Solving Discrete Logarithms on a 170-bit MNT Curve by Pairing Reduction

Aurore Guillevic and François Morain and Emmanuel Thomé

University of Calgary, PIMS–CNRS, LIX–École Polytechnique, Inria, Loria

SAC, August 12, 2016

Outline

Motivation: Pairing-based cryptography

The Number Field Sieve algorithm

GF(p^3): breaking a 508-bit MNT curve

Asymetric cryptography

Factorization (RSA cryptosystem)

Discrete logarithm problem (Diffie–Hellman, etc)

Given a finite cyclic group $(\mathbf{G}, \cdot)$, a generator g and $y \in \mathbf{G}$, compute x s.t. $y = g^x$.

Common choice of $\mathbf{G}$:
prime finite field $\mathbb{F}_p$ (since 1976), characteristic 2 finite field $\mathbb{F}_{2^n}$, elliptic curve $E(\mathbb{F}_p)$ (since 1985),
In particular at SAC 2016: Kummer surfaces and Four$\mathbb{Q}$ $E(\mathbb{F}_{p^2})$ (Smith and Longa talks)

Elliptic curves in cryptography

$$E : y^2 = x^3 + ax + b, \; a, b \in \mathbb{F}_p$$

- proposed in 1985 by Koblitz, Miller
- $E(\mathbb{F}_p)$ has an efficient group law (chord an tangent rule) $\rightarrow$ **G**
- $\#E(\mathbb{F}_p) = p + 1 - t$, trace t: $|t| \leqslant 2\sqrt{p}$

Need a prime-order (or with tiny cofactor) elliptic curve:

$$h \cdot \ell = \#E(\mathbb{F}_p), \quad \ell \text{ is prime}, \quad h \text{ tiny, e.g. } h = 1, 2$$

- compute t
- slow to compute in 1985: can use *supersingular curves* whose trace is known.

Supersingular elliptic curves

Example over $\mathbb{F}_p$, $p \geq 5$

$$E : y^2 = x^3 + x \ / \ \mathbb{F}_p, \quad p = 3 \bmod 4$$

s.t. $t = 0$, $\#E(\mathbb{F}_p) = p + 1$.
take p s.t. $p + 1 = 4 \cdot \ell$ where ℓ is prime.

Supersingular elliptic curves

Example over $\mathbb{F}_p$, $p \geq 5$

$$E : y^2 = x^3 + x \ / \ \mathbb{F}_p, \quad p = 3 \bmod 4$$

s.t. $t = 0$, $\#E(\mathbb{F}_p) = p + 1$.
take p s.t. $p + 1 = 4 \cdot \ell$ where ℓ is prime.

1993: Menezes-Okamoto-Vanstone and Frey-Rück attacks
There exists a pairing e that embeds the group $E(\mathbb{F}_p)$ into $\mathbb{F}_{p^2}$
where **DLP is much easier**.
Do not use supersingular curves.

Supersingular elliptic curves

Example over $\mathbb{F}_p$, $p \geq 5$

$$E : y^2 = x^3 + x \ / \ \mathbb{F}_p, \quad p = 3 \bmod 4$$

s.t. $t = 0$, $\#E(\mathbb{F}_p) = p + 1$.
take p s.t. $p + 1 = 4 \cdot \ell$ where ℓ is prime.

1993: Menezes-Okamoto-Vanstone and Frey-Rück attacks
There exists a pairing e that embeds the group $E(\mathbb{F}_p)$ into $\mathbb{F}_{p^2}$ where **DLP is much easier**.
Do not use supersingular curves.
But computing a pairing is **very slow**:
[Harasawa Shikata Suzuki Imai 99]: 161467s (112 days) on a 163-bit supersingular curve, where $\mathbf{G}_T \subset \mathbb{F}_{p^2}$ of 326 bits.

Pairing-based cryptography

1999: Frey–Muller–Rück: actually, Miller Algorithm can be **much faster**.

2000: *[Joux ANTS]* Computing a pairing can be done efficiently (1s on a supersingular 528-bit curve, $\mathbf{G}_T \subset \mathbb{F}_{p^2}$ of 1055 bits).

Weil or Tate pairing on an elliptic curve

Discrete logarithm problem with one more dimension.

$$e \ : \ E(\mathbb{F}_{p^n})[\ell] \times E(\mathbb{F}_{p^n})[\ell] \longrightarrow \mathbb{F}_{p^n}^*, \quad e([a]P, [b]Q) = e(P, Q)^{ab}$$

Pairing-based cryptography

1999: Frey–Muller–Rück: actually, Miller Algorithm can be **much faster**.

2000: *[Joux ANTS]* Computing a pairing can be done efficiently (1s on a supersingular 528-bit curve, $\mathbf{G}_T \subset \mathbb{F}_{p^2}$ of 1055 bits).

Weil or Tate pairing on an elliptic curve

Discrete logarithm problem with one more dimension.

$$e \; : \; E(\mathbb{F}_{p^n})[\ell] \times E(\mathbb{F}_{p^n})[\ell] \longrightarrow \mathbb{F}_{p^n}^*, \quad e([a]P, [b]Q) = e(P, Q)^{ab}$$

Attacks

Pairing-based cryptography

1999: Frey–Muller–Rück: actually, Miller Algorithm can be **much faster**.

2000: *[Joux ANTS]* Computing a pairing can be done efficiently (1s on a supersingular 528-bit curve, $\mathbf{G}_T \subset \mathbb{F}_{p^2}$ of 1055 bits).

Weil or Tate pairing on an elliptic curve

Discrete logarithm problem with one more dimension.

$$e \ : \ E(\mathbb{F}_{p^n})[\ell] \times E(\mathbb{F}_{p^n})[\ell] \longrightarrow \mathbb{F}_{p^n}^*, \quad e([a]P, [b]Q) = e(P, Q)^{ab}$$

Attacks

- inversion of e : hard problem (exponential)

Pairing-based cryptography

1999: Frey–Muller–Rück: actually, Miller Algorithm can be **much faster**.

2000: *[Joux ANTS]* Computing a pairing can be done efficiently (1s on a supersingular 528-bit curve, $\mathbf{G}_T \subset \mathbb{F}_{p^2}$ of 1055 bits).

Weil or Tate pairing on an elliptic curve

Discrete logarithm problem with one more dimension.

$$e \;:\; E(\mathbb{F}_{p^n})[\ell] \times E(\mathbb{F}_{p^n})[\ell] \longrightarrow \mathbb{F}_{p^n}^*, \quad e([a]P, [b]Q) = e(P, Q)^{ab}$$

Attacks

- inversion of e : hard problem (exponential)
- discrete logarithm computation in $E(\mathbb{F}_p)$: hard problem (exponential, in $O(\sqrt{\ell})$)

Pairing-based cryptography

1999: Frey–Muller–Rück: actually, Miller Algorithm can be **much faster**.

2000: *[Joux ANTS]* Computing a pairing can be done efficiently (1s on a supersingular 528-bit curve, $\mathbf{G}_T \subset \mathbb{F}_{p^2}$ of 1055 bits).

Weil or Tate pairing on an elliptic curve

Discrete logarithm problem with one more dimension.

$$e \;:\; E(\mathbb{F}_{p^n})[\ell] \times E(\mathbb{F}_{p^n})[\ell] \longrightarrow \mathbb{F}_{p^n}^*, \quad e([a]P, [b]Q) = e(P, Q)^{ab}$$

Attacks

- inversion of e : hard problem (exponential)
- discrete logarithm computation in $E(\mathbb{F}_p)$: hard problem (exponential, in $O(\sqrt{\ell})$)
- discrete logarithm computation in $\mathbb{F}_{p^n}^*$: **easier, subexponential** $\rightarrow$ take a large enough field

Common target groups $\mathbb{F}_{p^n}$

- $\mathbb{F}_{p^2}$ where $E/\mathbb{F}_p$ is a supersingular curve
- $\mathbb{F}_{p^3}, \mathbb{F}_{p^4}, \mathbb{F}_{p^6}$ where E is an ordinary MNT curve
  [Miyaji Nakabayashi Takano 01]
- $\mathbb{F}_{p^{12}}$ where E is a BN curve *[Barreto-Naehrig 05]*

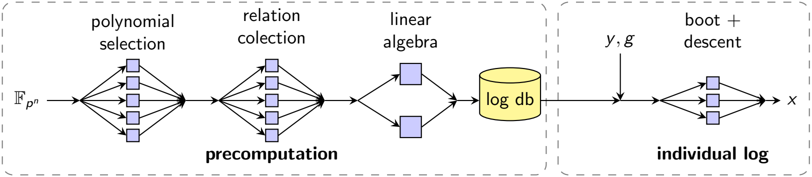
DLP hardness for a 3072-bit finite field:

- **hard** in $\mathbb{F}_p$ where p is a 3072-bit prime
- **easy** in $\mathbb{F}_{2^n}$ where $n = 3072$
  [Barbulescu, Gaudry, Joux, Thomé 14, Granger et al. 14]
- what about $\mathbb{F}_{p^3}$ where p is a 1024-bit prime?

Start the comparison for 512-bit finite fields.

NFS algorithm to compute discrete logarithms

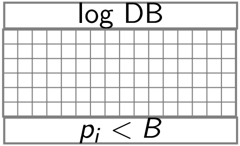
Input : finite field $\mathbb{F}_{p^n}$, generator g, target y
Output : discrete logarithm x of y in basis g, $g^x = y$



[graph: N. Heninger]

Relation collection and Linear algebra

1. Polynomial selection
2. Relation collection
3. Linear algebra



   - We know the log of *small* elements in $\mathbb{Z}[x]/(f(x))$ and $\mathbb{Z}[x]/(g(x))$
   - *small* elements are of the form $a_i - b_i x = \mathfrak{p}_i \in \mathbb{Z}[x]/(f(x))$, s.t. $|\,\mathrm{Norm}(\mathfrak{p}_i)| = p_i < B$

4. Individual discrete logarithm

NFS algorithm for DL in GF(p^n)

How to generate relations ?
Use *two* distinct rings $R_f = \mathbb{Z}[x]/(f(x))$, $R_g = \mathbb{Z}[x]/(g(x))$ and two maps ρ_f, ρ_g that map $x \in R_f$, resp. $x \in R_g$ to *the same* element $z \in \mathrm{GF}(p^n)$:

$$\begin{cases} \rho_f : x \in R_f \mapsto z, \\ \rho_g : x \in R_g \mapsto z \end{cases}$$

$$\begin{tikzcd}
& \mathbb{Z}[x] \arrow[dl] \arrow[dr] & \\
R_f = \mathbb{Z}[x]/(f(x)) \arrow[dr, "R_f \ni x \mapsto z"'] & & R_g = \mathbb{Z}[x]/(g(x)) \arrow[dl, "R_g \ni x \mapsto z"] \\
& \mathrm{GF}(p^n) = \mathrm{GF}(p)[z]/(\varphi(z)) &
\end{tikzcd}$$

Weak MNT curve, 170-bit prime p, 508-bit $\mathbb{F}_{p^3}$

[Miyaji Nakabayashi Takano 01]

$E/\mathbb{F}_p:\ y^2 = x^3 + ax + b$, where

$a =$ 0x22ffbb20cc052993fa27dc507800b624c650e4ff3d2

$b =$ 0x1c7be6fa8da953b5624efc72406af7fa77499803d08

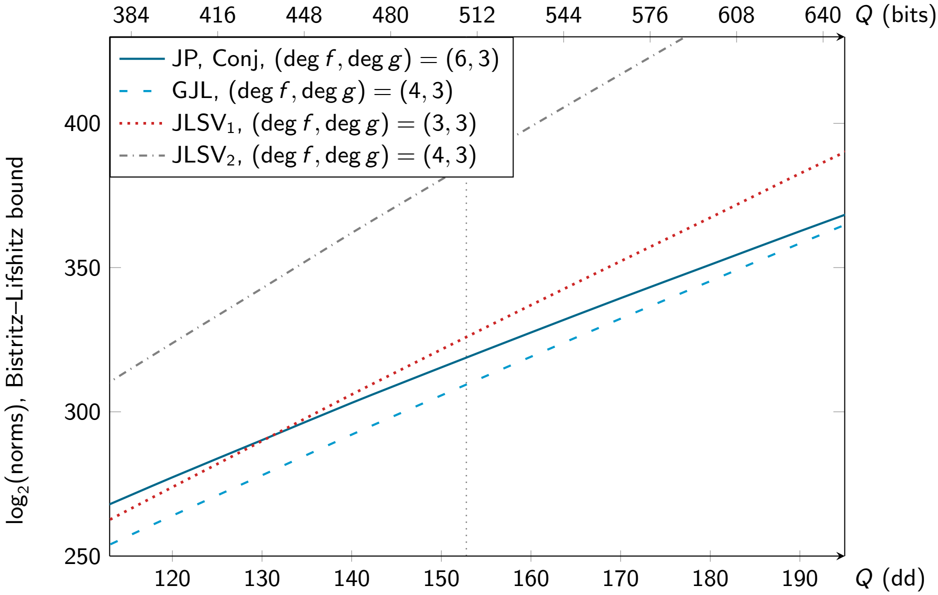
$p =$ 0x26dccacc5041939206cf2b7dec50950e3c9fa4827af

$\ell =$ 0xa60fd646ad409b3312c3b23ba64e082ad7b354d

such that

$$\begin{aligned} x_0 &= -\text{0x732c8cf5f983038060466} \\ t &= 6x_0 - 1 \\ p &= 12x_0^2 - 1 \\ \#E(\mathbb{F}_p) &= p + 1 - t = 7^2 \cdot 313 \cdot \ell \end{aligned}$$

Polynomial selection: norm estimates



Polynomial selection: norm estimates

Joux–Pierrot and Conjugation	319 bits	Galois aut. order 3
Generalized Joux–Lercier	310 bits	–
JouxLercierSmartVercauteren JLSV1	326 bits	Galois aut. order 3

Galois automorphism of order 3 $\rightarrow$ will obtain 3 times more relations for free

- JLSV1: $\sqrt{p} \approx 2^{85}$ possible polynomials f
- Conjugation: allow non-monic polynomials $\rightarrow \approx 2^{20}$ possible f

In both cases, take

$$\varphi(x, y) = x^3 - yx^2 - (y+3)x - 1$$

s.t. $\sigma : x \mapsto -1 - 1/x$ degree 3 Galois automorphism of $\mathbb{Q}[x]/(\varphi(x))$, K_f and K_g

Polynomial Selection

$$
\begin{aligned}
p &= 908761003790427908077548955758380356675829026531247 \\
&\quad \text{of 170 bits} \\
A &= 28y^2 + 16y - 109 \\
f &= 28x^6 + 16x^5 - 261x^4 - 322x^3 + 79x^2 + 152x + 28 \\
&\quad \|f\|_\infty = 8.33 \text{ bits} \\
\alpha(f) &= -2.9
\end{aligned}
$$

$$
\begin{aligned}
g &= 24757815186639197370442122x^3 + 40806897040253680471775183x^2 \\
&\quad -33466548519663911639551183x - 24757815186639197370442122 \\
&\quad \|g\|_\infty = 85.01 \text{ bits} \\
\alpha(g) &= -4.1 \\
&\quad \text{Murphy's E value:} \\
\mathbb{E}(f,g) &= 1.31 \cdot 10^{-12}
\end{aligned}
$$

Relation Collection: sieving

Smoothness bound $B = 50000000 (= 2^{25.6})$ on both sides
Special-q in $[B, 2^{27}]$

660 core-days (4-core Intel Xeon E5520 @ 2.27GHz).

$57 \cdot 10^6$ relations $\rightarrow$ filtered $\rightarrow$
1982791×1982784 matrix with weight $w(M) = 396558692$.
The whole matrix would have 7 more columns for taking the 7 Schirokaurer Maps into account.

Linear Algebra (`cado-nfs`)

8 sequences in Block-Wiedemann algorithm.
8 Krylov sequences 250 core-days, four 16-code nodes / sequence
finding linear matrix generator 3.1 core-days / 64 cores
building solution 170 core-days
Reconstructed virtual logarithms for 15196345 out of the 15206761 elements of the bases (99.9%).

423 core-days on a cluster Intel Xeon E5-2650, 2.4GHz

Individual discrete logarithm

Take $P_0 = [x_P, y_P] \in E(\mathbb{F}_p)$,
$x_P = \lfloor \pi 10^{50} \rfloor = 314159265358979323846264338327950288419716939937510$
$y_P = \sqrt{x_P^3 + a x_P + b} = 460095575547938627692618282835762310592027720907930$
and set $\mathrm{Target}_E = P = [7^2 \cdot 313] P_0$.

e is the reduced Tate pairing $e_\ell(P, Q)^{(p^3-1)/\ell}$

$E[\ell] \cong \mathbb{Z}/\ell\mathbb{Z} \oplus \mathbb{Z}/\ell\mathbb{Z} \simeq \langle G_1 \rangle \oplus \langle G_2 \rangle$ where
G_1 a generator of $E(\mathbb{F}_p)[\ell]$
G_2 a generator of second dim of r-torsion of $E(\mathbb{F}_{p^3})[\ell]$

Target in $\mathbb{F}_{p^3}$: $T = e(P, G_2)$, Basis: $g = e(G_1, G_2)$
Change $\mathbb{F}_{p^3} = \mathbb{F}_p[X]/(X^3 + X + 1)$ to $\mathbb{F}_p[Z]/(\varphi(Z))$

$$T = 0x11a2f1f13fa9b08703a033ee3c4321539156f865ee9 + 0x1098c3b7280ef2cf8b091d08197de0a9ba935ff79c6\ Z + 0x221205020e7729cb46166a9edfd5acb3bf59dd0a7d4\ Z^2$$

$$G_T = 0xd772111b150ec08f0ad89d987f1b037c630155608c + 0xf956cab6840c7e909abc29584f1aee48ccbd39d698\ Z + 0x205eb5b1e09f76bf0ef85efeaa3fdcb3827d43441b3\ Z^2$$

Individual discrete logarithm

Initial splitting: 32-core hours
preimage of g^{52154} in K_f has 59-bit-smooth norm
preimage of $g^{35313}T$ in K_f has 54-bit-smooth norm

Descent procedure: 13.4 hours.

Virtual log of g:
$\mathrm{vlog}(g) = 0x8c58b66f0d8b2e99a1c0530b2649ec0c76501c3$
virtual log of the target:
$\mathrm{vlog}(T) = 0x48a6bcf57cacca997658c98a0c196c25116a0aa$
Then $\log_g(T) = \mathrm{vlog}(T)/\mathrm{vlog}(g) \bmod \ell$.

$$\log(T) = \log(P) = 0x711d13ed75e05cc2ab2c9ec2c910a98288ec038 \bmod \ell\ .$$

Running-time comparison

record	relation col.	linear algebra	ind. log	total
Kleinjung 2007 530-bit $\mathbb{F}_p$	3.3 CPU-years 3.2 GHz Xeon64	14 years 3.2 GHz Xeon64	(few hours) 3.2 GHz Xeon64	17.3 years
BGGM 2014 529-bit $\mathbb{F}_{p^2}$	0.19 year 2.0 GHz E5-2650	30.3 *hours* NVidia GTX 680 graphic card	(few hours) 2.0 GHz E5-2650	0.2 year
BGGM 2015 512-bit $\mathbb{F}_{p^3}$	2.33 years 2.4 GHz Xeon E5-2650	15 years	(few days)	17.3 years
this work **508-bit $\mathbb{F}_{p^3}$**	**1.81 years** 2.27GHz 4-core Xeon E5520	**1.16 years*** 2.4GHz Xeon E5-2650	(2 days) 2.27GHz 4-core Xeon E5520	**2.97 years**

* **linear algebra** modulo $\ell \sim p$ (where $\ell \mid p+1-t$) instead of $\ell \sim p^{\varphi(n)} = p^2$, (+ better polynomials + smaller matrix)
$\rightarrow$ much faster than previous 512-bit $\mathbb{F}_{p^3}$.

State of the art in prime field: 768-bit $\mathbb{F}_p$
Kleinjung–Diem–Lenstra–Priplata–Stahlke 2016
5300 core-years on 2.2 GHz Xeon E5-2660

Future work

- 600-bit DL record in $\mathbb{F}_{p^3}, \mathbb{F}_{p^4}, \mathbb{F}_{p^6}, \mathbb{F}_{p^{12}}$ (with Gaudry, Grémy, Morain, Thomé)
- need new techniques for $\mathbb{F}_{p^4}, \mathbb{F}_{p^6}, \mathbb{F}_{p^{12}}$ (*[Barbulescu–Gaudry–Kleinjung]+[Kim]+[Sarkar–Singh]+[Jeong–Kim]*: Extended TNFS)
- implementation in `cado-nfs`

Consequences:
Increase the size of the target groups $\mathbb{F}_{p^n}$ in pairing-based cryptography

SAC in Newfoundland special: discrete log record in 180dd (593-bit) $\mathbb{F}_{p^3}$

Joint work with Pierick Gaudry and François Morain.

Setting

generic prime $p = \lfloor 10^{59}\pi \rfloor + 3569289$ of 60 decimal digits
118dd prime-order subgroup ℓ s.t. $39\ell = p^2 + p + 1$

Polynomial selection: Conjugation method (6 core-days)

$$
\begin{aligned}
f_0 &= 20x^6 - x^5 - 290x^4 - 375x^3 + 15x^2 + 121x + 20 \\
f_1 &= 1366383471413152347582603764 70x^3 - 297571133526942208465012783 13x^2 \\
&\quad -4396721547766399251212824077 23x - 1366383471413152347582603764 70 \\
\varphi &= \gcd(f_0, f_1) \mod p = x^3 - yx^2 - (y+3)x - 1, \\
&\quad \text{where } y \text{ is a root modulo } p \text{ of} \\
A(y) &= 20y^2 - y - 169
\end{aligned}
$$

discrete log record in 180dd (593-bit) $\mathbb{F}_{p^3}$

Relation collection: 9 core-years

Special-q lattice sieving, smoothness bound of $80\text{M} = 2^{26.25}$, large prime bound of 2^{28}.
Saved a factor 3 thanks to Galois σ.
Obtained 37705176 raw relations on side 0 and 36850254 on side 1.

Filtering

Duplicate removal: 48016023 unique relations (35.5% dup. rate)
Densification: 4.5M matrix, 200 coefs per row on average.

Linear algebra: 14 core-years

Block-Wiedemann algorithm with the 7 vectors of Schirokauer maps as input vectors for the $n = 7$ sequences.

Back-substitution (incl. Schirokauer maps): 32 core-days

Obtained the virtual logs of 98.7% of the ideals below 2^{28}.

discrete log record in 180dd (593-bit) $\mathbb{F}_{p^3}$

Generator of $\mathbb{F}_{p^3}$: $g = x + 5$
and target from the decimals of $e = \exp(1)$:

$$\begin{aligned} T = {} & 2718281828459045235360287471352662497757247093699959574966 \; x^2 \\ & + 7627724076630353547594571382178525166427427466391932003059 \; x \\ & + 2181741359662904357290033429526059563073813232862794349076 32 \end{aligned}$$

Individual discrete logarithm: 1 core-day

$\log_g(T) = 534299829823865777675362637916832228131271219214173907 44\backslash$
$427787489975365178688648857593262073182222695276991481809921 2 \bmod \ell$

Total time: 23 core-years

180dd integer factorization: 5.6 core-years (Gaudry's talk SAC'14)
NFS-DL in GF(p), 180dd p: 131 core-years (BGIJT 14)
NFS-DL in GF(p^2), 180dd p^2: 0.5 core-years (BGGM 15)

discrete log record in 180dd (593-bit) $\mathbb{F}_{p^3}$

Generator of $\mathbb{F}_{p^3}$: $g = x + 5$
and target from the decimals of $e = \exp(1)$:

$$\begin{aligned} T = {} & 271828182845904523536028747135266249775724709369995957496696\, x^2 \\ & + 762772407663035354759457138217852516642742746639193200305 99\, x \\ & + 218174135966290435729003342952605956307381323286279434907632 \end{aligned}$$

Individual discrete logarithm: 1 core-day

$\log_g(T) =$ 534299829823865777675362637916832228131271219214173907444\
427787489975365178688648857593262073182222695276991481809921 2 $\bmod \ell$

Total time: 23 core-years

180dd integer factorization:	5.6 core-years (Gaudry's talk SAC'14)
NFS-DL in GF(p), 180dd p:	131 core-years (BGIJT 14)
NFS-DL in GF(p^2), 180dd p^2:	0.5 core-years (BGGM 15)

Thank you & Merci !