KATHOLIEKE UNIVERSITEIT
LEUVEN

Hardware/Software Co-design for Hyperelliptic Curve Cryptography (HECC) on the 8051 µP

Lejla Batina, David Hwang, Alireza Hodjat,
Bart Preneel and Ingrid Verbauwhede

UCLA

1425

Outline

- Introduction and Motivation
- Hyperelliptic Curve Cryptography (HECC)
- Case study: HECC on the 8051 μP
- Results
- Conclusions
- Improvements and Future work

Introduction

- (H)ECC vs. RSA:
  - (H)ECC offers shorter certificates, lower power consumption, more “security per bit”
- HECC: known since 1988 but only recently showed its potential
  - even shorter operands then for ECC
- Fast finite field arithmetic => efficient PKC implementations

Motivation

- Emerging new applications: wireless applications, sensor networks, RFIDs, ...
  - resource limited
  - low-cost, low-power
- PKC is sometimes necessary
  - sw-only solutions are too slow on embedded platforms
  - hardware acceleration is required for computationally intensive operations
  - HW/SW co-design is the only answer

Hyperelliptic Curves (1/2)

A hyperelliptic curve of genus g over a finite field K:

$$C : y^2 + h(x)y = f(x) \quad in\ K[x, y],$$

where:

f and h are polynomials, $deg(h) \leq g$, $deg(f)=2g+1$

f is monic

some more conditions should be satisfied

Hyperelliptic Curves (2/2)

Type II curves:

$$C : y^2 + xy = x^5 + f_3 x^3 + x^2 + f_0 \quad in\ K[x, y],$$

A divisor on C is a formal sum of points on C i.e.

$D = \sum m_P P$ with a degree $\deg(D) = \sum m_P$

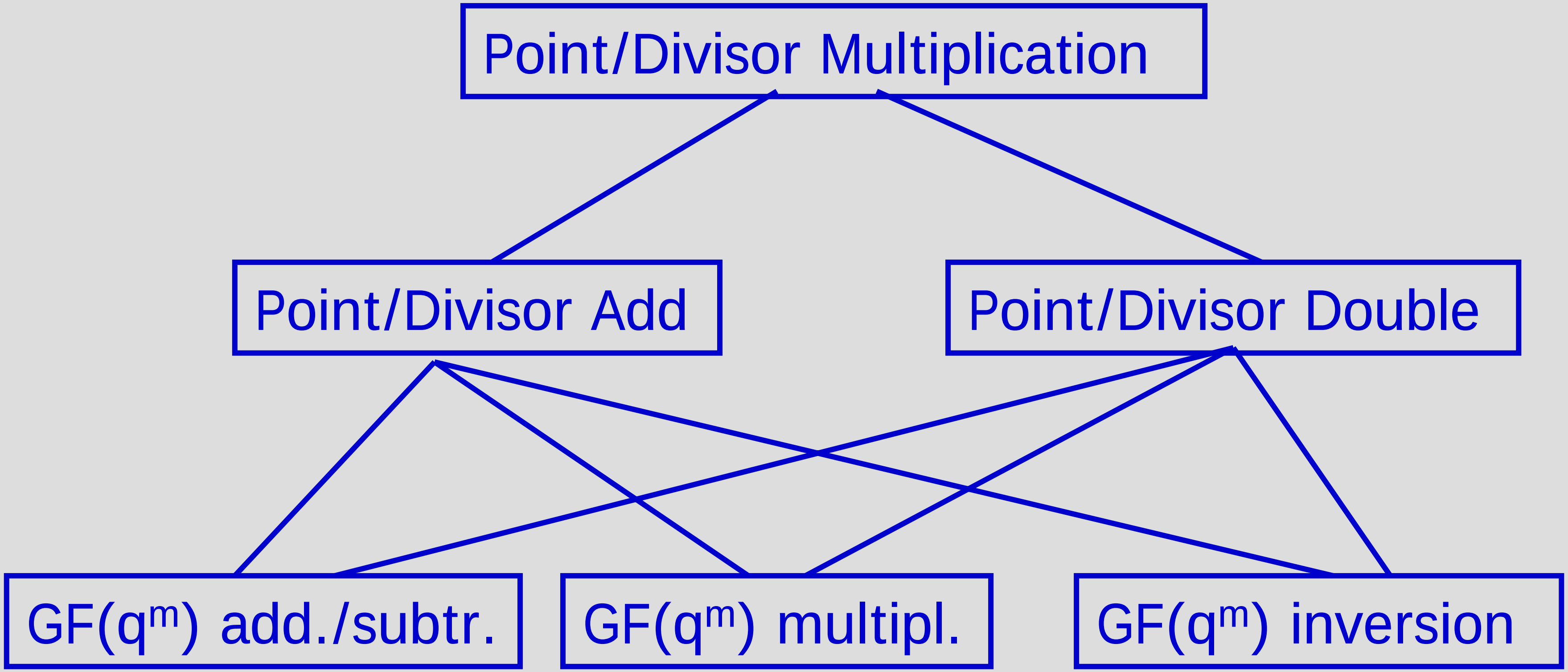
Div_0 – divisors of order 0

The Jacobian of C is defined by $Jac(C) = Div_0/P$, where P is the set of all principal divisors (a divisor D is called principal if $D = div(f)$ for some f)

$Jac(C)$ is an abelian group => DL system

Usual representation $D=[u,v]$, where u is monic of degree 2, $deg(v) < deg(u)$

(H)ECC hierarchy



Point/Divisor Multiplication

Point/Divisor Add

Point/Divisor Double

$GF(q^m)$ add./subtr.

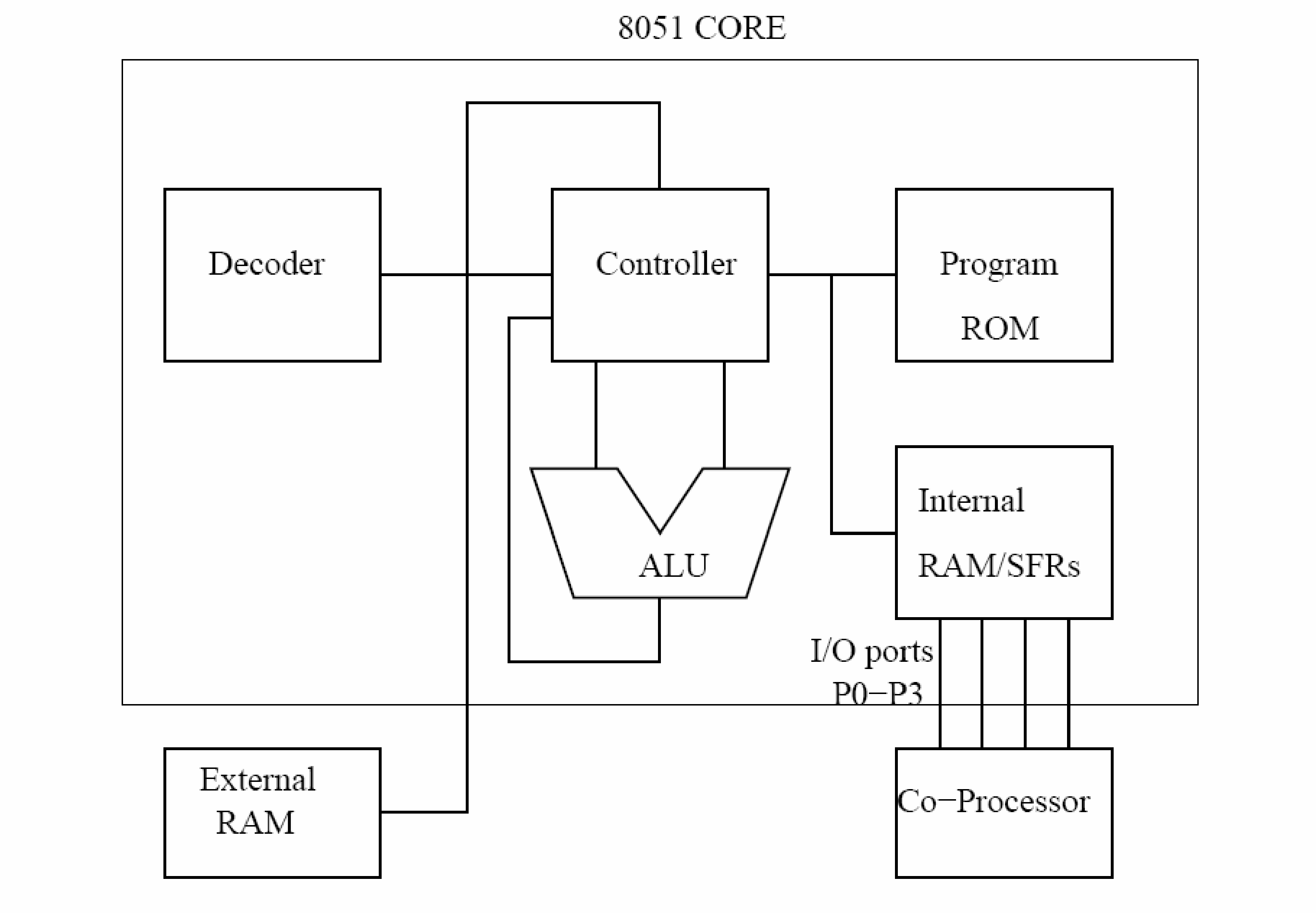
$GF(q^m)$ multipl.

$GF(q^m)$ inversion

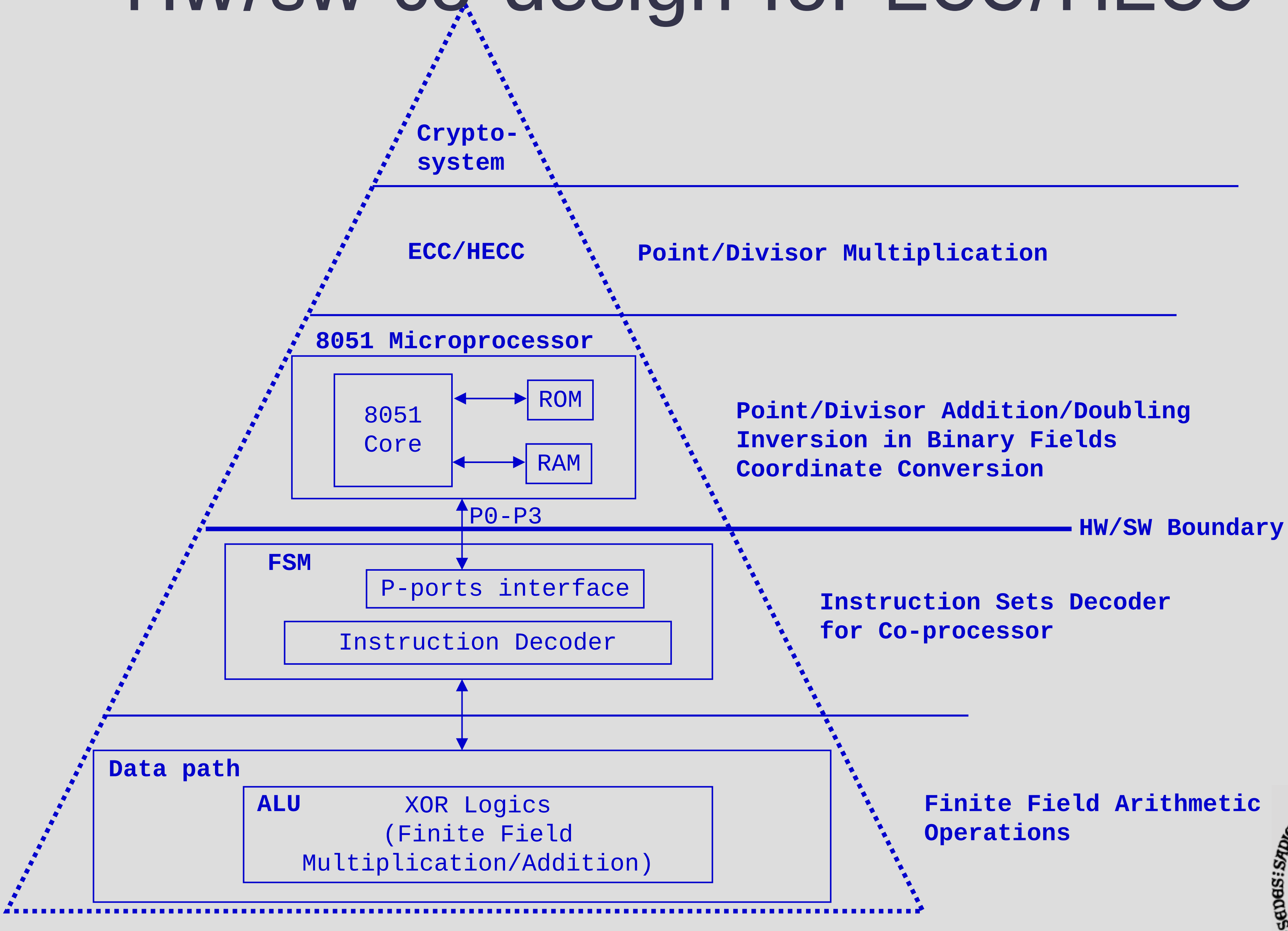
Case study: HECC on the μP

- 3 different implementations
  - pure software implementations
    - C implementations
    - mixed C/assembly
  - hardware/software models
    - software routines enhanced with binary field operations performed in hardware
    - 2 options for the data-path

The platform



Hw/sw co-design for ECC/HECC



Crypto-system

ECC/HECC

Point/Divisor Multiplication

8051 Microprocessor

8051 Core

ROM

RAM

Point/Divisor Addition/Doubling
Inversion in Binary Fields
Coordinate Conversion

P0-P3

HW/SW Boundary

FSM

P-ports interface

Instruction Decoder

Instruction Sets Decoder for Co-processor

Data path

ALU

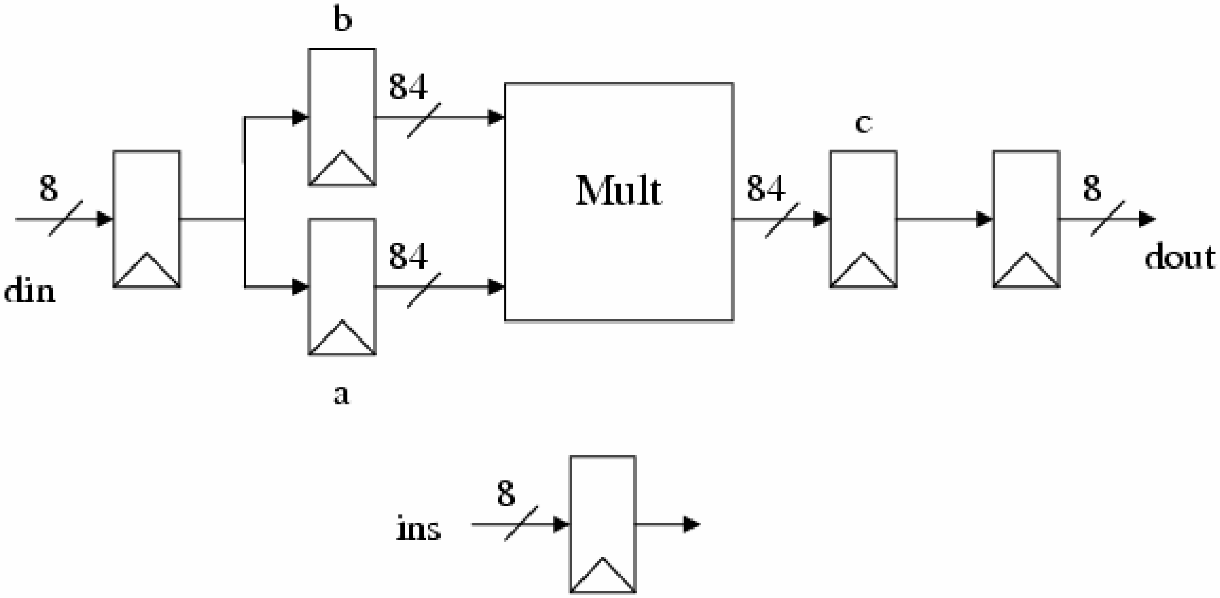
XOR Logics (Finite Field Multiplication/Addition)

Finite Field Arithmetic Operations

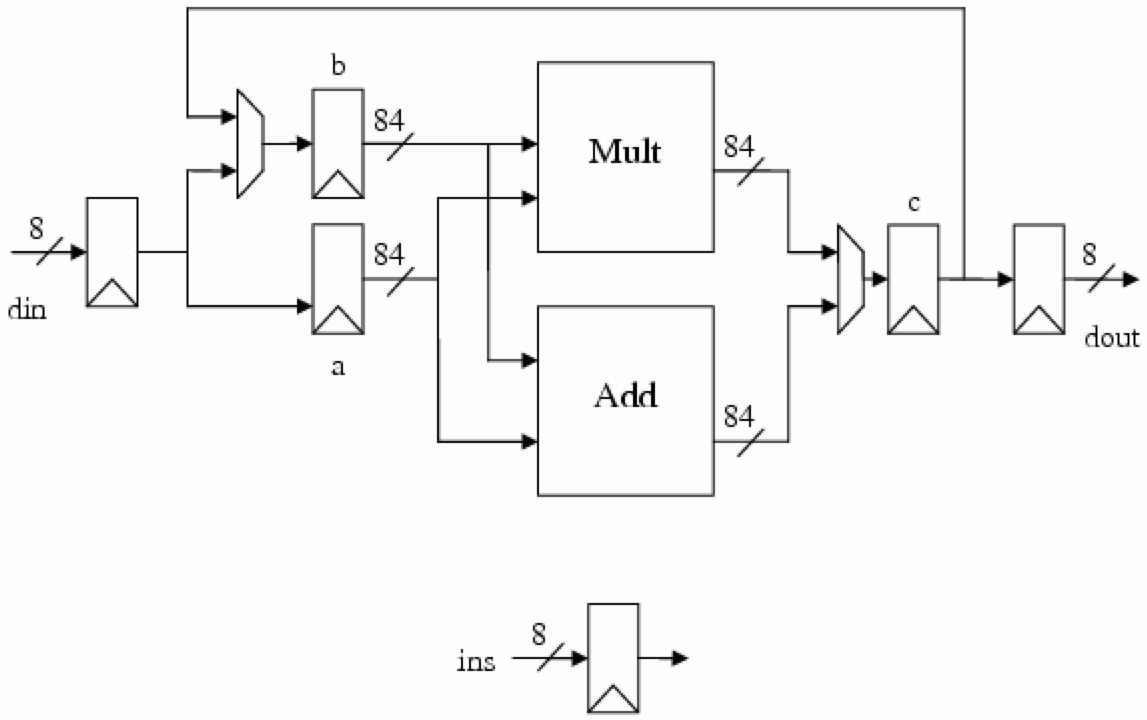
Implementations options

- Software Implementations
  - Pure C and C/assembly implementations
  - Compiled onto 8051 using Keil suite
  - Multiplication: comb-based
  - Inversion: Fermat
- 2 options for hardware/software implementations
  - A “multiplier-only” data-path
  - A “multiply-and-add” data-path

First hardware/software solution



Second solution: multiply-and-add data-path



Design environment - GEZEL

- Dalton 8051 ISS was used for simulation of software-only solution
- GEZEL – for HW/SW co-design
  - used as a HW description language
  - used to co-simulate the 8051 with a 12 MHz hw module

Results - field operations

Operation	Perf. [# cl. cyc.]	Perf. [ms]@12MHz	XRAM [Bytes]	ROM [Bytes]
Addition (SW)	38 K	3.2	54	608
Multipl. (SW)	650 K	54.1	122	2065
Addition (HW)	28.2 K	2.3	53	934
Multipl. (HW)	28.2 K	2.3	53	934
Inversion (HW)	788.5 K	65.7	75	1835
ab+c (HW)	30.5 K	2.5	44	942

Results - Divisor mult.

Implemen.	FPGA [#LUTs]	Perf. [s]@12MHz	XRAM [Bytes]	ROM [Bytes]
C Inv. in SW	3300	191.7	820	11754
C+ASM Inv.in SW	3300	64.9	820	12284
C+HW Inv. in SW	3600	52	820	11754
C+HW Inv.in HW	3600	4.1518	927	12789
C+HW Inv.in HW modif.d.p.	3781	2.488	936	11524

Comparison with ECC

Implemen.	FPGA [#LUTs]	Perf. [s]@12MHz	XRAM [Bytes]	ROM [Bytes]
ECC: SW	3300	144.5	980	7597
HECC: SW	3300	149.8	1186	13926
ECC: C+HW (1st)	3868	5.52	980	7597
HECC: C+HW (1st)	3600	4.1518	927	12789
ECC: C+HW (2nd)	4210	3.97	910	8739
HECC: C+HW (2nd)	3781	2.488	936	11524

Results: Summary

- Critical for performance are data transfers
- Modified data-path is more beneficial for HECC
- HECC: faster + less extra hardware
- Co-processor usage: less than 1% in both cases => more speed-up in performance is possible

Conclusions

- HW/SW co-design is a new alternative for low-power and low footprint devices
- HECC can be efficiently implemented on a small 8-bit processor
- Addition of a small HW module results in a substantial speed-up
- Parallelism for group operations can be efficiently exploited