Hardware-Software Co-Design for Security: ECC Processor Example

Arnaud Tisserand

CNRS, Lab-STICC

SILM Workshop, Nov. 2019

Introduction

Public-key (or asymmetric) cryptography (PKC):

- RSA
- (hyper-)elliptic curve cryptography ((H)ECC)
- post-quantum crypto (PQC)

Design, prototype and evaluate hardware/software (HW/SW) for PKC:

- HW: computation units, accelerators, crypto-processors
- SW: libraries, generators for HW, dedicated compiler for our processors

Objectives:

- high speed, reduced silicon area and energy consumption
- protections against side-channel and fault-injection attacks (SCA/FIA)
- HW: FPGA and ASIC implementations
- SW: embedded processors implementations

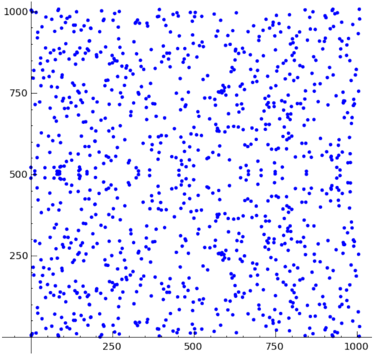
Elliptic Curve Cryptography (ECC)

Elliptic curve over GF(p):

$$E: \quad y^2 = x^3 + ax + b$$

Curve points representation:

- $P = (x, y)$ affine coordinates
  ☹ many field inversions
- $P = (x, y, z, \ldots)$ redundant coordinates
  ☺ significantly faster (e.g., Jacobian)



$y^2 = x^3 + 4x + 20$ over $\mathrm{GF}(1009)$

Scalar multiplication:

$$Q = [k]P = \underbrace{P + P + \cdots + P}_{k \text{ times}}$$

The most time consuming operation in protocols

where $P \in E$ and $k = (k_{n-1}k_{n-2}\ldots k_1k_0)_2$

k has 200–600 bits

Good and complete presentation in [14] and [10]

Scalar Multiplication

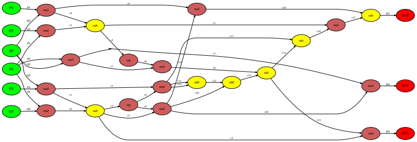
$$Q = [k]P = \underbrace{P + P + \cdots + P}_{k \text{ times}}$$

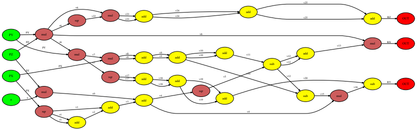
- $P \in E$
- $k = (k_{n-1}k_{n-2}\ldots k_1k_0)_2$

Double-and-add scalar multiplication algorithm:

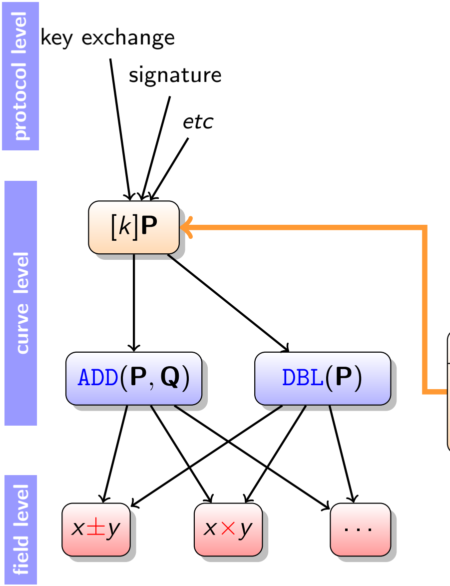
$Q \leftarrow \mathcal{O}$
for i **from** $n-1$ **to** 0 **do**
  $Q \leftarrow [2]Q$ (DBL)
  if $k_i = 1$ **then** $Q \leftarrow Q + P$ (ADD)
return Q

- scans each bit of k and performs corresponding curve-level operation
- average cost: $0.5n$ ADD $+ n$ DBL (security ➔ $\approx 0.5n$ ones in k)





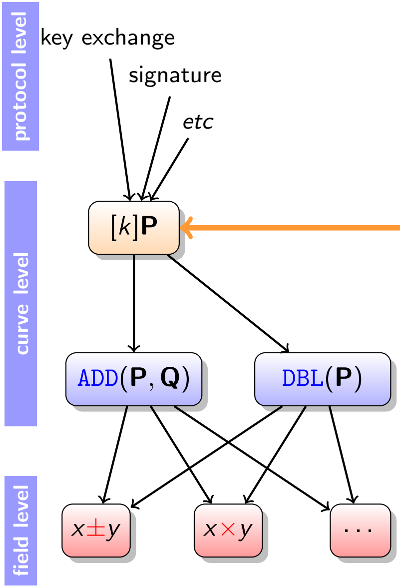
Side Channel Attacks

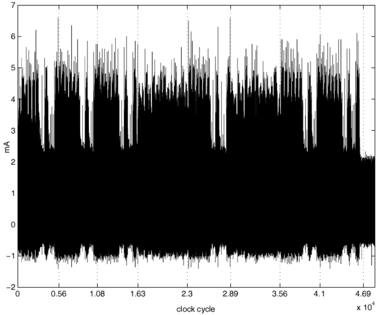


Scalar multiplication operation

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

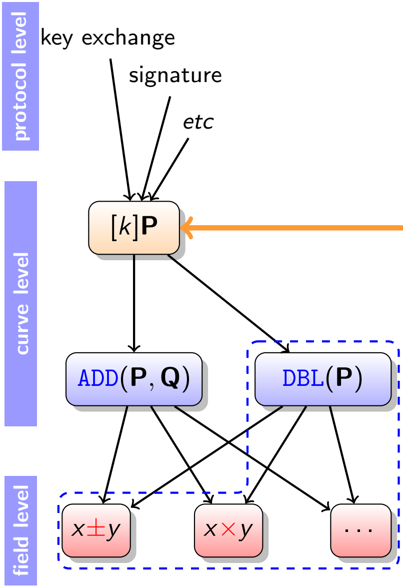


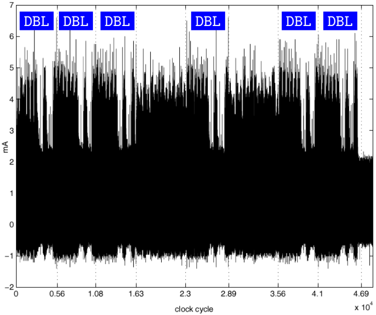


Scalar multiplication operation

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

Side Channel Attacks

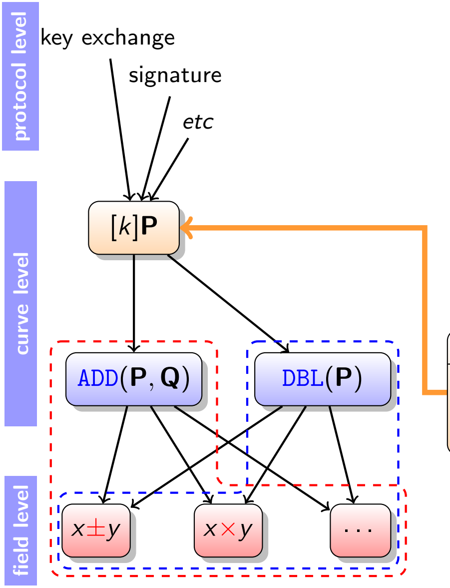


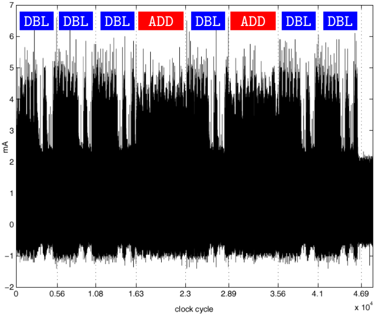


Scalar multiplication operation

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

Side Channel Attacks

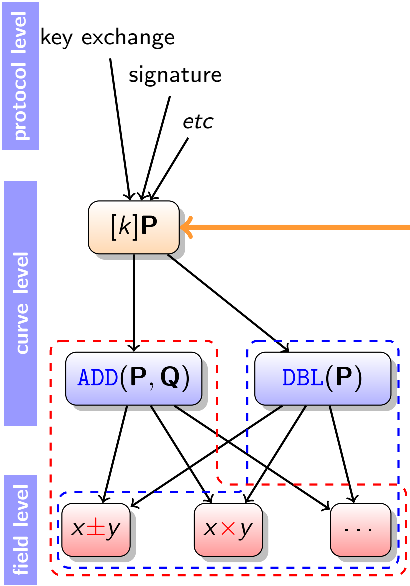


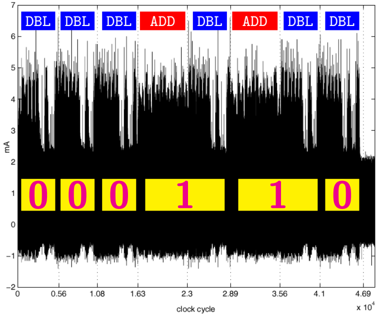


Scalar multiplication operation

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



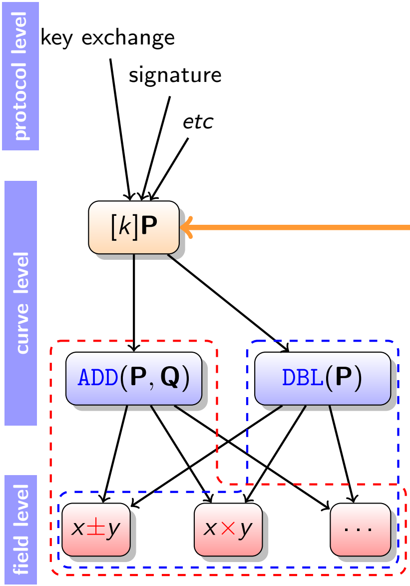


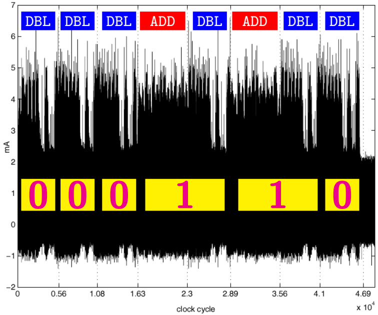
Scalar multiplication operation

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

- simple power analysis (& variants)

Side Channel Attacks



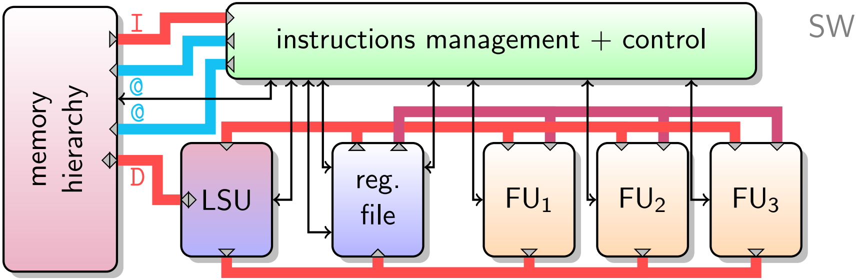


Scalar multiplication operation

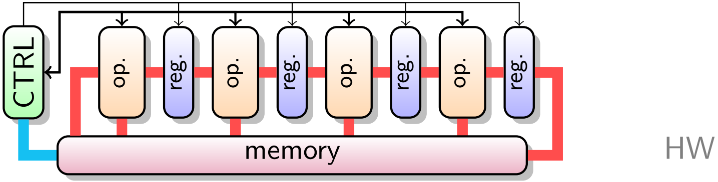
for i from 0 to $t-1$ do
  if $k_i = 1$ then $\mathbf{Q} = \mathtt{ADD}(\mathbf{P}, \mathbf{Q})$
  $\mathbf{P} = \mathtt{DBL}(\mathbf{P})$

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

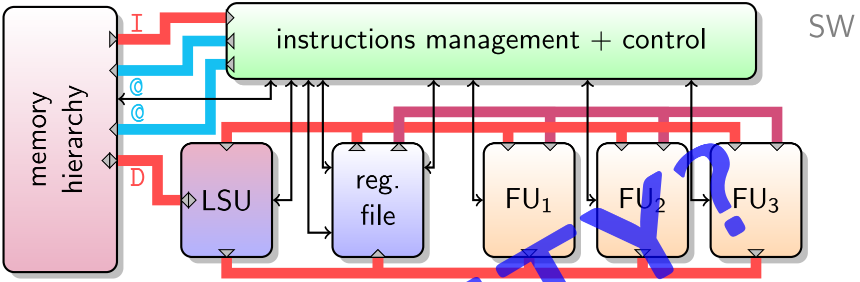
Software vs Hardware Support



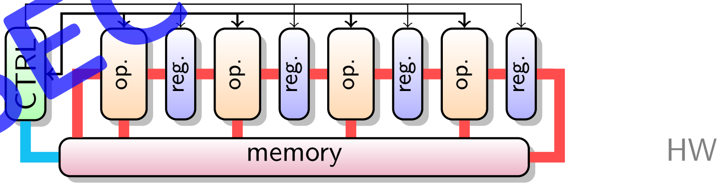
EXCELLENT	slow	large	large	moderate
FLEXIBILITY	SPEED	AREA	ENERGY	DEVEL. COST
limited	fast	small	small	HUGE



Software vs Hardware Support



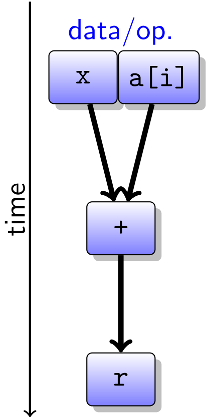
EXCELLENT	slow	large	large	moderate
FLEXIBILITY	SPEED	AREA	ENERGY	DEVEL. COST
limited	fast	small	small	HUGE



SECURITY?

Activity in a Processor

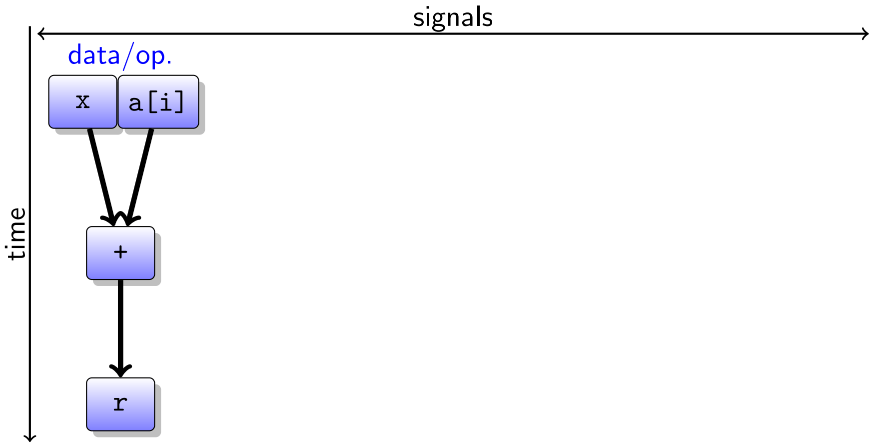
Operation to be executed: `r ← x + a[i]`



- AS: ALU status
- PIS: fetch, decode, pipeline management, bypasses, memory hierarchy, branch predictor, monitoring, etc.

Activity in a Processor

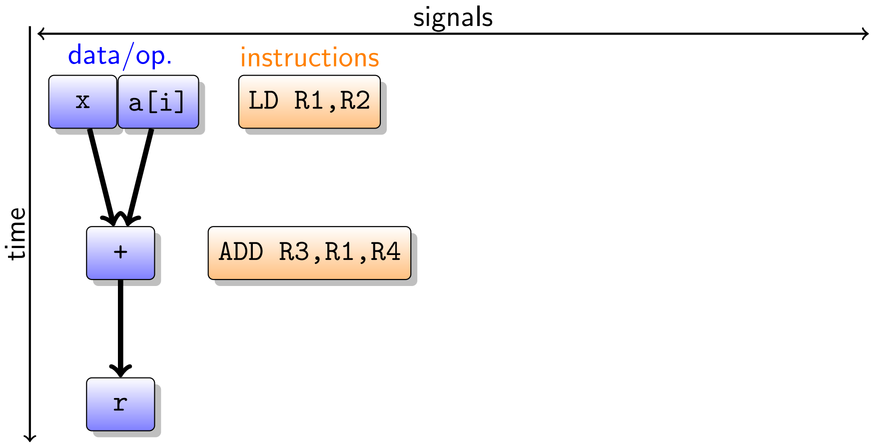
Operation to be executed: $r \leftarrow x + a[i]$



- AS: ALU status
- PIS: fetch, decode, pipeline management, bypasses, memory hierarchy, branch predictor, monitoring, etc.

Activity in a Processor

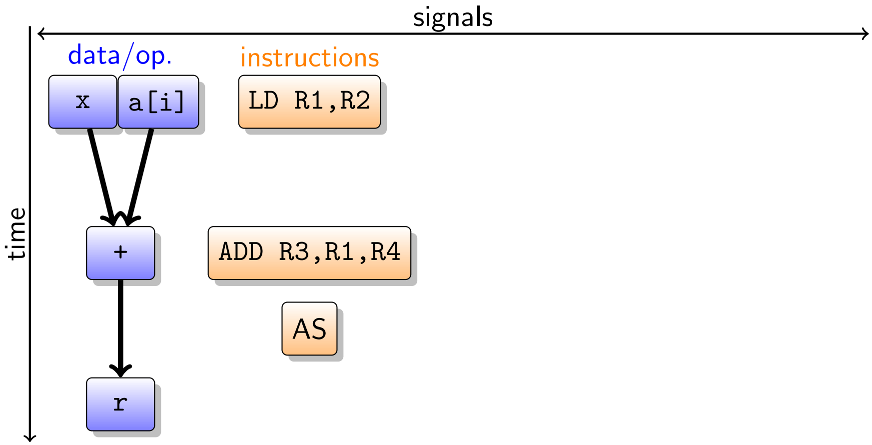
Operation to be executed: `r ← x + a[i]`



- AS: ALU status
- PIS: fetch, decode, pipeline management, bypasses, memory hierarchy, branch predictor, monitoring, etc.

Activity in a Processor

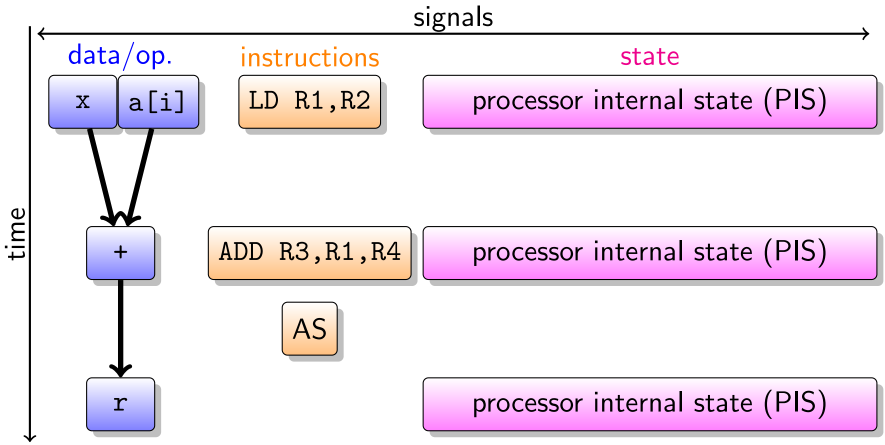
Operation to be executed: r ← x + a[i]



- AS: ALU status
- PIS: fetch, decode, pipeline management, bypasses, memory hierarchy, branch predictor, monitoring, etc.

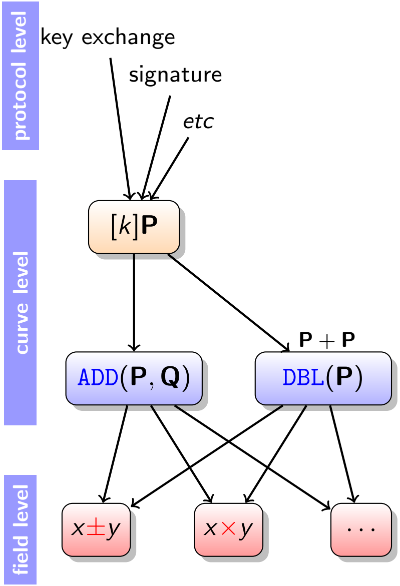
Activity in a Processor

Operation to be executed: `r ← x + a[i]`

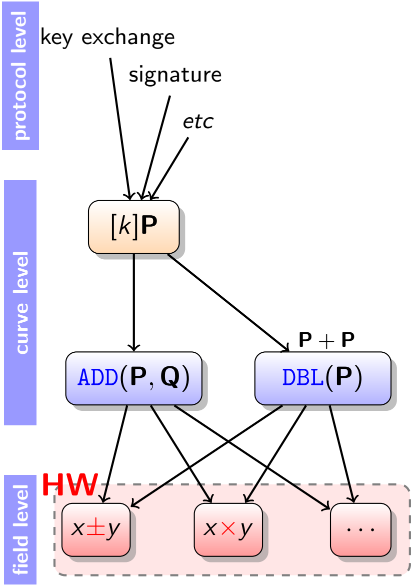


- AS: ALU status
- PIS: fetch, decode, pipeline management, bypasses, memory hierarchy, branch predictor, monitoring, etc.

Our Processor Specifications

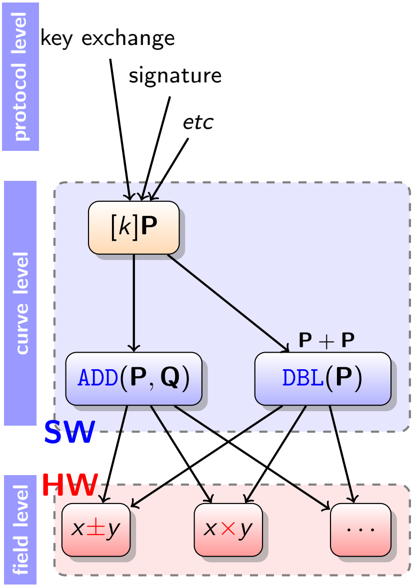


Our Processor Specifications



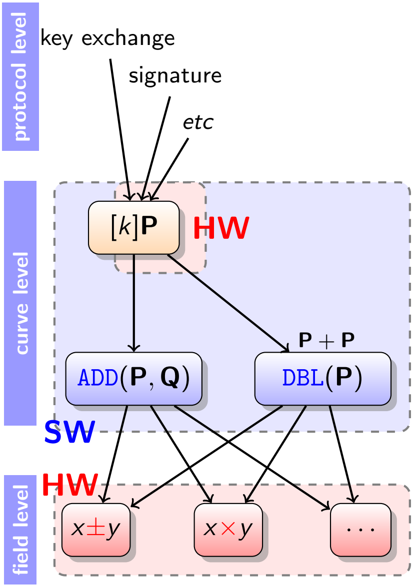
- Performances $\Longrightarrow$ hardware (**HW**)
  - dedicated functional units
  - internal parallelism
- Limited cost (embedded systems)
  - reduced silicon area
  - low energy (& power consumption)
  - large area used at each clock cycle

Our Processor 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

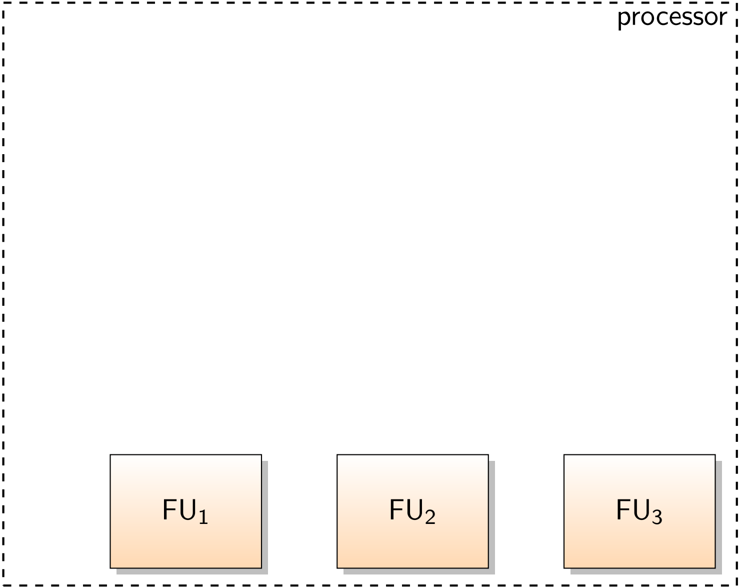
Our Processor Specifications



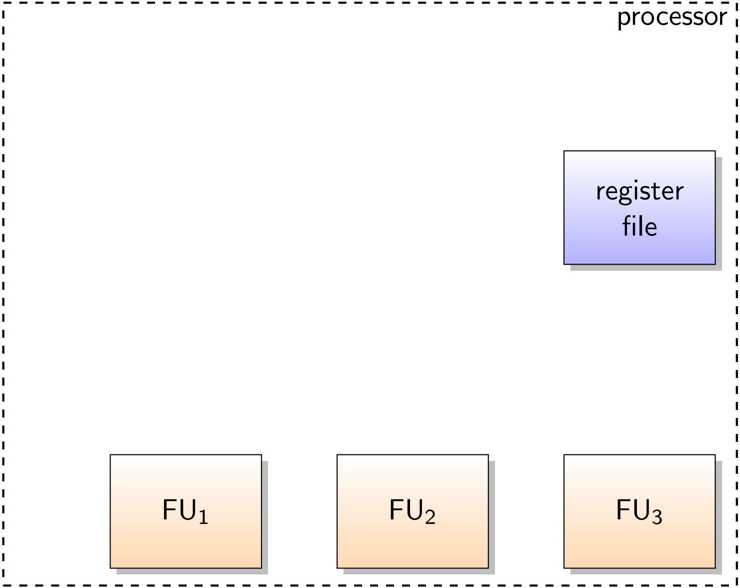
- Performances $\Longrightarrow$ hardware (**HW**)
  - dedicated functional units
  - internal parallelism
- Limited cost (embedded systems)
  - reduced silicon area
  - low energy (& power consumption)
  - large area used at each clock cycle
- Flexibility $\Longrightarrow$ software (**SW**)
  - curves,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

Processor Architecture

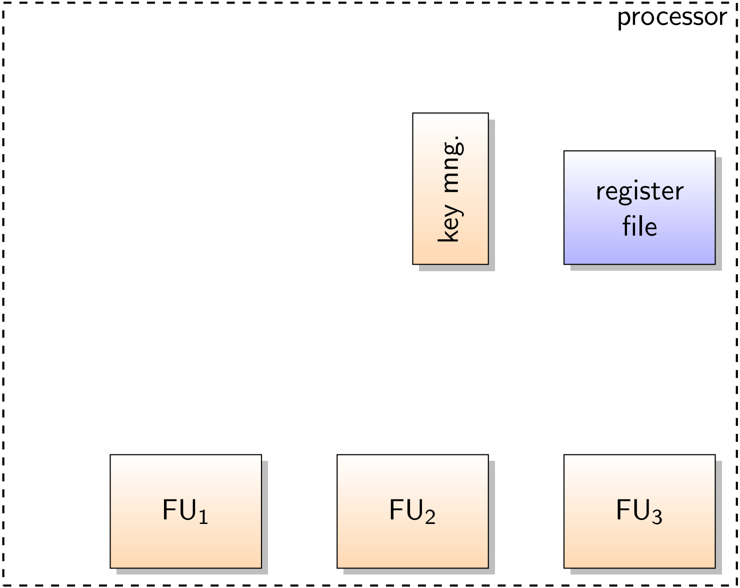
Processor Architecture



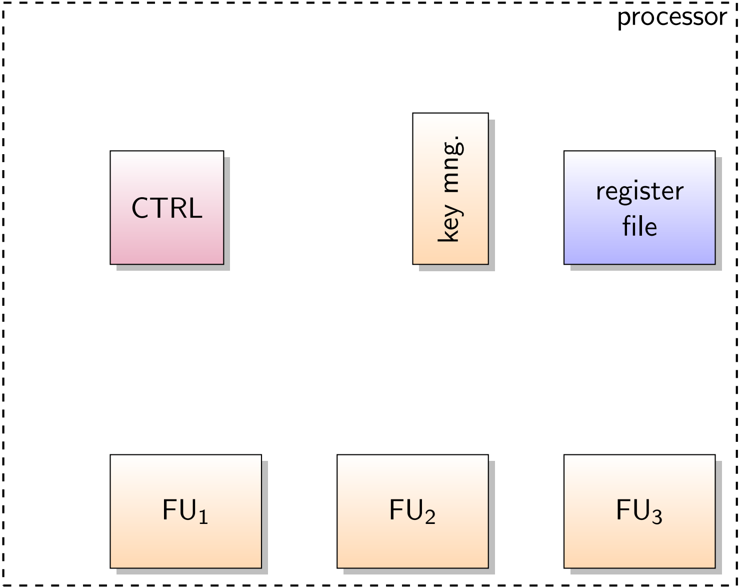
Processor Architecture



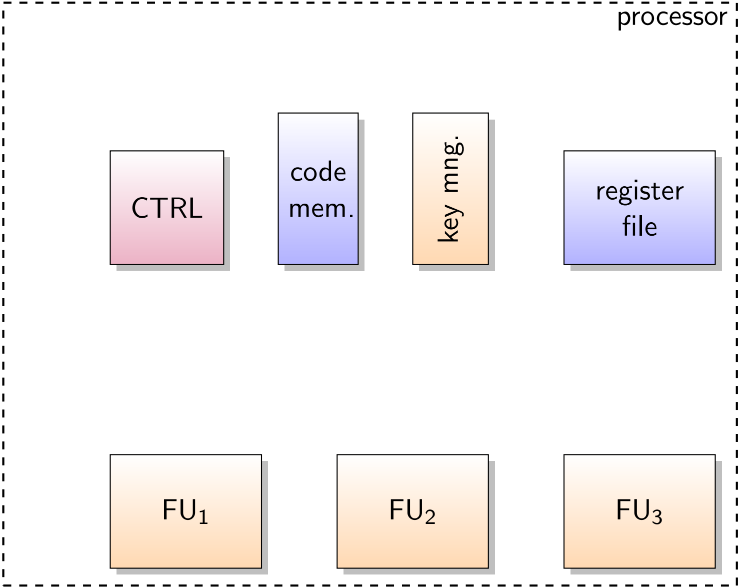
Processor Architecture



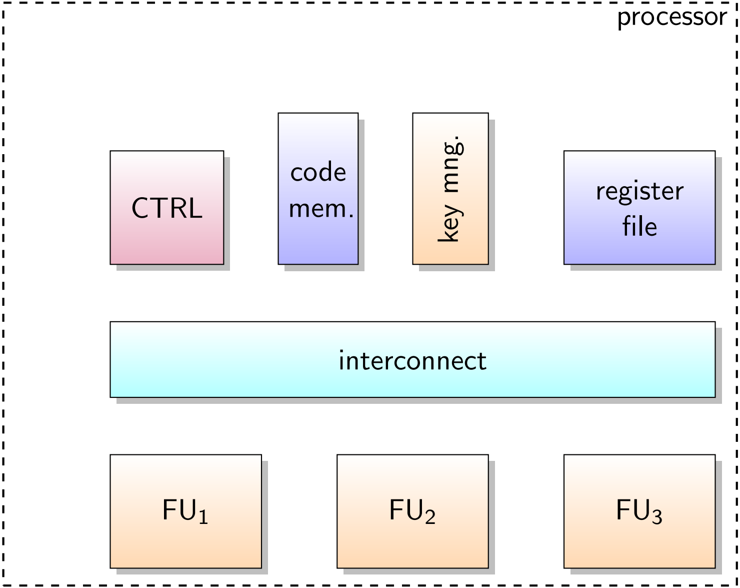
Processor Architecture



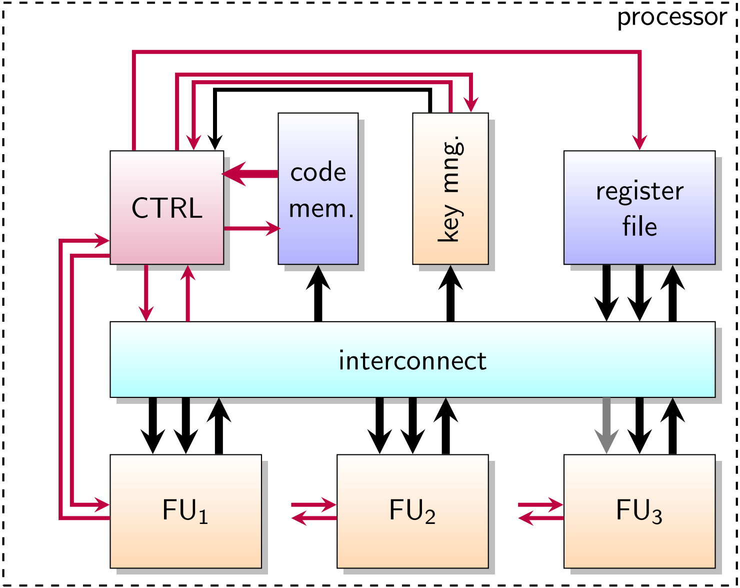
Processor Architecture



Processor Architecture

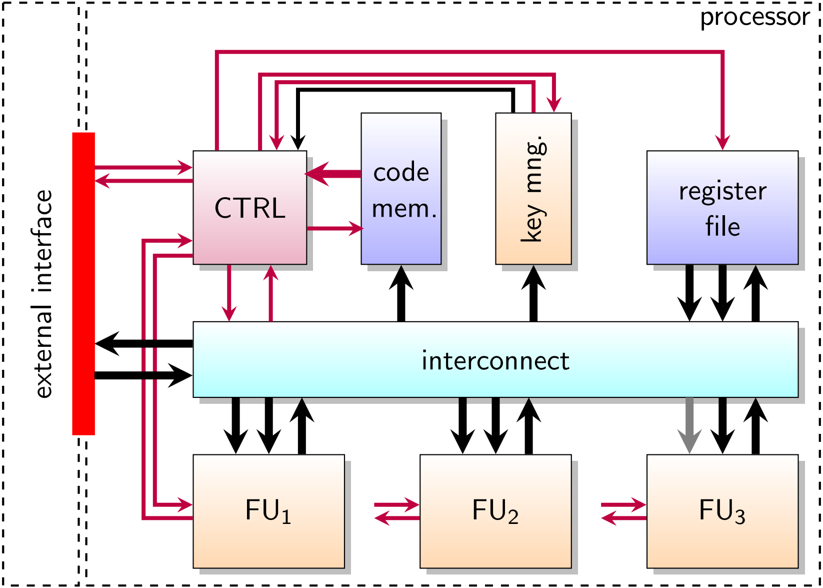


Processor Architecture



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

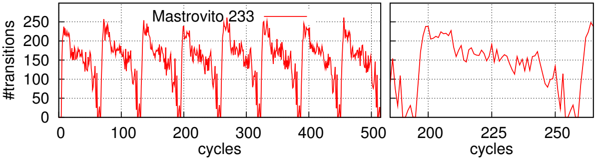
Processor Architecture



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

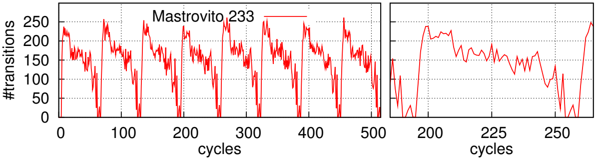
Protected $\mathbb{F}_{2^m}$ Multipliers

Unprotected



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

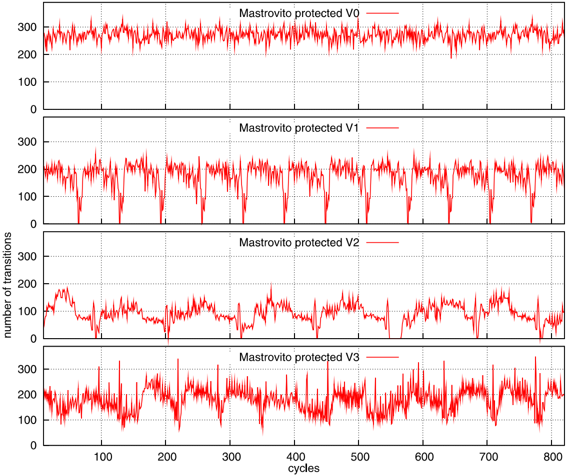
Unprotected



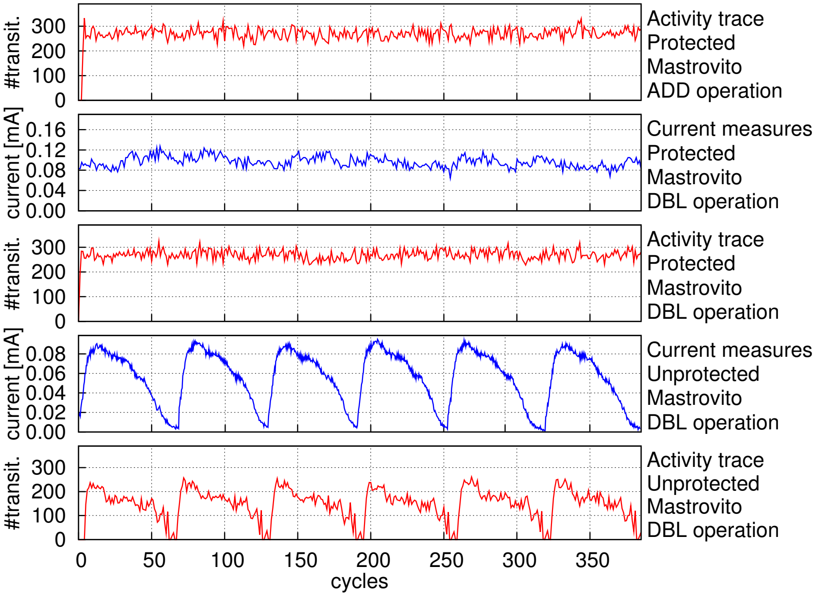
Protected

Overhead:

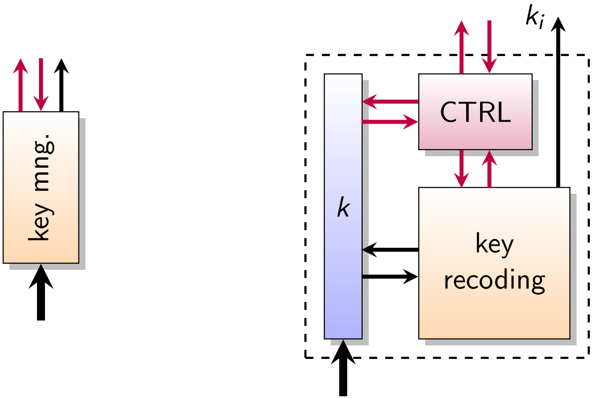
Area/time $< 10\,\%$



Protected Processor for $\mathbb{F}_{2^m}$



Key Management Unit



- On-the-fly recoding of k: binary, λ-NAF ($\lambda \in \{2,3,4,5\}$), variants (fixed/sliding), double-base [6] and multiple-base [7] number systems (w/wo randomization), addition chains [20], other ?
- Specific private path in the interconnect (no key leaks in RF or FUs)

Double-Base Number System

Standard radix-2 representation:

$$k = \sum_{i=0}^{t-1} k_i 2^i = \begin{array}{|c|c|c|c|c|c|}\hline k_{t-1} & k_{t-2} & \cdots & k_2 & k_1 & k_0 \\ \hline \end{array} \; t \text{ explicit digits}$$

Double-Base Number System

Standard radix-2 representation:

$$k = \sum_{i=0}^{t-1} k_i 2^i = \begin{array}{|c|c|c|c|c|c|} \multicolumn{1}{c}{2^{t-1}} & \multicolumn{1}{c}{2^{t-2}} & \multicolumn{1}{c}{\cdots} & \multicolumn{1}{c}{2^2} & \multicolumn{1}{c}{2^1} & \multicolumn{1}{c}{2^0} \\ \hline k_{t-1} & k_{t-2} & \cdots & k_2 & k_1 & k_0 \\ \hline \end{array}$$

implicit weights

t explicit digits

Digits: $k_i \in \{0, 1\}$, typical size: $t \in \{160, \ldots, 600\}$

Double-Base Number System

Standard radix-2 representation:

$$k = \sum_{i=0}^{t-1} k_i 2^i =$$

2^{t-1}	2^{t-2}	$\cdots$	2^2	2^1	2^0	implicit weights
k_{t-1}	k_{t-2}	$\cdots$	k_2	k_1	k_0	t explicit digits

Digits: $k_i \in \{0, 1\}$, typical size: $t \in \{160, \ldots, 600\}$

Double-Base Number System (DBNS):

$$k = \sum_{j=0}^{n-1} k_j 2^{a_j} 3^{b_j} =$$

Double-Base Number System

Standard radix-2 representation:

$$k = \sum_{i=0}^{t-1} k_i 2^i = \begin{array}{|c|c|c|c|c|c|} \hline k_{t-1} & k_{t-2} & \cdots & k_2 & k_1 & k_0 \\ \hline \end{array}$$

2^{t-1}	2^{t-2}	$\cdots$	2^2	2^1	2^0	implicit weights
k_{t-1}	k_{t-2}	$\cdots$	k_2	k_1	k_0	t explicit digits

Digits: $k_i \in \{0, 1\}$, typical size: $t \in \{160, \ldots, 600\}$

Double-Base Number System (DBNS):

$$k = \sum_{j=0}^{n-1} k_j 2^{a_j} 3^{b_j} =$$

k_{n-1}	$\cdots$	k_1	k_0	n $(2,3)-$terms
a_{n-1}	$\cdots$	a_1	a_0	explicit "digits"
b_{n-1}	$\cdots$	b_1	b_0	explicit ranks

$a_j, b_j \in \mathbb{N}$, $k_j \in \{1\}$ or $k_j \in \{-1, 1\}$, size $n \approx \log t$

Double-Base Number System

Standard radix-2 representation:

$$k = \sum_{i=0}^{t-1} k_i 2^i =$$

2^{t-1}	2^{t-2}	$\cdots$	2^2	2^1	2^0	implicit weights
k_{t-1}	k_{t-2}	$\cdots$	k_2	k_1	k_0	t explicit digits

Digits: $k_i \in \{0, 1\}$, typical size: $t \in \{160, \ldots, 600\}$

Double-Base Number System (DBNS):

$$k = \sum_{j=0}^{n-1} k_j 2^{a_j} 3^{b_j} =$$

k_{n-1}	$\cdots$	k_1	k_0	n $(2,3)-$terms
a_{n-1}	$\cdots$	a_1	a_0	explicit "digits"
b_{n-1}	$\cdots$	b_1	b_0	explicit ranks

$a_j, b_j \in \mathbb{N}$, $k_j \in \{1\}$ or $k_j \in \{-1, 1\}$, size $n \approx \log t$

DBNS is a very redundant and sparse representation: $1701 = (11010100101)_2$

$$\begin{aligned} 1701 &= 243 + 1458 = 2^0 3^5 + 2^1 3^6 = (1, 0, 5), (1, 1, 6) \\ &= 1728 - 27 = 2^6 3^3 - 2^0 3^3 = (1, 6, 3), (-1, 0, 3) \\ &= 729 + 972 = 2^0 3^6 + 2^2 3^5 = (1, 0, 6), (1, 2, 5) \\ &\ldots \end{aligned}$$

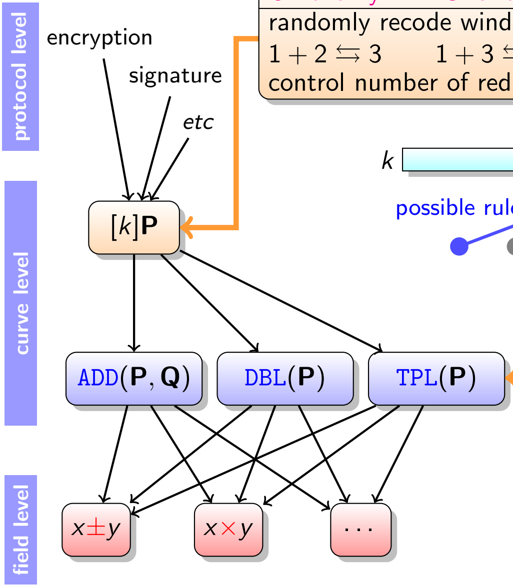
Randomized DBNS Recoding of the Scalar k

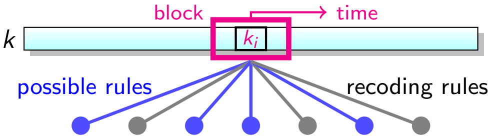
On-the-fly DBNS random recoding for the scalar k

randomly recode windows of the scalar k on-the-fly:

$1+2 \leftrightarrows 3 \qquad 1+3 \leftrightarrows 2^2 \qquad 1+2^3 \leftrightarrows 3^2 \qquad \ldots$

control number of reductions ($\leftarrow$) and expansions ($\rightarrow$)





Point tripling operation

$\mathbf{Q} = \mathtt{TPL}(\mathbf{P}) = \mathbf{P} + \mathbf{P} + \mathbf{P}$

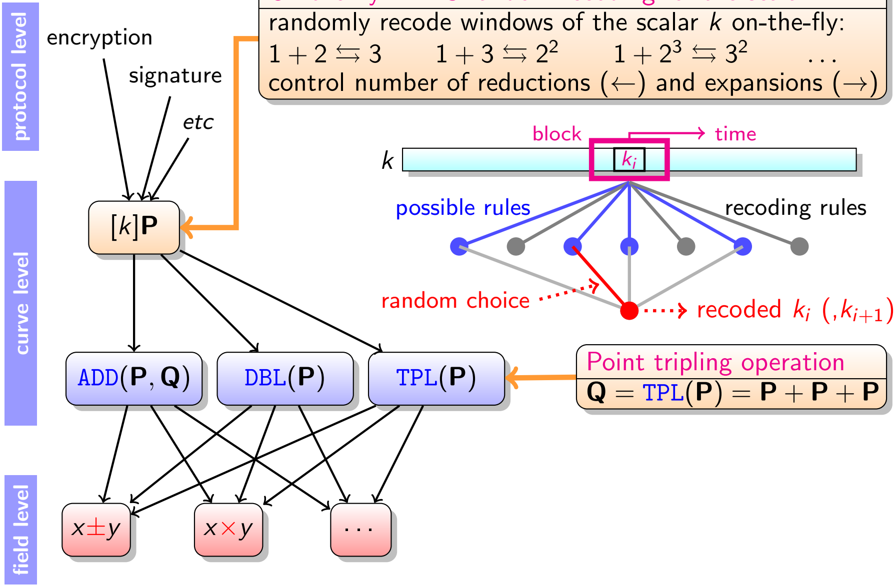
Randomized DBNS Recoding of the Scalar k

On-the-fly DBNS random recoding for the scalar k

randomly recode windows of the scalar k on-the-fly:

$1 + 2 \leftrightarrows 3 \qquad 1 + 3 \leftrightarrows 2^2 \qquad 1 + 2^3 \leftrightarrows 3^2 \qquad \dots$

control number of reductions ($\leftarrow$) and expansions ($\rightarrow$)



Point tripling operation

$\mathbf{Q} = \mathtt{TPL}(\mathbf{P}) = \mathbf{P} + \mathbf{P} + \mathbf{P}$

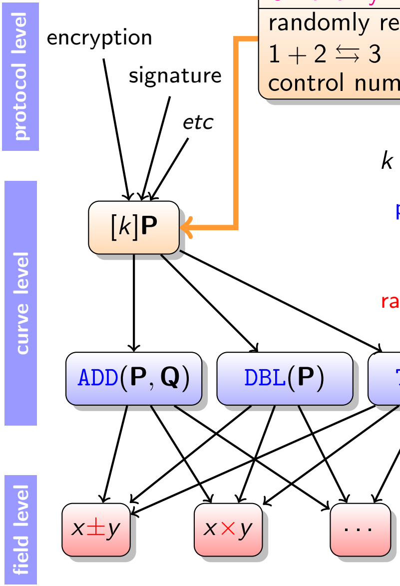
Randomized DBNS Recoding of the Scalar k

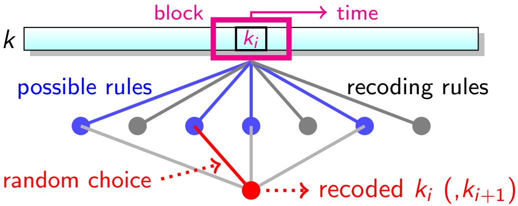
On-the-fly DBNS random recoding for the scalar k

randomly recode windows of the scalar k on-the-fly:

$1 + 2 \leftrightarrows 3 \qquad 1 + 3 \leftrightarrows 2^2 \qquad 1 + 2^3 \leftrightarrows 3^2 \qquad \ldots$

control number of reductions ($\leftarrow$) and expansions ($\rightarrow$)





Point tripling operation

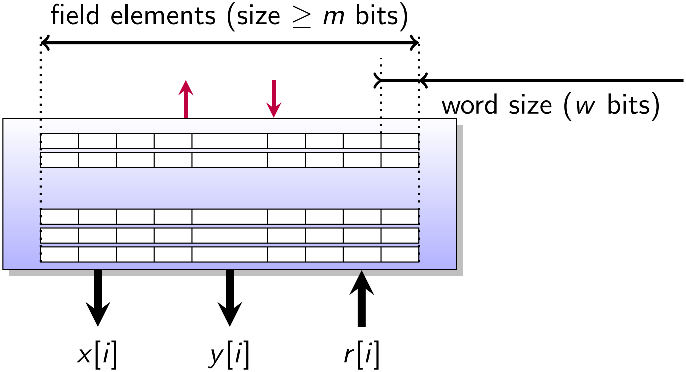
$\mathbf{Q} = \mathtt{TPL}(\mathbf{P}) = \mathbf{P} + \mathbf{P} + \mathbf{P}$

DBNS is redundant $\Rightarrow$ security $\nearrow$

DBNS is sparse $\Rightarrow$ 20–30 % speed $\nearrow$

Ref: [6]

Register File (≈ Dual Port Memory)

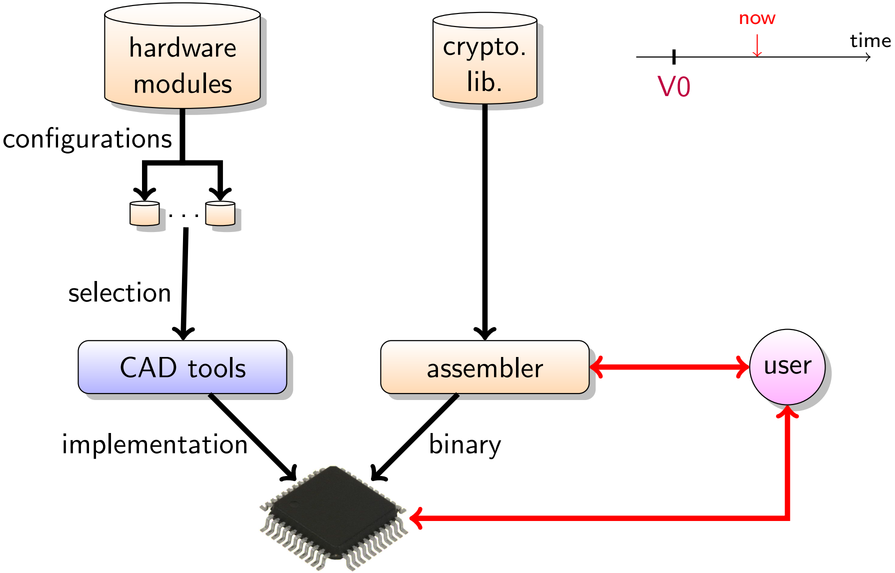


Control signals: addresses (port A, port B), read/write, write enable

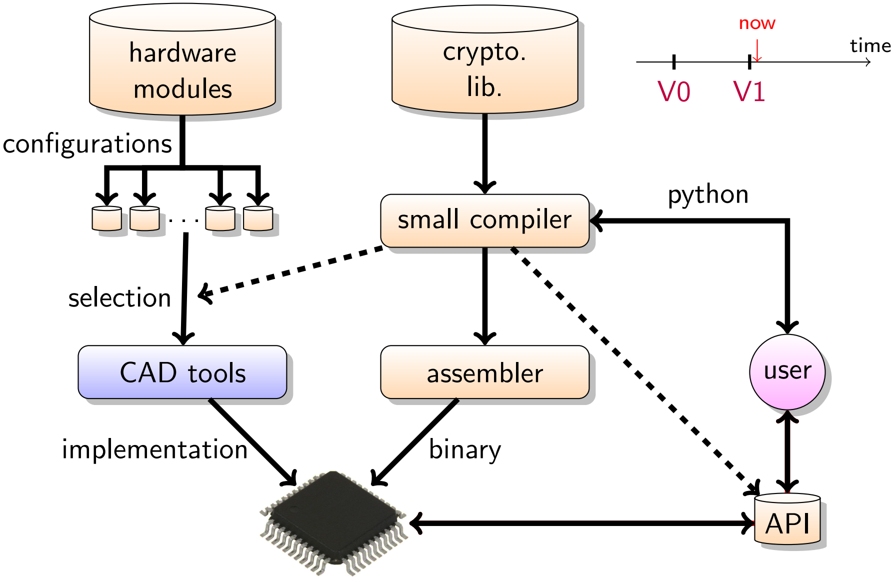
Specific addressing model for $\mathbb{F}_q$ elements through an intermediate address table with hardware loop

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

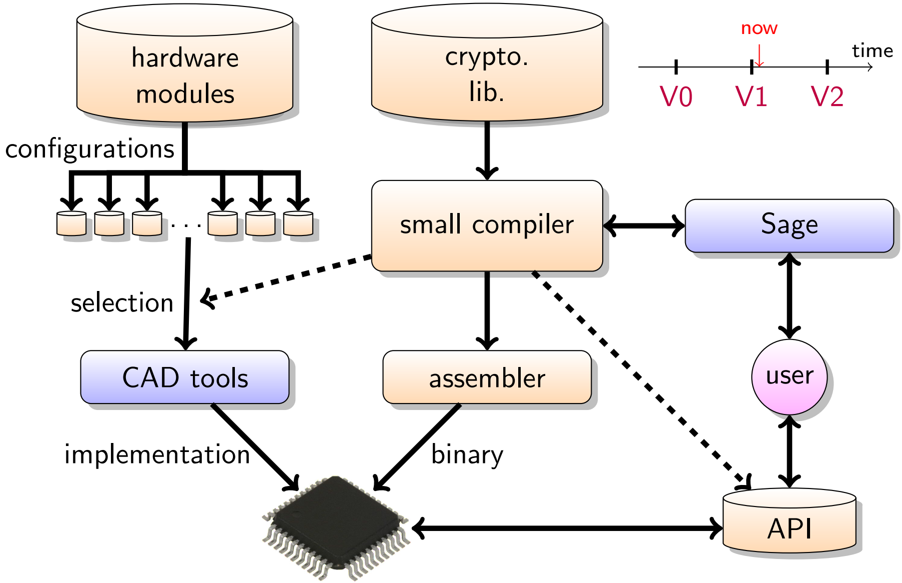
Developed Programming Tools



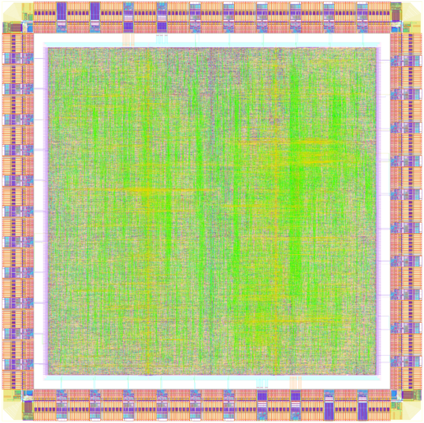
Developed Programming Tools



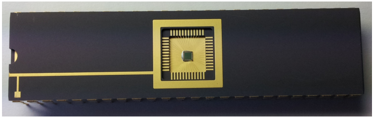
Developed Programming Tools



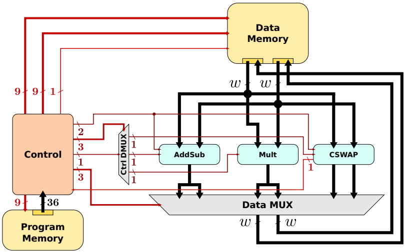
PAVOIS Integrated Circuit

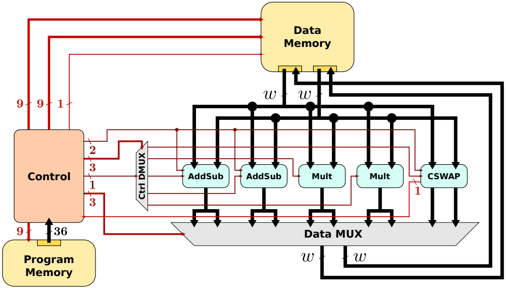


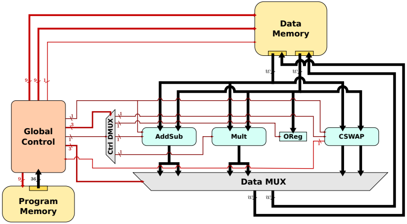
ECC 256 bits
GF(p) with p configurable
65 nm CMOS
1.5 mm^2
algo. & arith. protections
basic layout obfuscation

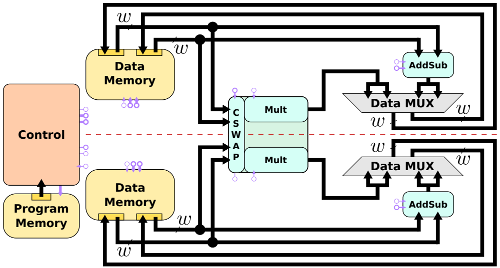


Cryptoprocessors for HECC









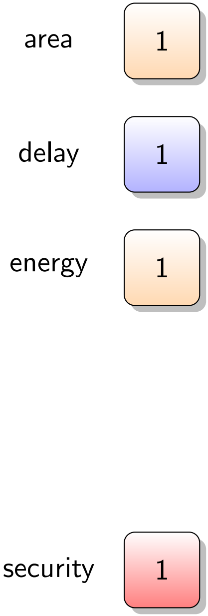
Our Long Term Objectives

Study the links between:

- cryptosystems
- arithmetic algorithms
- $\mathbb{F}_q$, pts representations
- architectures & units
- circuit optimisations

to ensure

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



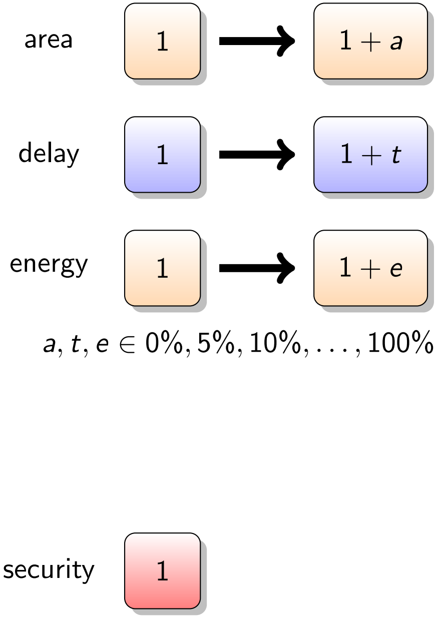
Our Long Term Objectives

Study the links between:

- cryptosystems
- arithmetic algorithms
- $\mathbb{F}_q$, pts representations
- architectures & units
- circuit optimisations

to ensure

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



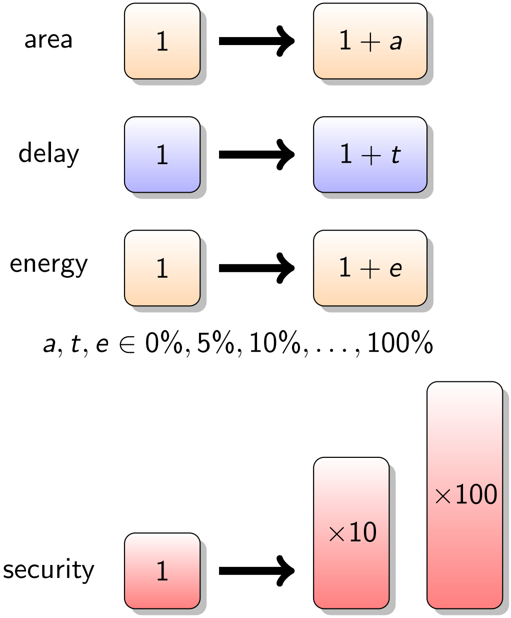
Our Long Term Objectives

Study the links between:

- cryptosystems
- arithmetic algorithms
- $\mathbb{F}_q$, pts representations
- architectures & units
- circuit optimisations

to ensure

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



The end, questions ?

Contact:

- `mailto:arnaud.tisserand@univ-ubs.fr`
- `http://www-labsticc.univ-ubs.fr/~tisseran`
- CNRS
  Lab-STICC, Centre Recherche UBS
  Rue St Maudé. BP 92116. 56321 Lorient cedex, France

Thank you

References I

[1] K. Bigou and A. Tisserand.
Improving modular inversion in RNS using the plus-minus method.
In G. Bertoni and J.-S. Coron, editors, *Proc. 15th International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, volume 8086 of *LNCS*, pages 233–249, Santa Barbara, CA, USA, August 2013. Springer.

[2] K. Bigou and A. Tisserand.
Single base modular multiplication for efficient hardware RNS implementations of ECC.
In T. Guneysu and H. Handschuh, editors, *Proc. 17th International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, volume 9293 of *LNCS*, pages 123–140, Saint-Malo, France, September 2015. Springer.

[3] K. Bigou and A. Tisserand.
Hybrid position-residues number system.
In J. Hormigo, S. Oberman, and N. Revol, editors, *Proc. 23rd Symposium on Computer Arithmetic (ARITH)*, pages 126–133, Santa Clara, CA, U.S.A, July 2016. IEEE Computer Society.

[4] A. Byrne, F. Crowe, W. P. Marnane, N. Meloni, A. Tisserand, and E. M. Popovici.
SPA resistant elliptic curve cryptosystem using addition chains.
Int. J. High Performance Systems Architecture, 1(2):133–142, October 2007.

[5] A. Byrne, N. Meloni, A. Tisserand, E. M. Popovici, and W. P. Marnane.
Comparison of simple power analysis attack resistant algorithms for an elliptic curve cryptosystem.
Journal of Computers, 2(10):52–62, 2007.

[6] 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.

[7] 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.

References II

[8] 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.

[9] J. Chen, A. Tisserand, E. M. 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.

[10] H. Cohen and G. Frey, editors.
Handbook of Elliptic and Hyperelliptic Curve Cryptography.
Discrete Mathematics and Its Applications. Chapman & Hall/CRC, July 2005.

[11] G. Gallin, T. U. Celik, and A. Tisserand.
Architecture level optimizations for kummer based HECC on FPGAs.
In Arpita Patra and Nigel P. Smart, editors, *Proc. 18th International Conference on Cryptology in India (IndoCrypt)*, volume 10698 of *LNCS*, pages 44–64, Chennai, India, December 2017. Springer.

[12] G. Gallin and A. Tisserand.
Hyper-threaded multiplier for HECC.
In *Proc. 51st Asilomar Conference on Signals, Systems and Computers*, pages 447–451, Pacific Grove, CA, USA, October 2017. IEEE.

[13] G. Gallin and A. Tisserand.
Generation of finely-pipelined GF(P) multipliers for flexible curve based cryptography on FPGAs.
IEEE Transactions on Computers, 69(11):1612–1622, November 2019.

[14] D. Hankerson, A. Menezes, and S. Vanstone.
Guide to Elliptic Curve Cryptography.
Springer, 2004.

References III

[15] A. Lucas and A. Tisserand.
Microcontroller implementation of simultaneous protections against observation and perturbation attacks for ECC.
In *Proc. 15th International Conference on Security and Cryptography (SECRYPT)*, Porto, Portugal, July 2018. Springer.

[16] 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.

[17] 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 271–276, Brno, Czech Republic, May 2012.

[18] 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.

[19] D. Pamula and A. Tisserand.
Fast and secure finite field multipliers.
In *Proc. 18th Euromicro Conference on Digital System Design (DSD)*, pages 653–660, Madeira, Portugal, August 2015.

[20] 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.

[21] A. Tisserand.
Hardware accelerators for ECC and HECC.
In *19th Workshop on Elliptic Curve Cryptography (ECC)*, Bordeaux, France, September 2015.
Invited talk.