Hardware Accelerators for ECC and HECC

Arnaud Tisserand

CNRS, IRISA laboratory, CAIRN research team

ECC
Bordeaux
Sep. 29–30, 2015

Summary

- Introduction
- Accelerator architecture and units
- Accelerator programming
- Implementation results: comparison ECC vs HECC on FPGA
- Conclusion & current/future works

Current Projects on (H)ECC Accelerators

PAVOIS project 2012–2016

Arithmetic Protections Against Physical Attacks for Elliptic Curve based Cryptography

- IRISA (Lannion)
- LIRMM (Perpignan, Montpellier & Toulon)

`http://pavois.irisa.fr/`

ANR 12 BS02 002

HAH project 2014–2017

Hardware and Arithmetic for Hyperelliptic Curves Cryptography

- IRISA (Lannion)
- IRMAR (Rennes)

`http://h-a-h.inria.fr/`

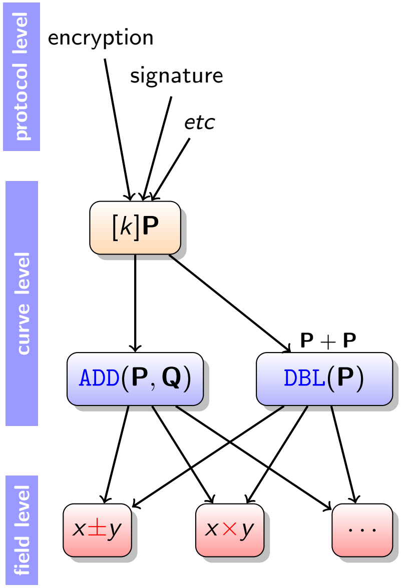
Labex

CominLabs

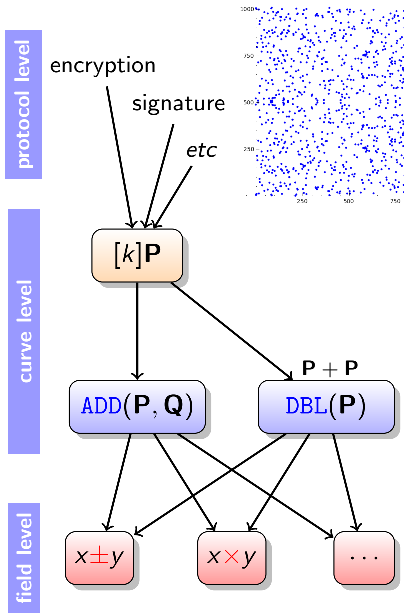
and

CENTRE HENRI LEBESGUE CENTRE DE MATHÉMATIQUES

Introduction



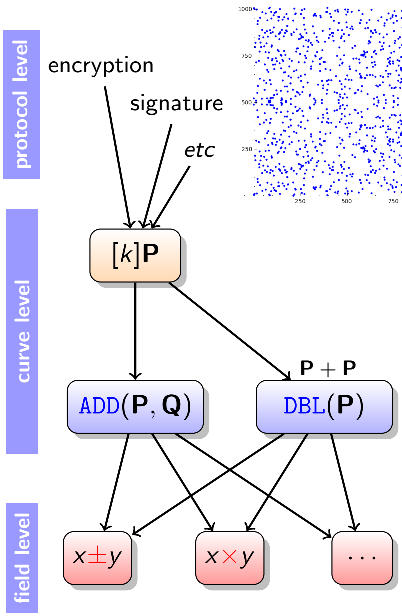
Introduction



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

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

Introduction



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

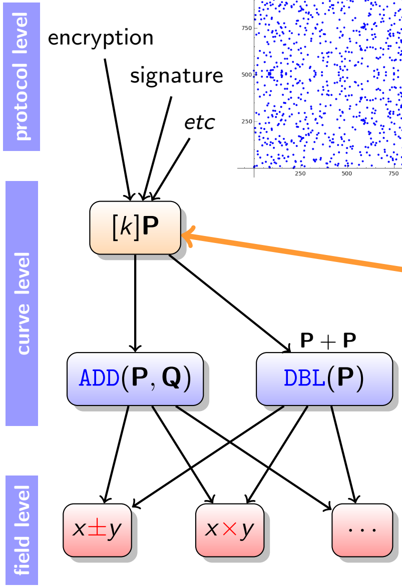
points: $\mathbf{P}$, $\mathbf{Q} = (x, y)$ or (x, y, z) or ...

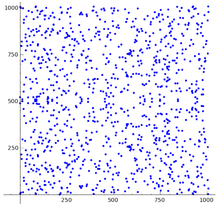
coordinates: $x, y, z \in \mathrm{GF}(\cdot)$

$\mathbb{F}_p$, $\mathbb{F}_{2^m}$, t : 80–600 bits

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

Introduction





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

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

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

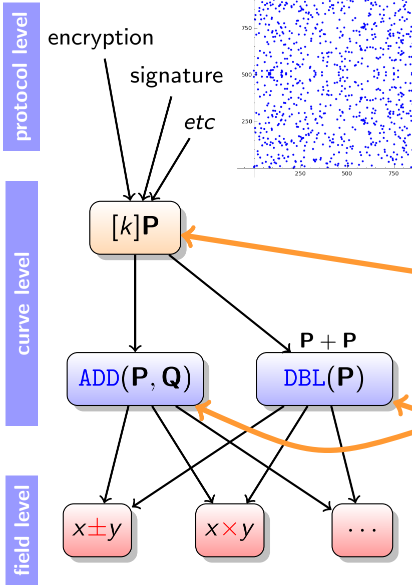
$\mathbb{F}_p$, $\mathbb{F}_{2^m}$, t : 80–600 bits

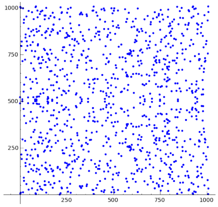
$k = (k_{t-1} k_{t-2} \dots k_1 k_0)_2 \in \mathbb{N}$

Scalar multiplication operation

```
for $i$ from 0 to $t-1$ do
    if $k_i = 1$ then $\mathbf{Q} = \mathrm{ADD}(\mathbf{P}, \mathbf{Q})$
    $\mathbf{P} = \mathrm{DBL}(\mathbf{P})$
```

Introduction





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

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

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

$\mathbb{F}_p$, $\mathbb{F}_{2^m}$, t : 80–600 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 $\mathbf{Q} = \texttt{ADD}(\mathbf{P}, \mathbf{Q})$
    $\mathbf{P} = \texttt{DBL}(\mathbf{P})$
```

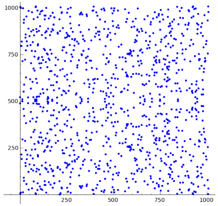
Point addition/doubling operations

sequence of finite field operations

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

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

Introduction



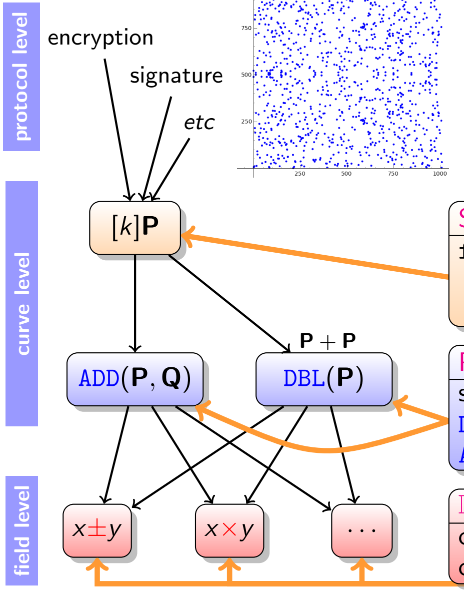
$E : y^2 = x^3 + 4x + 20$ over GF(1009)

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

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

$\mathbb{F}_p$, $\mathbb{F}_{2^m}$, t : 80–600 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 $\mathbf{Q} = \mathrm{ADD}(\mathbf{P}, \mathbf{Q})$
    $\mathbf{P} = \mathrm{DBL}(\mathbf{P})$
```

Point addition/doubling operations

sequence of finite field operations

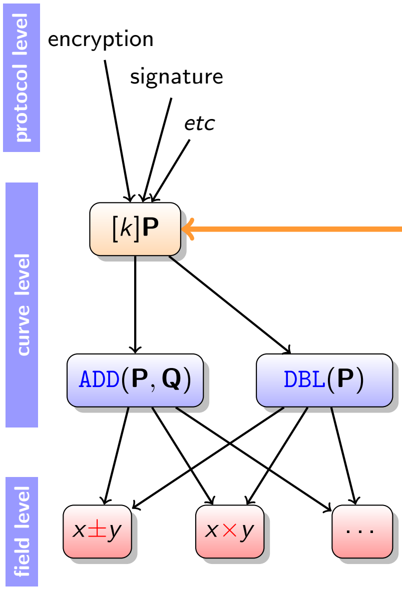
DBL: $v_1 = z_1^2, v_2 = x_1 - v_1, \ldots$

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

$\mathbb{F}_p$ or $\mathbb{F}_{2^m}$ operations

operation modulo large prime ($\mathbb{F}_p$)
or irreducible polynomial ($\mathbb{F}_{2^m}$)

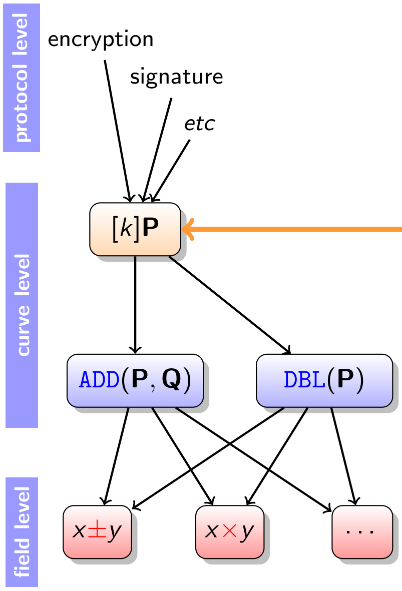
Side Channel Attacks

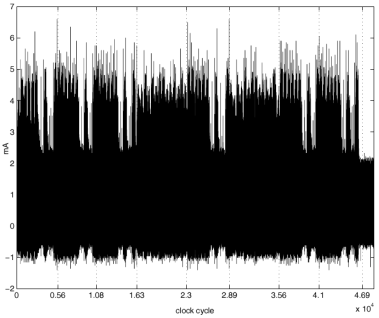


Scalar multiplication operation

```
for $i$ from 0 to $t-1$ do
    if $k_i = 1$ then $\mathbf{Q} = \texttt{ADD}(\mathbf{P},\mathbf{Q})$
    $\mathbf{P} = \texttt{DBL}(\mathbf{P})$
```

Side Channel Attacks

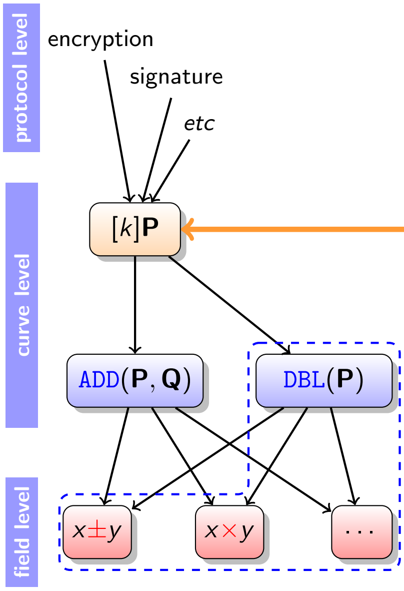


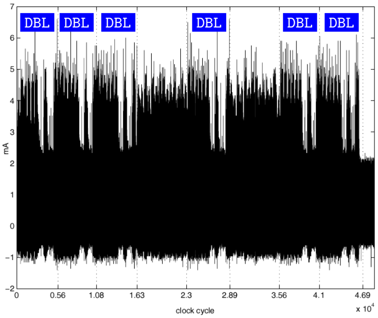


Scalar multiplication operation

```
for $i$ from 0 to $t-1$ do
    if $k_i = 1$ then $\mathbf{Q} = \texttt{ADD}(\mathbf{P}, \mathbf{Q})$
    $\mathbf{P} = \texttt{DBL}(\mathbf{P})$
```

Side Channel Attacks

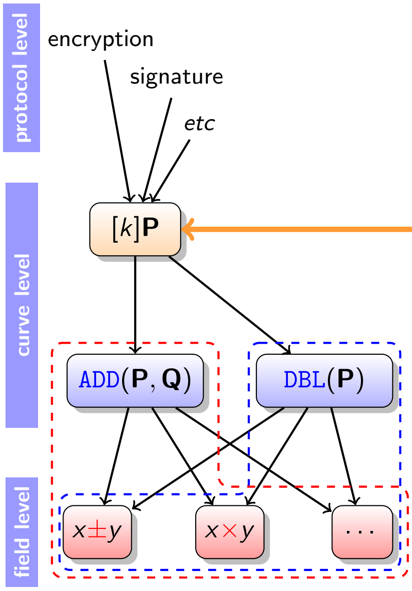


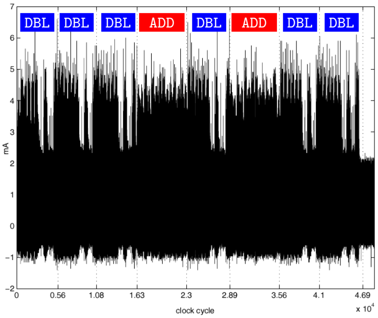


Scalar multiplication operation

```
for $i$ from 0 to $t-1$ do
    if $k_i = 1$ then $\mathbf{Q} = \mathtt{ADD}(\mathbf{P},\mathbf{Q})$
    $\mathbf{P} = \mathtt{DBL}(\mathbf{P})$
```

Side Channel Attacks

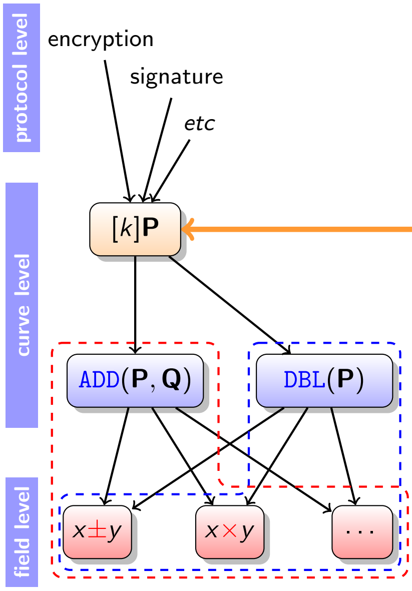


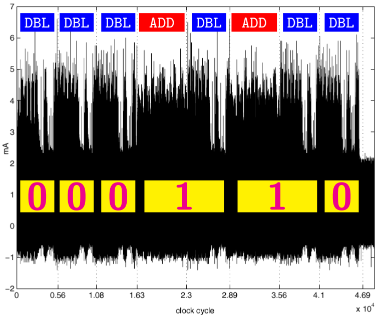


Scalar multiplication operation

```
for $i$ from 0 to $t-1$ do
    if $k_i = 1$ then $\mathbf{Q} = \texttt{ADD}(\mathbf{P}, \mathbf{Q})$
    $\mathbf{P} = \texttt{DBL}(\mathbf{P})$
```

Side Channel Attacks



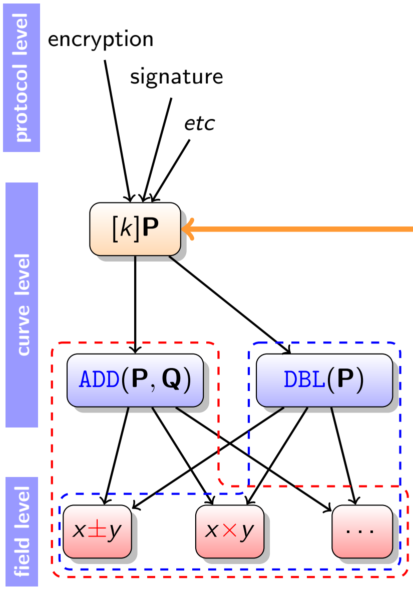


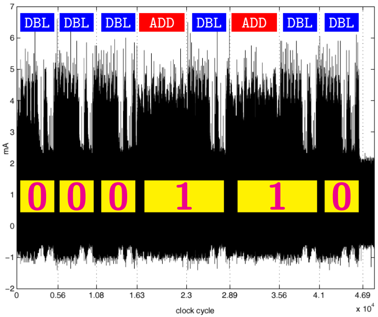
Scalar multiplication operation

```
for $i$ from 0 to $t-1$ do
    if $k_i = 1$ then $\mathbf{Q} = \texttt{ADD}(\mathbf{P},\mathbf{Q})$
    $\mathbf{P} = \texttt{DBL}(\mathbf{P})$
```

- simple power analysis (& variants)

Side Channel Attacks





Scalar multiplication operation

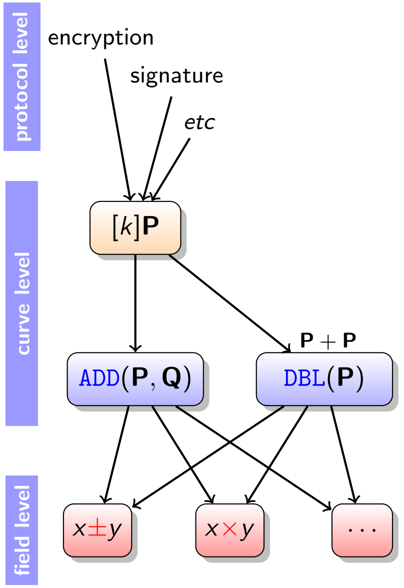
```
for $i$ from 0 to $t-1$ do
    if $k_i = 1$ then $\mathbf{Q} = \mathtt{ADD}(\mathbf{P},\mathbf{Q})$
    $\mathbf{P} = \mathtt{DBL}(\mathbf{P})$
```

- simple power analysis (& variants)
- differential power analysis (& variants)
- horizontal/vertical/. . . attacks

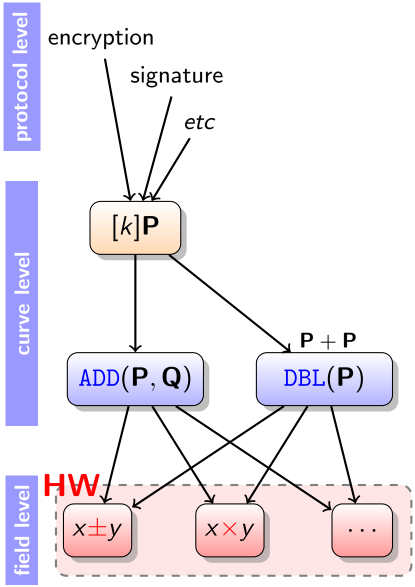
Objectives of Our Research Group

- Study and implementation of efficient hardware supports:
  - Cryptography over (hyper)-elliptic curves (H)ECC
  - Operations over finite fields $\mathbb{F}_p$ & $\mathbb{F}_{2^m}$ and curve points
  - Hardware targets: FPGAs and ASICs
  - Flexibility $\rightsquigarrow$ programmable in software
- Study and implementation of protections against physical attacks:
  - Passive attacks: measure of power consumption, electromagnetic radiations, timings
  - Active attacks: fault injection (*in progress*)
- Levels: algorithm, representation, operator, architecture, circuit
- Trade-offs between: performance, cost (area/energy), security
- Study, development and distribution of an *open source* (H)ECC accelerator and its programming tools

Accelerator Specifications

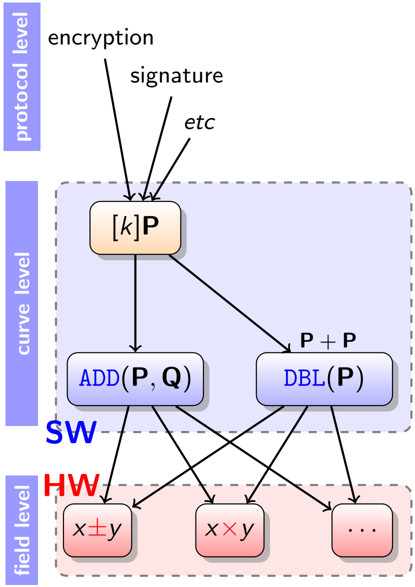


Accelerator Specifications



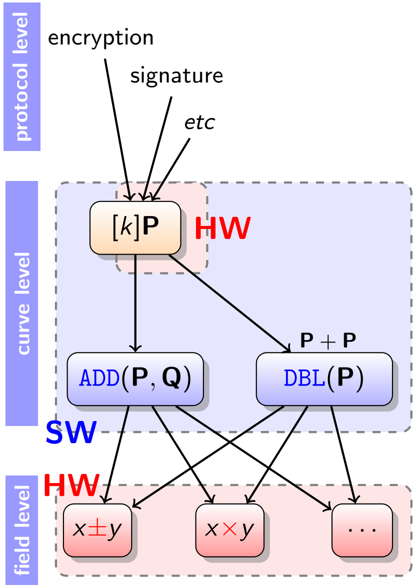
- Performances $\Longrightarrow$ hardware (**HW**)
  - dedicated functional units
  - internal parallelism
- Limited cost (embedded systems)
  - reduced silicon area
  - low energy (& power consumption)
  - large area used at each clock cycle

Accelerator Specifications



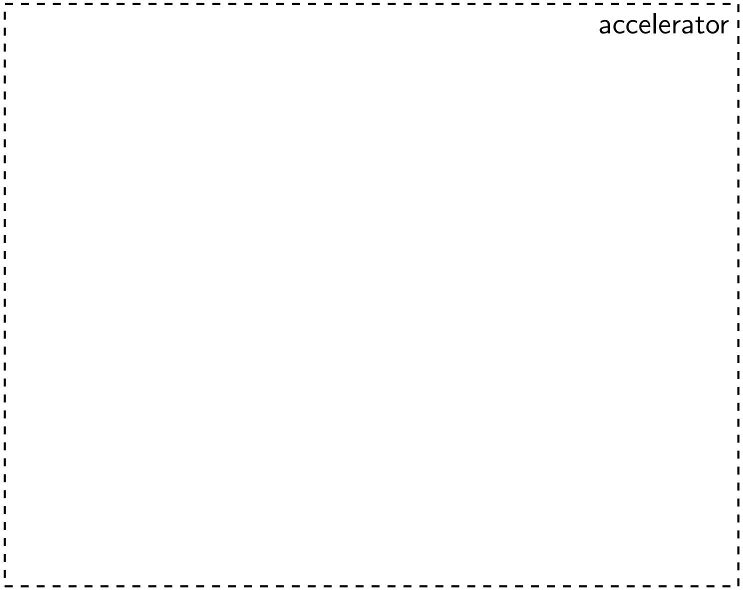
- Performances $\Longrightarrow$ hardware (**HW**)
  - dedicated functional units
  - internal parallelism
- Limited cost (embedded systems)
  - reduced silicon area
  - low energy (& power consumption)
  - large area used at each clock cycle
- Flexibility $\Longrightarrow$ software (**SW**)
  - curves, algorithms, representations (points/elements), k recoding, ...
  - at design time / at run time

Accelerator Specifications

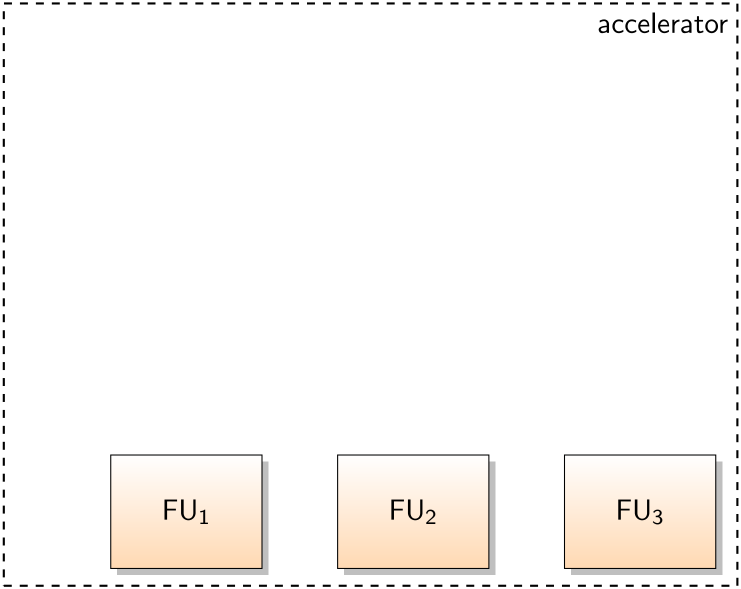


- Performances $\Longrightarrow$ hardware (**HW**)
  - dedicated functional units
  - internal parallelism
- Limited cost (embedded systems)
  - reduced silicon area
  - low energy (& power consumption)
  - large area used at each clock cycle
- Flexibility $\Longrightarrow$ software (**SW**)
  - curves, algorithms, representations (points/elements), k recoding, . . .
  - at design time / at run time
- Security against SCAs $\Longrightarrow$ **HW**
  - secure units ($\mathbb{F}_{2^m}$, $\mathbb{F}_p$)
  - secure key storage/management
  - secure control

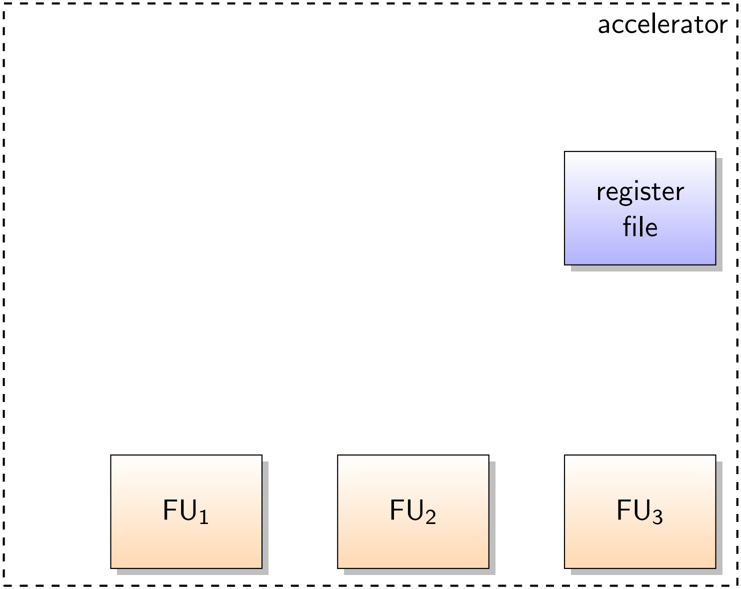
Accelerator Architecture



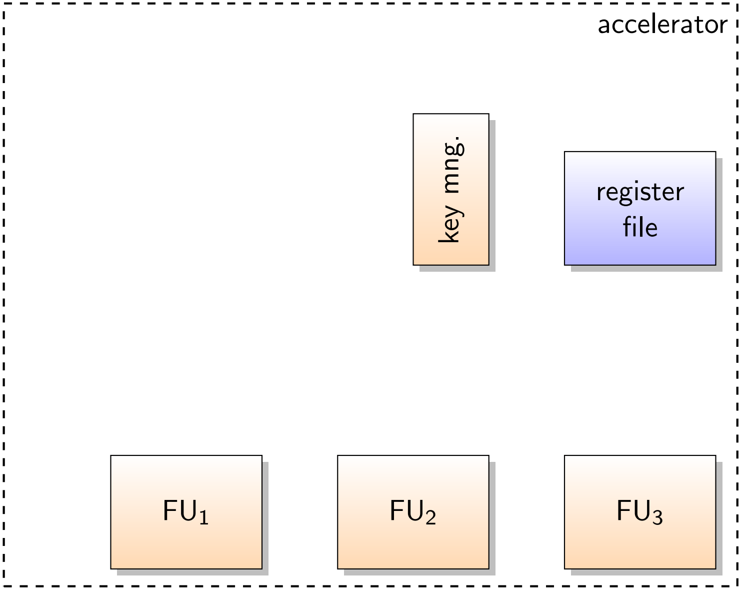
Accelerator Architecture



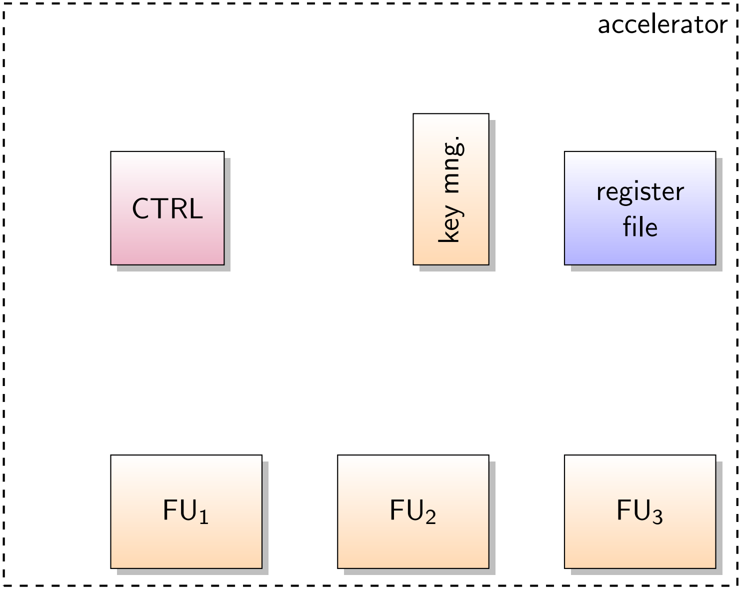
Accelerator Architecture



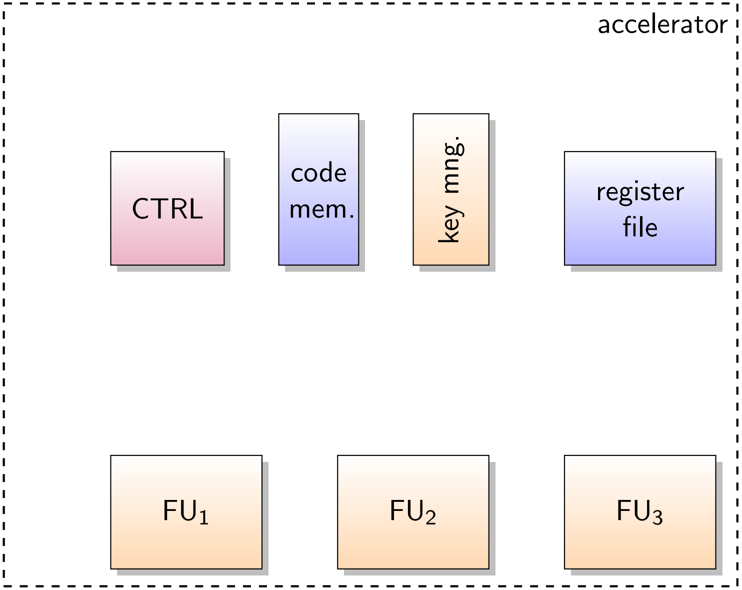
Accelerator Architecture



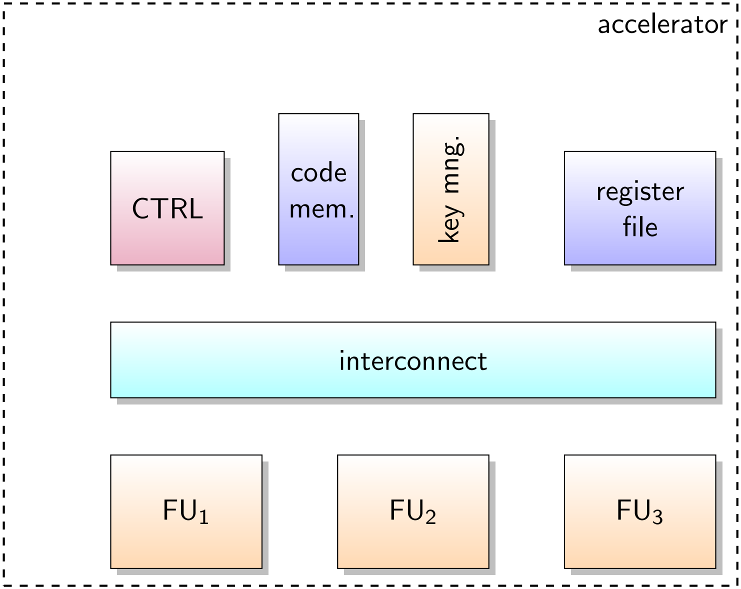
Accelerator Architecture



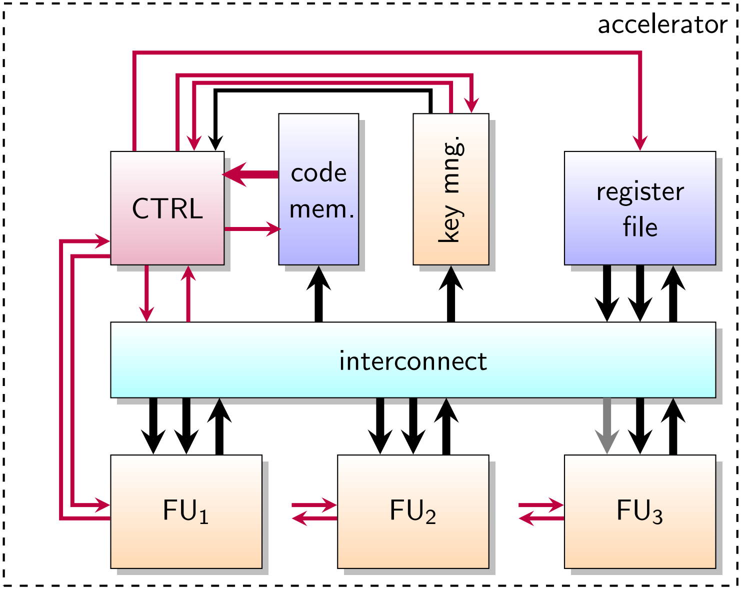
Accelerator Architecture



Accelerator Architecture

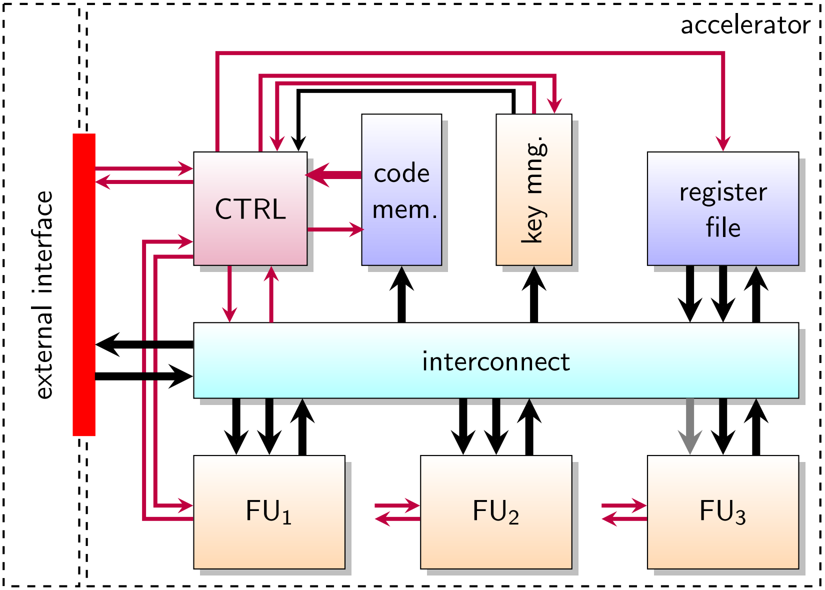


Accelerator Architecture



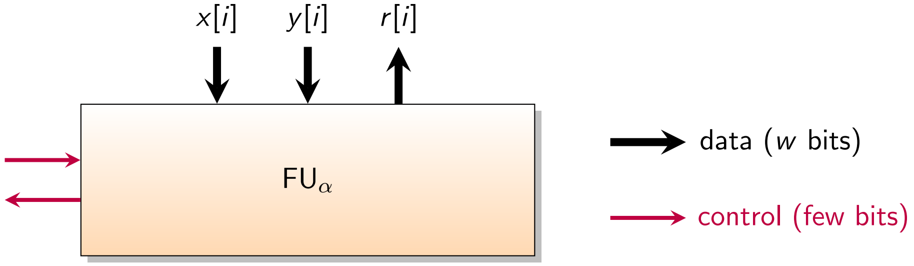
Data: *w*-bit $(32, \ldots, 128)$ except for *k* digits, **control**: a few bits per unit

Accelerator Architecture



Data: w-bit $(32, \ldots, 128)$ except for k digits, **control**: a few bits per unit

Functional Units for Field Level Operations



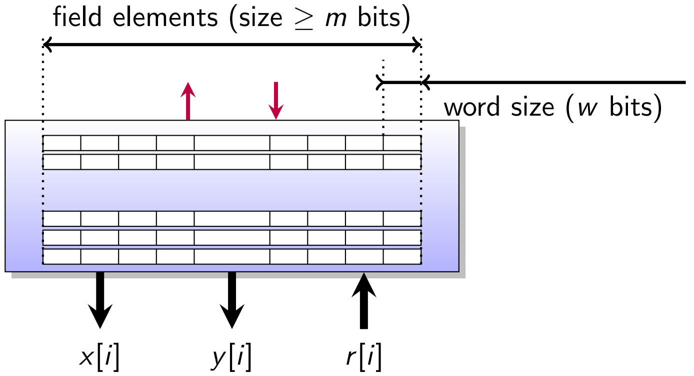
Notation: $x[i]$ is the i-th w-bit word of $x \in \mathbb{F}_q$

Units:

- $\mathbb{F}_p$: addition/subtraction, multiplication (2-step, Montgomery, variants), inversion
- $\mathbb{F}_{2^m}$ (polynomial basis, normal basis & variants): addition/subtraction, multiplication (Montgomery, Mastrovito, 2-step), square, inversion

Internal parameters: nb of sub-blocks, radix, pipelining scheme, countermeasure, mapping of local registers, output/input bypass, . . .

Register File (≈ Dual Port Memory)

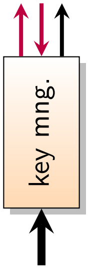


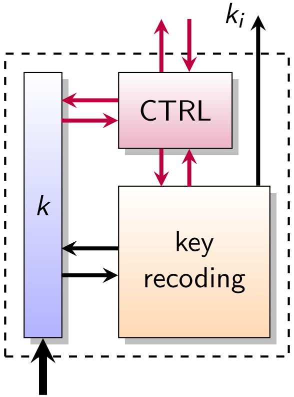
Control signals: addresses (port A, port B), read/write, write enable

Specific addressing model for $\mathbb{F}_q$ elements (through an intermediate address table with hardware loop)

- linear addresses, SW: `LOAD @`x $\Longrightarrow$ HW: loop $x[0], x[1], \ldots x[\ell-1]$
- randomized addresses

Key Management Unit





- On-the-fly recoding of k: binary, λ-NAF ($\lambda \in \{2, 3, 4, 5\}$), variants (fixed/sliding), double-base [1] and multiple-base [2] number systems (w/wo randomization), addition chains [12], other ?
- Specific private path in the interconnect (no key leaks in RF or FUs)

External Interface(s)

Under development:

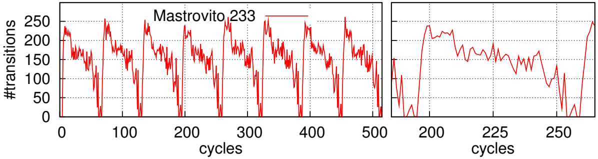
- Basic (neither clock rate nor width adaptation)
- ARM Cortex cores in Zynq 7 FPGAs (through AXI bus)
- MicroBlaze softcore processor for Xilinx FPGAs
  - AXI bus (V6+)
  - PLB bus (V2 – V5)
- Specific for a “small” ASIC pad ring

Future development:

- NIOS softcore processor for Altera FPGAs
- LEON softcore processor (depending on internal demand)

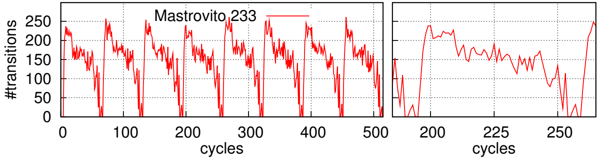
Protected $\mathbb{F}_{2^m}$ Multipliers

Unprotected



Protected $\mathbb{F}_{2^m}$ Multipliers

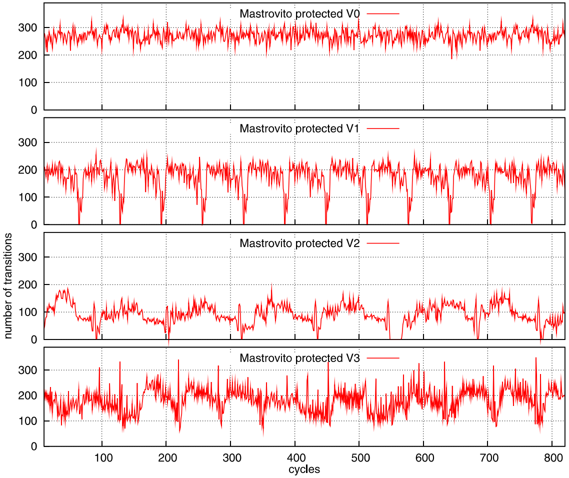
Unprotected



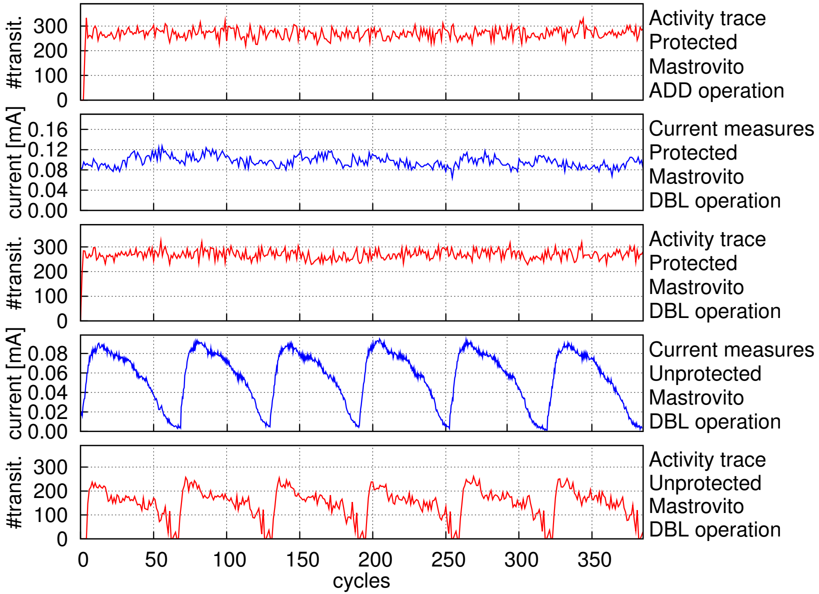
Protected

Overhead:
Area/time $< 10\,\%$

References:
PhD D. Pamula [8]
Articles: [11], [10], [9]

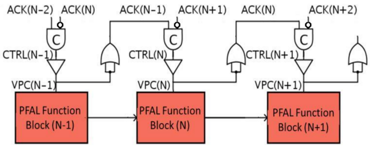


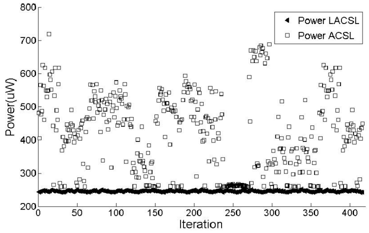
Protected (Old) Accelerator for $\mathbb{F}_{2^m}$

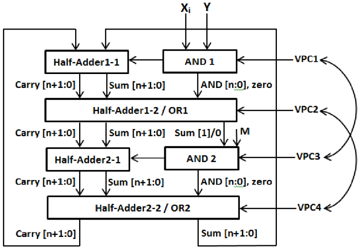


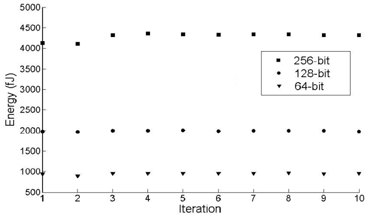
Warning: old dedicated accelerator (similar behavior is expected for our new one)

Circuit-Level Protections for Arithmetic Operators





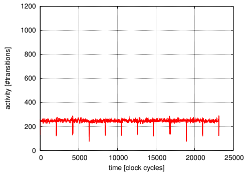
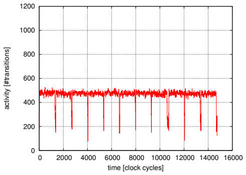
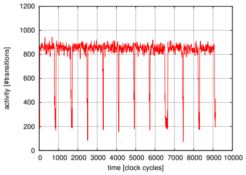
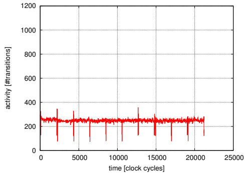
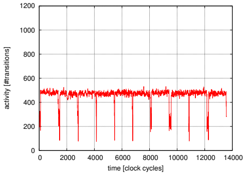
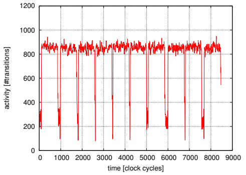




References: [4] and [3]

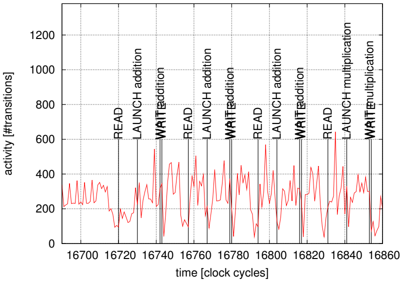
Units Impact on Side Channel Information (1/2)

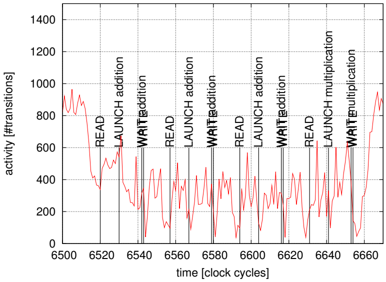
Activity traces measured with CABA[1] simulations for three configurations of the multiplier (1,2,4 sub-blocks of 32 bits) and a very small accelerator

	1	2	4
ADD			
DBL			

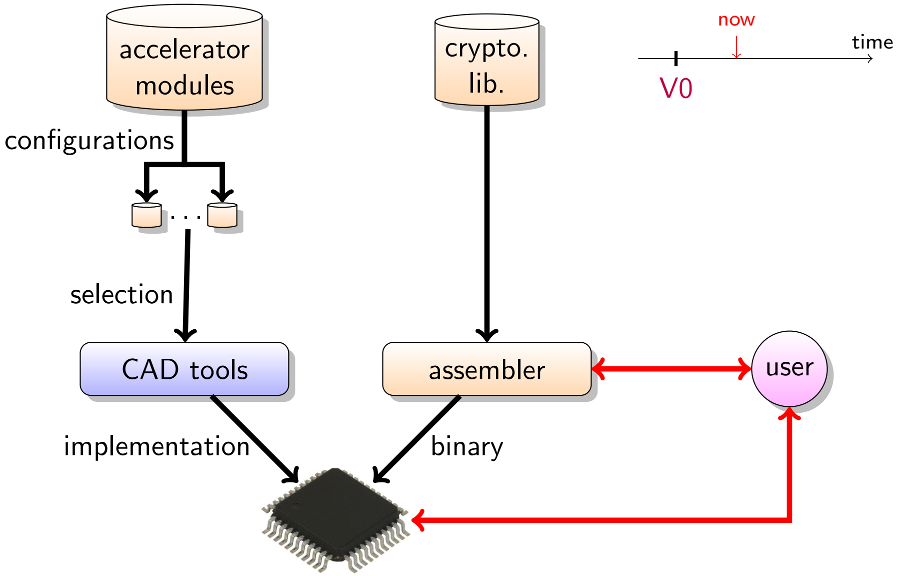
[1] Cycle Accurate Bit Accurate

Units Impact on Side Channel Information (2/2)

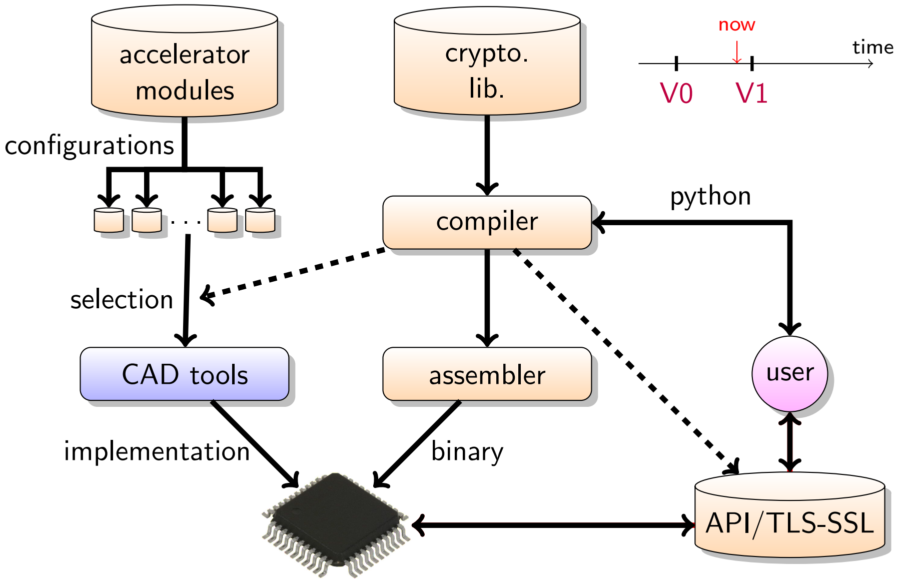




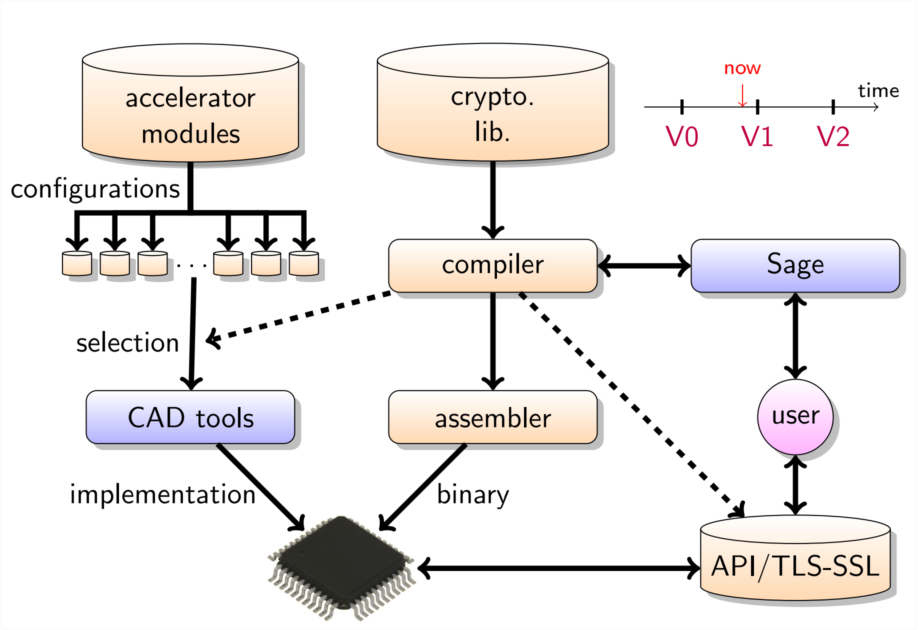
Developed Programming Tools



Developed Programming Tools



Developed Programming Tools



Instruction Set

READ	FUid	@Rid	@Rid	B/U
WRITE	FUid	@Rid		
LAUNCH	FUid	MODE		
WAIT	FUid			
SETADDRO	@Rid	OFFSET		
SETADDRN	@Rid	#WORD		
WRITEK	#WORD			
CALL	@DEST			
RET				
BZ	@DEST			
BNZ	@DEST			
JMP	@DEST			
CMPD	DIGIT			
SET	FLAGid			
TST	FLAGid			

Address Model in the Register File

RF requirements :

- 5–16 registers of m-bit $\mathbb{F}_q$ elements
- worst case: w small (16 bits) and m large (600 bits) $\Rightarrow$ 550$+$ words and 10-bit physical addresses

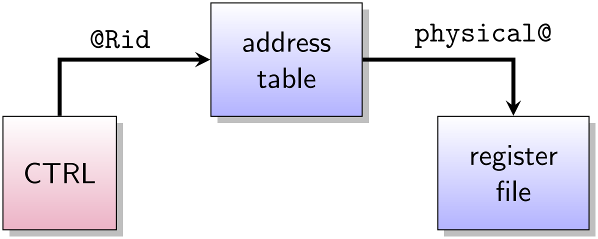
Address Model in the Register File

RF requirements :

- 5–16 registers of m-bit $\mathbb{F}_q$ elements
- worst case: w small (16 bits) and m large (600 bits) $\Rightarrow$ 550+ words and 10-bit physical addresses

$x \in \mathbb{F}_q$ is addressed by one entry (notation `@Rid`) of the intermediate address table (IAT) with 2 values:

- offset of the first word (e.g. $x[0]$)
- number of w-bit words



Code Memory

Behavior:

- Specific private path in the interconnect for code download (no leaks in RF or FUs)
- Code input can be disabled (ROM mode with code in the FPGA bitstream)
- Instruction `CALL`: push PC then jump to `@DEST`
- Instruction `RET`: jump to (pop) $+ 1$

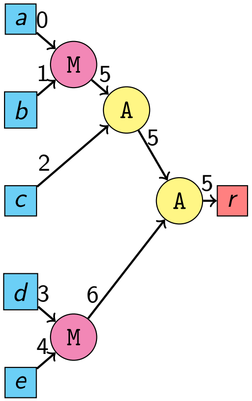
Memory mapping to be defined

Internal Parallelism Model

non-blocking instruction decoding (i.e. always do $PC \leftarrow PC + 1$ or $PC \leftarrow cst$) except for `WAIT` instruction

Example of operations sequence, its dependency graph and assembly code for 2 multipliers:

$r = ((a \times b) + c) + (d \times e))$



`read fu_mul_0, 0, 1`	read a & b
`launch fu_mul_0`	start ab
`read fu_mul_1, 3, 4`	lit d & e
`launch fu_mul_1`	start de
`wait fu_mul_0`	wait for ab
`write fu_mul_0, 5`	write ab
`set OPMODE, 0`	addition mode $(+)$
`read fu_add_sub_0, 5, 2`	read ab & c
`launch fu_add_sub_0`	start $(ab) + c$
`wait fu_mul_1`	wait for de
`write fu_mul_1, 6`	write de
`wait fu_add_sub_0`	wait for $(ab) + c$
`write fu_add_sub_0, 5`	write $(ab) + c$
`read fu_add_sub_0, 5, 6`	read $(ab) + c$ & de
`launch fu_add_sub_0`	start $((ab) + c) + (de)$
`wait fu_add_sub_0`	wait for $((ab) + c) + (de)$
`write fu_add_sub_0, 5`	write $((ab) + c) + (de)$

ECC Accelerator with Additions Chains

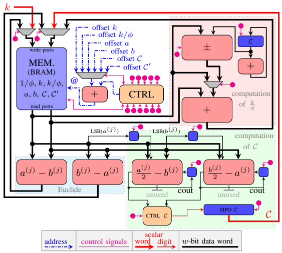
First full hardware implementation of recoding using additions chains

FPGA implementation

Spartan-6 XC6SLX9

192-bit $\mathbb{F}_p$

Very small config.

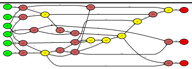
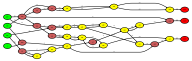
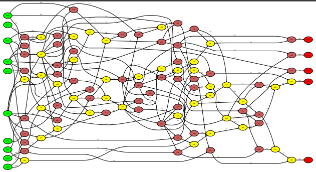
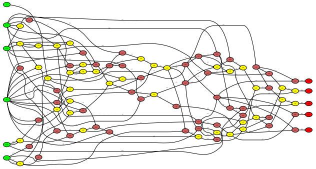


recoding method	BRAM	optim. target	area slices (FF/LUT)	freq. MHz	dura. ms	SCA prot.
EAC	3	area	534 (1813/1508)	132	35.8	Y
		speed	556 (1872/1523)	137	34.5	
DA	2	area	429 (1243/1134)	191	30	N
		speed	399 (1302/1222)	177	32.5	
ML	2	area	429 (1243/1134)	191	42.5	Y
		speed	399 (1302/1222)	177	45.8	
UF	2	area	429 (1243/1134)	191	50.4	Y
		speed	399 (1302/1222)	177	54.4	
NAF-3	2	area	422 (1280/1157)	181	25.2	N
		speed	423 (1321/1242)	175	26.1	
NAF-4	2	area	420 (1277/1161)	158	27.3	N
		speed	425 (1233/1246)	177	24.4	

EAC: Euclidean addition chains, DA: dbl-and-add, ML: Montgomery ladder, UF: unified formula

See details in [12]

Comparison ECC 256 vs HECC 128 (1/7)

	field $\mathbb{F}_p$	`ADD`	`DBL`
ECC	ℓ bits	 Cost: 12M + 2S	 Cost: 6M + 5S
HECC	$\frac{\ell}{2}$ bits	 Cost: 47M + 4S	 Cost: 38M + 6S

Configurations on a XC6SLX75 FPGA (details in [5]):

- $w = 32$ bits internal words
- 1 adder/subtracter, 1 inversion unit
- $n_{\mathtt{M}}$ multipliers (Montgomery) with $n_{\mathtt{B}}$ w-bit sub-blocks
- No DSP blocks
- ISE 14.6 Xilinx CAD tools, standard efforts (synthesis and P&R)

Comparison ECC 256 vs HECC 128 (2/7)

- Compared recoding techniques:
  - BIN: standard binary from left to right
  - NAF: non-adjacent form
  - λ-NAF: window methods with $\lambda \in \{3, 4\}$

- Implementation results for a full ECC accelerator ($n_{\mathtt{M}} = 1, n_{\mathtt{B}} = 1$):

Recoding	BIN	NAF	3-NAF	4-NAF
area slices (FF/LUT)	**565** (1321/1461)	**570** (1340/1479)	**571** (1344/1495)	**503** (1348/1489)
freq. (MHz)	225	228	237	217

All other results are reported for 4-NAF

Comparison ECC 256 vs HECC 128 (3/7)

Impact of the number/size of multipliers on the area and frequency:

	n_{M}	BRAM	$n_{\mathrm{B}} = 1$		$n_{\mathrm{B}} = 2$		$n_{\mathrm{B}} = 4$	
			area slices (FF/LUT)	freq. MHz	area slices (FF/LUT)	freq. MHz	area slices (FF/LUT)	freq. MHz
ECC	1	3	547 (1374/1460)	231	573 (1476/1625)	233	673 (1674/1875)	233
	2	3	722 (1776/1903)	220	811 (1979/2210)	227	942 (2377/2701)	220
	3	3	810 (2174/2236)	221	915 (2480/2698)	215	1130 (3077/3430)	214
	4	3	952 (2569/2656)	215	1100 (2977/3282)	217	1512 (3771/4293)	216
	5	3	1064 (2982/3136)	210	1405 (3492/3902)	206	1722 (4487/5122)	209
HECC	1	4	514 (1336/1374)	235	549 (1434/1513)	234		
	2	4	646 (1716/1783)	220	737 (1912/2055)	234		
	3	4	732 (2092/2075)	224	826 (2386/2485)	225		
	4	4	870 (2476/2424)	218	1022 (2868/2987)	214		
	5	4	976 (2865/2773)	219	1115 (3355/3465)	210		
	6	4	1089 (3233/3092)	203	1240 (3821/3908)	208		
	7	4	1145 (3601/3426)	213	1372 (4287/4365)	205		
	8	4	1281 (3981/3809)	191	1552 (4765/4890)	183		
	9	4	1379 (4363/4051)	202	1691 (5245/5277)	199		
	10	4	1543 (4739/4435)	196	1856 (5719/5801)	198		
	11	4	1547 (5114/4750)	189	1936 (6192/6240)	198		
	12	4	1738 (5499/5128)	191	2100 (6675/6771)	188		

Comparison ECC 256 vs HECC 128 (4/7)

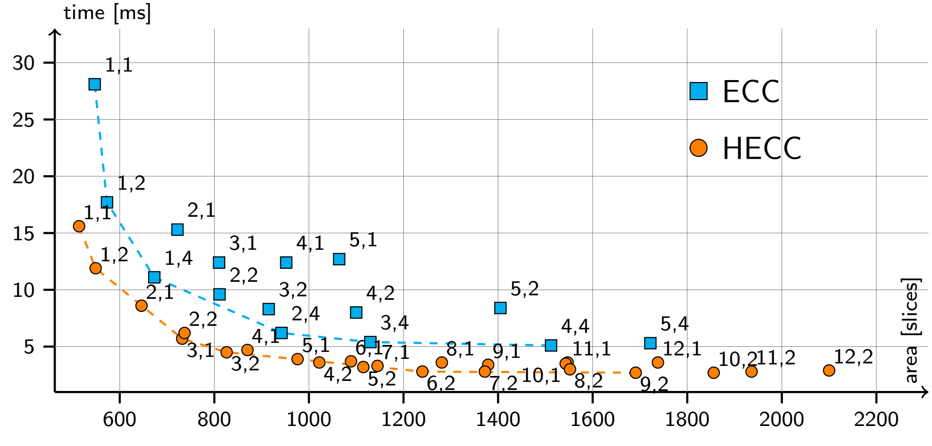
Impact of the number/size of multipliers on the average time (ms):

	n_B	n_M											
		1	2	3	4	5	6	7	8	9	10	11	12
HECC	1	15.6	8.6	5.7	4.7	3.9	3.7	3.3	3.6	3.4	3.5	3.6	3.6
	2	11.9	6.2	4.5	3.6	3.2	2.8	2.8	3.0	2.7	2.7	2.8	2.9
ECC	1	28.1	15.3	12.4	12.4	12.7							
	2	17.7	9.6	8.3	8.0	8.4							
	4	11.1	6.2	5.4	5.1	5.3							

Standard deviation for 1000 $[k]P$:

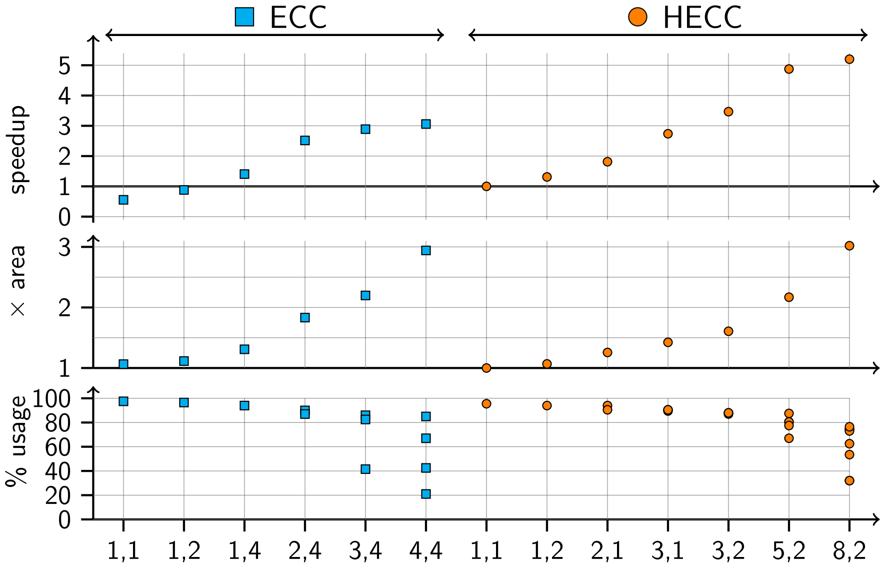
configuration	ECC (1,1)	ECC (3,4)	HECC (1,1)	HECC (6,2)
average time [ms]	28.1	5.4	15.6	2.8
standard deviation [ms]	0.289	0.056	0.324	0.045

Comparison ECC 256 vs HECC 128 (5/7)



On average HECC is 40 % faster than ECC for a similar silicon cost

Comparison ECC 256 vs HECC 128 (6/7)



Comparison ECC 256 vs HECC 128 (7/7)

Source	FPGA	area slices / DSP blocks	freq. MHz	duration $[k]\mathbf{P}$ ms
ECC 1,2	Spartan 6	573 / 0	233	17.7
ECC 1,4		673 / 0	233	11.1
ECC 2,4		942 / 0	220	6.2
ECC 3,4		1 130 / 0	214	5.4
[7]	Virtex-5	1 725 / 37	291	0.38
	Virtex-4	4 655 / 37	250	0.44
[6]	Virtex-4	13 661 / 0	43	9.2
		20 123 / 0	43	7.7

Conclusion & Current/Future Works

- HECC is efficient in hardware (40 % speedup vs ECC)
- Flexible architecture and tools for research activities
- Advanced recoding schemes are efficient in hardware

Current/future works:

- Hardware implementation of halving based method(s)
- Protections against fault injection
- HECC extensions of the accelerator (and tools)
- ASIC (CMOS 65nm) implementation of the accelerator
- Side channel evaluation of (some) proposed protections
- HW/SW Code distribution under free license
- More advanced architecture/circuit level protections
- Collaboration with other research groups

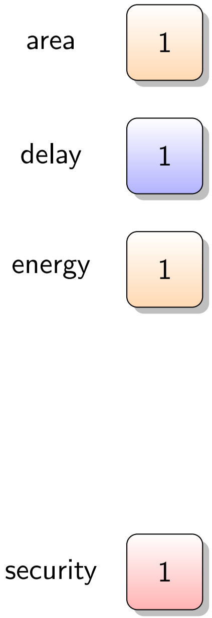
Our Long Term Objectives

Study the links between:

- curves
- arithmetic algorithms
- $\mathbb{F}_q$, pts representations
- architecture & units
- circuit styles

to ensure

- high security against
  - theoretical attacks
  - physical attacks
- low design cost
- low silicon cost
- low energy(/power)
- high performances
- high flexibility



Our Long Term Objectives

Study the links between:

- curves
- arithmetic algorithms
- $\mathbb{F}_q$, pts representations
- architecture & units
- circuit styles

to ensure

- high security against
  - theoretical attacks
  - physical attacks
- low design cost
- low silicon cost
- low energy(/power)
- high performances
- high flexibility

$$\begin{tikzcd} \text{area} & \boxed{1} \arrow[r] & \boxed{1+a} \\ \text{delay} & \boxed{1} \arrow[r] & \boxed{1+t} \\ \text{energy} & \boxed{1} \arrow[r] & \boxed{1+e} \end{tikzcd}$$

$a, t, e \in 0\%, 5\%, 10\%, \ldots, 100\%$

security $\boxed{1}$

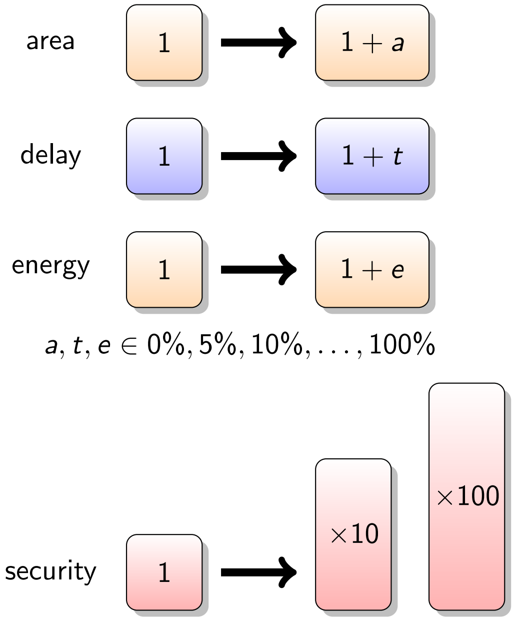
Our Long Term Objectives

Study the links between:

- curves
- arithmetic algorithms
- $\mathbb{F}_q$, pts representations
- architecture & units
- circuit styles

to ensure

- high security against
  - theoretical attacks
  - physical attacks
- low design cost
- low silicon cost
- low energy(/power)
- high performances
- high flexibility



References I

T. Chabrier, D. Pamula, and A. Tisserand.
Hardware implementation of DBNS recoding for ECC processor.
In *Proc. 44rd Asilomar Conference on Signals, Systems and Computers*, pages 1129–1133, Pacific Grove, California, U.S.A., November 2010. IEEE.

T. Chabrier and A. Tisserand.
On-the-fly multi-base recoding for ECC scalar multiplication without pre-computations.
In A. Nannarelli, P.-M. Seidel, and P. T. P. Tang, editors, *Proc. 21st Symposium on Computer Arithmetic (ARITH)*, pages 219–228, Austin, TX, U.S.A, April 2013. IEEE Computer Society.

J. Chen, A. Tisserand, E. Popovici, and S. Cotofana.
Asynchronous charge sharing power consistent montgomery multiplier.
In J. Sparso and E Yahya, editors, *Proc. 21st IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC)*, pages 132–138, Mountain View, California, USA, May 2015.

J. Chen, A. Tisserand, E. M. Popovici, and S. Cotofana.
Robust sub-powered asynchronous logic.
In J. Becker and M. R. Adrover, editors, *Proc. 24th International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, pages 1–7, Palma de Mallorca, Spain, September 2014. IEEE.

G. Gallin, A. Tisserand, and N. Veyrat-Charvillon.
Comparaison expérimentale d'architectures de crypto-processeurs pour courbes elliptiques et hyper-elliptiques.
In *Actes Conférence d'informatique en Parallélisme, Architecture et Système (ComPAS)*, Lille, France, June 2015.
Prix meilleur papier track architecture.

S. Ghosh, M. Alam, D. Roychowdhury, and I.S. Gupta.
Parallel crypto-devices for GF(p) elliptic curve multiplication resistant against side channel attacks.
Computers and Electrical Engineering, 35(2):329–338, March 2009.

References II

Y. Ma, Z. Liu, W. Pan, and J. Jing.
A high-speed elliptic curve cryptographic processor for generic curves over GF(p).
In *Proc. 20th International Workshop on Selected Areas in Cryptography (SAC)*, volume 8282 of *LNCS*, pages 421–437, Burnaby, BC, Canada, August 2013. Springer.

D. Pamula.
Arithmetic Operators on GF(2^m) for Cryptographic Applications: Performance - Power Consumption - Security Tradeoffs.
Phd thesis, University of Rennes 1 and Silesian University of Technology, December 2012.

D. Pamula, E. Hrynkiewicz, and A. Tisserand.
Analysis of GF(2^{233}) multipliers regarding elliptic curve cryptosystem applications.
In *11th IFAC/IEEE International Conference on Programmable Devices and Embedded Systems (PDeS)*, pages 252–257, Brno, Czech Republic, May 2012.

D. Pamula and A. Tisserand.
GF(2^m) finite-field multipliers with reduced activity variations.
In *4th International Workshop on the Arithmetic of Finite Fields*, volume 7369 of *LNCS*, pages 152–167, Bochum, Germany, July 2012. Springer.

D. Pamula and A. Tisserand.
Fast and secure finite field multipliers.
In *Proc. Euromicro Conference on Digital System Design (DSD)*, pages 1–8, Funchal, Portugal, August 2015.

J. Proy, N. Veyrat-Charvillon, A. Tisserand, and N. Meloni.
Full hardware implementation of short addition chains recoding for ECC scalar multiplication.
In *Actes Conférence d'informatique en Parallélisme, Architecture et Système (ComPAS)*, Lille, France, June 2015.

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