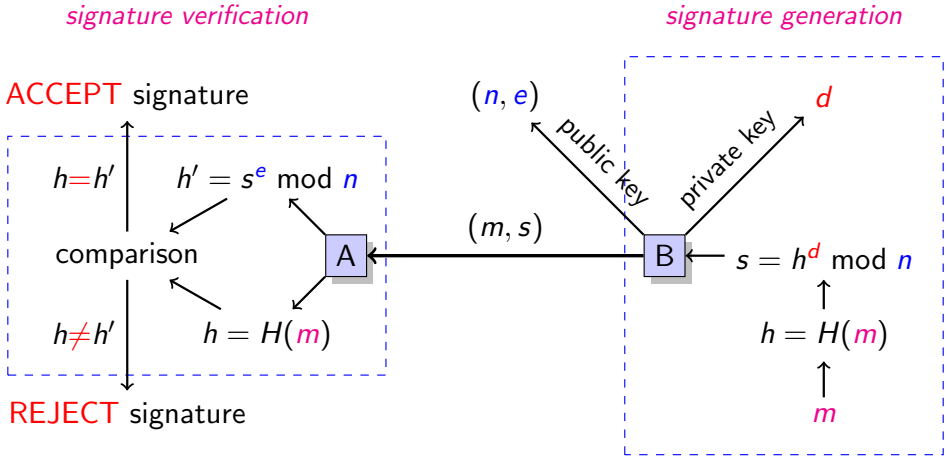
RSA key pair generation:

1. generate primes p and q (length $l/2$)
2. compute $n = pq$ and $\phi = (p-1)(q-1)$
3. select e such that $1 < e < \phi$ and $\gcd(e, \phi) = 1$
4. compute d satisfying $1 < d < \phi$ and $ed \equiv 1 \bmod \phi$

Security:

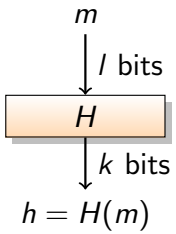
- integer factorization problem: compute (p,q) knowing just n is hard
- minimal key size recommendation: 1024 bits

RSA Signature



H a cryptographic hash function

Cryptographic Hash Functions (1/2)



- message m (arbitrary block of data)
- variable size l
- hash or digest h
- fixed size k (in practice $k << l$)

Security properties of cryptographic hash functions:

- preimage resistance (one way function): $h \nrightarrow m \mid h = H(m)$
- second preimage resistance[1]: $m_1 \nrightarrow m_2 \neq m_1 \mid H(m_1) = H(m_2)$
- collision resistance: finding (m_1, m_2) such that $m_1 \neq m_2$ and $H(m_1) = H(m_2)$ is very hard

Examples: MD5, WHIRLPOOL, SHA-1, SHA-2, SHA-3 (selection 2010)

Cryptographic Hash Functions (2/2)

Examples using `openssl`:

```
> echo "string" | openssl dgst -sha224 -hex
```

string	digest
0123456789	95ae4be607f065743ac1c81d1180591a919d08b8d4765e176b26f214
1123456789	5c527cd1341a4338f09086e71d1a0d69f818d74a828c974b9433524a
0123446789	94e5aa1d275dc3a21c76d28b011f4ea6121fa228af3ec7fa329da44f
01234567 8	b04c6b0b1d663ad0c00d749441747cc6df211ea6c98f4fd2dbf283ff

[Trapdoor] One Way Function

One way function: $f : x \mapsto y = f(x)$

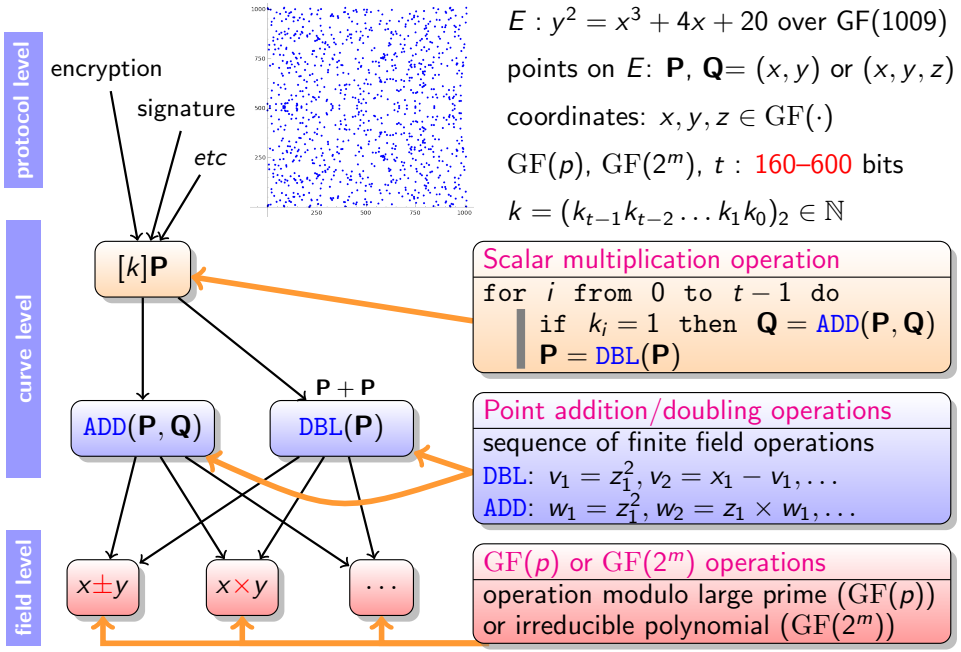
- given x, computing y is easy
- given y, computing x is very hard

Trapdoor one way function: $f : x \mapsto y = f(x)$

- given x, computing y is easy
- given y, computing x is very hard
- given some (secret) information and y, computing x is easy

Example: p and q primes, computing $n = pq$ is easy but finding (p, q) knowing just n is very hard

Elliptic Curve Cryptography (ECC)



$E : y^2 = x^3 + 4x + 20$ over GF(1009)

points on E: $\mathbf{P}$, $\mathbf{Q}$= (x, y) or (x, y, z)

coordinates: $x, y, z \in \mathrm{GF}(\cdot)$

$\mathrm{GF}(p)$, $\mathrm{GF}(2^m)$, t : 160–600 bits

$k = (k_{t-1}k_{t-2} \ldots k_1 k_0)_2 \in \mathbb{N}$

Scalar multiplication operation

```
for i from 0 to t − 1 do
    if k_i = 1 then Q = ADD(P, Q)
    P = DBL(P)
```

Point addition/doubling operations

sequence of finite field operations

DBL: $v_1 = z_1^2, v_2 = x_1 - v_1, \ldots$

ADD: $w_1 = z_1^2, w_2 = z_1 \times w_1, \ldots$

$\mathrm{GF}(p)$ or $\mathrm{GF}(2^m)$ operations

operation modulo large prime ($\mathrm{GF}(p)$)
or irreducible polynomial ($\mathrm{GF}(2^m)$)

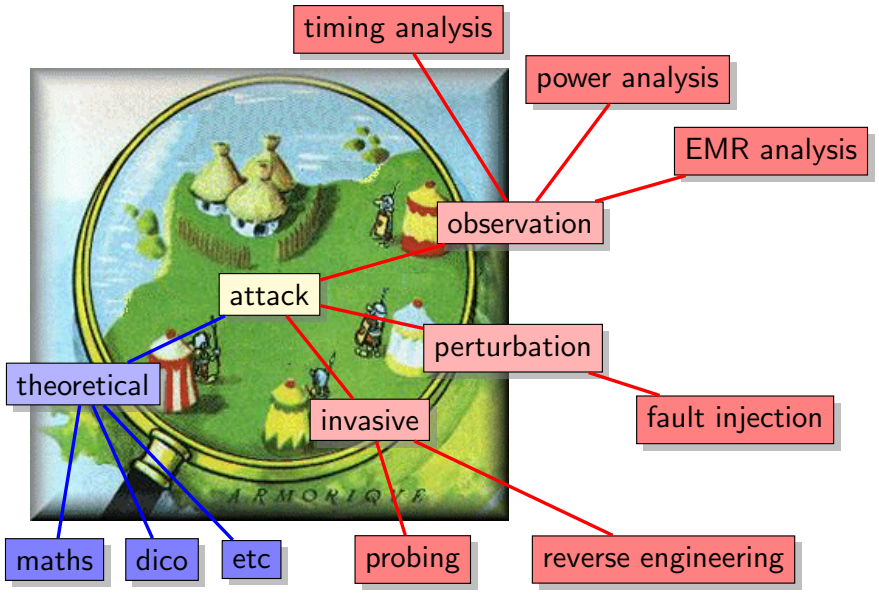
Key Size vs Security Level

security level	RSA	ECC	
		$\mathrm{GF}(p)$	$\mathrm{GF}(2^m)$
	$\lvert n\rvert$ [bits]	$\lvert p\rvert$ [bits]	m [bits]
56	512	112	113
64	704	128	131
80	1024	160	163
96	1536	192	193
112	2048	224	233
128	3072	256	283
192	7680	384	409
256	15360	521	571

- Security level of h: the best known algorithm takes 2^h steps for breaking the cryptosystem
- RSA: $\mathbb{Z}/n\mathbb{Z}$ with $n = pq$, p and q primes
- ECC: $\mathrm{GF}(p)$ with p prime or $\mathrm{GF}(2^m)$

Source: SEC2 recommendations from Certicom (v1.0, Jan. 2000)

Various Types of Attacks



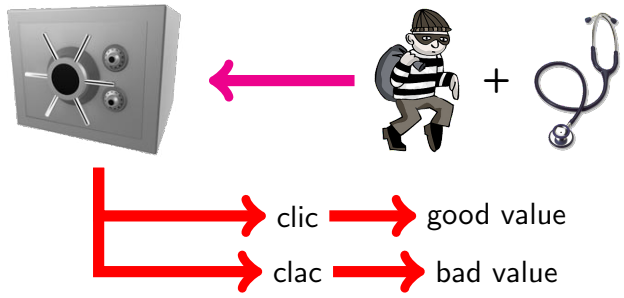
EMR = Electromagnetic radiation

Side Channel Attacks

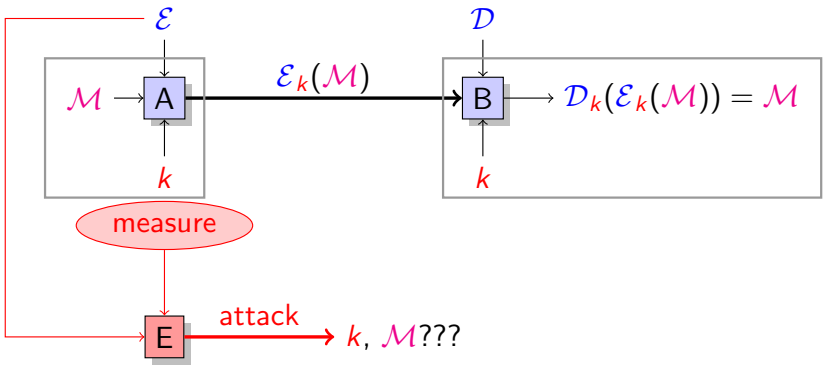
Attack: attempt to find, without any knowledge about the secret:

- the message (or parts of the message)
- informations on the message
- the secret (or parts of the secret)

"Old style" side channel attacks:



Side Channel Analysis/Attacks (SCA)



General principle: measure external parameter(s) on running device in order to deduce internal informations

What Should be Measured?

Answer: everything that can "enter" and/or "get out" in/from the device

- power consumption
- electromagnetic radiation
- temperature
- sound
- computation time
- number of cache misses
- number and type of error messages
- ...

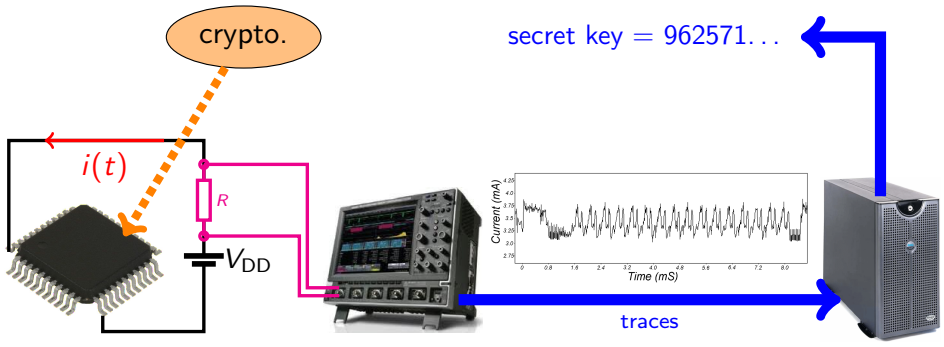
The measured parameters may provide informations on:

- global behavior (temperature, power, sound...)
- local behavior (EMR, # cache misses...)

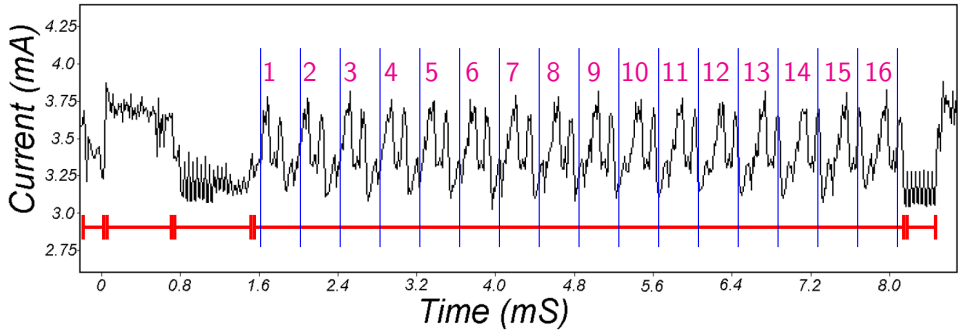
Power Consumption Analysis

General principle:

1. measure the current $i(t)$ in the cryptosystem
2. use those measurements to "deduce" secret informations



"Read" the Traces



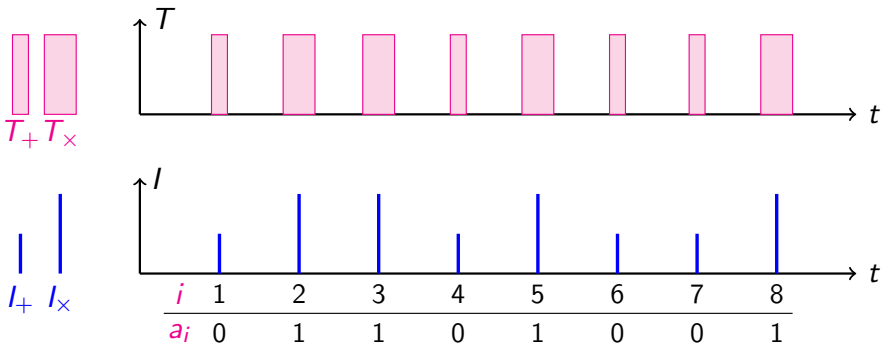
- algorithm → decomposition into steps
- detect loops
  - constant time for the loop iterations
  - non-constant time for the loop iterations

Source: [8] Kocher, Jaffe and Jun. **Differential Power Analysis**, Crypto99

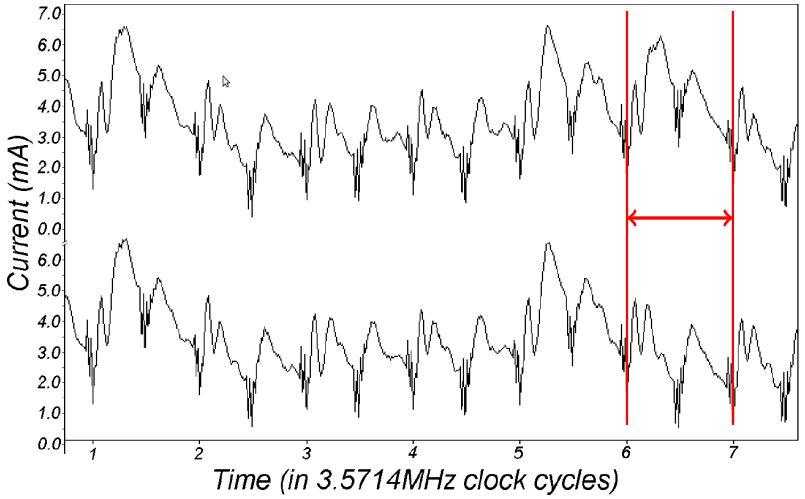
Differences & External Signature

An algorithm has a current signature and a time signature:

```
r = c0
for i from 1 to n do
    if a_i = 0 then
        r = r+c1
    else
        r = r×c2
```



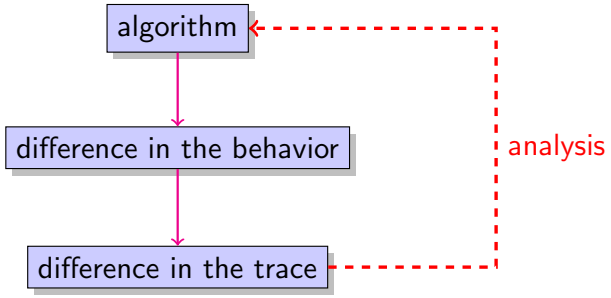
Simple Power Analysis (SPA)



Source: [8]

SPA in Practice

General principle:

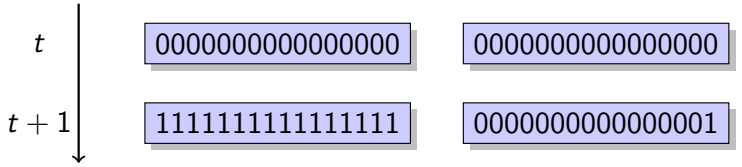


Methods: interpretation of the differences in

- control signals
- computation time
- operand values
- ...

Limits of the SPA

Example of behavior difference: (activity into a register)

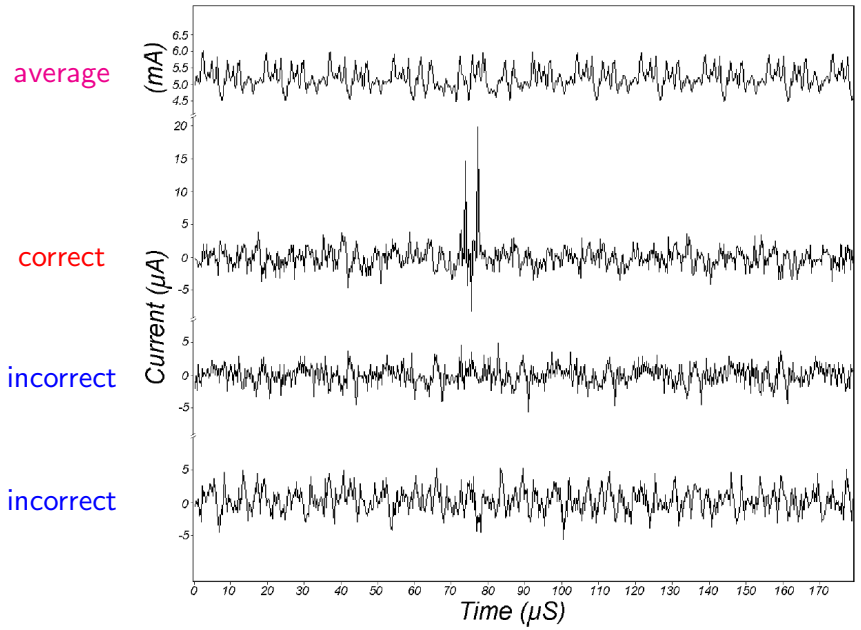


Important: a small difference may be evaluated has a noise during the measurement → traces cannot be distinguished

Question: what can be done when differences are too small?

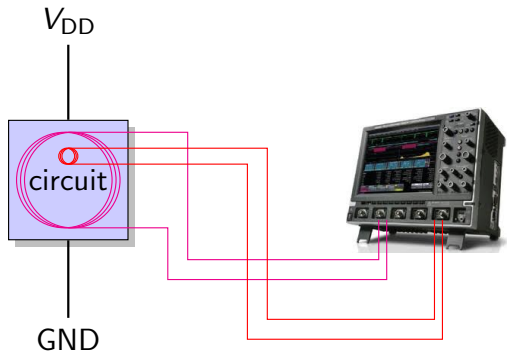
Answer: use statistics over several traces

DPA Example



Electromagnetic Radiation Analysis (1/2)

General principle: use a probe to measure the EMR



EMR measurement:

- global EMR with a large probe
- local EMR with a microprobe

Electromagnetic Radiation Analysis (2/2)

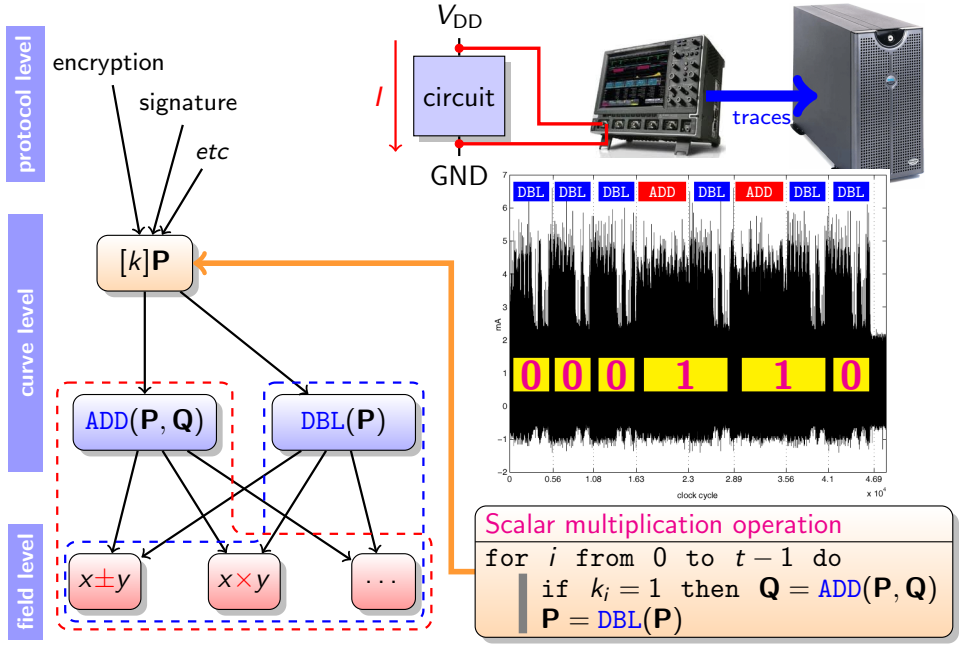
EMR analysis methods:

- simple electromagnetic analysis: SEMA
- differential electromagnetic analysis: DEMA

Local EMR analysis may be used to determine internal architecture details, and then select weak parts of the circuit for the attack

→ X-Y table

Basic Power Analysis Attack on ECC



Random Number Generators (RNG)

Pseudo random number generator (PRNG):

- deterministic algorithms
- very high throughput and good statistical properties
- various algorithms → quality/throughput/cost tradeoffs

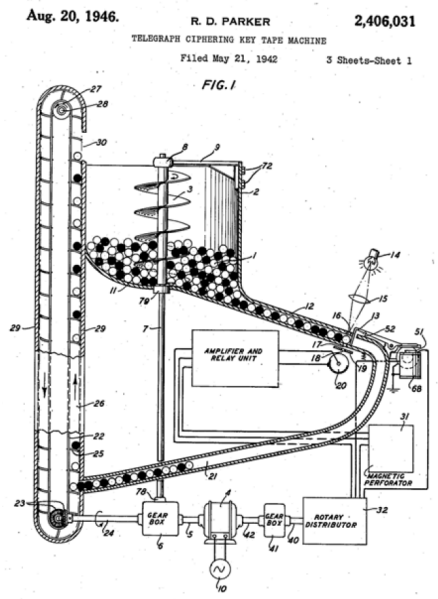
True random number generator (TRNG):

- non-deterministic algorithms (physical random source)
- limited throughput
- quality = *func*(environment parameters, ...) → attacks

Hybrid random number generator (HRNG):

- HRNG = TRNG + PRNG
- very high speed and very good quality
- selection needs more research

Historical Hardware TRNGs



ATT Patent 1946, source: P. Kohlbrenner

TRNGs Selection

Physical noise source:

- quantum physics
- radioactive decay
- atmospheric noise
- thermal/Johnson noise
- jitter in ring oscillator sampling
- meta-stability
- noises in circuits: 1/f, shot, popcorn, crosstalk, ...
- ...

Characteristics:

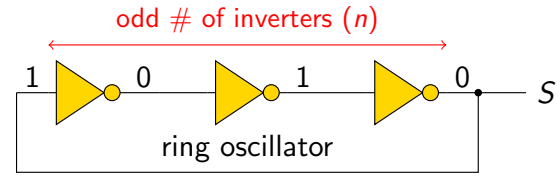
- throughput (? Mb/s)
- randomness quality (bias, entropy/bit, stability, effects of environment variations, ...)
- security → fully integrated in the chip
- cost (silicon area, power consumption)

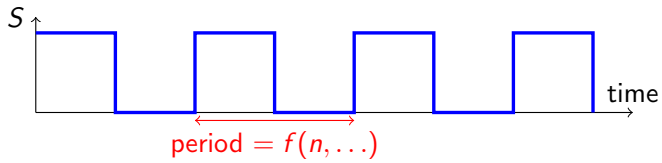
VLSI implement.

Free Running Ring Oscillator

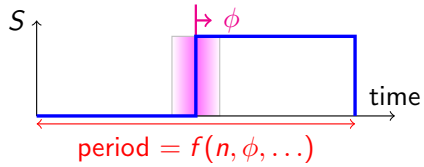
inverter:

in	out
0	1
1	0

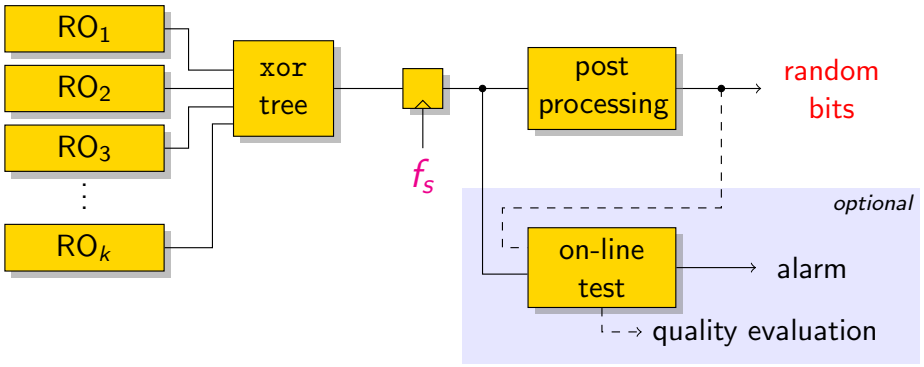




ϕ random jitter
(*timing/phase instability*)



Example of Ring Oscillator (RO) Based TRNG



Description:

- k free running ring oscillators
- f_s is the sampling frequency
- post processing: enhance statistical parameters
- on-line quality test (environment variations, attacks, ...)

Post Processing

Purpose: enhance statistical parameters of the output sequence

- reduce bias $Pr(x = 1) = 0.5 + \epsilon$ (AIS 31: $\epsilon < 0.0173$)
- increase entropy per bit (the real randomness)

Typical post processing methods:

- Von Neumann correction

input bits	(0,0)	(0,1)	(1,0)	(1,1)
output bit	*none*	1	0	*none*

- Linear feedback shift register (LFSR)
- Hash function (e.g. SHA)
- Ciphering (e.g. AES)
- Resilient function (e.g. error code computations)
- ...

Trade-off: entropy per bit, data rate, cost, quality

RO Based TRNG Example

[11]

Description:

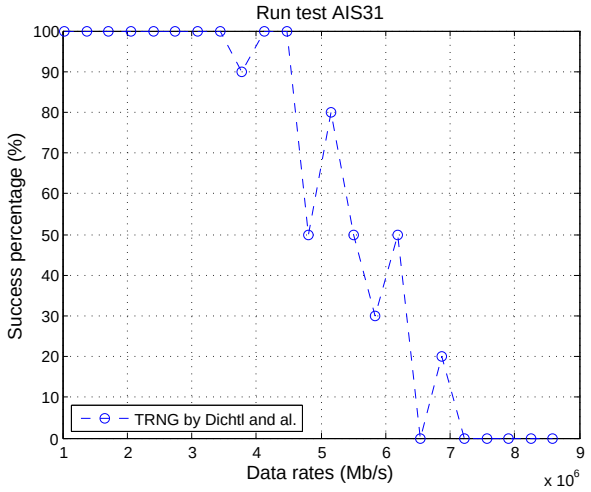
- $k = 114$ RO of 13 inverters
- resilient function: BCH(256, 13, 113) code
- mathematical model (but not realistic assumptions)
- data rate 2.5 Mb/s on FPGA

Problems:

- very complex calibration (external measurement of the jitter!!!)
- too many transitions in the `xor` tree
- setup/hold violations in the flip-flop
- ...

Example of Measurements on FPGAs

TRNG from [6] (Altera Stratix II):



Part III

Processors and Co-Processors

Cryptographic Processors

Cryptographic Co-Processors & Accelerators

Trusted Platform Module (TPM)

Cryptographic Processors

- Mechanical devices

  Example 1: Confederate Cipher Disc used during the American Civil War (1861-1865)

  Example 2: CD-57 portable, produced in 1957

- Electromechanical devices

  Example: Enigma used during the 2nd World War

- Electronic circuits

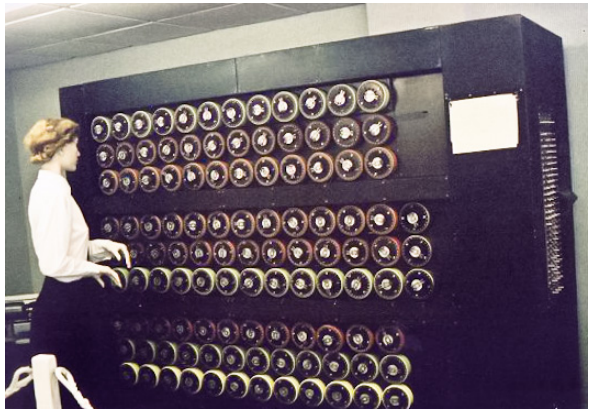
  1970s for bank applications

- Tomorrow?

Images sources: `http://fr.wikipedia.org/` & `http://cryptomuseum.com/`

The Bombe

Electromechanical device designed for deciphering (*i.e.* breaking) Enigma



Bletchley Park (`http://www.bletchleypark.org.uk/`)

Image source: `http://fr.wikipedia.org/`

IBM 4765 PCIe Cryptographic Coprocessor

AES, DES, TDES

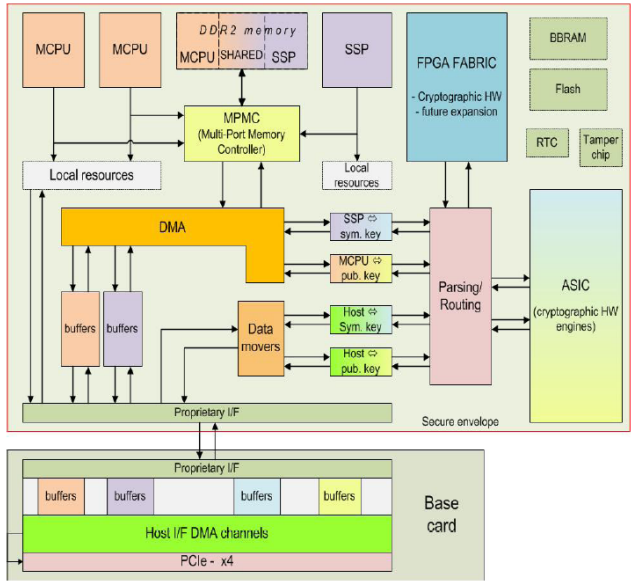
RSA sign. $\leq$4096b

ECDSA sign. p521

SHA-1, SHA-2, ...

key management

...



Images source: `https://www-03.ibm.com/security/cryptocards/pciecc/pdf/PCIe_Spec_Sheet.pdf`

NIST certification: `http://csrc.nist.gov/groups/STM/cmvp/documents/140-1/140sp/140sp1505.pdf`

Trusted Platform Module (TPM)

Hardware device for authentication and accreditation of the platform

- Boot process integrity: → system start-up is tamper free

  Verification of (many) stored measurements from previous boots
  Verification of code & behavior: BIOS, chip-set, peripherals, firmwares, boot loader, kernel, ...

- Data protection: → robust against software and physical attacks

  Security keys, passwords, certificates

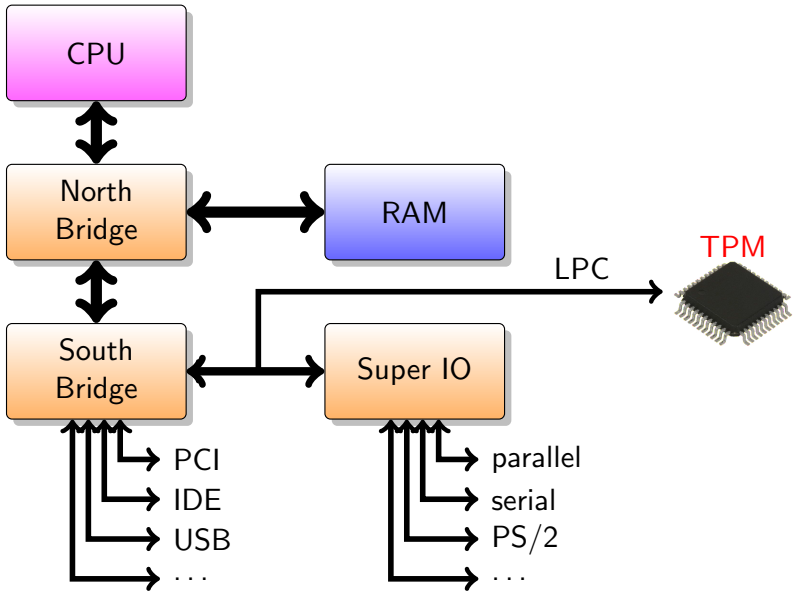
- Improved Security support for operating system:

  Encryption, hash functions, RNG, key generation & management
  Memory protection, session isolation, protected partition, security support for virtual machines

- Device physically locked to the motherboard

Specifications: Trusted Computing Group (TCG, created in 1999)
`http://www.trustedcomputinggroup.org/`

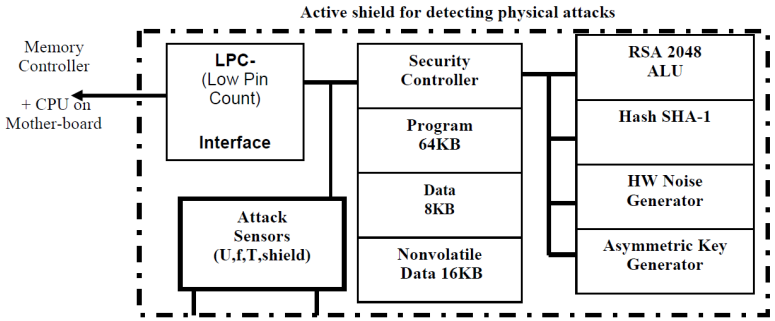
TPM Integration in a PC Platform



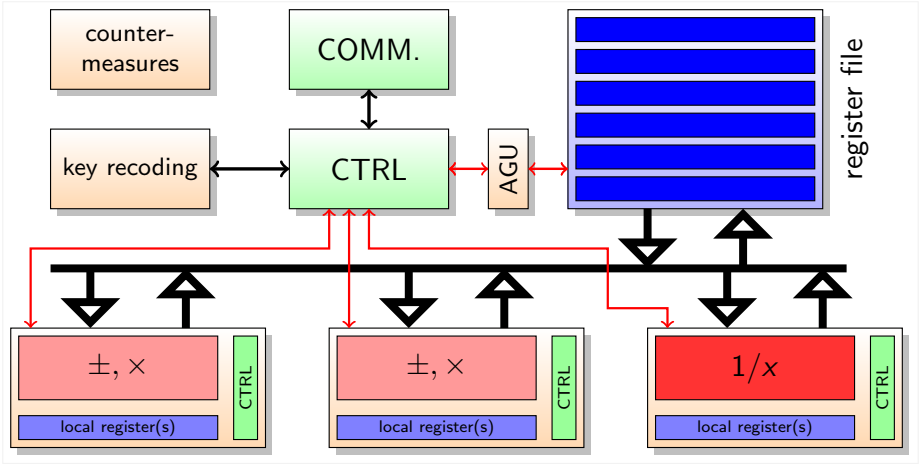
LPC: low pin count

TPM Example from Infineon

TPM Block diagram (from Infineon white paper [5]):



Our ECC (Co)Processor



- Functional units: $\pm, \times, 1/x$ for $\mathrm{GF}(p)$ or $\mathrm{GF}(2^m)$, key recoding
- Memory: main register file + internal registers in FUs
- Control: operations (curve and field levels) schedule, parameters management, active countermeasures...

Part IV

Instruction Set Extensions

Instruction Set

Programming interface of a processor:

- Instruction set (or instruction set architecture)
  - Control flow operations
  - Computations operations (ALU, floating-point)
  - Memory and data handling operations
- Registers features and organization
- Memory mapping
- Virtual memory support
- Virtual machines support
- Interruption and exception handling
- Permissions
- I/O space organization
- ...

Instruction Set Extensions

Objectives:

- Improve efficiency for (useful) operations and memory handling:
  multimedia, signal processing, codecs, cryptography, ...
- Increase internal parallelism:
  vector, matrix, SIMD (single instruction multiple data)
- Efficient support for specific operations:
  dedicated hardware operators

Programming models:

- Compiler assisted generation:
  compiler identifies patterns to be mapped on the extended IS
- Optimized library based design:
  replace a standard and generic library by a specific library for the target processor extended IS
- Intrinsics:
  access to low-level instructions from a high-level programming language

Examples of Instruction Set Extensions

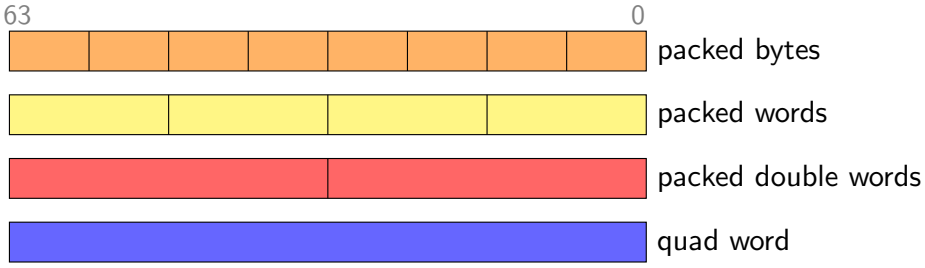
For x86 architectures:

- MMX (1996, Intel) +57 instructions and +8 registers 64b
- 3D Now (1997, AMD,) +21 instructions and +8 registers 64b (FP/MMX)
- SSE (1999, Intel) +70 instructions and +8 registers 128b
- SSE2 (2001, Intel) +144 instructions and +8 registers 128b
- SSE3 (2004, Intel) +13 instructions
- SSE4 (2006, Intel) +54 instructions
- AVX (2008, Intel) +12 instructions and registers 128→256b
- AES (2008)
- F16C (2009, AMD)
- XOP (2009, AMD)
- FMA (2011)
- BMI (2012)

Instruction set extensions have been proposed for other architectures: MAX-1 for PA-RISC, VIS for Sparc, AltiVec for Apple-IBM-Motorola), MIPS-3D for MIPS, NEON for ARM, ...

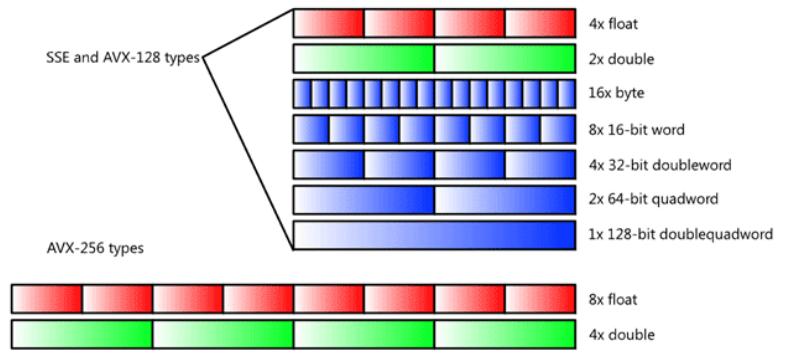
MMX Example

- 8 new 64b registers (in 80b floating-point ones): `MM0`, `MM1`, ..., `MM7`
- 4 new vector data types:



- 57 new instructions: (examples)
  - logic: `por`, `pand`, `pxor`, `psrlw`, `psrld`, `psrlq`, `psraw`, `psraq`, ...
  - maths: `paddb`, `paddsb`, `paddusb`, `paddw`, `paddsw`, `paddusw`, `paddd`, `pmulhw`, `pmullw`, ...
  - comparisons: `pcmpeqb`, `pcmpeqw`, ...
  - data movement: `movd`, `movq`, ...
  - data packing: `packsswb`, `packssdw`, `punpckhbw`, ...

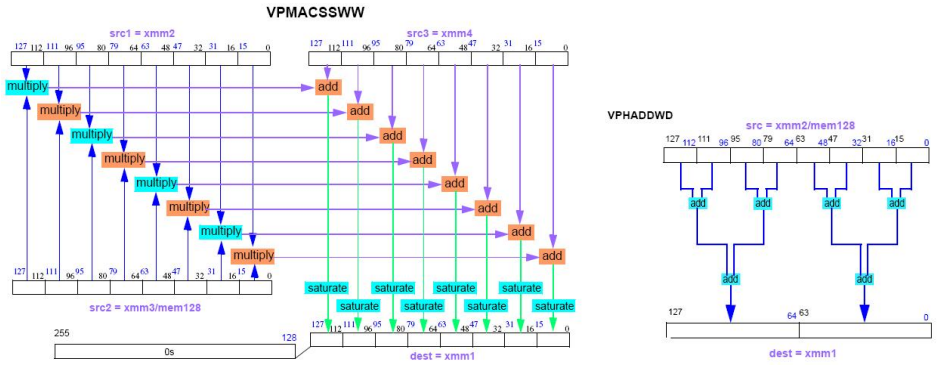
AVX Example



Example of new instructions:

- `VBROADCASTSS`, `VBROADCASTSD`, `VBROADCASTF128`: broadcast 32b, 64b or 128b word to ALL elements
- permutations
- shuffles

XOP Example



Images source: `https://chessprogramming.wikispaces.com/XOP`

Addition of Long Operands

Addition of long operands is very important in:

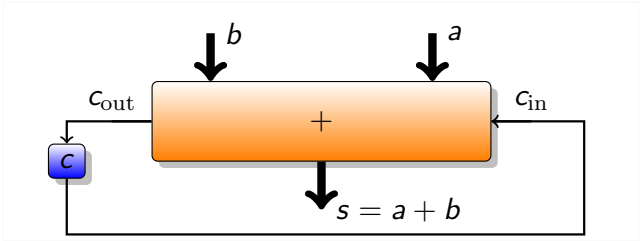
- multiprecision arithmetic, *e.g.* GMP, MPRI, MPFR, MPFI libraries
- asymmetric cryptography:
  - RSA 1024–8192 bits integers and modular arithmetic
  - ECC 160–600 bits finite fields elements $\mathrm{GF}(p)$ and $\mathrm{GF}(2^m)$
  - Fully Homomorphic Encryption ($> 10^5$ bits integers or polynomials)
  - El Gamal signature, Diffie-Hellman key exchange, ...

Addition of long integers is not efficient using a standard ISA:

- Addition of w-bit words ($w \in \{32, 64\}$) produces a w-bit sum
- Carry out (c_{out}) handled as a flag (not an accessible value)
- Bad branch prediction since $\mathrm{Proba}(c_{\text{out}} = 1) \approx \frac{1}{2}$

Hardware Extension for Large Additions

Proposed solution: local flip-flop to store carries "between" sub-words



- c_{out} produced at step i is used as c_{in} at step $i + 1$
- very cheap, no more control issues
- $c_{\text{in}} = 0$ for the first sub-word
- New instruction `ADC` (*i.e.* add with carry)

Problem: how can I use this instruction from a high level programming language?

Software Usage of ADC Instruction

Programming models:

- At assembly language level: explicit use of the `ADC` instruction

  not very popular

- At compiler level:

  ```
  s = a + b + (c==1 ?  1 :  0)
  ```

  is identified as a call to the `ADC` instruction

  does not work!

- Using intrinsic: fake function call (replaced by assembler code)

```
unsigned char _addcarry_u64 (
        unsigned char c_in,
        unsigned __int64 a,
        unsigned __int64 b,
        unsigned __int64 * out )
```

Hyper Short Introduction to $\mathrm{GF}(2^m)$ Arithmetic

- Element of the field: $A = \sum_{i=0}^{m-1} a_i x^i$ with $a_i \in \{0, 1\}$

$$A = 1 \cdot x^3 + 1 \cdot x^2 + 0 \cdot x + 1 \text{ and } B = 0 \cdot x^3 + 1 \cdot x^2 + 1 \cdot x + 1$$

A: | 1 | 1 | 0 | 1 | B: | 0 | 1 | 1 | 1 |

- Field addition: component wise addition in GF(2) (*i.e.* bit wise XOR)

$A + B$: | 1 | 0 | 1 | 0 |

- Field multiplication: polynomial multiplication

$A \times B$: | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |

In many cases, we need arithmetic modulo an irreducible polynomial

Extensions for $\mathrm{GF}(2^m)$ Arithmetic

SSE2-4, AVX support:

- `PXOR` instruction: 128-bit exclusive OR
  → addition in $\mathrm{GF}(2^m)$

- `PCLMULQDQ` instruction: 64-bit $\times$ 64 $\rightarrow$ 128-bit product
  → product in $\mathrm{GF}(2^m)$

  Carry-less multiplication

  Other instructions: `PCLMULLQLQDQ`, `PCLMULHQLQDQ`, `PCLMULLQHQDQ`, `PCLMULHQHQDQ`

- inversion in the field: euclidean algorithm with $\mathrm{GF}(2^m)$ addition, bit manipulation instructions

Future extensions (?): 256 $\times$ 256 $\rightarrow$ 512 bits

Advanced Encryption Standard (AES)

Established by NIST in 2001

Symmetric encryption

Block size: 128 bits

key length	#round
128	10
192	12
256	14

Based on substitution-permutation network

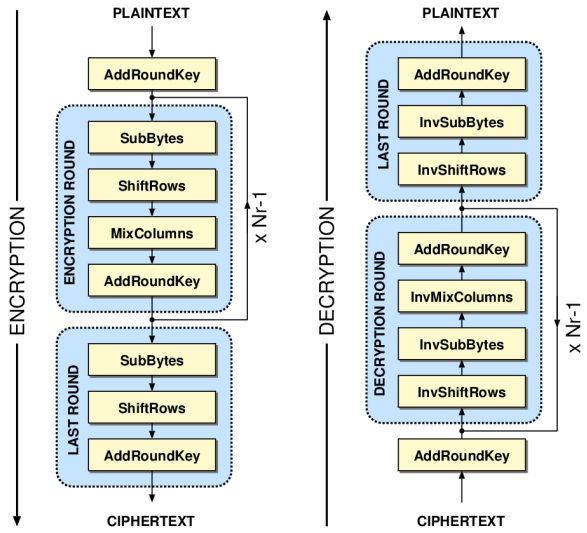
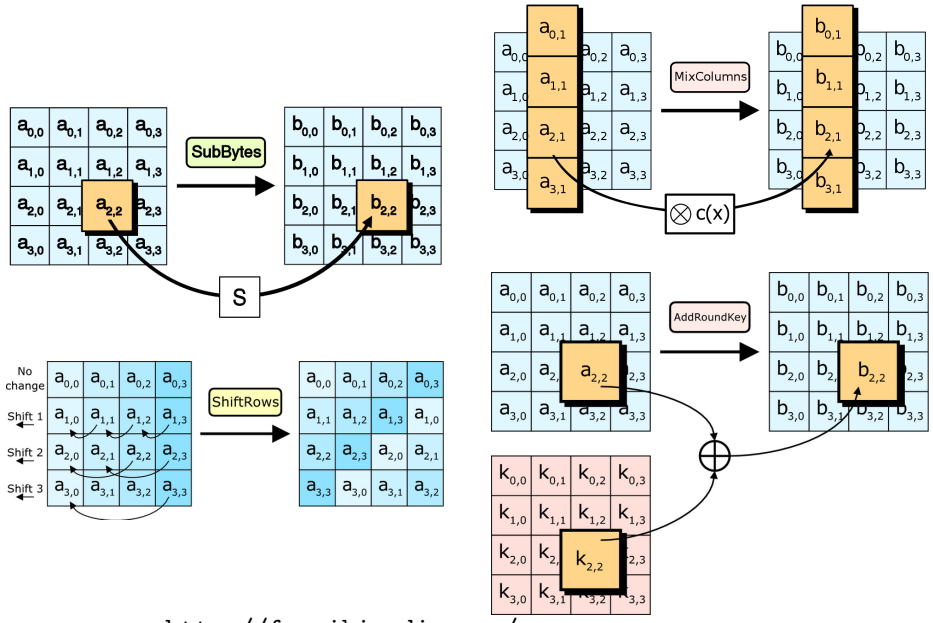


Image source: `http://fr.wikipedia.org/`

NIST: National Institute of Standards and Technology

AES Round Operations



Images source: `http://fr.wikipedia.org/`

Instruction Set Extension for AES

- `AESKEYGENASSIST` key generation (partial operations)
- `AESENC` one round of encryption
- `AESENCLAST` last round of encryption
- `AESDEC` one round of decryption
- `AESDECLAST` last round of decryption
- `AESIMC` inverse mix columns
- `PCLMULQDQ` $\mathrm{GF}(2^m)$ multiplication (carry less)

Example

Source:

Shay Gueron

Intel's New AES Instructions for Enhanced Performance and Security

Proc. Fast Software Encryption (FSE) 2009

`http://iacr.org/archive/fse2009/56650054/56650054.pdf`

```
void AES_128_CBC_Decrypt_C_4_blocks (...) {
    __m128i RKEY_DECRYPT [11];
    __m128i tmp1, tmp2, tmp3, tmp4, feedback;
    __m128i z1, z2, z3, z4;
    int j, k;
    for (k=0; k<11; k++) {
        RKEY_DECRYPT [10-k] =
        _mm_load_si128 ( (__m128i*)&Key_Schedule_Decrypt [4*k]);
    }
    feedback = _mm_load_si128 ( (__m128i*)&IV [0]);

    z1 = _mm_load_si128 ( (__m128i*)&CIPHERTEXT[0]);
    z2 = _mm_load_si128 ( (__m128i*)&CIPHERTEXT[4]);
    z3 = _mm_load_si128 ( (__m128i*)&CIPHERTEXT[8]);
    z4 = _mm_load_si128 ( (__m128i*)&CIPHERTEXT[12]);

    tmp1 = _mm_xor_si128(z1,RKEY_DECRYPT[0]);
    tmp2 = _mm_xor_si128(z2,RKEY_DECRYPT[0]);
    tmp3 = _mm_xor_si128(z3,RKEY_DECRYPT[0]);
    tmp4 = _mm_xor_si128(z4,RKEY_DECRYPT[0]);

    for(j=1; j <10; j++) {
        tmp1 = _mm_aesdec_si128 (tmp1, RKEY_DECRYPT [j]);
        tmp2 = _mm_aesdec_si128 (tmp2, RKEY_DECRYPT [j]);
        tmp3 = _mm_aesdec_si128 (tmp3, RKEY_DECRYPT [j]);
        tmp4 = _mm_aesdec_si128 (tmp4, RKEY_DECRYPT [j]);
    }
    tmp1 = _mm_aesdeclast_si128 (tmp1, RKEY_DECRYPT [10]);
    tmp2 = _mm_aesdeclast_si128 (tmp2, RKEY_DECRYPT [10]);
    tmp3 = _mm_aesdeclast_si128 (tmp3, RKEY_DECRYPT [10]);
    tmp4 = _mm_aesdeclast_si128 (tmp4, RKEY_DECRYPT [10]);

    tmp4 = _mm_xor_si128(tmp4,z3);
    tmp3 = _mm_xor_si128(tmp3,z2);
    tmp2 = _mm_xor_si128(tmp2,z1);
    tmp1 = _mm_xor_si128(tmp1,feedback);

    _mm_store_si128 ((__m128i*)&DECRYPTED_TEXT[0], tmp1);
    _mm_store_si128 ((__m128i*)&DECRYPTED_TEXT[4], tmp2);
    _mm_store_si128 ((__m128i*)&DECRYPTED_TEXT[8], tmp3);
    _mm_store_si128 ((__m128i*)&DECRYPTED_TEXT[12], tmp4);
}
```


Hardware Support for Security: Pros/Cons

At cluster/network/system level: (autonomous) dedicated processors

- ☺ very high security (isolation, strong storage, protections against SCAs)
- ☹ cost, not flexible, requires system level integration (client/server prg. model)

At computer level: co-processors and accelerators

- ☺ very high security (isolation, protections against SCAs)
- ☹ not flexible, requires computer level integration

At processor/core level: instruction set extensions

- ☺ flexible
- ☹ security against SCAs

Conclusion & Future Prospects

Security support in processors:

- Requires hardware blocks (software is not secure enough)
- Use secured libraries at OS, cryptographic and application levels
- Important features: isolation, authentication, identification, strong cryptographic primitives (cipher, hash, RNG, key management)
- Important threats: attacks at ALL levels (protocols, software, IP, OS, maths, physical...)

Future security support:

- Advanced TPMs
- Advanced security co-processors, accelerators, IPs
- End-to-end trust chain
- Security solutions vs economic models vs social aspects

  privacy protection, anti-trust, small vs huge industries, ...
- ...

References I

M. Alioto, L. Giancane, G. Scotti, and A. Trifiletti.
Leakage power analysis attacks: A novel class of attacks to nanometer cryptographic circuits.
IEEE Transactions on Circuits and Systems I, 57(2):355–367, February 2010.

R. Anderson, M. Bond, J. Clulow, and S. Skorobogatov.
Cryptographic processors – a survey.
Technical report, University of Cambridge, Computer Laboratory, Cambridge, UK, August 2005.
Paper version [3].

R. Anderson, M. Bond, J. Clulow, and S. Skorobogatov.
Cryptographic processors – a survey.
Proceedings of the IEEE, 94(2):357–369, February 2006.
Research report [2].

H. Bar-El, H. Choukri, D. Naccache, M. Tunstall, and C. Whelan.
The sorcerer's apprentice guide to fault attacks.
Proceedings of the IEEE, 94(2):370–382, February 2006.

H. Brandl and T. Rosteck.
Technology, implementation and application of the trusted computing group standard (TCG).
White paper, Infineon, 2004.

References II

M. Dichtl and J. D. Golic.
High-speed true random number generation with logic gates only.
In *Proc. Cryptographic Hardware and Embedded Systems (CHES)*, volume 4727 of *LNCS*, pages 45–62. Springer, September 2007.

P. C. Kocher.
Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems.
In *Proc. Advances in Cryptology (CRYPTO)*, volume 1109 of *LNCS*, pages 104–113. Springer, August 1996.

P. C. Kocher, J. Jaffe, and B. Jun.
Differential power analysis.
In *Proc. Advances in Cryptology (CRYPTO)*, volume 1666 of *LNCS*, pages 388–397. Springer, August 1999.

F. Koeune and F.-X. Standaert.
A tutorial on physical security and side-channel attacks.
In *5th International School on Foundations of Security Analysis and Design (FOSAD)*, volume 3655 of *LNCS*, pages 78–108. Springer-Verlag, 2005.

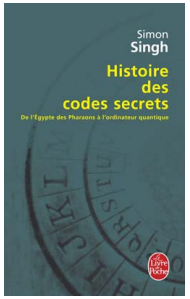
L. Lin and W. Burleson.
Leakage-based differential power analysis (LDPA) on sub-90nm CMOS cryptosystems.
In *Proc. IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 252–255, Seattle, WA, USA, May 2008.

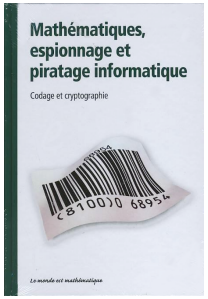
References III

B. Sunar, W. J. Martin, and D. R. Stinson.
A provably secure true random number generator with built-in tolerance to active attacks.
IEEE Transactions on Computers, 56(1):109–119, January 2007.

Good Books (in French)

Histoire des codes secrets
Simon Singh
1999
Livre de poche

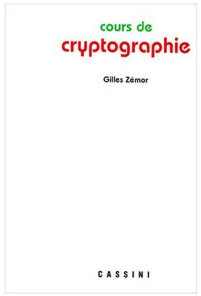




Mathématiques, espionnage et piratage informatique
Joan Gomez
2010
Le monde est mathématique, RBA

Good Books (in French)

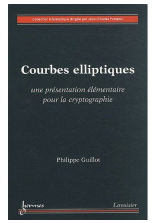
Cryptographie appliquée
Bruce Schneier
1997, 2ème édition
Wiley
ISBN: 2–84180–036–9



Cours de cryptographie
Gilles Zémor
2000
Cassini
ISBN: 2–84225–020–6

Good Books (in French)

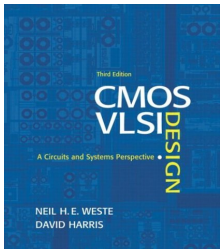
Courbes elliptiques
Philippe Guillot
2010
Hermes
ISBN: 978-2-7462-2392-9

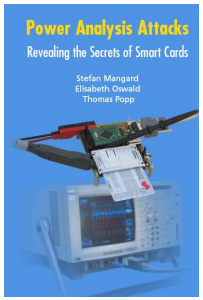


Micro et nano-électronique
Bases, Composants, Circuits
Hervé Fanet
2006
Dunod
ISBN: 2–10–049141–5

Good Books (in English)

CMOS VLSI Design
A Circuits and Systems Perspective
Neil Weste and David Harris
3rd edition, 2004
Addison Wesley
ISBN: 0–321–14901–7

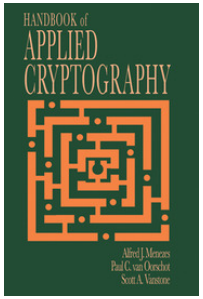




Power Analysis Attacks
Revealing the Secrets of Smart Cards
Stefan Mangard, Elisabeth Oswald and
Thomas Popp
2007
Springer
ISBN:978-0-387-30857-9

Good Books (in English)

Handbook of Applied Cryptography
Alfred J. Menezes, Paul C. van Oorschot and
Scott A. Vanstone
2001
CRC Press
ISBN:0-8493-8523-7
Web: `http://cacr.uwaterloo.ca/hac/`



The end, questions ?

Contact:

- mailto:arnaud.tisserand@irisa.fr
- http://people.irisa.fr/Arnaud.Tisserand/
- CAIRN Group http://www.irisa.fr/cairn/
- IRISA Laboratory, CNRS–INRIA–Univ. Rennes 1
  6 rue Kerampont, CS 80518, F-22305 Lannion cedex, France

Thank you