

Hardware for Collision Search on Elliptic Curve over $GF(2^m)$

Philippe Bulens (S), Gueric Meurice de Dormale
and Jean-Jacques Quisquater

{bulens, gmeurice, quisquater}@dice.ucl.ac.be

UCL Crypto Group - DICE



Outline

- Pollard's Rho Method (quick)
- Collision Search Architecture
- EC Arithmetic Core
- Implementation Results
- Comparison Hard/Soft
- Conclusion
- Further Work



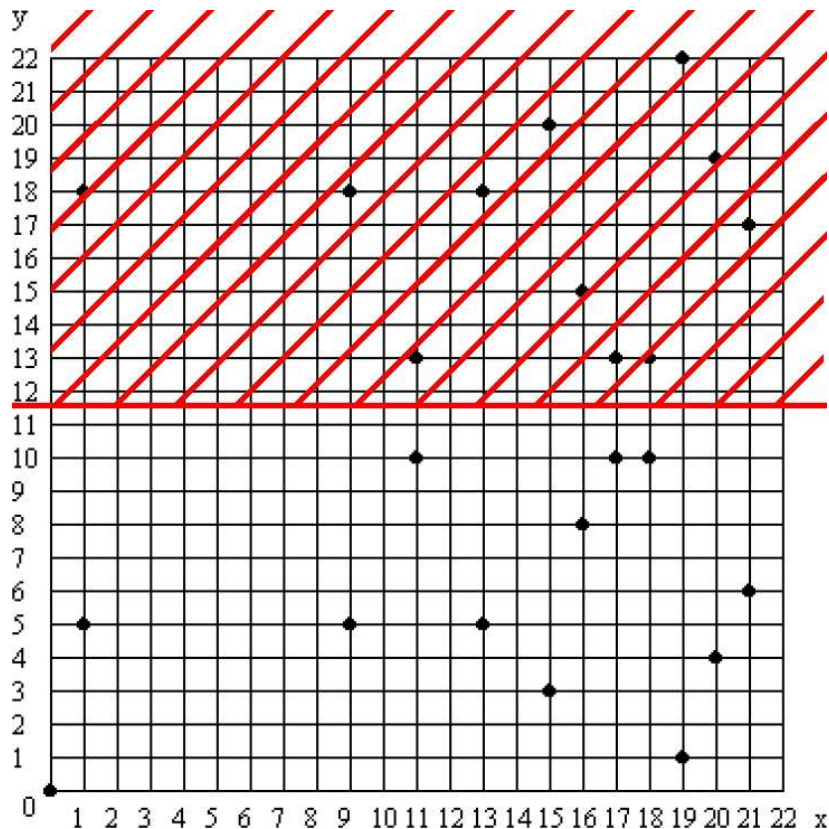
Pollard's Rho Method

- Parallelized Pollard's rho method (MPPR)
- Adding walks (Teske '00)
- Increased number of partitions (Teske '00)
- Distinguished points (Rivest '82)



Pollard's Rho Method (cont'd)

- Negation and Frobenius Map (Wiener & Zuccherato 98)



Anomalous binary curves:

$$y^2 + xy = x^3 + ax^2 + b$$

with $b = 1$, $a = \{0,1\}$

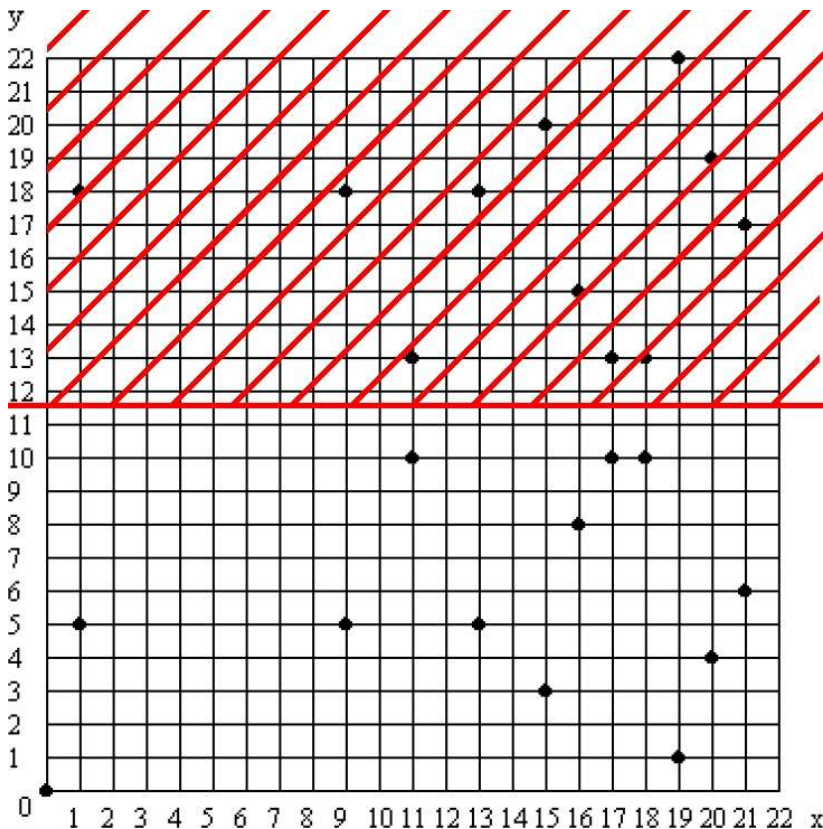
Frobenius:

$$\tau R = (x^2, y^2) = 2R$$



Pollard's Rho Method (cont'd)

- Negation and Frobenius Map (Wiener & Zuccherato 98)



Anomalous binary curves:

$$y^2 + xy = x^3 + ax^2 + b$$

with $b = 1$, $a = \{0,1\}$

Compute the $m-1$ Frobenius

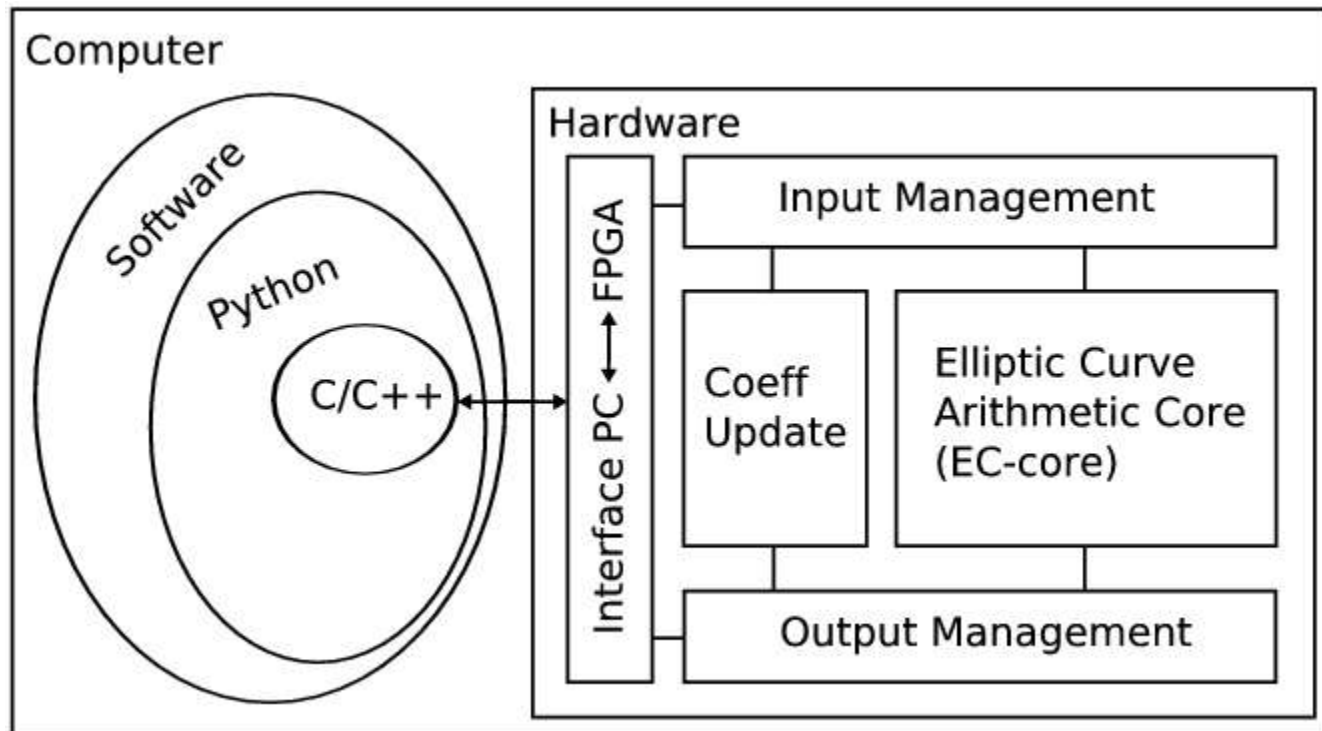
$$\tau^i R = (x^{2^i}, y^{2^i})$$

and keep the « smallest »

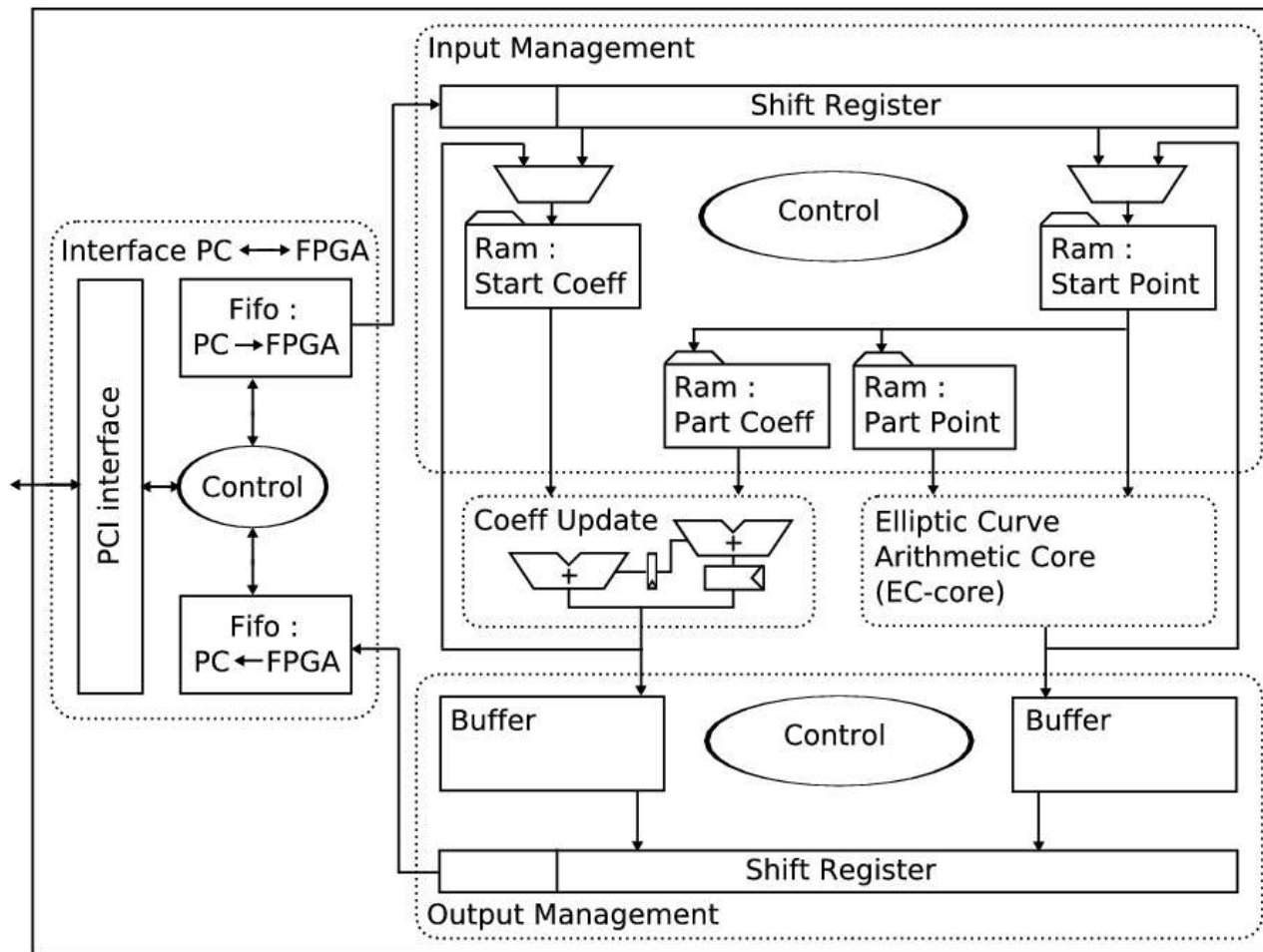
$$\Rightarrow \frac{1}{M} \cdot \sqrt{\frac{\pi \cdot n}{2 \cdot 2 \cdot m}} + \frac{1}{\theta}$$



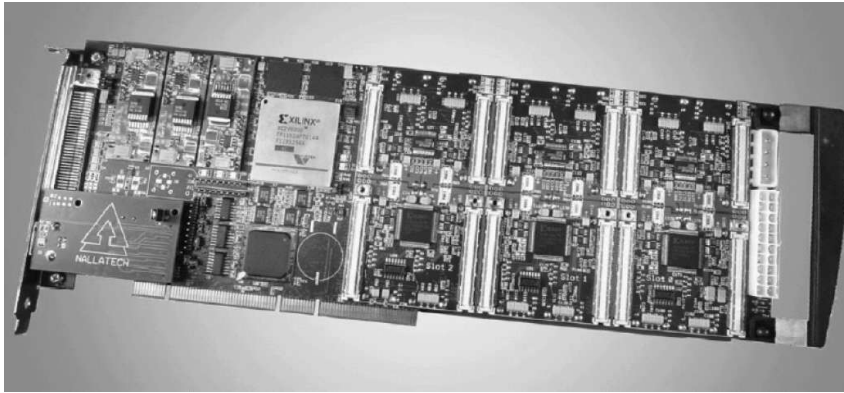
Collision Search Architecture



Collision Search Architecture



Setup



- Xilinx Virtex2 (XC2V6000-4ff1152)
- 200 MHz (RAM and Mult: 100MHz)

- PC: P4 1.8 GHz and 1Gb RAM
- Python, SWIG, NTL, C/C++, Pari/GP, etc.

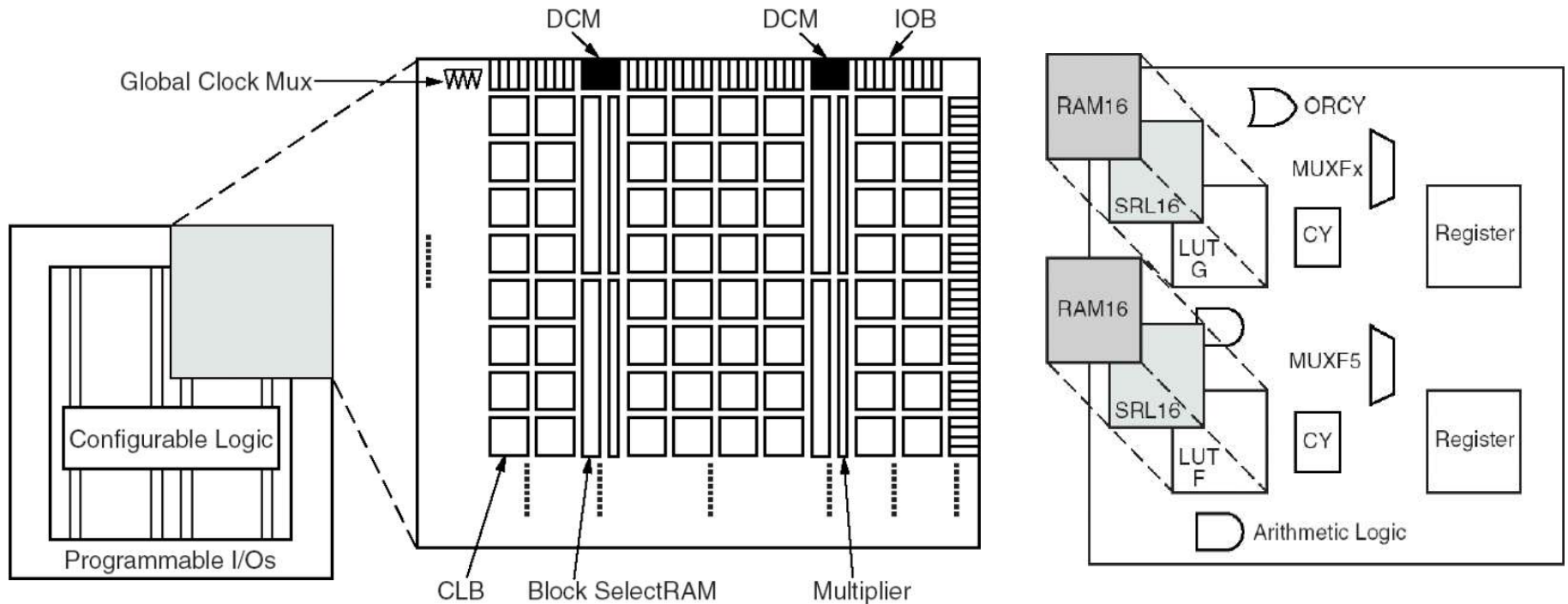
Arithmetic Core

- Usually: inversion-free formulae
- Need Inversion → Approach really different
- Assumptions
 - Fields without ONB
 - Low-weight binary irred. poly. (NIST,SECG)
 - Cheap modular reduction
- Goal: Highest throughput / area



Targeted Platform

- FPGA : Field Programmable Gate Array



Full Pipeline Approach

- A full pipeline need big FPGAs but,
 - Data-driven
 - Economy of scale
 - Specialized for each features
 - Every resources used every cycles
- 1 chain for each register stage
- 1 point addition/cycle / pipeline



Multiplication

- Sub-quadratic complexity \rightarrow Karatsuba
- $\text{Mult}(2^l \times 2^l) \rightarrow 3^k \text{ Mults}(2^{l-k} \times 2^{l-k}) + \text{Add}$
Our problems: not power of two !
- Input extension $l = \lceil \log_2(m) \rceil + \text{logic optim.}$
- $\text{Mult}(L \times L) \rightarrow 3^k \text{ Mults}(L/2^k \times L/2^k) + \text{Add}$
 - $m=79 \rightarrow 80 \times 80$ (by 10×10) + logic optim.
 - $m=163 \rightarrow 160 \times 160$ (by 10×10) + 2 160×3 + 3×3
- + Register balancing $\rightarrow \text{Area} < (m/2)^2$ slices



Division – Inversion

- Main part of the arithmetic core
- High throughput with realistic area
 - Not easy !
- Extended Euclidean algorithm (binary)
 - Transformation of the GCD()
- Little Fermat's theorem
 - Modular exponentiation



Euclidean Algorithm

- 4 variables (U-V, R-S) and $2m-1$ iterations
- Iterative circuit: $4 m/2$ slices
- Unrolled Divider

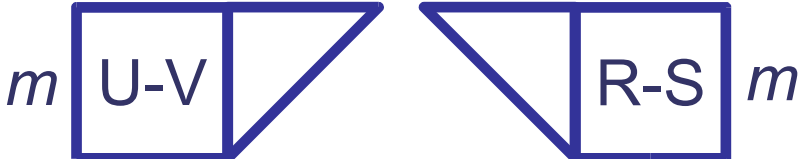


– Area = $3 m^2 /2 + 2 m^2$ slices



Euclidean Algorithm (cont'd)

- Unrolled almost Montgomery inverse

– Size: 

– Area = $3m^2$ slices $\rightarrow m^2 / 2$ slices saved

- Inverse: need a map to a domain

Cost is $\geq (m/2)^2$ slices

+ 1 mult = $(m/2)^2$ slices for division \rightarrow Useless

\rightarrow Euclidean division is expensive ! ($7m^2 / 2$)



Little Fermat's Theorem

- $\beta^{-1} \equiv \beta^{(2^m-1)-1} \equiv \beta^{2^m-2}$
- Min(# Mults) by chain technique (Itoh-Tsujii)
- # Mults = $\lfloor \log_2(m-1) \rfloor + W(m-1) - 1$
- Normal Basis (NB) ?
 - Square \rightarrow NB, Mult \rightarrow PB
 - NB alone ? \rightarrow Cost of Mult !
 - NB + PB ? \rightarrow Cost of change of basis !
- \rightarrow PB alone (repeated squares are OK)



Montgomery Trick

- $a^{-1}, b^{-1} ? \rightarrow (ab)^{-1} \times a = b^{-1} ; (ab)^{-1} \times b = a^{-1}$
- 1 Inv \rightarrow $1/n$ Inv + $3(n-1)$ Mults (1 PPA)
- n Inv \rightarrow 1 Inv + $3(n-1)$ Mults (n PPA)
- $1/n$ Inv: feasible but complicated
- If Inv \equiv 10 Mults (M), for p PPA:
 - 1 Inv $\rightarrow (12 + 5(p-1))M$
 - $\frac{1}{2}$ Inv $\rightarrow (10 + 8(p-1))M$
 - \rightarrow For $p \geq 2$, use 1 Inv



Hardware Results

GF(2 ⁷⁹) PPAR on V2	Freq. (MHz)	Area (Slices)	BRAM
Multiplication	170	2324	0
Inversion	154	15849	0
Point Addition	154	18335	0
Whole Management	124	4361	30
Whole design	100	22236	30

- Virtex4 (XC4LX200)
 - 500 MHz: no restriction for Mult and RAMS



Hardware Results

GF(2 ⁷⁹) PPAR on V2	Freq. (MHz)	Area (Slices)	BRAM
Multiplication	170	2324	0
Inversion	154	15849	0
Point Addition	154	18335	0
Whole Management	124	4361	30
Whole design	100	22236	30

GF(2 ¹⁶³) Synth on V4	Freq. (MHz)	Area (Slices)	BRAM
Multiplication	257	5940	0
Inversion	245	58885	0
Point Addition	242	71469	0
Whole Management	228	8793	60
Whole Design	≈ 200	≈ 80000	60



Software Results

- Shoup's NTL library (C++)
- Xeon 3.2 GHz
- For ECC2-79 / 109 / 163

m	79	109	163
16 · 1000 PA (s)	22.39	32.78	66.062
Throughput (PA/s)	714	488	242



Hardware vs Software

$$\text{ERT} = \frac{\text{ENO}}{\text{Throughput}} \quad \text{where ENO} = \frac{1}{M} \cdot \sqrt{\frac{\pi \cdot n}{2}} + \frac{1}{\theta}$$

$$\text{ERT}(h, 79) = \frac{\sqrt{\frac{\pi \cdot 2^{79}}{2}} + 2^{26}}{100 \cdot 10^6} = \frac{\frac{1}{66} \cdot \sqrt{\frac{\pi \cdot 2^{79}}{2}} + 2^{26}}{\frac{100 \cdot 10^6}{66}}$$

	Our Hardware	Our Software
GF(2 ⁷⁹)	3 hours	43 years
GF(2 ¹⁶³)	700 · 10 ⁶ years	540 · 10 ¹² years



Hardware vs Software

$$\text{ERT} = \frac{\text{ENO}}{\text{Throughput}} \quad \text{where ENO} = \frac{1}{M} \cdot \sqrt{\frac{\pi \cdot n}{2}} + \frac{1}{\theta}$$

	Our Hardware	Our Software
GF(2 ⁷⁹)	3 hours	43 years
GF(2 ¹⁰⁹)	5 years	2 · 10 ⁶ years
GF(2 ¹⁶³)	700 · 10 ⁶ years	540 · 10 ¹² years

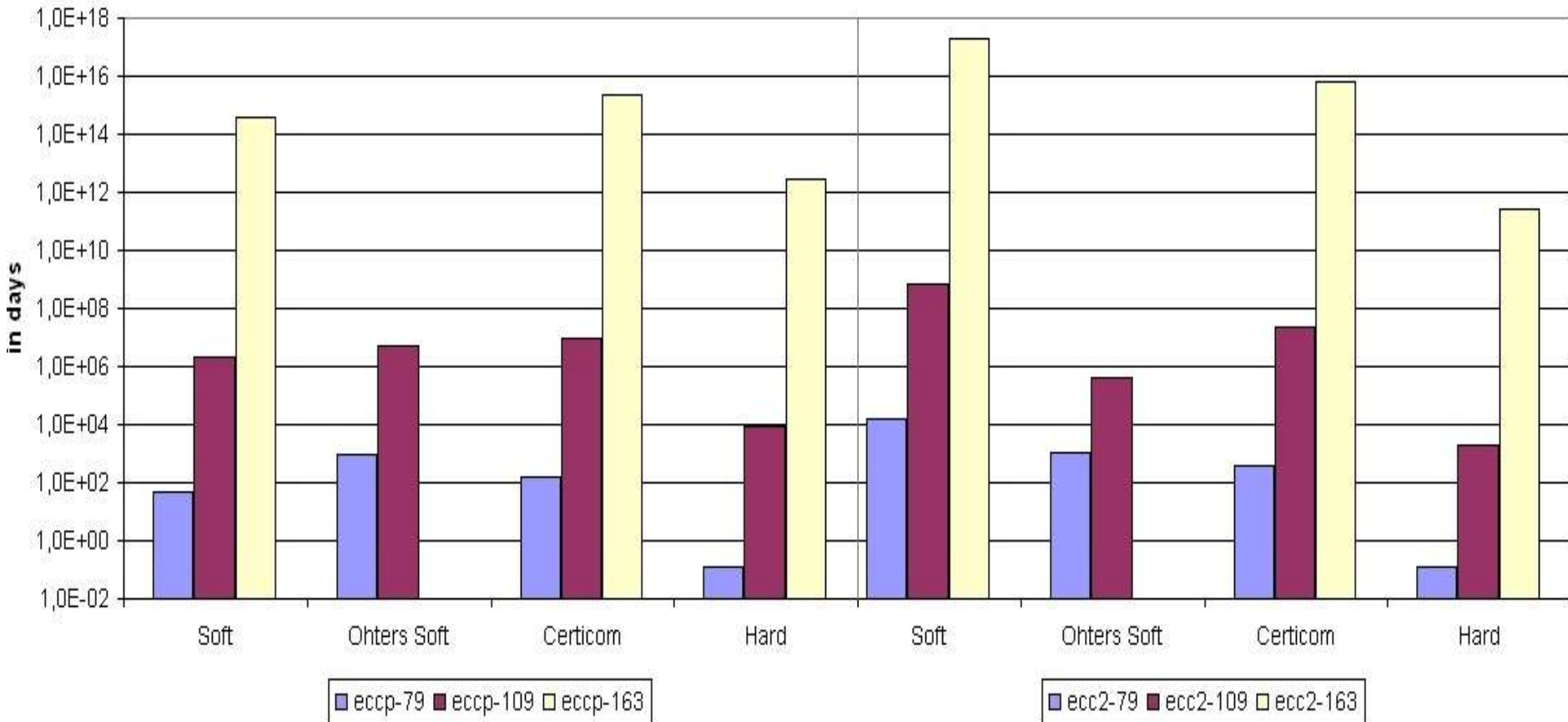
	Other Software	Certicom
GF(2 ⁷⁹)	3 years	1 year
GF(2 ¹⁰⁹)	1200 years	60 · 10 ³ years
GF(2 ¹⁶³)	later	17 · 10 ¹² years



Hardware vs Software (cont'd)

Expected Running Time in GF(p)

Expected Running Time in GF(2m)



Conclusion

- Collision search architecture
 - $GF(2^m)$
 - Single FPGA, full pipeline architecture
 - High throughput
- $ERT(h, 109) \approx 5$ years
 - 240 times faster than in software
- $ERT(h, 163) \approx 700 \cdot 10^6$ years
 - ... unlikely to be broken soon



Further Work

- Completely debug the ECC2-79 example
- Enable the negation map
- Include the Frobenius map (ECC2K)
- Gaussian normal basis
- Build prime field arithmetic core



Open Discussion

- Questions, comments, remarks, etc. are more than welcome.

