A Few More Index Calculus Algorithms For The Elliptic Curve Discrete Logarithm Problem

Daniela Mueller

Joint work with Gary McGuire

School of Mathematics and Statistics
University College Dublin, Ireland

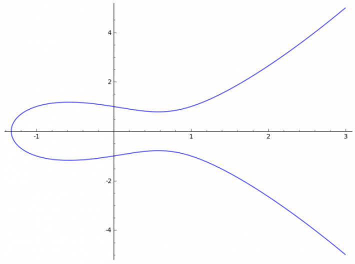
CAEN 2018

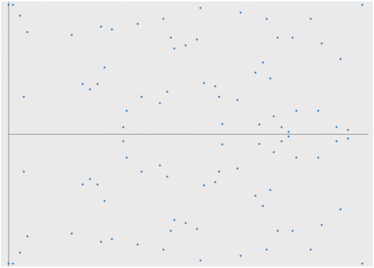
IRISH RESEARCH COUNCIL
An Chomhairle um Thaighde in Éirinn

UCD School of Mathematics and Statistics
Scoil na Matamaitice agus na Staitisticí UCD

What is the Elliptic Curve Discrete Logarithm Problem?

Elliptic curve in short Weierstrass form: $E : y^2 = x^3 + Ax + B$ over finite field. We will focus on prime order fields.





ECDLP: Given points $P, Q \in E(\mathbb{F}_q)$, find s such that $Q = sP$ (if it exists).

Index Calculus Algorithm for ECDLP (Gaudry)

Let $P, Q \in E(\mathbb{F}_q)$ such that there exists s that solves $Q = sP$.

Factor Base

Define a factor base $\mathcal{F} \subseteq E(\mathbb{F}_q)$.

Point Decomposition $\Rightarrow$ Relation

Generate points $R = aP + bQ$, where a, b are random, and try to write $R = P_1 + P_2 + \ldots + P_m$, where $P_1, \ldots, P_m \in \mathcal{F}$.

Linear Algebra Step

$$\begin{pmatrix} \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots \end{pmatrix} \begin{pmatrix} log(P_1) \\ log(P_2) \\ \cdots \\ log(P_{|\mathcal{F}|}) \end{pmatrix} = \begin{pmatrix} a_1 \\ a_2 \\ \cdots \\ \cdots \\ a_k \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \\ \cdots \\ \cdots \\ b_k \end{pmatrix} log(Q)$$

Find $v = (v_1, \ldots, v_k)$ such that $vM = 0$, $\sum_i a_i v_i \neq 0$ and $\sum_i b_i v_i$ is invertible. Then $s = (\sum_i a_i v_i)/(\sum_i b_i v_i)$.

Summation Polynomials

Let E be an elliptic curve over a field K.

Definition

For $m \geq 2$, define the m$^{\text{th}}$ summation polynomial $S_m = S_m(X_1, X_2, \ldots, X_m)$ of E by the following property:

Let $x_1, x_2, \ldots, x_m \in \overline{K}$, then $S_m(x_1, x_2, \ldots, x_m) = 0$

if and only if

$\exists y_1, y_2, \ldots, y_m \in \overline{K}$ such that $(x_i, y_i) \in E(\overline{K})$ $\forall i$ and $(x_1, y_1) + (x_2, y_2) + \ldots + (x_m, y_m) = \mathcal{O}$.

Summation Polynomials

Theorem (I. Semaev)

Let E be an elliptic curve given by $y^2 = x^3 + Ax + B$ over a field K with characteristic $\neq 2, 3$.
Then the summation polynomials are given by

$S_2(X_1, X_2) = X_1 - X_2$,

$S_3(X_1, X_2, X_3) = (X_1 - X_2)^2 X_3^2 - 2((X_1 + X_2)(X_1 X_2 + A) + 2B)X_3 + ((X_1 X_2 - A)^2 - 4B(X_1 + X_2))$,

$S_m(X_1, \ldots, X_m) =$
$Res_Y(S_{m-k}(X_1 \ldots, X_{m-k-1}, Y), S_{k+2}(X_{m-k} \ldots, X_m, Y))$
for $m \geq 4$ and any $m - 3 \geq k \geq 1$.

Furthermore, the polynomials S_m, $m \geq 3$, are symmetric and of degree 2^{m-2} in each variable, and absolutely irreducible.

4th Summation Polynomial S_4

Elliptic Curve defined by $y^2 = x^3 + 3x + 3$ over $\mathbb{F}_5$

```
X1^4*X2^4*X3^4 + X1^4*X2^4*X3^3*X4 + X1^4*X2^4*X3^2*X4^2 + X1^4*X2^4*X3*X4^3 + X1^4*X2^4*X4^4 + X1^4*X2^3*X3^4*X4 + 4*X1^4*X2^3*X3^3*X4^2 +
3*X1^4*X2^3*X3^3 + 4*X1^4*X2^3*X3^2*X4^3 + 2*X1^4*X2^3*X3^2*X4 + X1^4*X2^3*X3^2 + X1^4*X2^3*X3*X4^4 + 2*X1^4*X2^3*X3*X4^2 + 3*X1^4*X2^3*X3*X4 +
3*X1^4*X2^3*X4^3 + X1^4*X2^3*X4^2 + X1^4*X2^2*X3^4*X4^2 + 4*X1^4*X2^2*X3^3*X4^3 + 2*X1^4*X2^2*X3^3*X4 + X1^4*X2^2*X3^3 + X1^4*X2^2*X3^2*X4^4 +
2*X1^4*X2^2*X3^2*X4^2 + 2*X1^4*X2^2*X3^2*X4 + 4*X1^4*X2^2*X3^2 + 2*X1^4*X2^2*X3*X4^3 + 2*X1^4*X2^2*X3*X4^2 + X1^4*X2^2*X3*X4 + 4*X1^4*X2^2*X3 +
X1^4*X2^2*X4^3 + 4*X1^4*X2^2*X4^2 + 4*X1^4*X2^2*X4 + 4*X1^4*X2^2 + X1^4*X2*X3^4*X4^3 + X1^4*X2*X3^3*X4^4 + 2*X1^4*X2*X3^3*X4^2 + 3*X1^4*X2*X3^3*X4 +
2*X1^4*X2*X3^2*X4^3 + 2*X1^4*X2*X3^2*X4^2 + X1^4*X2*X3^2*X4 + 4*X1^4*X2*X3^2 + 3*X1^4*X2*X3*X4^3 + X1^4*X2*X3*X4^2 + 2*X1^4*X2*X3*X4 + 4*X1^4*X2*X4^2 +
4*X1^4*X2 + X1^4*X3^4*X4^4 + 3*X1^4*X3^3*X4^3 + X1^4*X3^3*X4^2 + X1^4*X3^2*X4^3 + 4*X1^4*X3^2*X4^2 + 4*X1^4*X3^2*X4 + 4*X1^4*X3^2 + 4*X1^4*X3*X4^2 +
4*X1^4*X3 + 4*X1^4*X4^2 + 4*X1^4*X4 + X1^4 + X1^3*X2^4*X3^4*X4 + 4*X1^3*X2^4*X3^3*X4^2 + 3*X1^3*X2^4*X3^3 + 4*X1^3*X2^4*X3^2*X4^3 + 2*X1^3*X2^4*X3^2*X4 +
X1^3*X2^4*X3^2 + X1^3*X2^4*X3*X4^4 + 2*X1^3*X2^4*X3*X4^2 + 3*X1^3*X2^4*X3*X4 + 3*X1^3*X2^4*X4^3 + X1^3*X2^4*X4^2 + 4*X1^3*X2^3*X3^4*X4^2 +
3*X1^3*X2^3*X3^4 + 3*X1^3*X2^3*X3^3*X4+ X1^3*X2^3*X3^3 + 4*X1^3*X2^3*X3^2*X4^4 + X1^3*X2^3*X3^2*X4^2 + 3*X1^3*X2^3*X3^2*X4 + X1^3*X2^3*X3^2 +
3*X1^3*X2^3*X3*X4^3 + 3*X1^3*X2^3*X3*X4^2 + 2*X1^3*X2^3*X3 + 3*X1^3*X2^3*X4^4 + X1^3*X2^3*X4^3 + X1^3*X2^3*X4^2 + 2*X1^3*X2^3*X4 + 2*X1^3*X2^3 +
4*X1^3*X2^2*X3^4*X4^3 + 2*X1^3*X2^2*X3^4*X4 + X1^3*X2^2*X3^4 + 4*X1^3*X2^2*X3^3*X4^4 + X1^3*X2^2*X3^3*X4^2 + 3*X1^3*X2^2*X3^3*X4 + X1^3*X2^2*X3^3 +
X1^3*X2^2*X3^2*X4^3 + 4*X1^3*X2^2*X3^2*X4^2 + 3*X1^3*X2^2*X3^2*X4 + 3*X1^3*X2^2*X3^2 + 2*X1^3*X2^2*X3*X4^4 + 3*X1^3*X2^2*X3*X4^3 + 3*X1^3*X2^2*X3*X4^2
+ 4*X1^3*X2^2*X3*X4 + X1^3*X2^2*X4^4 + X1^3*X2^2*X4^3 + 3*X1^3*X2^2*X4^2 + X1^3*X2^2 + X1^3*X2*X3^4*X4^4 + 2*X1^3*X2*X3^4*X4^2 + 3*X1^3*X2*X3^4*X4 +
3*X1^3*X2*X3^3*X4^3 + 3*X1^3*X2*X3^3*X4^2 + 2*X1^3*X2*X3^3 + 2*X1^3*X2*X3^2*X4^4 + 3*X1^3*X2*X3^2*X4^3 + 3*X1^3*X2*X3^2*X4^2 + 4*X1^3*X2*X3^2*X4 +
3*X1^3*X2*X3*X4^4 + 4*X1^3*X2*X3*X4^2 + 4*X1^3*X2*X3*X4 + 2*X1^3*X2*X4^3 + 2*X1^3*X2 + 3*X1^3*X3^4*X4^3 + X1^3*X3^4*X4^2 + 3*X1^3*X3^3*X4^4 +
X1^3*X3^3*X4^3 + X1^3*X3^3*X4^2 + 2*X1^3*X3^3*X4 + 2*X1^3*X3^3 + X1^3*X3^2*X4^4 + X1^3*X3^2*X4^3 + 3*X1^3*X3^2*X4^2 + X1^3*X3^2 + 2*X1^3*X3*X4^3 +
2*X1^3*X3 + 2*X1^3*X4^3 + X1^3*X4^2 + 2*X1^3*X4 + 4*X1^3 + X1^2*X2^4*X3^4*X4^2 + 4*X1^2*X2^4*X3^3*X4^3 + 2*X1^2*X2^4*X3^3*X4 + X1^2*X2^4*X3^3 +
X1^2*X2^4*X3^2*X4^4 + 2*X1^2*X2^4*X3^2*X4^2 + 2*X1^2*X2^4*X3^2*X4 + 4*X1^2*X2^4*X3^2 + 2*X1^2*X2^4*X3*X4^3 + 2*X1^2*X2^4*X3*X4^2 + X1^2*X2^4*X3*X4 +
4*X1^2*X2^4*X3 + X1^2*X2^4*X4^3 + 4*X1^2*X2^4*X4^2 + 4*X1^2*X2^4*X4 + 4*X1^2*X2^4 + 4*X1^2*X2^3*X3^4*X4^3 + 2*X1^2*X2^3*X3^4*X4 + X1^2*X2^3*X3^4 +
4*X1^2*X2^3*X3^3*X4^4 + X1^2*X2^3*X3^3*X4^2 + 3*X1^2*X2^3*X3^3*X4 + X1^2*X2^3*X3^3 + X1^2*X2^3*X3^2*X4^3 + 4*X1^2*X2^3*X3^2*X4^2 + 3*X1^2*X2^3*X3^2*X4
+ 3*X1^2*X2^3*X3^2 + 2*X1^2*X2^3*X3*X4^4 + 3*X1^2*X2^3*X3*X4^3 + 3*X1^2*X2^3*X3*X4^2 + 4*X1^2*X2^3*X3*X4 + X1^2*X2^3*X4^4 + X1^2*X2^3*X4^3 +
3*X1^2*X2^3*X4^2 + X1^2*X2^3 + X1^2*X2^2*X3^4*X4^4 + 2*X1^2*X2^2*X3^4*X4^2 + 2*X1^2*X2^2*X3^4*X4 + 4*X1^2*X2^2*X3^4 + X1^2*X2^2*X3^3*X4^3 +
4*X1^2*X2^2*X3^3*X4^2 + 3*X1^2*X2^2*X3^3*X4 + 3*X1^2*X2^2*X3^3 + 2*X1^2*X2^2*X3^2*X4^4 + 4*X1^2*X2^2*X3^2*X4^3 + 2*X1^2*X2^2*X3^2*X4^2 +
2*X1^2*X2^2*X3^2*X4 + 4*X1^2*X2^2*X3^2 + 2*X1^2*X2^2*X3*X4^4 + 3*X1^2*X2^2*X3*X4^3 + 2*X1^2*X2^2*X3*X4^2 + 3*X1^2*X2^2*X3*X4 + 2*X1^2*X2^2*X3 +
4*X1^2*X2^2*X4^4 + 3*X1^2*X2^2*X4^3 + 4*X1^2*X2^2*X4^2 + 2*X1^2*X2^2*X4 + 4*X1^2*X2^2 + 2*X1^2*X2*X3^4*X4^3 + 2*X1^2*X2*X3^4*X4^2 + X1^2*X2*X3^4*X4 +
4*X1^2*X2*X3^4 + 2*X1^2*X2*X3^3*X4^4 + 3*X1^2*X2*X3^3*X4^3 + 3*X1^2*X2*X3^3*X4^2 + 4*X1^2*X2*X3^3*X4 + 2*X1^2*X2*X3^2*X4^4 + 3*X1^2*X2*X3^2*X4^3 +
2*X1^2*X2*X3^2*X4^2 + 3*X1^2*X2*X3^2*X4 + 2*X1^2*X2*X3^2 + X1^2*X2*X3*X4^4 + 4*X1^2*X2*X3*X4^3 + 3*X1^2*X2*X3*X4^2 + 4*X1^2*X2*X3*X4 + 3*X1^2*X2*X3 +
4*X1^2*X2*X4^4 + 2*X1^2*X2*X4^2 + 3*X1^2*X2*X4 + X1^2*X2 + X1^2*X3^4*X4^3 + 4*X1^2*X3^4*X4^2 + 4*X1^2*X3^4*X4 + 4*X1^2*X3^4 + X1^2*X3^3*X4^4 +
X1^2*X3^3*X4^3 + 3*X1^2*X3^3*X4^2 + X1^2*X3^3 + 4*X1^2*X3^2*X4^4 + 3*X1^2*X3^2*X4^3 + 4*X1^2*X3^2*X4^2 + 2*X1^2*X3^2*X4 + 4*X1^2*X3^2 + 4*X1^2*X3*X4^4 +
2*X1^2*X3*X4^2 + 3*X1^2*X3*X4 + X1^2*X3 + 4*X1^2*X4^4 + X1^2*X4^3 + 4*X1^2*X4^2 + X1^2*X4 + X1*X2^4*X3^4*X4^3 + X1*X2^4*X3^3*X4^4 + 2*X1*X2^4*X3^3*X4^2 +
3*X1*X2^4*X3^3*X4 + 2*X1*X2^4*X3^2*X4^3 + 2*X1*X2^4*X3^2*X4^2 + X1*X2^4*X3^2*X4 + 4*X1*X2^4*X3^2 + 3*X1*X2^4*X3*X4^3 + X1*X2^4*X3*X4^2 + 2*X1*X2^4*X3*X4 +
4*X1*X2^4*X4^2 + 4*X1*X2^4 + X1*X2^3*X3^4*X4^4 + 2*X1*X2^3*X3^4*X4^2 + 3*X1*X2^3*X3^4*X4 + 3*X1*X2^3*X3^3*X4^3 + 3*X1*X2^3*X3^3*X4^2 + 2*X1*X2^3*X3^3 +
2*X1*X2^3*X3^2*X4^4 + 3*X1*X2^3*X3^2*X4^3 + 3*X1*X2^3*X3^2*X4^2 + 4*X1*X2^3*X3^2*X4 + 3*X1*X2^3*X3*X4^4 + 4*X1*X2^3*X3*X4^2 + 4*X1*X2^3*X3*X4 +
2*X1*X2^3*X4^3 + 2*X1*X2^3 + 2*X1*X2^2*X3^4*X4^3 + 2*X1*X2^2*X3^4*X4^2 + X1*X2^2*X3^4*X4 + 4*X1*X2^2*X3^4 + 2*X1*X2^2*X3^3*X4^4 + 3*X1*X2^2*X3^3*X4^3 +
3*X1*X2^2*X3^3*X4^2 + 4*X1*X2^2*X3^3*X4 + 2*X1*X2^2*X3^2*X4^4 + 3*X1*X2^2*X3^2*X4^3 + 2*X1*X2^2*X3^2*X4^2 + 3*X1*X2^2*X3^2*X4 + 2*X1*X2^2*X3^2 +
X1*X2^2*X3*X4^4 + 4*X1*X2^2*X3*X4^3 + 3*X1*X2^2*X3*X4^2 + 4*X1*X2^2*X3*X4 + 3*X1*X2^2*X3 + 4*X1*X2^2*X4^4 + 2*X1*X2^2*X4^2 + 3*X1*X2^2*X4 + X1*X2^2 +
3*X1*X2*X3^4*X4^3 + X1*X2*X3^4*X4^2 + 2*X1*X2*X3^4*X4 + 3*X1*X2*X3^3*X4^4 + 4*X1*X2*X3^3*X4^2 + 4*X1*X2*X3^3*X4 + X1*X2*X3^2*X4^4 + 4*X1*X2*X3^2*X4^3 +
3*X1*X2*X3^2*X4^2 + 4*X1*X2*X3^2*X4 + 3*X1*X2*X3^2 + 2*X1*X2*X3*X4^4 + 4*X1*X2*X3*X4^3 + 4*X1*X2*X3*X4^2 + 2*X1*X2*X3*X4 + 3*X1*X2*X3 + 3*X1*X2*X4^2 +
3*X1*X2*X4 + 4*X1*X3^4*X4^2 + 4*X1*X3^4 + 2*X1*X3^3*X4^3 + 2*X1*X3^3 + 4*X1*X3^2*X4^4 + 2*X1*X3^2*X4^2 + 3*X1*X3^2*X4 + X1*X3^2 + 3*X1*X3*X4^2 + 3*X1*X3*X4 +
4*X1*X4^4 + 2*X1*X4^3 + X1*X4^2+ X2^4*X3^4*X4^4 + 3*X2^4*X3^3*X4^3 + X2^4*X3^3*X4^2 + X2^4*X3^2*X4^3 + 4*X2^4*X3^2*X4^2 + 4*X2^4*X3^2*X4 + 4*X2^4*X3^2 +
4*X2^4*X3*X4^2 + 4*X2^4*X3 + 4*X2^4*X4^2 + 4*X2^4*X4 + X2^4 + 3*X2^3*X3^4*X4^3 + X2^3*X3^4*X4^2 + 3*X2^3*X3^3*X4^4 + X2^3*X3^3*X4^3 + X2^3*X3^3*X4^2 +
2*X2^3*X3^3*X4 + 2*X2^3*X3^3 + X2^3*X3^2*X4^4 + X2^3*X3^2*X4^3 + 3*X2^3*X3^2*X4^2 + X2^3*X3^2 + 2*X2^3*X3*X4^3 + 2*X2^3*X3 + 2*X2^3*X4^3 + X2^3*X4^2 +
2*X2^3*X4 + 4*X2^3 + X2^2*X3^4*X4^3 + 4*X2^2*X3^4*X4^2 + 4*X2^2*X3^4*X4 + 4*X2^2*X3^4 + X2^2*X3^3*X4^4 + X2^2*X3^3*X4^3 + 3*X2^2*X3^3*X4^2 + X2^2*X3^3 +
4*X2^2*X3^2*X4^4 + 3*X2^2*X3^2*X4^3 + 4*X2^2*X3^2*X4^2 + 2*X2^2*X3^2*X4 + 4*X2^2*X3^2 + 4*X2^2*X3*X4^4 +2*X2^2*X3*X4^2 + 3*X2^2*X3*X4 + X2^2*X3 +
4*X2^2*X4^4 + X2^2*X4^3 + 4*X2^2*X4^2 + X2^2*X4 + 4*X2*X3^4*X4^2 + 4*X2*X3^4+ 2*X2*X3^3*X4^3 + 2*X2*X3^3 + 4*X2*X3^2*X4^4 + 2*X2*X3^2*X4^2 + 3*X2*X3^2*X4 +
X2*X3^2 + 3*X2*X3*X4^2 + 3*X2*X3*X4 + 4*X2*X4^4 + 2*X2*X4^3 + X2*X4^2 + 4*X3^4*X4^2 + 4*X3^4*X4 + X3^4 + 2*X3^3*X4^3 + X3^3*X4^2 + 2*X3^3*X4 + 4*X3^3 +
4*X3^2*X4^4 + X3^2*X4^3 + 4*X3^2*X4^2 + X3^2*X4 + 4*X3*X4^4 + 2*X3*X4^3 + X3*X4^2 + X4^4 + 4*X4^3
```

4th Summation Polynomial S_4

$$
\begin{aligned}
S_4(x_1, x_2, x_3, x_4) &= \mathrm{Res}_Y(S_3(x_1, x_2, Y), S_3(x_3, x_4, Y)) \\
&= \det \begin{pmatrix} a_2 & a_1 & a_0 & 0 \\ 0 & a_2 & a_1 & a_0 \\ b_2 & b_1 & b_0 & 0 \\ 0 & b_2 & b_1 & b_0 \end{pmatrix} \\
&= a_2(b_0(a_2 b_0 - 2a_0 b_2 - a_1 b_1) + a_0 b_1^2) \\
&\quad + b_2(a_1(a_1 b_0 - a_0 b_1) + a_0^2 b_2).
\end{aligned}
$$

where $S_3(x_1, x_2, Y) = a_2 Y^2 + a_1 Y + a_0$
and $S_3(x_3, x_4, Y) = b_2 Y^2 + b_1 Y + b_0$.

We can evaluate this using only 21 multiplications and 24 additions.

Evaluating the Summation Polynomials

Theorem
Evaluating S_m at a point $(x_1, \ldots, x_m)$ in $\mathbb{F}_p^m$ can be done in $O(\log^2 p)$ steps for $m \ll p$.

Idea:
Use $S_m(x_1, \ldots, x_m) = Res_Y(S_3(x_1, x_2, Y), S_{m-1}(x_3, \ldots, x_m, Y))$ for $m \geq 4$ and $x_1, \ldots, x_m \in \mathbb{F}_p$.

This allows us to evaluate S_9 and S_{10} even if we can't compute them.

Our Variant of Index Calculus for the ECDLP

Let $P, Q \in E(\mathbb{F}_p)$ such that there exists k that solves $Q = kP$.

Factor Base (see also Amadori, Pintore, Sala 2017)

Compute random integers $a_1, ..., a_s$, $b_1, ..., b_s$. Then our factor base is $\mathcal{F} = \{a_1 P + b_1 Q, ..., a_s P + b_s Q\}$.

Find A Relation

Choose a multiset $\{x_1, \ldots, x_m\}$ with each $x_i \in \{x | (x, y) \in \mathcal{F}\}$ and check if $S_m(x_1, \ldots, x_m) = 0$.

Solving Step (see also APS17)

If $S_m(x_1, \ldots, x_m) = 0$ for some $\{x_1, \ldots, x_m\}$, then there exist y_i such that $(x_1, y_1) + \ldots + (x_m, y_m) = \mathcal{O}$ with (x_i, y_i) or $-(x_i, y_i)$ in $\mathcal{F}$. Substituting each $\pm(x_i, y_i)$ with $\pm(a_i P + b_i Q)$, we get a relation of the form $\sum_{i=1}^{m} \pm a_i P + \sum_{i=1}^{m} \pm b_i Q = \mathcal{O}$ and can solve for the discrete logarithm of Q, provided $\sum_{i=1}^{m} \pm b_i$ is invertible modulo the order of E.

Complexity

Theorem

The complexity of our algorithm is $O(p \log^2 p)$ *for* $m \ll s$ *and* $m \ll p$ *and* $\frac{s^{m-1}}{m!} \geq \log p$.

Still exponential, but better than other index calculus type algorithms for the ECDLP over prime fields which use Gröbner bases.

Our algorithm is embarrassingly parallel.

Another variant

Instead of evaluating S_m, choose a multiset of $m-1$ points from the factor base and check if the sum of those points is in the factor base.

Theorem

The complexity of this algorithm is $O(p)$ for $m \ll s$ and $s \geq (m-2)\log^2 p$.

Full article: Cryptology ePrint Archive, Report 2017/1262