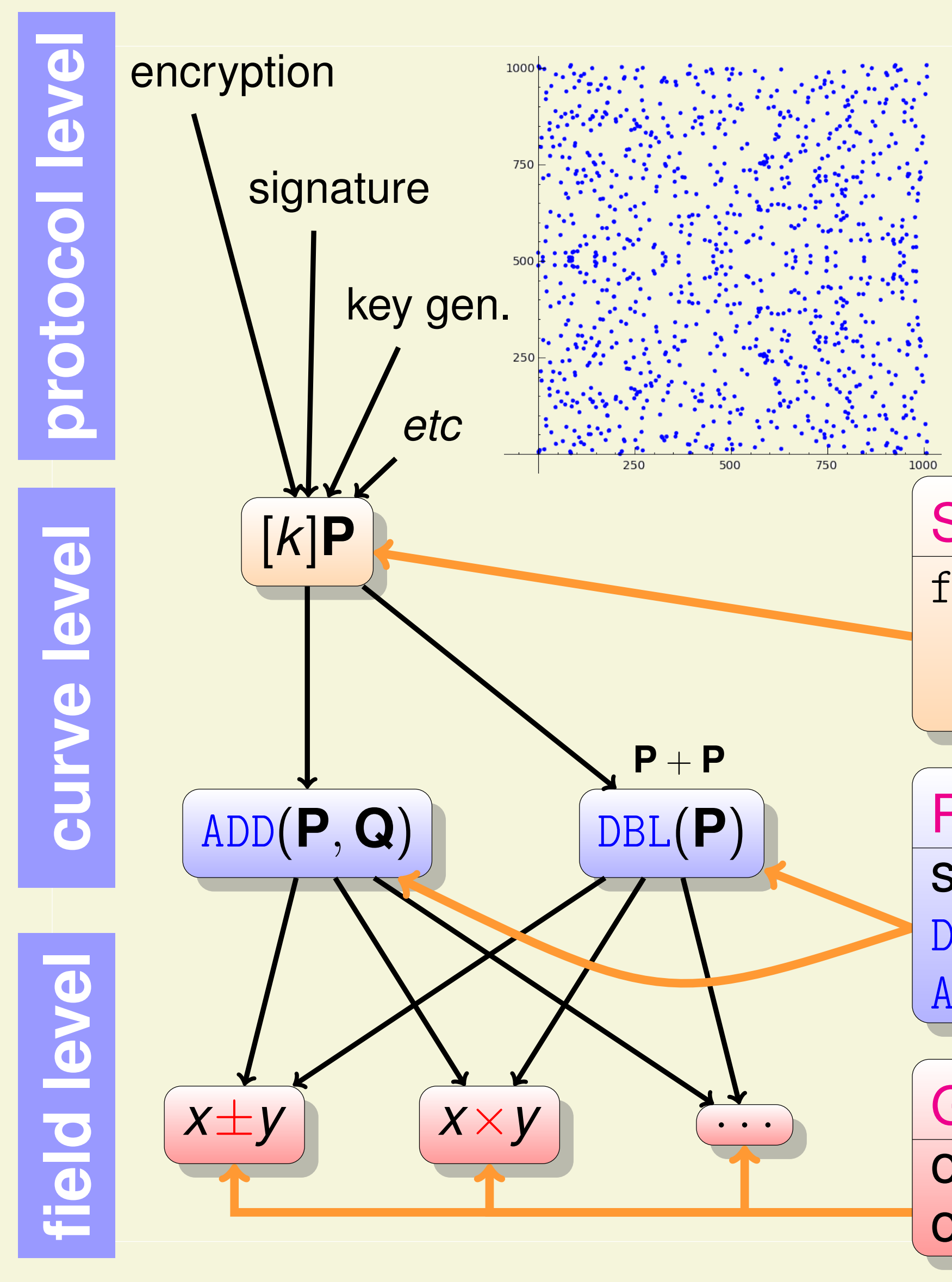
Hardware and Arithmetic for Hyperelliptic Curves Cryptography

Arnaud Tisserand, Gabriel Gallin

HAH Project, IRISA–IRMAR
Hardware and Arithmetic for Hyperelliptic Curves Cryptography

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

GF(p), GF(2^m), t : 160–600 bits

$k = (k_{t-1}k_{t-2}\dots k_1k_0)_2 \in \mathbb{N}$

Scalar multiplication operation

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

Point addition/doubling operations

sequence of finite field operations

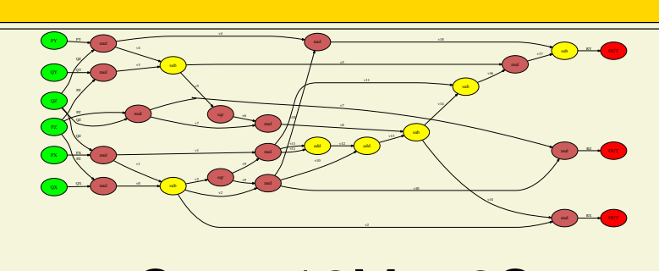
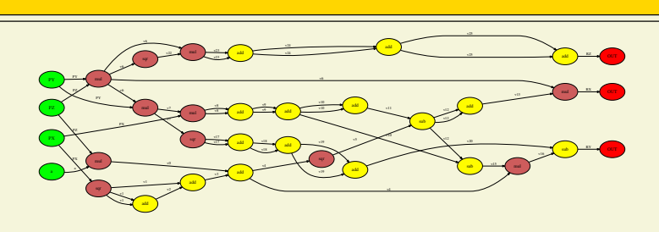
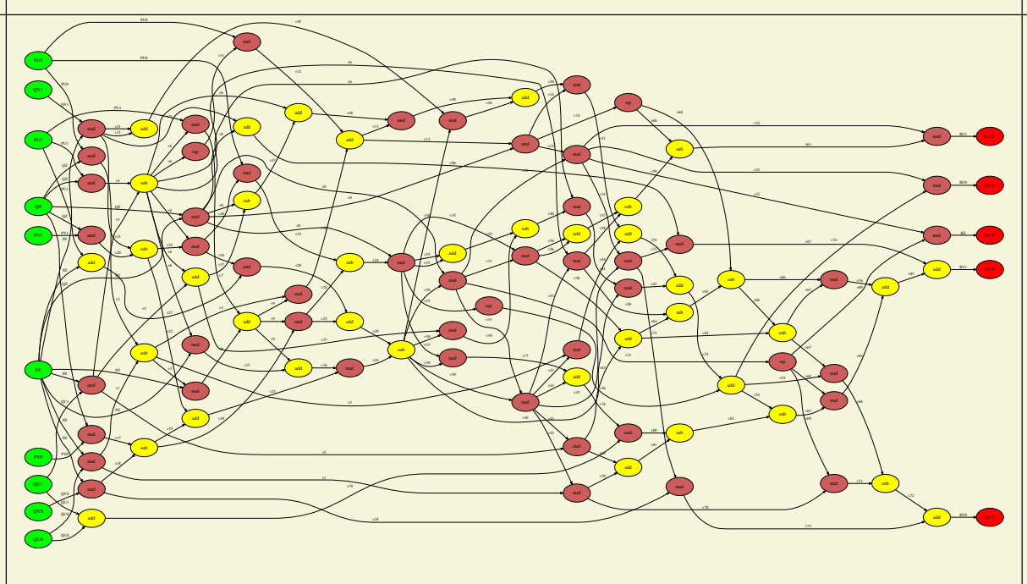
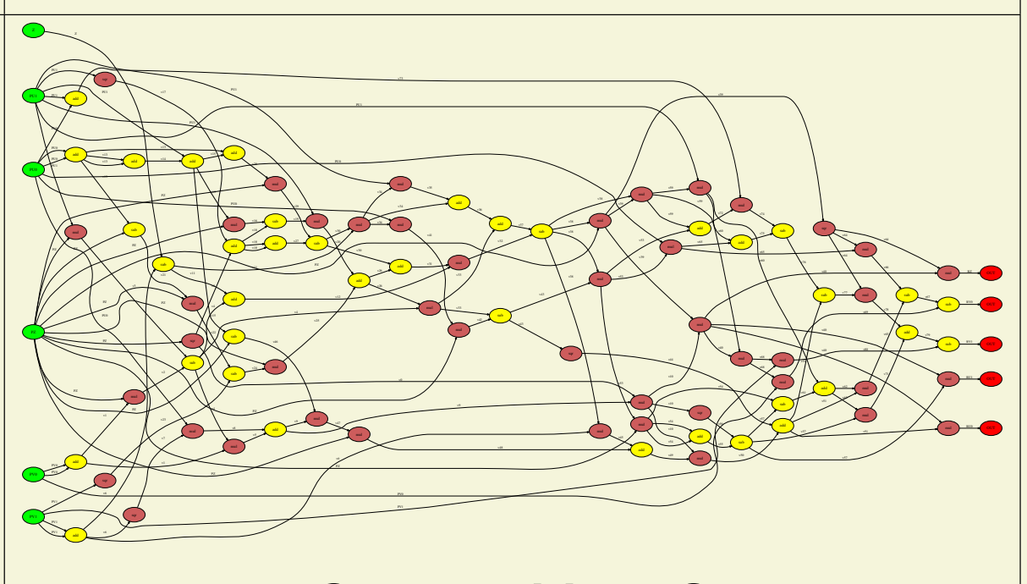
DBL: $v_1 = z_1^2, v_2 = x_1 - v_1, \dots$

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

GF(p) or GF(2^m) operations

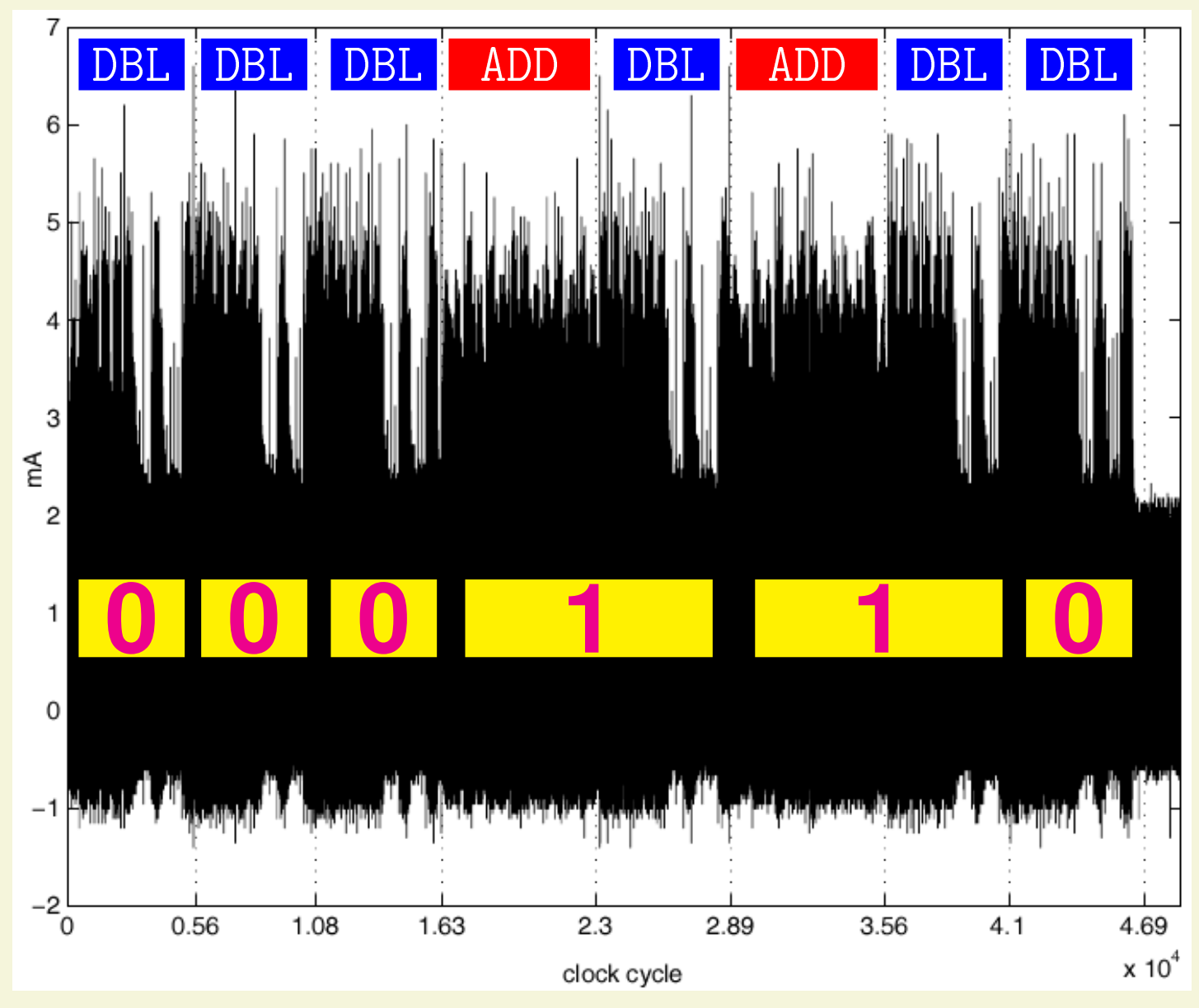
operation modulo large prime (GF(p)) or irreducible polynomial (GF(2^m))

2. From ECC to HECC

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

Examples of computation expressions for projective coordinates

3. Side Channel Attacks (SCAs)



Side channels:

- Power consumption
- Electromagnetic radiation
- Computation timings

Attacks:

- Simple analysis
- Differential analysis (statistics)
- Templates and learning

4. Protections & Counter-Measures Against SCAs

- Uniform comp. durations
- Uniform power/EM profile
- Random behavior
- Circuit reconfiguration
- detection/correction codes
- Add noise (!)

Example: use redundant number systems

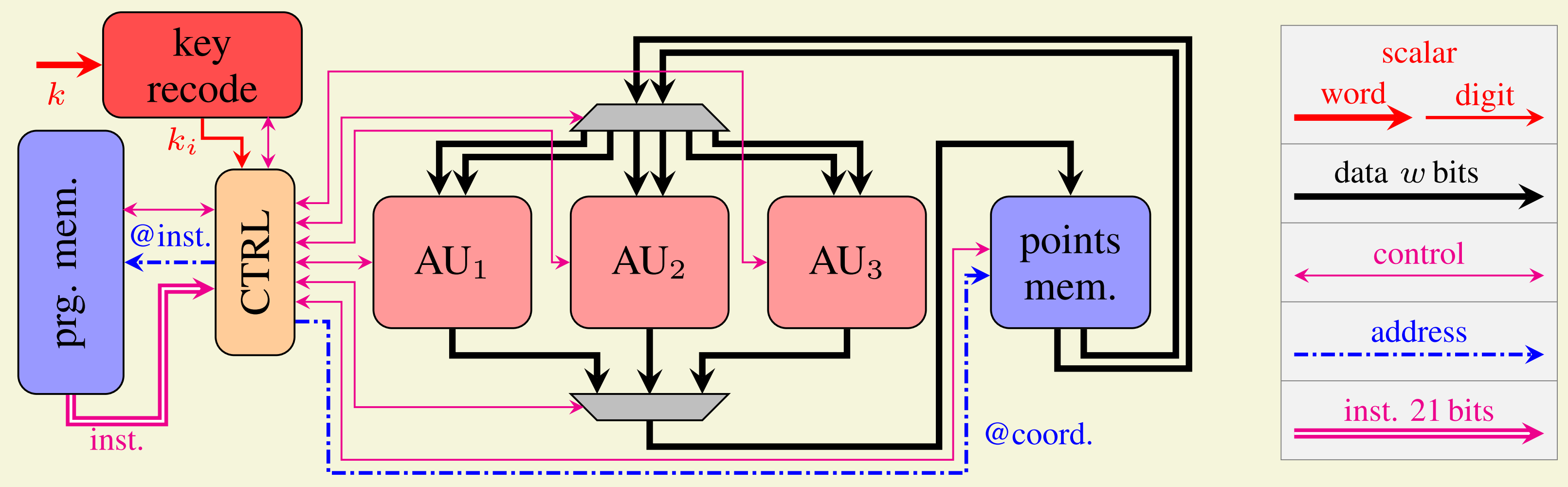
$$\begin{tikzcd}
 & & & k \arrow[dlll] \arrow[dll] \arrow[dl] \arrow[d] \arrow[dr] \arrow[drr] & & & \\
R_1(k) \arrow[d] & R_2(k) \arrow[d] & R_3(k) \arrow[d] & R_4(k) \arrow[d] & R_5(k) \arrow[d] & R_6(k) \arrow[d] & \cdots \\
{[R_1(k)]\mathbf{P}} & {[R_2(k)]\mathbf{P}} & {[R_3(k)]\mathbf{P}} & {[R_4(k)]\mathbf{P}} & {[R_5(k)]\mathbf{P}} & {[R_6(k)]\mathbf{P}} & \cdots
\end{tikzcd}$$

Random recoding: $\forall i \quad [R_i(k)]\mathbf{P} = [k]\mathbf{P}$

5. HAH Project Objectives

- Efficient algorithms and representations for HECC
- HECC protections against SCAs (passive and active)
- Fast, low-power and secure hardware implementations (open source hardware code and programming tools)
- Intensive security evaluation using our SCA setup

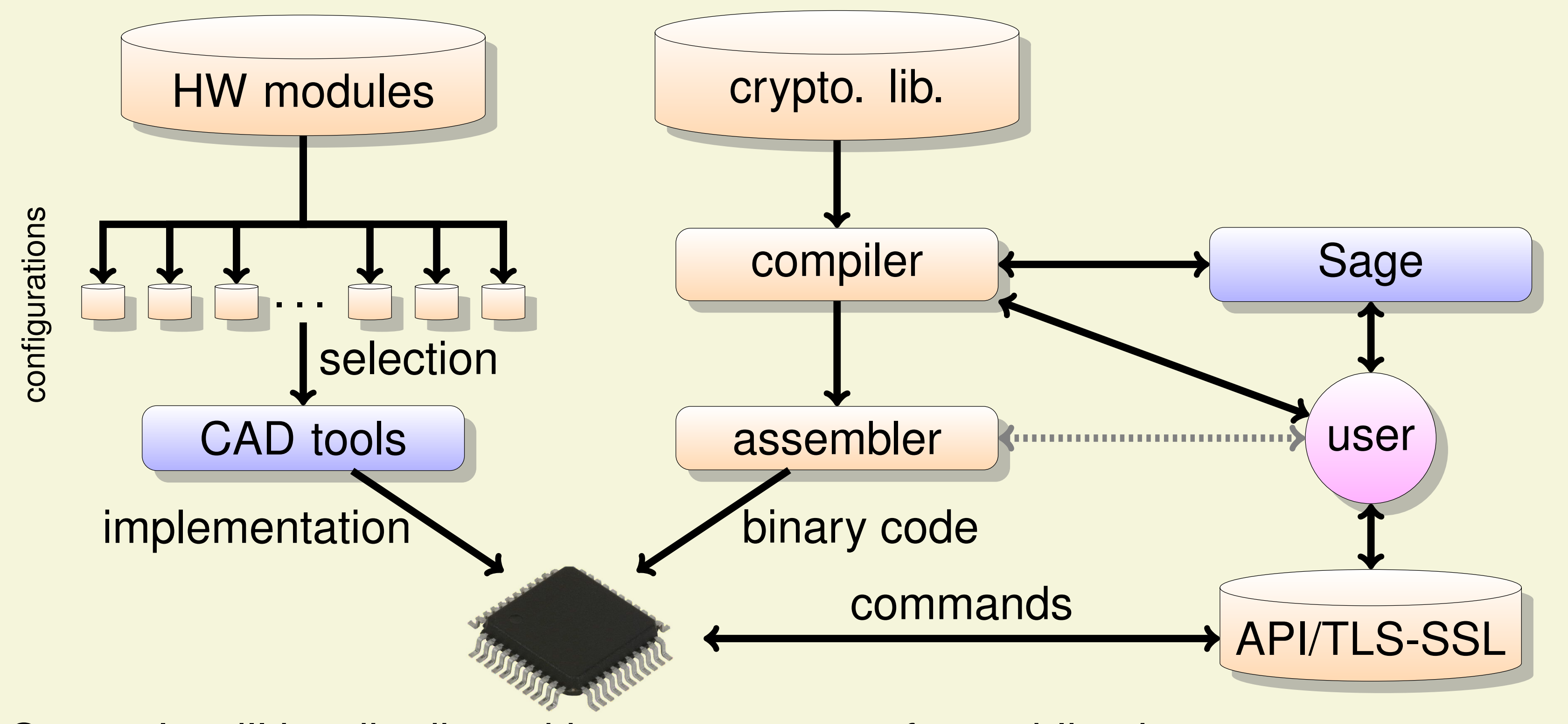
6. Developed Crypto-Processor(s)



- Arithmetic Units (AUs): $\pm$, $\times$, $\div$ over GF(p)/GF(2^m) various configurations (area vs speed, internal protection)
- Various key recoding methods (and dedicated units)
- Configuration: field size, internal word size, #AUs, type(AUs)
- Circuit/architecture level protections

Remark: crypto-processor development started in PAVOIS ANR project

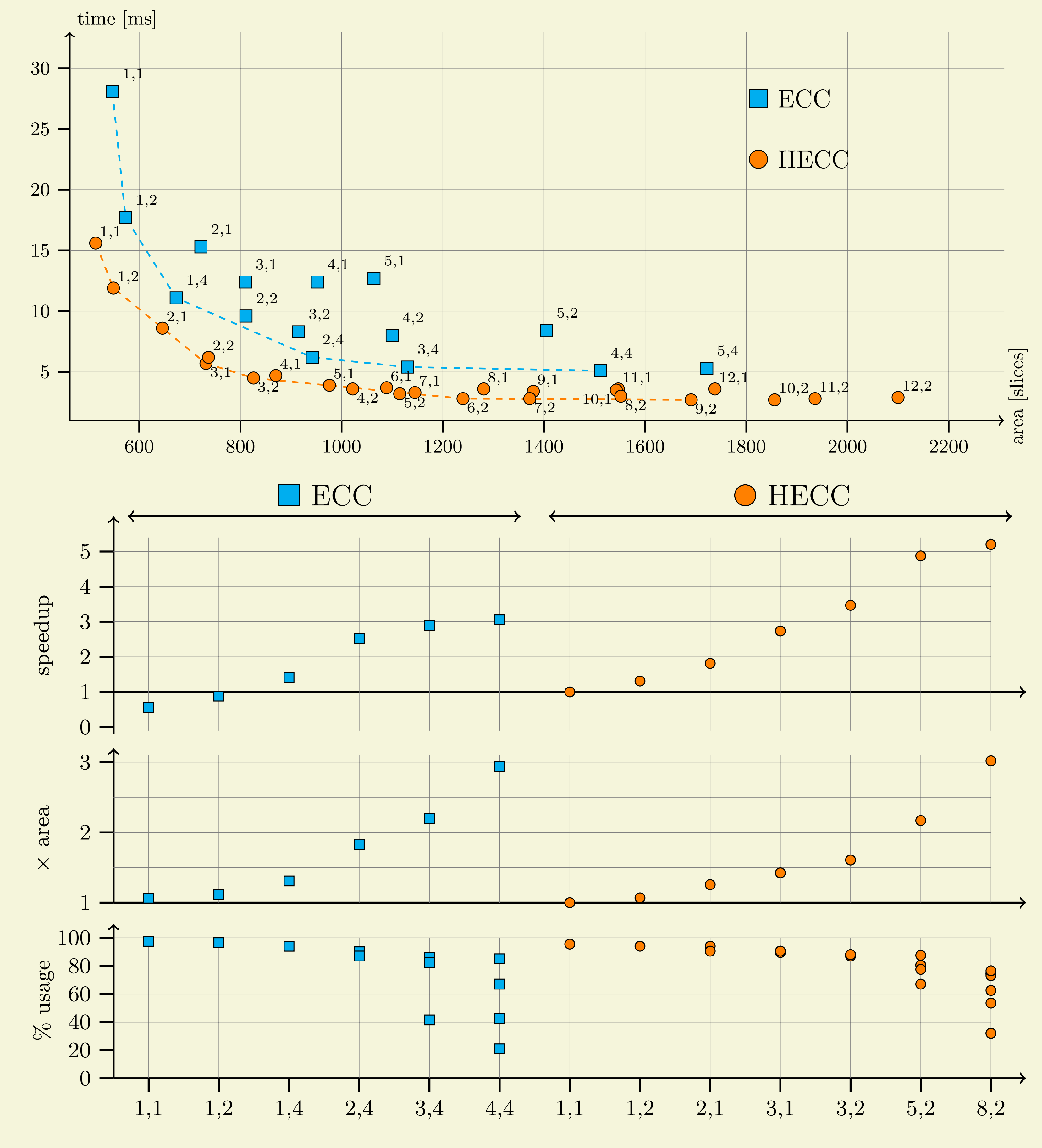
7. Programming Tools for Our Crypto-Processor(s)



Our tools will be distributed in open source after publication

8. Implementation Results on FPGA

XC6SLX75 FPGA, GF(p), 256-bit ECC or 128-bit HECC, internal word size $w = 32$ bits



parameters (x, y)=(number of multiplers, multiplier width)

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