

Pollard Rho on the PlayStation 3

Joppe W. Bos¹

Marcelo E. Kaihara¹

Peter L. Montgomery²

¹EPFL IC LACAL, CH-1015 Lausanne, Switzerland
{joppe.bos, marcelo.kaihara}@epfl.ch

²Microsoft Research, One Microsoft Way, Redmond, WA 98052, USA
peter.montgomery@microsoft.com

- 1 Preliminaries
 - The ECDLP
 - Pollard Rho
 - The Cell Broadband Engine
- 2 Fast SIMD Arithmetic Algorithms
- 3 Results
- 4 Performance Comparison



- Elliptic curve cryptography (ECC)
 - Popular public-key approach
 - Smaller key sizes
 - Widely standardized
 - Standard for Efficient Cryptography (112-521)
 - Wireless Transport Layer Security Specification (112-224)
 - Digital Signature Standard (FIPS 186-3) (192-521)

- Elliptic curve cryptography (ECC)
 - Popular public-key approach
 - Smaller key sizes
 - Widely standardized
 - Standard for Efficient Cryptography (112-521)
 - Wireless Transport Layer Security Specification (112-224)
 - Digital Signature Standard (FIPS 186-3) (192-521)
- Security of elliptic curve schemes
 - Elliptic curve discrete logarithm problem
 - Pollard rho discrete logarithm algorithm ($\mathcal{O}(\sqrt{n})$)
 - Largest solved instance is for 109-bit prime field (2002)

Related work (Pollard rho targeted at prime fields)

Estimates for solving the ECDLP for various sizes using using FPGAs:
T. Güneysu, C. Paar, and J. Pelzl. Special-purpose hardware for solving the elliptic curve discrete logarithm problem. *ACM Transactions on Reconfigurable Technology and Systems*, 1(2):1-21, 2008.

Related work (Pollard rho targeted at prime fields)

Estimates for solving the ECDLP for various sizes using using FPGAs:
T. Güneysu, C. Paar, and J. Pelzl. Special-purpose hardware for solving the elliptic curve discrete logarithm problem. *ACM Transactions on Reconfigurable Technology and Systems*, 1(2):1-21, 2008.

What we did

Evaluate the security of the 112-bit standard.

Target: Low-priced and broadly available multi-core Cell architecture.

- Design SIMD arithmetic algorithms.
- Implement Pollard rho exploiting the features of the Cell architecture.
- Optimize arithmetic for the 112-bit prime
- Set a new record by solving this ECDLP.

The ECDLP

The setting:

- E is an elliptic curve over $\mathbb{F}_p$ with p prime.
- $P \in E(\mathbb{F}_p)$ a point of order n .
- $Q = k \cdot P \in \langle P \rangle$.

Problem: Given E, p, n, P and Q what is k ?

Pollard rho

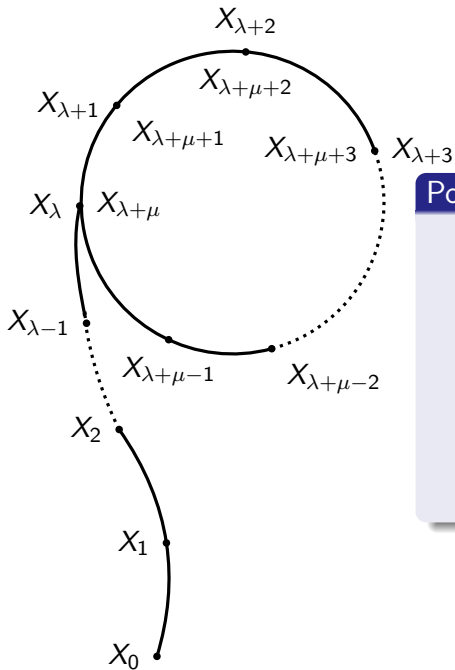
The most efficient algorithm in the literature (for generic curves) is Pollard rho. The underlying idea of this method is to search for two distinct pairs $(c_i, d_i), (c_j, d_j) \in \mathbb{Z}/n\mathbb{Z} \times \mathbb{Z}/n\mathbb{Z}$ such that

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

$$(c_i - c_j) \cdot P = (d_j - d_i) \cdot Q = (d_j - d_i)k \cdot P$$

$$k \equiv (c_i - c_j)(d_j - d_i)^{-1} \bmod n$$

J. M. Pollard. Monte Carlo methods for index computation (mod p). *Mathematics of Computation*, 32:918-924, 1978.



Pollard Rho

- “Walk” through the set $\langle P \rangle$.
- $X_i = c_i \cdot P + d_i \cdot Q$
- Iteration function $f : \langle P \rangle \rightarrow \langle P \rangle$
- This sequence eventually collides.
- Expected number of steps

(iterations): $\sqrt{\frac{\pi \cdot |\langle P \rangle|}{2}}$

Survey of Various Optimizations

- *r*-adding walks

E. Teske. On random walks for Pollard's rho method. *Mathematics of Computation*, 70(234):809-825, 2001.

Survey of Various Optimizations

- *r*-adding walks

E. Teske. On random walks for Pollard's rho method. *Mathematics of Computation*, 70(234):809-825, 2001.

- Parallel version, distinguish points and send them to a central server

P. C. van Oorschot and M. J. Wiener. Parallel collision search with cryptanalytic applications. *Journal of Cryptology*, 12(1):1-28, 1999.

Survey of Various Optimizations

- r -adding walks

E. Teske. On random walks for Pollard's rho method. *Mathematics of Computation*, 70(234):809-825, 2001.

- Parallel version, distinguish points and send them to a central server

P. C. van Oorschot and M. J. Wiener. Parallel collision search with cryptanalytic applications. *Journal of Cryptology*, 12(1):1-28, 1999.

- Simultaneous Inversion, trade inversions for multiplications

P. L. Montgomery. Speeding the Pollard and elliptic curve methods of factorization. *Mathematics of Computation*, 48:243-264, 1987.

Survey of Various Optimizations

- *r*-adding walks

E. Teske. On random walks for Pollard's rho method. *Mathematics of Computation*, 70(234):809-825, 2001.

- Parallel version, distinguish points and send them to a central server

P. C. van Oorschot and M. J. Wiener. Parallel collision search with cryptanalytic applications. *Journal of Cryptology*, 12(1):1-28, 1999.

- Simultaneous Inversion, trade inversions for multiplications

P. L. Montgomery. Speeding the Pollard and elliptic curve methods of factorization. *Mathematics of Computation*, 48:243-264, 1987.

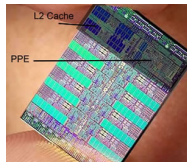
- Negation Map (*not used*)

M. J. Wiener and R. J. Zuccherato. Faster attacks on elliptic curve cryptosystems. In *Selected Areas in Cryptography*, volume 1556 of LNCS, pages 190-200, 1998.

The PlayStation 3

Cell architecture in the PlayStation 3 (@ 3.2 GHz):

- Broadly available (24.6 million)
- Relatively cheap (US\$ 300)



The Cell contains

- eight “**Synergistic Processing Elements**” (SPEs)
six available to the user in the PS3
- one “Power Processor Element” (PPE)
- the Element Interconnect Bus (EIB)
a specialized high-bandwidth circular data bus



Cell architecture, the SPEs

The SPEs contain

- a Synergistic Processing Unit (SPU)
 - Access to 128 registers of 128-bit
 - SIMD operations
 - Dual pipeline (odd and even)
 - Rich instruction set
 - In-order processor
- 256 KB of fast local memory (Local Store)
- Memory Flow Controller (MFC)
 - Direct Memory Access (DMA) controller
 - Handles synchronization operations to the other SPUs and the PPU
 - DMA transfers are independent of the SPU program execution

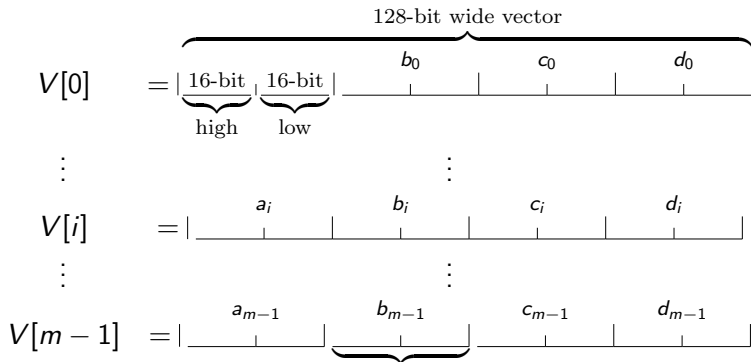
Programming Challenges

- Memory
 - The executable **and** all data should fit in the LS
 - Or perform manual DMA requests to the main memory (max. 214 MB)
- Branching
 - No “smart” dynamic branch prediction
 - Instead “prepare-to-branch” instructions to redirect instruction prefetch to branch targets
- Instruction set limitations
 - $16 \times 16 \rightarrow 32$ bit multipliers (4-SIMD)
- Dual pipeline
 - One odd and one even instruction can be dispatched per clock cycle.

Integer Representation

Four $(16 \cdot m)$ -bit integers A, B, C, D represented in m vectors.

$$X = \sum_{i=0}^{m-1} x_i \cdot 2^{16 \cdot i}$$



the most significant position of X_1 is located in
either the lower or higher 16-bit of the 32-bit word

When using simultaneous inversion and affine Weierstrass representation

6	modular multiplications
6	modular subtractions
$\frac{1}{M}$	inversion

are needed in order to perform one step when running M curves in parallel.

Performance bottleneck: modular multiplication

112-bit target

The prime 112-bit p in the target curve $E(\mathbb{F}_p)$ is

$$p = \frac{2^{128} - 3}{11 \cdot 6949}$$

Idea: Redundant representation modulo $\tilde{p} = 2^{128} - 3 = 11 \cdot 6949 \cdot p$

Note:
$$x \cdot 2^{128} \equiv x \cdot 3 \pmod{\tilde{p}}$$

Fast Modular Multiplication

Do the multiplication and reduction separately.

Fast schoolbook Multiplication

- If $0 \leq a, b, c, d < 2^{16}$, then $a \cdot b + c + d < 2^{32}$.
- Use the multiply-and-add instruction and an extra addition of carries.
- Branch-free implementation.

Fast Modular Multiplication

Do the multiplication and reduction separately.

Fast schoolbook Multiplication

- If $0 \leq a, b, c, d < 2^{16}$, then $a \cdot b + c + d < 2^{32}$.
- Use the multiply-and-add instruction and an extra addition of carries.
- Branch-free implementation.

Fast reduction

$$\begin{array}{ccc} R : \mathbb{Z}/2^{256}\mathbb{Z} & \rightarrow & \mathbb{Z}/2^{256}\mathbb{Z} \\ x & \mapsto & (x \bmod 2^{128}) + 3 \cdot \lfloor \frac{x}{2^{128}} \rfloor \end{array}$$

$$x = x_H \cdot 2^{128} + x_L \equiv x_L + 3 \cdot x_H = R(x) \bmod \tilde{p}$$

Proposition

For independent random 128-bit non-negative integers x and y there is overwhelming probability that $0 \leq R(R(x \cdot y)) < \tilde{p}$.

Counter-examples easy to construct: $0 \leq R(R(x)) < 2^{128} + 9$

During the whole run not a single faulty reduction.

Modular Inversion

- Almost Montgomery inverse: $x^{-1} \cdot 2^k \bmod p$ for some known k
- A normalization phase where the factor $2^k \bmod p$ is removed
- “Almost” branch free implementation
- SIMD implementation, work on four variables simultaneously

Modular Inversion

- Almost Montgomery inverse: $x^{-1} \cdot 2^k \bmod p$ for some known k
- A normalization phase where the factor $2^k \bmod p$ is removed
- “Almost” branch free implementation
- SIMD implementation, work on four variables simultaneously

Distinguish Point Property and r -Adding Walks

Need to uniquely determine the partition number and DTP property.

$P = (x, y)$, $0 \leq x < \tilde{p}$. Compute $z = x \pmod{p}$

One 16-bit step of Montgomery reduction

$$z = x \cdot 2^{-16} \pmod{p}$$

LACAL setup

- Physically in the cluster room:
190 PS3s
- 6×4 PS3s in the PlayLaB
(attached to the cluster)
- 5 PS3 in our offices for
programming purposes
- $\Rightarrow$ 219 PS3s in total.



Performance Results

Operation	#cycles per operation	Quantity per iteration	#cycles per iteration
Modular multiplication	53	6	318
Modular subtraction	5	6	30
Montgomery reduction	24	1	24
Modular inversion	4941	$\frac{1}{400}$	12
Miscellaneous	69	1	69
Total	453		

Performance Results

Operation	#cycles per operation	Quantity per iteration	#cycles per iteration
Modular multiplication	53	6	318
Modular subtraction	5	6	30
Montgomery reduction	24	1	24
Modular inversion	4941	$\frac{1}{400}$	12
Miscellaneous	69	1	69
Total	453		

Hence, our 214-PS3 cluster:

- computes $9.1 \cdot 10^9 \approx 2^{33}$ iterations per second
- works on $> 0.5M$ curves in parallel

XC3S1000 FPGAs [1]

- FPGA-results of EC over 96- and 128-bit generic prime fields for COPACOBANA [2]
- Can host up to 120 FPGAs (US\$ 10,000)

Our implementation

- Targeted at 112-bit prime curve.
- Use 128-bit multiplication + fast reduction modulo $\tilde{p}$
- For US\$ 10,000 buy 33 PS3s.

[1] T. Güneysu, C. Paar, and J. Pelzl. Special-purpose hardware for solving the elliptic curve discrete logarithm problem. *ACM Transactions on Reconfigurable Technology and Systems*, 1(2):1-21, 2008.

[2] S. Kumar, C. Paar, J. Pelzl, G. Pfeiffer, and M. Schimmler. Breaking ciphers with COPACOBANA a cost-optimized parallel code breaker. In CHES 2006, volume 4249 of LNCS, pages 101-118, 2006.

Comparison

	96 bits	128 bits
COPACOBANA	$4.0 \cdot 10^7$	$2.1 \cdot 10^7$
+ Moore's law	$7.9 \cdot 10^7$	$4.2 \cdot 10^7$
+ Negation map	$1.1 \cdot 10^8$	$5.9 \cdot 10^7$
PS3	$4.2 \cdot 10^7$	
33 PS3	$1.4 \cdot 10^9$	

Table: Iterations per second

33 PS3 / COPACOBANA (96 bits): **12.4 times faster**

33 PS3 / COPACOBANA (128 bits): **23.8 times faster**

Note: 33 dual-threaded PPE were not used!

The 112-bit Solution

The point P of order n is given in the standard.

The x-coordinate of Q was chosen as $\lfloor (\pi - 3)10^{34} \rfloor$.

- Expected #iterations $\sqrt{\frac{\pi \cdot n}{2}} \approx 8.4 \cdot 10^{16}$
- January 13, 2009 – July 8, 2009 (not running continuously)
- When run continuously using the latest version of our code, the same calculation would have taken 3.5 months.

$P =$	(188281465057972534892223778713752,	3419875491033170827167861896082688)
$Q =$	(1415926535897932384626433832795028,	3846759606494706724286139623885544)
$n =$	4451685225093714776491891542548933	

$$Q = 312521636014772477161767351856699 \cdot P$$

Conclusions

- We presented modular arithmetic algorithms using SIMD instruction.
- We showed the potential of the Cell architecture for cryptanalysis.
- Pollard rho SIMD-implementation
- Set a new record for solving the ECDLP.

Conclusions

- We presented modular arithmetic algorithms using SIMD instruction.
- We showed the potential of the Cell architecture for cryptanalysis.
- Pollard rho SIMD-implementation
- Set a new record for solving the ECDLP.

It is unwise to use the (standardized) elliptic curves over 112-bit fields in practice!



```
[ From hang full down disabled channel ]
john@rooted: ~$ cluster-nodes
[ From hang full down disabled channel ]
=====
16 10.123.4567.1@nab.icrpf?
15 10.123.4567.1@nab.icrpf?
=====
14 10.123.4567.1@nab.icrpf?
13 10.123.4567.1@nab.icrpf?
12 10.123.4567.1@nab.icrpf?
11 10.123.4567.1@nab.icrpf?
10 10.123.4567.1@nab.icrpf?
9 10.123.4567.1@nab.icrpf?
=====
7 10.123.4567.1@nab.icrpf?
6 10.123.4567.1@nab.icrpf?
5 10.123.4567.1@nab.icrpf?
4 10.123.4567.1@nab.icrpf?
3 10.123.4567.1@nab.icrpf?
2 10.123.4567.1@nab.icrpf?
1 10.123.4567.1@nab.icrpf?
=====
Master: 0 Slave: 0
john@rooted: ~$
```