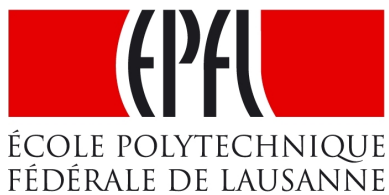
ECM at Work

Joppe W. Bos and Thorsten Kleinjung

Laboratory for Cryptologic Algorithms
EPFL, Station 14, CH-1015 Lausanne, Switzerland



Motivation

The elliptic curve method for integer factorization is used

- in the cofactorization phase of NFS ($\approx 100 - 200$-bits)
- to factor large numbers (Mersenne, Cunningham etc)

Motivation

The elliptic curve method for integer factorization is used

- in the cofactorization phase of NFS ($\approx 100 - 200$-bits)
- to factor large numbers (Mersenne, Cunningham etc)

Edwards curves vs Montgomery curves

 faster EC-arithmetic  more memory is required

Motivation

The elliptic curve method for integer factorization is used

- in the cofactorization phase of NFS ($\approx 100 - 200$-bits)
- to factor large numbers (Mersenne, Cunningham etc)

Edwards curves vs Montgomery curves

 faster EC-arithmetic  more memory is required

Difficult to run Edwards-ECM fast on memory-constrained devices

This presentation: *slightly* **faster**, *memory* **efficient** Edwards ECM

Edwards Curves (based on work by Euler & Gauss)

- Edwards curves
- Twisted Edwards curves
- Inverted Edwards coordinates
- Extended twisted Edwards coordinates

A twisted Edwards curve is defined ($ad(a-d) \neq 0$)

$$ax^2 + y^2 = 1 + dx^2y^2 \quad \text{and} \quad (ax^2 + y^2)z^2 = z^4 + dx^2y^2$$

2007: H. M. Edwards. A normal form for elliptic curves. Bulletin of the American Mathematical Society
2007: D. J. Bernstein and T. Lange. Faster addition and doubling on elliptic curves. Asiacrypt
2008: H. Hisil, K. K.-H. Wong, G. Carter, and E. Dawson. Twisted Edwards curves revisited. Asiacrypt

Edwards Curves (based on work by Euler & Gauss)

- Edwards curves
- Twisted Edwards curves
- Inverted Edwards coordinates
- Extended twisted Edwards coordinates

A twisted Edwards curve is defined ($ad(a-d) \neq 0$)

$$ax^2 + y^2 = 1 + dx^2y^2 \quad \text{and} \quad (ax^2 + y^2)z^2 = z^4 + dx^2y^2$$

Elliptic Curve Point Addition $\begin{cases} a = -1\text{: } 8\text{M} \\ a = -1, z_1 = 1\text{: } 7\text{M} \end{cases}$

Elliptic Curve Point Duplication: $a = -1$: $3\text{M} + 4\text{S}$

2007: H. M. Edwards. A normal form for elliptic curves. Bulletin of the American Mathematical Society
2007: D. J. Bernstein and T. Lange. Faster addition and doubling on elliptic curves. Asiacrypt
2008: H. Hisil, K. K.-H. Wong, G. Carter, and E. Dawson. Twisted Edwards curves revisited. Asiacrypt

Elliptic Curve Method (ECM)

Try and factor $n = p \cdot q$ with $1 < p < q < n$. Repeat:

- Pick a random point P and construct an elliptic E over $\mathbf{Z}/n\mathbf{Z}$ containing P
- Compute $Q = kP \in E(\mathbf{Z}/n\mathbf{Z})$ for some $k \in \mathbf{Z}$
- If $\#E(\mathbf{F}_p) \mid k$ (and $\#E(\mathbf{Z}/q\mathbf{Z}) \nmid k$) then Q and the neutral element become the same modulo p
- $p = \gcd(n, Q_z)$

In practice given a bound $B_1 \in \mathbf{Z}$: $k = \mathrm{lcm}(1, 2, \ldots, B_1)$

H. W. Lenstra Jr. Factoring integers with elliptic curves. Annals of Mathematics, 1987.

Elliptic Curve Method (ECM)

Try and factor $n = p \cdot q$ with $1 < p < q < n$. Repeat:

- Pick a random point P and construct an elliptic E over $\mathbf{Z}/n\mathbf{Z}$ containing P
- Compute $Q = kP \in E(\mathbf{Z}/n\mathbf{Z})$ for some $k \in \mathbf{Z}$
- If $\#E(\mathbf{F}_p) \mid k$ (and $\#E(\mathbf{Z}/q\mathbf{Z}) \nmid k$) then Q and the neutral element become the same modulo p
- $p = \gcd(n, Q_z)$

In practice given a bound $B_1 \in \mathbf{Z}$: $k = \operatorname{lcm}(1, 2, \ldots, B_1)$

$$O(\exp((\sqrt{2} + o(1))(\sqrt{\log p \log\log p}))M(\log n))$$

where $M(\log n)$ represents the complexity of multiplication modulo n and the $o(1)$ is for $p \to \infty$.

H. W. Lenstra Jr. Factoring integers with elliptic curves. Annals of Mathematics, 1987.

EC-multiplication

Notation: **A** (EC-additions), **D** (EC-duplications), **R** (residues in memory)
M (modular multiplications), S (modular squaring)

	Montgomery	Edwards
EC-multiplication method	PRAC	e.g. signed sliding w-bit windows
#**R**	14	$4(2^{w-1}) + 4 + 2$

D. J. Bernstein, P. Birkner, T. Lange, and C. Peters. ECM using Edwards curves. Cryptology ePrint Archive, Report 2008/016
D. J. Bernstein, P. Birkner, and T. Lange. Starfish on strike. Latincrypt, 2010

EC-multiplication

Notation: **A** (EC-additions), **D** (EC-duplications), **R** (residues in memory)
M (modular multiplications), S (modular squaring)

	Montgomery	Edwards
EC-multiplication method	PRAC	e.g. signed sliding w-bit windows
#**R**	14	$4(2^{w-1}) + 4 + 2$
Performance	$\#(\mathrm{S} + \mathrm{M})/\mathrm{bit} \approx$ 8-9	$B_1 \to \infty \begin{cases} \#\mathbf{A}/\mathrm{bit} \to 0, \\ \#\mathbf{R} \to \infty \end{cases}$ $\to$ (3M + 4S) / bit

D. J. Bernstein, P. Birkner, T. Lange, and C. Peters. ECM using Edwards curves. Cryptology ePrint Archive, Report 2008/016
D. J. Bernstein, P. Birkner, and T. Lange. Starfish on strike. Latincrypt, 2010

B1	GMP-ECM			
	#S	#M	#S+#M	
256	1 066	2 025	3 091	
512	2 200	4 210	6 400	
1 024	4 422	8 494	12 916	
12 288	53 356	103 662	157 018	
49 152	214 130	417 372	631 502	
262 144	1 147 928	2 242 384	3 390 312	
1 048 576	4 607 170	9 010 980	13 618 150	
	EECM-MPFQ ($a = -1$)			
256	1 436	1 608	3 044	
512	2 952	3 138	6 090	
1 024	5 892	6 116	12 008	
12 288	70 780	67 693	138 473	
49 152	283 272	260 372	543 644	
262 144	1 512 100	1 351 268	2 863 368	
1 048 576	6 050 208	5 306 139	11 356 347	

B1	GMP-ECM			
	#S	#M	#S+#M	#**R**
256	1 066	2 025	3 091	14
512	2 200	4 210	6 400	14
1 024	4 422	8 494	12 916	14
12 288	53 356	103 662	157 018	14
49 152	214 130	417 372	631 502	14
262 144	1 147 928	2 242 384	3 390 312	14
1 048 576	4 607 170	9 010 980	13 618 150	14
	EECM-MPFQ ($a = -1$)			
256	1 436	1 608	3 044	38
512	2 952	3 138	6 090	62
1 024	5 892	6 116	12 008	134
12 288	70 780	67 693	138 473	1 046
49 152	283 272	260 372	543 644	2 122
262 144	1 512 100	1 351 268	2 863 368	9 286
1 048 576	6 050 208	5 306 139	11 356 347	32 786

Elliptic Curve *Constant* Scalar Multiplication

In practice people use the same B_1 for many numbers:
Can we do better for a fixed B_1?

Elliptic Curve *Constant* Scalar Multiplication

In practice people use the same B_1 for many numbers:
Can we do better for a fixed B_1?

B. Dixon and A. K. Lenstra. Massively parallel elliptic curve factoring. Eurocrypt 1992.

- **Observation:** Low Hamming-weight integers $\rightarrow$ fewer EC-additions
- **Idea:** Search for low-weight prime products
  Partition the set of primes in subsets of cardinality of most three
- **Result:** Lowered the weight by $\approx$ a factor three

Elliptic Curve *Constant* Scalar Multiplication

In practice people use the same B_1 for many numbers:
Can we do better for a fixed B_1?

B. Dixon and A. K. Lenstra. Massively parallel elliptic curve factoring. Eurocrypt 1992.

- **Observation:** Low Hamming-weight integers $\rightarrow$ fewer EC-additions
- **Idea:** Search for low-weight prime products
  Partition the set of primes in subsets of cardinality of most three
- **Result:** Lowered the weight by $\approx$ a factor three

$$1028107 \cdot 1030639 \cdot 1097101 = 11624960862233886 73$$
$$w(1028107) = 10, \quad w(1030639) = 16,$$
$$w(1097101) = 11, \quad w(11624960862233886 73) = 8$$

Elliptic Curve *Constant* Scalar Multiplication

We try the opposite approach ($c(s) := \#\mathbf{A}$ in the addition chain)

- Generate integers s with "good" $\mathbf{D}/\mathbf{A}$ ratio
- Test for B_1-smoothness and factor these integers $s = \prod_i \hat{s}_i$

J. Franke, T. Kleinjung, F. Morain, and T. Wirth. Proving the primality of very large numbers with fastECPP. Algorithmic Number Theory 2004

Elliptic Curve *Constant* Scalar Multiplication

We try the opposite approach ($c(s) := \#\mathbf{A}$ in the addition chain)

- Generate integers s with "good" $\mathbf{D}/\mathbf{A}$ ratio
- Test for B_1-smoothness and factor these integers $s = \prod_i \hat{s}_i$

  J. Franke, T. Kleinjung, F. Morain, and T. Wirth. Proving the primality of very large numbers with fastECPP. Algorithmic Number Theory 2004
- Combine integers s_j such that

$$\prod_i s_i = \prod_i \prod_j \hat{s}_{i,j} = k = \operatorname{lcm}(1, \ldots, B_1) = \prod_\ell p_\ell$$

  i.e. all the $\hat{s}_{i,j}$ match all the p_ℓ
- Such that $\sum_i c(s_i = \prod_j \hat{s}_{i,j}) < c'(\prod_i \prod_j \hat{s}_{i,j}) = c'(k)$

Addition/subtraction chain

Addition/subtraction chain resulting in s

$$a_r = s, \ldots, a_1, a_0 = 1$$

s.t. every $a_i = a_j \pm a_k$ with $0 \leq j, k < i$

Avoid unnecessary computations

- Only double the last element
  $A_{3,0}, D_0, D_0, D_0 \rightarrow (3, 2, 2, 2, 1)$ vs $A_{1,0}, D_0 \rightarrow (3, 2, 1)$
- Only add or subtract to the last integer in the sequence
  (*Brauer chains* or *star addition chains*)
  This avoids computing the addition of two previous values without using this result

Addition chains with restrictions

Reduce the number of duplicates

Idea: Only add or subtract an even number from an odd number *and* after an addition (or subtraction) always perform a duplication

Generation

Start with $u_0 = 1$ (and end with an $\pm$),

$$u_{i+1} = \begin{cases} 2u_i \\ u_i \pm u_j \quad \text{for } j < i \text{ and } u_i \equiv 0 \not\equiv u_j \bmod 2 \end{cases}$$

Addition chains with restrictions

Reduce the number of duplicates

Idea: Only add or subtract an even number from an odd number *and* after an addition (or subtraction) always perform a duplication

Generation

Start with $u_0 = 1$ (and end with an $\pm$),

$$u_{i+1} = \begin{cases} 2u_i \\ u_i \pm u_j \quad \text{for } j < i \text{ and } u_i \equiv 0 \not\equiv u_j \bmod 2 \end{cases}$$

Given **A** EC-additions and **D** EC-duplications this approach generates

$$\binom{\mathbf{D}-1}{\mathbf{A}-1} \cdot \mathbf{A}! \cdot 2^{\mathbf{A}} \text{ integers}$$

Brauer chains vs Restricted chains ($\mathbf{A} = 3, \mathbf{D} = 50$)

$140 \cdot \#\text{Restricted chain} \approx \#\text{Brauer chain}$

$1.09 \cdot \text{uniq}(\#\text{Restricted chains}) \approx \text{uniq}(\#\text{Brauer chains})$

No storage

Only add or subtract the input

Less integers are generated: $\binom{\mathbf{D}-1}{\mathbf{A}-1} \cdot 2^{\mathbf{A}}$

Brauer chains vs Restricted chains ($\mathbf{A} = 3, \mathbf{D} = 50$)

$140 \cdot \#\text{Restricted chain} \approx \#\text{Brauer chain}$

$1.09 \cdot \text{uniq}(\#\text{Restricted chains}) \approx \text{uniq}(\#\text{Brauer chains})$

No storage

Only add or subtract the input

Less integers are generated: $\binom{\mathbf{D}-1}{\mathbf{A}-1} \cdot 2^{\mathbf{A}}$

Combining the smooth-integers

- Greedy approach (use good $\mathbf{D}/\mathbf{A}$ ratios first)
- Selection process is randomized
- Score according to the size of the prime divisors
- Left-overs are done using brute-force

$2.9 \cdot 10^9$-smoothness testing

No-storage setting			Low-storage setting		
A	**D**	#ST	**A**	**D**	#ST
1	$5 - 200$	$3.920 \cdot 10^2$	1	$5 - 250$	$4.920 \cdot 10^2$
2	$10 - 200$	$7.946 \cdot 10^4$	2	$10 - 250$	$2.487 \cdot 10^5$
3	$15 - 200$	$1.050 \cdot 10^7$	3	$15 - 250$	$1.235 \cdot 10^8$
4	$20 - 200$	$1.035 \cdot 10^9$	4	$20 - 221$	$3.714 \cdot 10^{10}$
5	$25 - 200$	$8.114 \cdot 10^{10}$	5	$25 - 152$	$2.429 \cdot 10^{12}$
6	$30 - 124$	$2.858 \cdot 10^{11}$	**5**	$\mathbf{153 - 220}$	$\mathbf{1.460 \cdot 10^{11}}$
7	$35 - 55$	$2.529 \cdot 10^{10}$	**6**	$\mathbf{60 - 176}$	$\mathbf{2.513 \cdot 10^{11}}$
Total		$3.932 \cdot 10^{11}$			$2.864 \cdot 10^{12}$

$2.9 \cdot 10^9$-smoothness tests on our mini-cluster using 4.5 GB memory
(5×8 Intel Xeon CPU E5430 2.66GHz)
Results obtained in ≈ 6 months

$2.9 \cdot 10^9$-smoothness testing

No-storage setting			Low-storage setting		
A	**D**	#ST	**A**	**D**	#ST
1	$5 - 200$	$3.920 \cdot 10^2$	1	$5 - 250$	$4.920 \cdot 10^2$
2	$10 - 200$	$7.946 \cdot 10^4$	2	$10 - 250$	$2.487 \cdot 10^5$
3	$15 - 200$	$1.050 \cdot 10^7$	3	$15 - 250$	$1.235 \cdot 10^8$
4	$20 - 200$	$1.035 \cdot 10^9$	4	$20 - 221$	$3.714 \cdot 10^{10}$
5	$25 - 200$	$8.114 \cdot 10^{10}$	5	$25 - 152$	$2.429 \cdot 10^{12}$
6	$30 - 124$	$2.858 \cdot 10^{11}$	**5**	$\mathbf{153 - 220}$	$\mathbf{1.460 \cdot 10^{11}}$
7	$35 - 55$	$2.529 \cdot 10^{10}$	**6**	$\mathbf{60 - 176}$	$\mathbf{2.513 \cdot 10^{11}}$
Total		$3.932 \cdot 10^{11}$			$2.864 \cdot 10^{12}$

Smooth integers

B_1	No-storage	Low-Storage	Total
3 000 000	$1.99 \cdot 10^9$	$7.00 \cdot 10^9$	$8.99 \cdot 10^9$
2 900 000 000	$1.05 \cdot 10^{10}$	$3.47 \cdot 10^{10}$	$4.53 \cdot 10^{10}$

Example $B_1 = 256$, No-Storage

#D	#A	product	addition chain
11	1	$89 \cdot 23$	$S_0 D^{11}$
14	2	$197 \cdot 83$	$S_0 D^5 S_0 D^9$
15	2	$193 \cdot 191$	$S_0 D^{12} A_0 D^3$
15	2	$199 \cdot 19 \cdot 13$	$A_0 D^{14} A_0 D^1$
18	1	$109 \cdot 37 \cdot 13 \cdot 5$	$A_0 D^{18}$
19	2	$157 \cdot 53 \cdot 7 \cdot 3 \cdot 3$	$S_0 D^6 S_0 D^{13}$
21	3	$223 \cdot 137 \cdot 103$	$A_0 D^{10} A_0 D^{10} A_0 D^1$
23	3	$179 \cdot 149 \cdot 61 \cdot 5$	$S_0 D^{13} A_0 D^5 S_0 D^5$
28	1	$127 \cdot 113 \cdot 43 \cdot 29 \cdot 5 \cdot 3$	$S_0 D^{28}$
30	3	$181 \cdot 173 \cdot 167 \cdot 11 \cdot 7 \cdot 3$	$A_0 D^{11} A_0 D^{16} A_0 D^3$
33	5	$211 \cdot 73 \cdot 67 \cdot 59 \cdot 47 \cdot 3$	$S_0 D^6 A_0 D^2 A_0 D^{11} S_0 D^3 S_0 D^{11}$
36	4	$241 \cdot 131 \cdot 101 \cdot 79 \cdot 31 \cdot 11$	$A_0 D^2 A_0 D^{16} A_0 D^{16} A_0 D^2$
41	4	$233 \cdot 229 \cdot 163 \cdot 139 \cdot 107 \cdot 17$	$S_0 D^9 S_0 D^4 S_0 D^{11} S_0 D^{17}$
49	5	$251 \cdot 239 \cdot 227 \cdot 151 \cdot 97 \cdot 71 \cdot 41$	$S_0 D^3 S_0 D^{29} A_0 D^4 A_0 D^8 A_0 D^5$
8	0	2^8	D^8
361	38	**Total**	

Results

Cost \ B_1	256	512	1024	12,288	49,152	262,144
			EECM-MPFQ			
#M	1,608	3,138	6,116	67,693	260,372	1,351,268
#S	1,436	2,952	5,892	70,780	283,272	1,512,100
#M+#S	3,044	6,090	12,008	138,473	543,644	2,863,368
A	69	120	215	1,864	6,392	29,039
D	359	738	1,473	17,695	70,818	378,025
#**R**	38	62	134	1,046	2,122	9,286
			No Storage Setting			
#M	1,400	2,842	5,596	65,873	262,343	1,389,078
#S	1,444	2,964	5,912	70,768	283,168	1,511,428
#M+#S	2,844	5,806	11,508	136,641	545,511	2,900,506
A	38	75	141	1,564	6,113	31,280
D	361	741	1,478	17,692	70,792	377,857
#**R**	10	10	10	10	10	10
			Low Storage Setting			
#M	1,383	2,783	5,481	64,634	255,852	1,354,052
#S	1,448	2,964	5,908	70,740	283,056	1,510,796
#M+#S	2,831	5,747	11,389	135,374	538,908	2,864,848
A	35	66	124	1,366	5,127	25,956
D	362	741	1,477	17,685	70,764	377,699
#**R**	22	22	22	26	26	26

Results

Cost \ B_1	256	512	1024	12,288	49,152	262,144
			EECM-MPFQ			
#M	1,608	3,138	6,116	67,693	260,372	1,351,268
#S	1,436	2,952	5,892	70,780	283,272	1,512,100
#M+#S	3,044	6,090	12,008	138,473	543,644	2,863,368
A	69	120	215	1,864	6,392	29,039
D	359	738	1,473	17,695	70,818	378,025
#**R**	38	62	134	1,046	2,122	9,286
			No Storage Setting			
#M	1,400	2,842	5,596	65,873	262,343	1,389,078
#S	1,444	2,964	5,912	70,768	283,168	1,511,428
#M+#S	2,844	5,806	11,508	136,641	545,511	2,900,506
A	38	75	141	1,564	6,113	31,280
D	361	741	1,478	17,692	70,792	377,857
#**R**	10	10	10	10	10	10
			Low Storage Setting			
#M	1,383	2,783	5,481	64,634	255,852	1,354,052
#S	1,448	2,964	5,908	70,740	283,056	1,510,796
#M+#S	2,831	5,747	11,389	135,374	538,908	2,864,848
A	35	66	124	1,366	5,127	25,956
D	362	741	1,477	17,685	70,764	377,699
#**R**	22	22	22	26	26	26

Results

Cost \ B_1	256	512	1024	12,288	49,152	262,144
			EECM-MPFQ			
#M	1,608	3,138	6,116	67,693	260,372	1,351,268
#S	1,436	2,952	5,892	70,780	283,272	1,512,100
#M+#S	3,044	6,090	12,008	138,473	543,644	2,863,368
A	69	120	215	1,864	6,392	29,039
D	359	738	1,473	17,695	70,818	378,025
#**R**	38	62	134	1,046	2,122	9,286
			No Storage Setting			
#M	1,400	2,842	5,596	65,873	262,343	1,389,078
#S	1,444	2,964	5,912	70,768	283,168	1,511,428
#M+#S	2,844	5,806	11,508	136,641	545,511	2,900,506
A	38	75	141	1,564	6,113	31,280
D	361	741	1,478	17,692	70,792	377,857
#**R**	10	10	10	10	10	10
			Low Storage Setting			
#M	1,383	2,783	5,481	64,634	255,852	1,354,052
#S	1,448	2,964	5,908	70,740	283,056	1,510,796
#M+#S	2,831	5,747	11,389	135,374	538,908	2,864,848
A	35	66	124	1,366	5,127	25,956
D	362	741	1,477	17,685	70,764	377,699
#**R**	22	22	22	26	26	26