On Summation Polynomials for Elliptic Curves

Workshop BunnyTN 7

Alessandro Amadori

University of Trento

16th November, 2016

1. ECDLP
2. Summation Polynomials
3. Our results

The Elliptic Curve Discrete Logarithm Problem

The Discrete Logarithm Problem

- In 1976, Diffie and Hellman invented new cryptographic protocols which base their security on the Discrete Logarithm Problem.

The Discrete Logarithm Problem (DLP)

Let G be a finite cyclic group, let g be a generator of G and h a generic element of G. The *Discrete Logarithm Problem* consists in computing $x \in \mathbb{Z}$ such that

$$g^x = h.$$

- In 1984, Koblitz and Miller independently proposed to employ Elliptic Curves for Cryptosystems basing their security on the DLP.

Algorithms solving the DLP

Finite Fields:

- Index Calculus is the most efficient algorithm solving the DLP over finite fields.
- Subexponential complexity for FFDLP: efficiency relies on fast factorisation algorithms.

Elliptic Curves:

- Same arguments do not hold for elliptic curves: no analogous factorisation in elliptic curves.
- Algorithms do not exploit the structure of the curves: the most efficient known is Pollard's ρ which has exponential complexity and works on generic groups.

Index Calculus: a quick Overview

Given g generator of $\mathbb{F}_p^*$ and $h \in \mathbb{F}_p^*$ such that $g^x = h$, the steps of the index calculus are

1. choose a factor base of s prime numbers;
2. collect s powers of g that factorise in the set of prime numbers;
3. using linear algebra, solve the discrete logarithm problems with respect to the base of prime numbers;
4. combine relations obtained from linear algebra;
5. using the logarithms previously computed, obtain x.

Summation Polynomials

Summation Polynomials

Summation polynomials were introduced by I. Semaev in 2004 as proposal to solve the point decomposition problem for elliptic curves.

Definition

Let E be an elliptic curve defined over a field $\mathbb{K}$, of any characteristic, with $\overline{\mathbb{K}}$ as its algebraic closure. For any integer $m \geq 2$ the m-th summation polynomial $\mathcal{S}_m \in \mathbb{K}[X_1, \ldots, X_m]$ is such that, given $x_1, \ldots, x_m \in \overline{\mathbb{K}}$, then $\mathcal{S}_m(x_1, \ldots, x_m) = 0$ if and only if there exist $y_1, \ldots, y_m \in \overline{\mathbb{K}}$ for which $(x_1, y_1), \ldots, (x_m, y_m) \in E(\overline{\mathbb{K}})$ and

$$(x_1, y_1) + \ldots + (x_m, y_m) = \infty$$

Summation Polynomials

Let E be an elliptic curve defined over a field $\mathbb{K}$, with $char(\mathbb{K}) \neq 2, 3$, by $y^2 = x^3 + Ax + B$, with $A, B \in \mathbb{K}$. Let $m \geq 2$ be an integer number. Then

- $\mathcal{S}_2(X_1, X_2) = X_1 - X_2$
- $\mathcal{S}_3(X_1, X_2, X_3) =$
  $(X_1 - X_2)^2 X_3^2 - 2[(X_1 + X_2)(X_1 X_2 + A) + 2B]X_3 + (X_1 X_2 - A)^2 - 4B(X_1 + X_2)$

and for all $m \geq 4$ and $1 \leq k \leq m - 3$ it holds

$$\mathcal{S}_m(X_1, \ldots, X_m) = Res_X(\mathcal{S}_{m-k}(X_1, \ldots, X_{m-k-1}, X), \mathcal{S}_{k+2}(X_{m-k}, \ldots, X_m, X)).$$

Summation polynomials

Summation polynomials have the following properties:

- $\mathcal{S}_m$ is symmetric of degree 2^{m-2} in each variable X_i,
- they are absolutely irreducible,
- $\mathcal{S}_m(X_1, \ldots, X_m) = \mathcal{S}_{m-1}^2(X_1, \ldots, X_{m-1})X_m^{2^{m-2}} + \ldots$ for any $m \geq 3$.

For fields of characteristic 2 and 3, adjustments are needed, but the previous properties still hold.

A sketch of the algorithm

The proposal for an Index Calculus algorithm for elliptic curves is due to Semaev (2004) and Gaundry (2009), that suggest that it should be made of the following steps.

Take an elliptic curve E defined over $\mathbb{F}_p$, a point $P \in E(\mathbb{F}_p)$, an integer w and $Q := wP$. Fix $m \geq 2$ and

- for two random integers u and v, set $R := uP + vQ = (x, y)$;
- find $x_1, \ldots, x_m \in \mathbb{F}_p$ of $\mathcal{S}_{m+1}(x_1, \ldots, x_m, x) = 0$ and compute a relation
$$(x_1, y_1) + \ldots + (x_m, y_m) + R = \infty;$$
- collect several point relations and combine them using linear algebra.

The solutions are sought with components in a subset $\mathcal{V} \subset \mathbb{F}_p$ of size around $\sqrt[m]{p}$.
Generalisations can be made for curves defined over $\mathbb{F}_q$.

Useful Observations

- It can happen that some of the points appearing in the point relation have their y-coordinate in $\mathbb{F}_{q^2} \setminus \mathbb{F}_q$. The sum of all such points is a point of order 2.
- To find roots of the polynomial $\mathcal{S}_{m+1}(X_1, \ldots, X_m, x)$ (combined with the field equations and the constaint of belonging in $\mathcal{V}$), a Gröbner basis algorithm should be employed (F4).
- For elliptic curves defined over $\mathbb{F}_q = \mathbb{F}_{p^n}$, the roots are sought with components in a random vector subspace $\mathcal{V} \subset \mathbb{F}_q$ of dimension $\lceil \frac{n}{m} \rceil$ using the Weil descent technique.

Simplifying computations

Due to the high degree of $\mathcal{S}_{m+1}$, it can be really hard to find its roots.

In 2015 Semaev and Karabina proposed to split a summation polynomial equation into a polynomial system: solving the polynomial equation $\mathcal{S}_{m+1}(X_1, \ldots, X_m, x) = 0$ looking for solutions with components in $\mathcal{V}$ is equivalent to solve the polynomial system

$$\begin{cases} \mathcal{S}_3(X_1, X_2, U_1) & = 0 \\ \mathcal{S}_3(U_i, U_{i+1}, X_{i+2}) & = 0 \qquad 1 \leq i \leq m-3 \\ \mathcal{S}_3(U_{m-2}, X_m, x) & = 0 \end{cases}$$

looking for solutions in $x_1, \ldots, x_m \in \mathcal{V}$ and $u_1, \ldots u_{m-2} \in \mathbb{F}_q$. The system is made of polynomials of total degree 4. Moreover the probability of finding a solution with $x_1, \ldots, x_m \in \mathcal{V}$ is about

$$\frac{|V|^m}{q \cdot m!}$$

F4 algorithm solving the system

- Use F4 algorithm to compute the Gröbner basis and then compute the roots.
- In case $\mathbb{F}_q = \mathbb{F}_{p^n}$, first apply the Weil descent.
- The complexity of F4 is

$$(2m-2)^{\omega D_{reg}}$$

where ω is the linear algebra constant and D_{reg} is a parameter that depends on the degrees of the polynomial equations of the system. In the case of $\mathbb{F}_q = \mathbb{F}_{p^n}$, after applying the Weil descent, the complexity of F4 becomes

$$[n(m-1)]^{\omega D_{reg}}$$

The First Fall Degree

Generally D_{reg} is approximated with another parameter: the first fall degree.

Definition

Let R be a polynomial ring over a field $\mathbb{K}$ and let $F := \{f_1, \ldots, f_s\} \subset R$ be a set of polynomials of total degrees $\leq D$. The first fall degree of F is the smallest D, denoted by $D_{firstfall}$, such that there exist r polynomials $g_i \in R$ such that $\max_{i \in \{1,\ldots,r\}}(\deg(f_i) + \deg(g_i)) = D_{firstfall}$ and $\deg(\sum_{i=1}^{r} g_i f_i) < D_{firstfall}$, but $\sum_{i=1}^{r} g_i f_i \neq 0$.

- **First Fall Degree Assumption**: $D_{reg} = D_{firstfall}$.

Complexity for the polynomial System

- For the polynomial system

$$\begin{cases} \mathcal{S}_3(X_1, X_2, U_1) & = 0 \\ \mathcal{S}_3(U_i, U_{i+1}, X_{i+2}) & = 0 \\ \mathcal{S}_3(U_{m-2}, X_m, x) & = 0 \end{cases} \qquad 1 \leq i \leq m-3$$

  in characteristic 2, the first fall degree is 4.

- The complexity of F4 for the previous system is

$$[n(m-1)]^{4\omega}$$

  in characteristic 2.

- No generic bounds for the first fall degree of the previous system are known in other characteristics.

Collecting relations

To compute the Discrete Logarithm we have to combine point relations. Suppose that we have found

$$P_1^{(i)} + \ldots + P_m^{(i)} = u_i P + v_i Q = R_i$$

where $u_i \neq u_j$ and $v_i \neq v_j$ if $i \neq j$.
The goal is to obtain a sufficient number of relations such that they are *dependent*, i.e. there exist integers $\mu_1, \ldots, \mu_s$ for which

$$\sum_{j=1}^{s} \mu_j (P_1^{(j)} + \ldots + P_m^{(j)}) = \sum_{j=1}^{s} \mu_j R_j = \infty$$

obtaining the congruence

$$w(\sum_{j=1}^{s} \mu_j v_j) \equiv -\sum_{j=1}^{s} \mu_j u_j \mod ord(P)$$

for which w can be computed. Around $|\mathcal{V}| \approx \sqrt[m]{q}$ point relations are expected to be computed

Algorithmic Complexity

The complexity is given by the sum of the two main parts of the algorithm: computation of relations and combination of relations. Consider elliptic curves defined over $\mathbb{F}_q = \mathbb{F}_{p^n}$, set $k = \lceil \frac{n}{m} \rceil$.

$$\frac{m!}{p^{mk-n}} p^k n^{D_{ff}\omega} + p^{2k} \approx p^{c\sqrt{n \log n}}$$

with $c = \frac{2}{\sqrt{2 \log p}}$ and $m \approx \sqrt{\frac{2n \log p}{\log n}}$.
In case $p = 2$ and the value of $m \approx \sqrt{\frac{2n \log 2}{\log n}}$,

$$2^{c\sqrt{n \log n}}$$

where $c = \frac{2}{\sqrt{2 \log 2}} \approx 1.44$, which is faster than Pollard's ρ starting from a certain n.

OUR RESULTS

Consider an elliptic curve E defined over a finite field $\mathbb{F}_p$, p odd prime, and fix an integer $m \geq 3$. Set $s := \lceil \sqrt[m]{p} \rceil$. We construct the set $\mathcal{V}$ with a different approach. Define the set $\mathcal{F}$ as:

$$\mathcal{F} = \{R_i \in E(\mathbb{F}_p) | R_i = u_i P + v_i Q, i \in \{1, \ldots, s\}\}.$$

with u_i and v_i random integers and define $\mathcal{V}$ as the set of x-coordinates of the points in $\mathcal{F}$.
After defining $\mathcal{V}$ we try to solve the single system with $t \in \{3, \ldots, m\}$

$$\begin{cases} \mathcal{S}_3(X_1, X_2, U_1) & = 0 \\ \mathcal{S}_3(U_i, U_{i+1}, X_{i+2}) & = 0 \qquad 1 \leq i \leq t-4 \\ \mathcal{S}_3(U_{t-3}, X_{t-1}, X_t) & = 0. \end{cases}$$

looking for solutions $x_1, \ldots, x_t \in \mathcal{V}$.

Once a solution with components in $\mathcal{V}$ is found, then, we can write the following point relation:

$$\pm P_{i_1} \pm \ldots \pm P_{i_t} = \infty$$

where $P_{i_1}, \ldots, P_{i_t} \in \mathcal{F}$. We obtain:

$$\pm(u_{i_1} P + v_{i_1} Q) \pm \ldots \pm (u_{i_t} P + v_{i_t} Q) = \infty,$$

which gives the following linear congruence

$$(\pm u_{i_1} \pm \ldots \pm u_{i_t}) + (\pm v_{i_1} \pm \ldots \pm v_{i_t}) w \equiv 0 \mod ord(P)$$

Following this procedure, only one point relation is needed (instead of about $\sqrt[m]{p}$) and so no linear algebra steps are employed.

Experimental results and asymptotical complexity

We compared the time resolutions of our proposal with Semaev's algorithm (as well as with Pollard's ρ).

bit(p)	m	our method (s)	Semaev's method (s)	Pollard's ρ (s)
11	3	0.000 - 0.290	1504.070 - 8020.840	0.000
12	3	0.020 - 3.560	23773.500 - 55610.460	0.000
13	3	0.090 - 3.310	89843.100 - 550387.840	0.000

We estimated the complexity of our proposed algorithm as:

$$2^{\frac{3D_{ff}+1}{2}\sqrt{\log p}\log\log p}$$

with $m \approx \sqrt{\log p}$.

Thank you for your attention!