Collision Search for Elliptic Curve Discrete Logarithm over GF(2^m) with FPGA

Workshop on Cryptographic Hardware and Embedded Systems (CHES 2007)
September 2007

Guerric Meurice de Dormale*,
Philippe Bulens, Jean-Jacques Quisquater

Université catholique de Louvain
UCL/DICE Crypto Group
Belgium

Outline

- Motivations
- The attack
- Arithmetic choices & architecture
- Results & cost assessment
- Conclusion

Basics

- Cryptography
- Public-key schemes
- Elliptic Curve Cryptography (ECC)
- Underlying hard problem: ECDLP

  Given P and Q = $k \cdot$ P, find k

Why attacking systems?

- Feasibility

  Cost reachable for a given adversary?

  ➔ Security of a given set of parameters

- Forecast

  How long data will remain secure?

- Means
  - **Hardware**-based cost assessment (FPGA)
  - **Cost-effective** algorithms and architectures

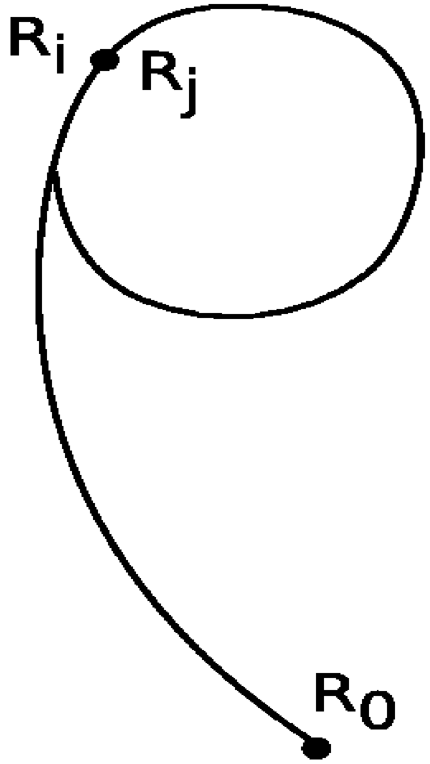
Solving ECDLP

- Solving general instances ➔ Pollard rho
  - – Find a collision by random walks
  - – Keep track of points in P,Q basis

$P = k \cdot Q$

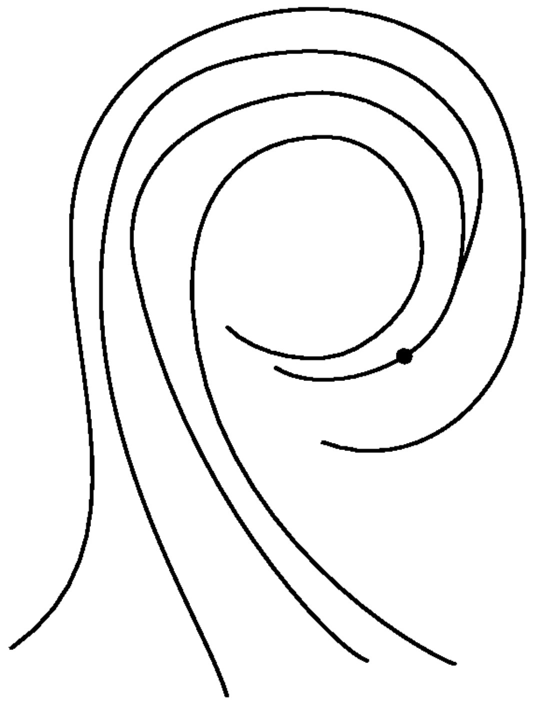
$c_i P + d_i Q = c_j P + d_j Q$

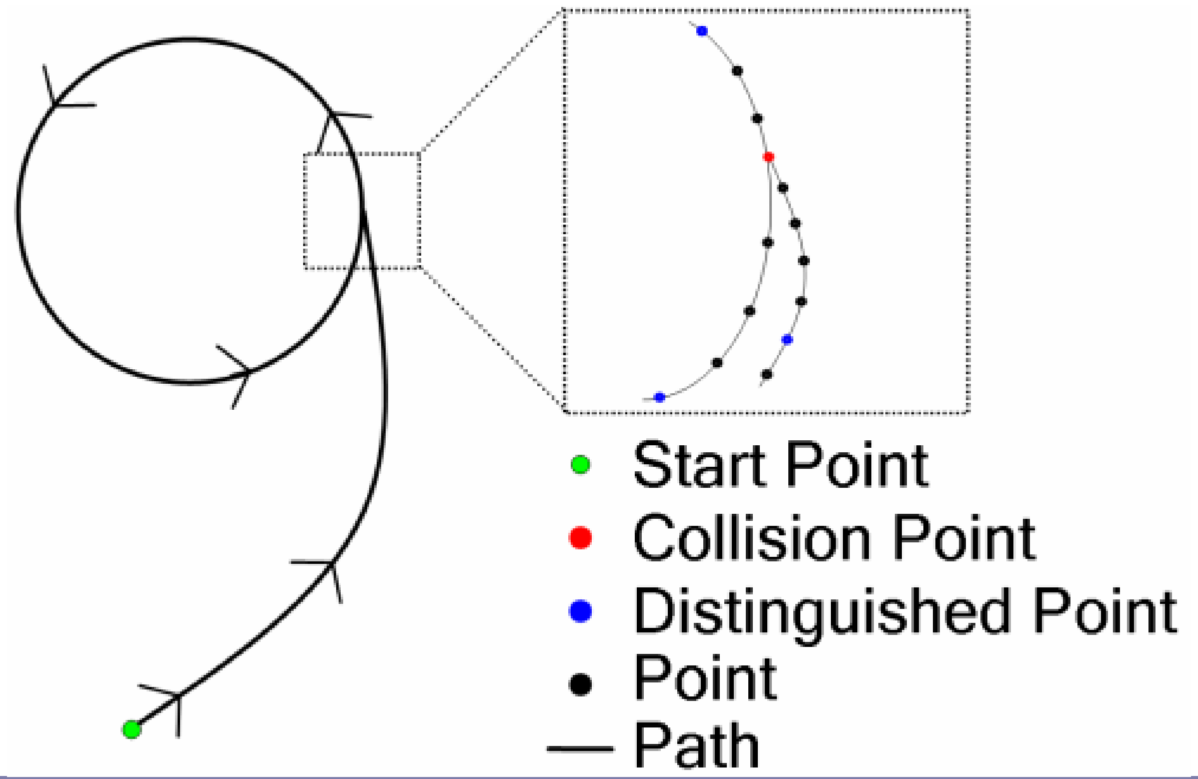
$\rightarrow k = (c_j - c_i) / (d_i - d_j) \bmod \#P$



Pollard ρ improvements

- Parallelized ρ + distinguished points
- More partitions & adding walks





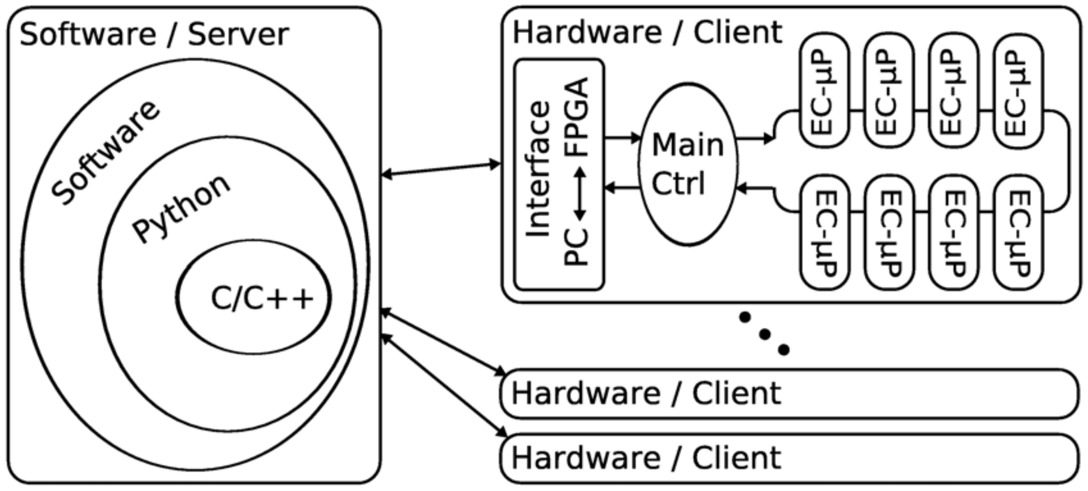
Point coordinates

- Point addition in high-speed domain
  - High-speed division: expensive!
  - → Projective coordinates: less expensive
- Parallelized ρ + DP: need invariant!
  - Check DP criteria
  - Apply pseudo-random mapping
  - $P(x,y) \rightarrow P(X,Y,Z)$ with $x = X/Z$ and $y = Y/Z$
  - → Cheapest coordinates: affine

Proposals

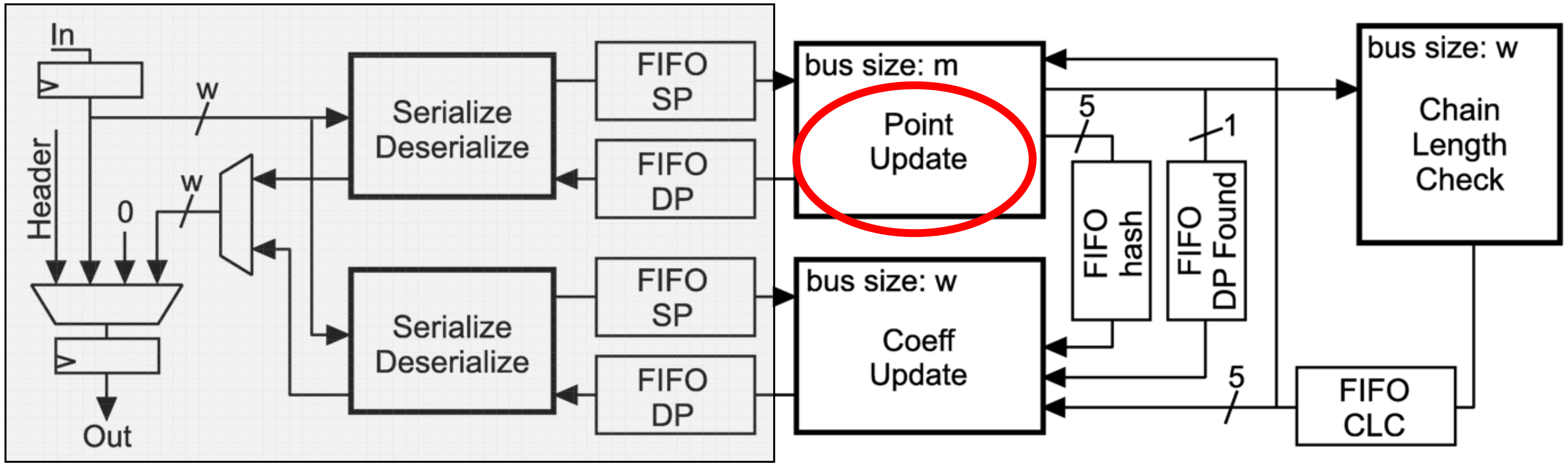
- Previous works
  - Software (Certicom's challenges)
  - Hardware for GF(*p*) curves
  - Rough ASIC extrapolation for (small) GF(2^m)
- Our work
  - *Real* FPGA results
  - Recommended polynomials (NIST, SECG)
  - Polynomial basis $p(z) = z^{163} + z^7 + z^6 + z^3 + 1$

Whole system



Through./$

EC-µP



Modular arithmetic

- Squarer `1000101000001000101`
  - Recommended p(*z*) → very cheap
- Multiplier
  - Digit-serial by parallel (moderate throughput)
  - Parallel using Karatsuba (high throughput)
- Inverter – divider
  - Euclidean divider
    - Nice for low throughput
    - Impractical for high throughput

Modular arithmetic

- Inverter – divider
  - Euclidean Montgomery inverter
    - More expensive for low & high throughput
  - Fermat's little inverter ($a^{-1} = a^{2^m-2} \bmod p(z)$)
    - Few multiplications with IT → nice for high throughput
- Mult/inverter trade-offs with Montgomery trick

a^{-1}, b^{-1} ? $\rightarrow$ $(a \times b)^{-1} \times a = b^{-1}$

$(a \times b)^{-1} \times b = a^{-1}$

4 strategies

- Tiny

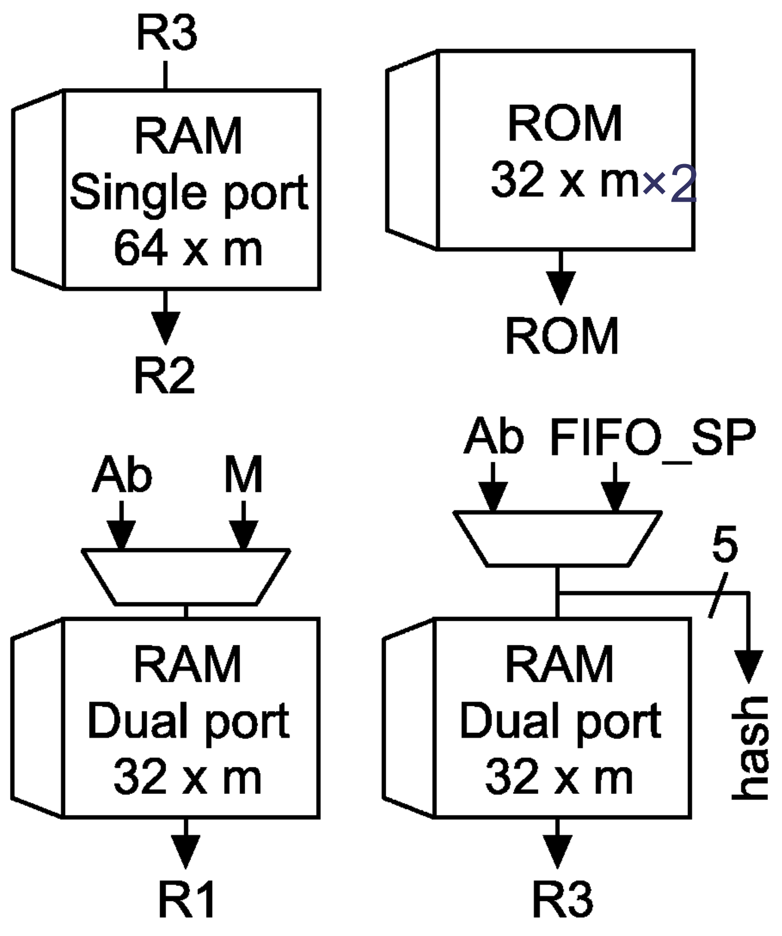
  1 ALU for all operations
- Small

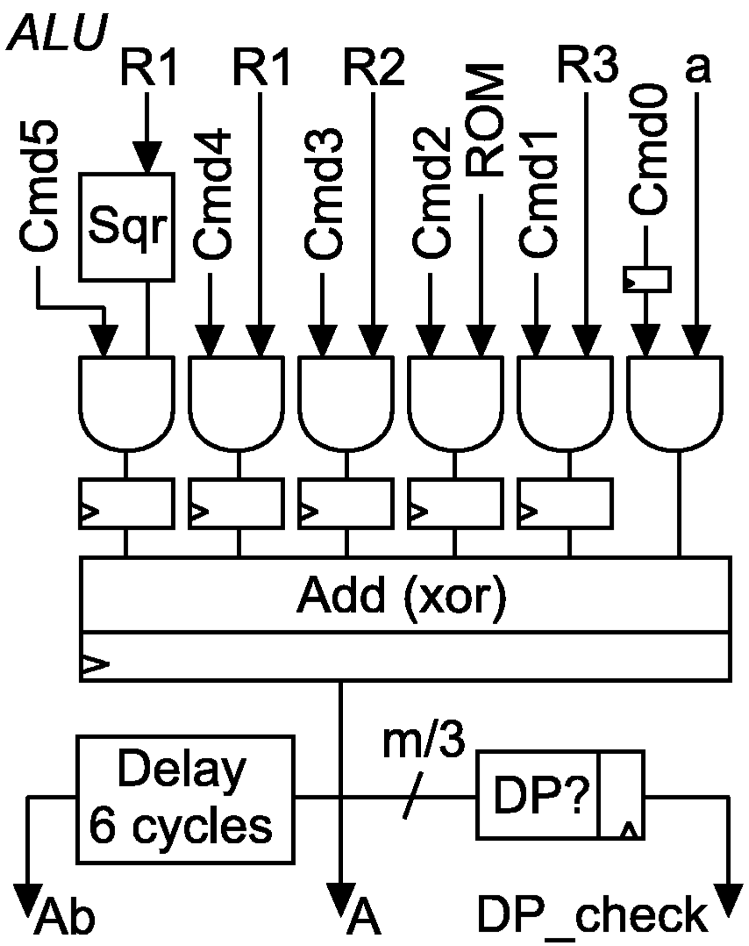
  1 serial multiplier, 1 serial divider
- Medium

  1 parallel multiplier, dedicated repeated squarers
- Large

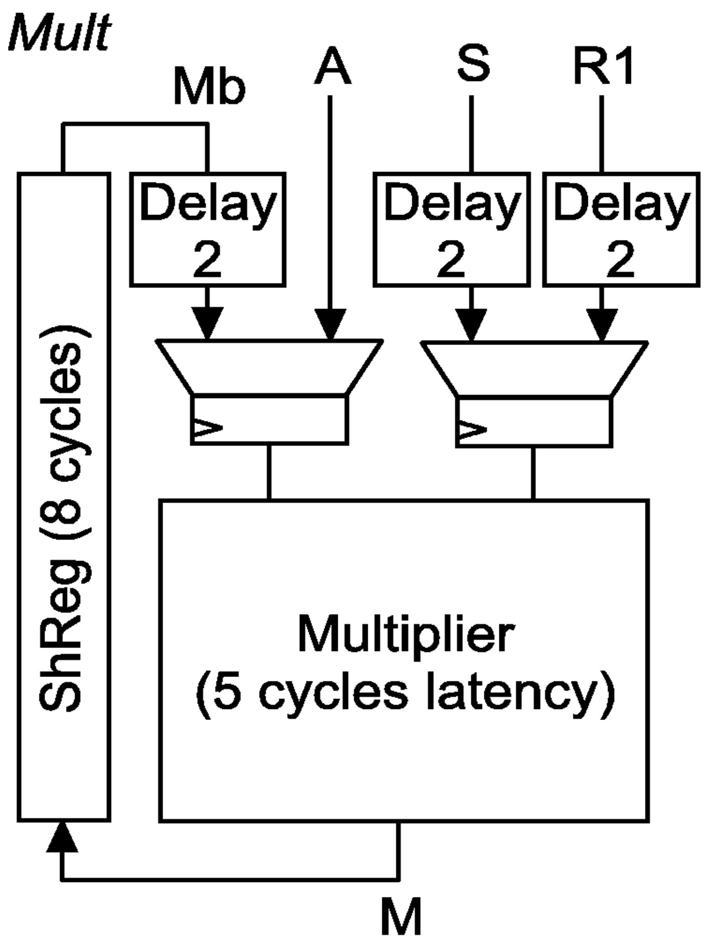
  Fully unrolled Fermat inverter and multipliers

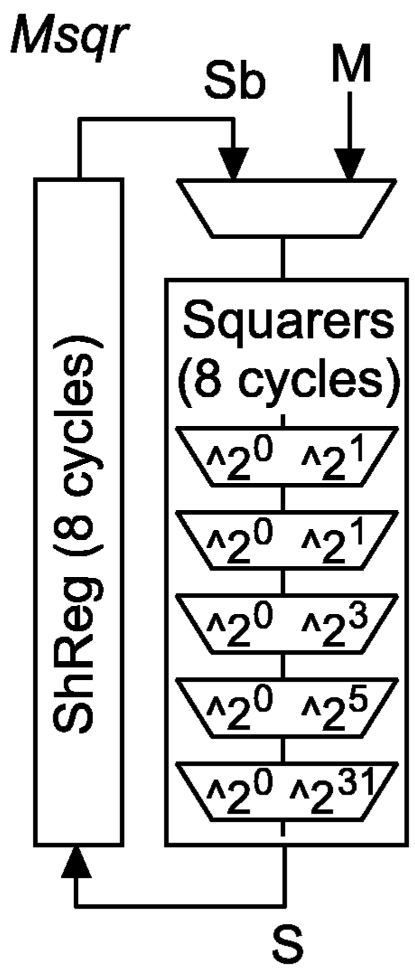
Medium processor





Medium processor





Computation:

…

M = Mb × R1

S = 5 Sqr(M)

M = Mb × S

S = 10 Sqr(M)

…

Sqr() i times

i=0,1,2,5,10,20,40,81

i'=0,1,2,5,10,40,41

Medium: results

Freq = 100 Mhz, elec price = 0.1 US$/kWh

m	113	131	163
FPGA	S3E1600-5	S3E1200-4	S3E1600-5
Area [kSlices]	13.9 (95%)	7.9 (90%)	10.9 (75%)
Area [bRAMs]	18 (50%)	21 (75%)	25 (70%)
Throughput [PA/s]	**2×10^7**	**10^7**	**$9 .10^6$**
Thr./cost [PA/s$]	$6 .10^5$	$4.8 .10^5$	$2.7 .10^5$
Consumption [W]	4.2	3.2	3.8
Elec. price [$/1 year]	3.7	2.8	3.3

Cost assessment

- Attack on *m*=163 in 1 year
  - Spartan3E-1600 COPACOBANA (10k$, 1.2 kW)
  - 125 .10^6 devices ➔ $1.4 10^{12}
  - 1/10th is for power!
- **Rough** 90 *nm* ASIC extrapolation *m*=163
  - Area: 20, speed: 3.5, consumption: 14
  - Die size Spartan3E-1600: 2.5 × 2.5 *mm*
  - 300 *mm* wafer cost: 2 × 30k$ ➔ $2.2 10^9
  - Half is for power!

Cost assessment

- Attack on m=113 (SECG) in 1 year
  - – 2 COPACOBANA → $22,000
- Comparison with GF(2^{109}) in software
  - – Computer price: $150, consumption: 250W
  - → Purchase price: 35, consumption: 500
- Comparison with GF(p) 160-bit (Guneysu *et al.* fpga'07)
  - → Throughput ratio: 50

Further work

- Launch a real attack on COPACOBANA
- Montgomery trick for medium architecture
- Use of *negation* and *Frobenius* map
- Attack GF(*p*) curves using FPGA Mult.

Conclusion

- Attacks against 163-bit GF(2^m) curves seems impractical
- Attacks against 113-bit GF(2^m) curves is feasible (\$22,000 / 1 year)
- Confirm that:
  - HW more efficient than SW (power!)
  - GF(2^m) faster than GF(p)

Questions ?

http://www.dice.ucl.ac.be/crypto