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/

HAH project 2014–2017

Hardware and Arithmetic for Hyperelliptic Curves Cryptography

- IRISA (Lannion)
- IRMAR (Rennes)

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

Addition and doubling operations

Scalar multiplication operation

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

Point addition/doubling operations

sequence of finite field operations

$\text{DBL}: v_1 = z_2, v_2 = x_1 - v_1, ...$

$\text{ADD}: w_1 = z_2, w_2 = z_1 \times w_1, ...$

Field operations

operation modulo large prime ($F_p$) or irreducible polynomial ($F_{2^m}$)
Introduction

Encryption
Signature
Etc

$E : y^2 = x^3 + 4x + 20$ over $GF(1009)$
Points: $P, Q = (x, y)$ or $(x, y, z)$ or ...
Introduction

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

points: $P, Q = (x, y)$ or $(x, y, z)$ or ... 
coordinates: $x, y, z \in \text{GF}(\cdot)$

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

$k = (k_{t-1}k_{t-2} \ldots k_1k_0)2 \in \mathbb{N}$
**Introduction**

Encryption

Signature

etc

protocol level

\[ [k]P \]

signature

encryption

curve level

field level

\[ E : y^2 = x^3 + 4x + 20 \text{ over } GF(1009) \]

points: \( P, Q = (x, y) \) or \((x, y, z)\) or ...

coordinates: \( x, y, z \in GF(\cdot) \)

\( \mathbb{F}_p, \mathbb{F}_{2^m}, t : 80-600 \text{ bits} \)

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

Scalar multiplication operation

for \( i \) from 0 to \( t - 1 \) do

\[ \text{if } k_i = 1 \text{ then } Q = \text{ADD}(P, Q) \]

\( P = \text{DBL}(P) \)
Introduction

Encryption
Signature
etc

Protocol level

Field level

Curve level

Scalar multiplication operation

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

Point addition/doubling operations

Sequence of finite field operations

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

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

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

Points: $P, Q = (x, y)$ or $(x, y, z)$ or ... coordinates: $x, y, z \in \text{GF}(\cdot)$

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

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

$A. Tisserand, CNRS–IRISA–CAIRN. Hardware Accelerators for ECC and HECC
Introduction

\[ E : y^2 = x^3 + 4x + 20 \text{ over } \text{GF}(1009) \]

points: \( P, Q = (x, y) \) or \((x, y, z)\) or \ldots

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

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

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

Scalar multiplication operation

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

Point addition/doubling operations

sequence of finite field operations

DBL: \( v_1 = z_1^2, v_2 = x_1 - v_1, \ldots \)
ADD: \( w_1 = z_1^2, w_2 = z_1 \times w_1, \ldots \)

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

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

- Encryption
- Signature
- etc

Scalar multiplication operation

\[
\text{for } i \text{ from } 0 \text{ to } t - 1 \text{ do }
\begin{align*}
& \text{if } k_i = 1 \text{ then } Q = \text{ADD}(P, Q) \\
& P = \text{DBL}(P)
\end{align*}
\]
Side Channel Attacks

Scalar multiplication operation

for \( i \) from 0 to \( t - 1 \) do
  if \( k_i = 1 \) then \( Q = ADD(P, Q) \)
  \( P = DBL(P) \)
Side Channel Attacks

 Scalar multiplication operation

\[
\text{for } i \text{ from 0 to } t - 1 \text{ do }
\begin{align*}
\text{if } k_i = 1 \text{ then } Q &= \text{ADD}(P, Q) \\
P &= \text{DBL}(P)
\end{align*}
\]
Side Channel Attacks

**Protocol Level**
- Encryption
- Signature
- etc

**Curve Level**
- $[k]P$
- $\text{ADD}(P, Q)$
- $\text{DBL}(P)$

**Field Level**
- $x \pm y$
- $x \times y$
- ...

Scalar multiplication operation

\[
\text{for } i \text{ from } 0 \text{ to } t - 1 \text{ do } \\
\text{if } k_i = 1 \text{ then } Q = \text{ADD}(P, Q) \\
P = \text{DBL}(P)
\]
Side Channel Attacks

encryption
signature

protocol level

signature

etc

curve level

Scalar multiplication operation

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

- simple power analysis (& variants)

field level

A. Tisserand, CNRS–IRISA–CAIRN. Hardware Accelerators for ECC and HECC
Side Channel Attacks

Scalar multiplication operation

\[
\text{for } i \text{ from } 0 \text{ to } t - 1 \text{ do }
\]

- if \( k_i = 1 \) then \( Q = \text{ADD}(P, Q) \)
- \( P = \text{DBL}(P) \)

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

A. Tisserand, CNRS–IRISA–CAIRN. Hardware Accelerators for ECC and HECC
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 $\sim$ 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

- Protocol level:
  - Encryption
  - Signature
  - etc.

- Curve level:
  - $[k]P$
  - ADD($P, Q$)
  - DBL($P$)
  - $P + P$
  - $x \pm y$
  - $x \times y$
  - ...

- Field level:

- 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

- Software (SW):
  - Curves, algorithms, representations (points/elements)
  - K-recoding, ...
  - At design time / at run time

- Security against SCAs:
  - Secure units ($F_{2^m}, F_p$)
  - Secure key storage/management
  - Secure control

A. Tisserand, CNRS–IRISA–CAIRN. Hardware Accelerators for ECC and HECC
Accelerator Specifications

- Performances $\rightarrow$ 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

A. Tisserand, CNRS–IRISA–CAIRN. *Hardware Accelerators for ECC and HECC*
Accelerator Specifications

- **Performances** $\rightarrow$ **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** $\rightarrow$ **software** (SW)
  - curves, algorithms, representations (points/elements), $k$ recoding, …
  - at design time / at run time

---

A. Tisserand, CNRS–IRISA–CAIRN. *Hardware Accelerators for ECC and HECC*
Accelerator Specifications

- Performances $\implies$ 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 $\implies$ software (SW)
  - curves, algorithms, representations (points/elements), $k$ recoding, \ldots
  - at design time / at run time

- Security against SCAs $\implies$ HW
  - secure units ($\mathbb{F}_{2^m}, \mathbb{F}_p$)
  - secure key storage/management
  - secure control
Accelerator Architecture

Data:
- Bit (32, ..., 128) except for k digits
  - Control: a few bits per unit

A. Tisserand, CNRS–IRISA–CAIRN. *Hardware Accelerators for ECC and HECC*
Accelerator Architecture

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

A. Tisserand, CNRS–IRISA–CAIRN. *Hardware Accelerators for ECC and HECC*
Data: \( w \)-bit (32, ..., 128) except for \( k \) digits,
control: a few bits per unit
Accelerator Architecture

- External interface
- Key management
- Register file
- FU1
- FU2
- FU3

Data: w-bit (32, ..., 128) except for k digits,
Control: a few bits per unit

A. Tisserand, CNRS–IRISA–CAIRN. Hardware Accelerators for ECC and HECC
Accelerator Architecture

CTRL

key mng.

register file

FU₁

FU₂

FU₃

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

A. Tisserand, CNRS–IRISA–CAIRN. Hardware Accelerators for ECC and HECC
Accelerator Architecture

CTRL

codemem.

key mng.

register file

FU₁

FU₂

FU₃

Data: \( w \)-bit (32, ..., 128) except for \( k \) digits,

Control: a few bits per unit

A. Tisserand, CNRS–IRISA–CAIRN. Hardware Accelerators for ECC and HECC
Accelerator Architecture

- CTRL
- code mem.
- key mng.
- register file
- interconnect
- FU₁
- FU₂
- FU₃

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

A. Tisserand, CNRS–IRISA–CAIRN. Hardware Accelerators for ECC and HECC
**Data:** \( w \)-bit (32, \ldots, 128) except for \( k \) digits, **control:** a few bits per unit

A. Tisserand, CNRS–IRISA–CAIRN. *Hardware Accelerators for ECC and HECC*
Accelerator Architecture

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

A. Tisserand, CNRS–IRISA–CAIRN. Hardware Accelerators for ECC and HECC
Functional Units for Field Level Operations

\[ x[i] \quad y[i] \quad r[i] \]

FU\(_\alpha\)

\[ \rightarrow \text{data (}w\text{ bits)} \]

\[ \rightarrow \text{control (few bits)} \]

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)

field elements (size $\geq m$ bits)

word size ($w$ bits)

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: $\text{LOAD } \odot x \implies$ HW: loop $x[0], x[1], \ldots x[\ell - 1]$
- randomized addresses
• **On-the-fly recoding** of $k$: binary, $\lambda$-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

A. Tisserand, CNRS–IRISA–CAIRN. *Hardware Accelerators for ECC and HECC*
Protected $\mathbb{F}_{2^m}$ Multipliers

Unprotected

Protected

Overhead: Area/time $< 10\%$

References:
PhD D. Pamula [8]
Articles: [11], [10], [9]
Warning: old dedicated accelerator (similar behavior is expected for our new one)
Circuit-Level Protections for Arithmetic Operators

References: [4] and [3]

A. Tisserand, CNRS–IRISA–CAIRN. Hardware Accelerators for ECC and HECC
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

\(^1\) Cycle Accurate Bit Accurate
Units Impact on Side Channel Information (2/2)

A. Tisserand, CNRS–IRISA–CAIRN. *Hardware Accelerators for ECC and HECC*
Developed Programming Tools

- Developed Programming Tools
  - accelerator modules
  - configurations
  - selection
  - CAD tools
  - implementation
  - assembler
  - binary
  - crypto. lib.
  - now
  - time
  - V0

A. Tisserand, CNRS–IRISA–CAIRN. Hardware Accelerators for ECC and HECC
Developed Programming Tools

- accelerator modules
- crypto. lib.
- compiler
- assembler
- CAD tools
- configuration
- selection
- implementation
- binary
- user
- API/TLS-SSL
Developed Programming Tools

- accelerator modules
- crypto. lib.
- compiler
- Sage
- user
- API/TLS-SSL

configurations

selection

implementation

binary
# Instruction Set

<table>
<thead>
<tr>
<th>Instruction</th>
<th>FUId</th>
<th>@Rid</th>
<th>@Rid</th>
<th>B/U</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ</td>
<td>FUId</td>
<td>@Rid</td>
<td>@Rid</td>
<td>B/U</td>
</tr>
<tr>
<td>WRITE</td>
<td>FUId</td>
<td>@Rid</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LAUNCH</td>
<td>FUId</td>
<td></td>
<td>MODE</td>
<td></td>
</tr>
<tr>
<td>WAIT</td>
<td>FUId</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SETADDRO</td>
<td>@Rid</td>
<td>OFFSET</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SETADDRN</td>
<td>@Rid</td>
<td>#WORD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WRITEK</td>
<td>#WORD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CALL</td>
<td>@DEST</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RET</td>
<td></td>
<td>@DEST</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BZ</td>
<td>@DEST</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BNZ</td>
<td>@DEST</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>JMP</td>
<td>@DEST</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CMPD</td>
<td></td>
<td>DIGIT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SET</td>
<td></td>
<td>FLAGid</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TST</td>
<td></td>
<td>FLAGid</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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

A. Tisserand, CNRS–IRISA–CAIRN. Hardware Accelerators for ECC and HECC
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:

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

1. read fu_mul_0, 0, 1
2. launch fu_mul_0
3. read fu_mul_1, 3, 4
4. launch fu_mul_1
5. wait fu_mul_0
6. write fu_mul_0, 5
7. set OPMODE, 0
8. read fu_add_sub_0, 5, 2
9. launch fu_add_sub_0
10. wait fu_mul_1
11. write fu_mul_1, 6
12. wait fu_add_sub_0
13. write fu_add_sub_0, 5
14. read fu_add_sub_0, 5, 6
15. launch fu_add_sub_0
16. wait fu_add_sub_0
17. write fu_add_sub_0, 5

read $a \& b$
start $ab$
lit $d \& e$
start $de$
wait for $ab$
write $ab$
addition mode (+)
read $ab \& c$
start $(ab) + c$
write $de$
wait for $(ab) + c$
write $(ab) + c$
read $(ab) + c \& de$
start $((ab) + c) + (de)$
wait for $((ab) + c) + (de)$
write $((ab) + c) + (de)$

A. Tisserand, CNRS–IRISA–CAIRN. *Hardware Accelerators for ECC and HECC*
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.

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

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

See details in [12]

A. Tisserand, CNRS–IRISA–CAIRN. Hardware Accelerators for ECC and HECC
Comparison ECC 256 vs HECC 128 (1/7)

<table>
<thead>
<tr>
<th></th>
<th>field $\mathbb{F}_p$</th>
<th>ADD</th>
<th>DBL</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECC</td>
<td>$\ell$ bits</td>
<td>![ADD Diagram]</td>
<td>![DBL Diagram]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cost: $12M + 2S$</td>
<td>Cost: $6M + 5S$</td>
</tr>
<tr>
<td>HECC</td>
<td>$\frac{\ell}{2}$ bits</td>
<td>![ADD Diagram]</td>
<td>![DBL Diagram]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cost: $47M + 4S$</td>
<td>Cost: $38M + 6S$</td>
</tr>
</tbody>
</table>

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

- $w = 32$ bits internal words
- 1 adder/subtractor, 1 inversion unit
- $n_M$ multipliers (Montgomery) with $n_B$ $w$-bit sub-blocks
- No DSP blocks
- ISE 14.6 Xilinx CAD tools, standard efforts (synthesis and P&R)
• Compared recoding techniques:
  ▶ BIN: standard binary from left to right
  ▶ NAF: non-adjacent form
  ▶ $\lambda$-NAF: window methods with $\lambda \in \{3, 4\}$

• Implementation results for a full ECC accelerator ($n_M = 1$, $n_B = 1$):

<table>
<thead>
<tr>
<th>Recoding</th>
<th>BIN</th>
<th>NAF</th>
<th>3-NAF</th>
<th>4-NAF</th>
</tr>
</thead>
<tbody>
<tr>
<td>area slices</td>
<td>565 (1321/1461)</td>
<td>570 (1340/1479)</td>
<td>571 (1344/1495)</td>
<td>503 (1348/1489)</td>
</tr>
<tr>
<td>freq. (MHz)</td>
<td>225</td>
<td>228</td>
<td>237</td>
<td>217</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>$n_M$</th>
<th>BRAM</th>
<th>$n_B = 1$</th>
<th>freq.</th>
<th>$n_B = 2$</th>
<th>freq.</th>
<th>$n_B = 4$</th>
<th>freq.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>area</td>
<td>MHz</td>
<td>area</td>
<td>MHz</td>
<td>area</td>
<td>MHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>slices (FF/LUT)</td>
<td></td>
<td>slices (FF/LUT)</td>
<td></td>
<td>slices (FF/LUT)</td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>( n_B )</th>
<th>( n_M )</th>
<th>( n_{n M} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>HECC</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>15.6</td>
<td>8.6</td>
</tr>
<tr>
<td>2</td>
<td>11.9</td>
<td>6.2</td>
</tr>
<tr>
<td>ECC</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>28.1</td>
<td>15.3</td>
</tr>
<tr>
<td>2</td>
<td>17.7</td>
<td>9.6</td>
</tr>
<tr>
<td>4</td>
<td>11.1</td>
<td>6.2</td>
</tr>
</tbody>
</table>

Standard deviation for 1000 \([k]P\):

<table>
<thead>
<tr>
<th>configuration</th>
<th>ECC (1,1)</th>
<th>ECC (3,4)</th>
<th>HECC (1,1)</th>
<th>HECC (6,2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>average time [ms]</td>
<td>28.1</td>
<td>5.4</td>
<td>15.6</td>
<td>2.8</td>
</tr>
<tr>
<td>standard deviation [ms]</td>
<td>0.289</td>
<td>0.056</td>
<td>0.324</td>
<td>0.045</td>
</tr>
</tbody>
</table>
On average HECC is 40% faster than ECC for a similar silicon cost
## Comparison ECC 256 vs HECC 128 (7/7)

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

area 1
delay 1
energy 1
security 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 and physical attacks
- low design cost
- low silicon cost
- low energy/power
- high performances
- high flexibility

\[
\begin{align*}
\text{area} & : 1 \quad \rightarrow \quad 1 + a \\
\text{delay} & : 1 \quad \rightarrow \quad 1 + t \\
\text{energy} & : 1 \quad \rightarrow \quad 1 + e \\
\text{security} & : 1
\end{align*}
\]

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

\[
\text{area} \quad 1 \quad \rightarrow \quad 1 + a
\]

\[
\text{delay} \quad 1 \quad \rightarrow \quad 1 + t
\]

\[
\text{energy} \quad 1 \quad \rightarrow \quad 1 + e
\]

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

\[
\text{security} \quad 1 \quad \rightarrow \quad \times 10 \quad \times 100
\]
References I

T. Chabrier, D. Pamula, and A. Tisserand.
Hardware implementation of DBNS recoding for ECC processor.

T. Chabrier and A. Tisserand.
On-the-fly multi-base recoding for ECC scalar multiplication without pre-computations.

Asynchronous charge sharing power consistent montgomery multiplier.

Robust sub-powered asynchronous logic.

G. Gallin, A. Tisserand, and N. Veyrat-Charvillon.
Comparaison expérimentale d’architectures de crypto-processeurs pour courbes elliptiques et hyper-elliptiques.

Parallel crypto-devices for GF(p) elliptic curve multiplication resistant against side channel attacks.
Y. Ma, Z. Liu, W. Pan, and J. Jing.
A high-speed elliptic curve cryptographic processor for generic curves over GF(p).

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.

D. Pamula and A. Tisserand.
GF(2^m) finite-field multipliers with reduced activity variations.

D. Pamula and A. Tisserand.
Fast and secure finite field multipliers.

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 Kerampoint, CS 80518, F-22305 Lannion cedex, France

Thank you