On the discrete logarithm problem

Fabrice Mouhartem

With Fré Vercauteren in the COSIC team
Outline

1. Introduction
   - Presentation of cryptology
   - Presentation of the discrete logarithm
   - State of the art

2. Presentation of the different algorithms
   - The Function Field Sieve
   - The BGJT algorithm
   - GKZ descent

3. Results
Introduction

- Presentation of cryptology
- Presentation of the discrete logarithm
- State of the art

Presentation of the different algorithms

- The Function Field Sieve
- The BGJT algorithm
- GKZ descent

Results
## Presentation of Cryptology

<table>
<thead>
<tr>
<th>Protocols</th>
<th>SSL</th>
<th>PGP</th>
<th>Bitcoin</th>
</tr>
</thead>
</table>

Cryptology is based on different layers. We are interested in the last layer.
Presentation of Cryptology

- Protocols: SSL, PGP, Bitcoin
- Security schemes: El Gamal, RSA, YASH

Cryptology is based on different layers.
Presentation of Cryptology

- **Protocols**: SSL, PGP, Bitcoin
- **Security schemes**: El Gamal, RSA, YASH
- **Hard problems**: DLP, Factorisation, RLWE

- Cryptology is based on different layers
- We are interested in the last layer
Presentation of Cryptology

- Protocols: SSL, PGP, Bitcoin
- Security schemes: El Gamal, RSA, YASH
- Hard problems: DLP, Factorisation, RLWE

- Cryptology is based on different layers
- We are interested in the last layer
Definition

Given a triple \((G, g, h)\) where \(G = \langle g \rangle\) is a group and \(h = g^x\) we want to compute the value of \(x\).
Presentation of the Discrete Logarithm

Definition

Given a triple \((G, g, h)\) where \(G = \langle g \rangle\) is a group and \(h = g^x\) we want to compute the value of \(x\).

Main properties:

- Many protocols rely on it to ensure their security
Presentation of the Discrete Logarithm

Definition

Given a triple \((G, g, h)\) where \(G = \langle g \rangle\) is a group and \(h = g^x\) we want to compute the value of \(x\).

Main properties:

- Many protocols rely on it to ensure their security
- In the most general case, we can solve it in \(L(1/3)\)
Presentation of the Discrete Logarithm

Definition

Given a triple \((G, g, h)\) where \(G = \langle g \rangle\) is a group and \(h = g^x\) we want to compute the value of \(x\).

Main properties:

- Many protocols rely on it to ensure their security
- In the most general case, we can solve it in \(L(1/3)\)
- We can embed the group into the multiplicative group of a finite field \(\mathbb{F}_{p^\ell}\)
Presentation of the Discrete Logarithm

Definition

Given a triple \((G, g, h)\) where \(G = \langle g \rangle\) is a group and \(h = g^x\) we want to compute the value of \(x\).

Main properties:

- Many protocols rely on it to ensure their security
- In the most general case, we can solve it in \(L(1/3)\)
- We can embed the group into the multiplicative group of a finite field \(\mathbb{F}_{p^\ell}\)
- In the case where \(p \ll p^\ell\), we can solve the DLP in quasi-polynomial time.
State of the Art

Chronology:

- Introduction of the DLP: 1976
- FFS to solve DLP in subexponential time: $L(1/2)$: 1979
- Coppersmith algorithm: $L(1/3)$: 1984
- Joux $L(1/4 + o(1))$ algorithm: 2013
- BGJT quasi polynomial algorithm: 2013
- GKZ quasi polynomial algorithm: Mai 2014

Actual record: Granger, Kleinjung, Zumbrägel, $F_{2^{9234}}$ in 400,000 core hours.
State of the Art

Chronology:

<table>
<thead>
<tr>
<th>Algorithm Description</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction of the DLP</td>
<td>1976</td>
</tr>
<tr>
<td>FFS to solve DLP in subexponential time: $L(1/2)$</td>
<td>1979</td>
</tr>
<tr>
<td>Coppersmith algorithm: $L(1/3)$</td>
<td>1984</td>
</tr>
<tr>
<td>Joux $L(1/4 + o(1))$ algorithm</td>
<td>2013</td>
</tr>
<tr>
<td>BGJT quasi polynomial algorithm</td>
<td>2013</td>
</tr>
<tr>
<td>GKZ quasi polynomial algorithm</td>
<td>Mai 2014</td>
</tr>
</tbody>
</table>

Actual record: Granger, Kleinjung, Zumbrägel, $\mathbb{F}_{2^{9234}}$ in 400,000 core hours.
Outline

1 Introduction
   - Presentation of cryptology
   - Presentation of the discrete logarithm
   - State of the art

2 Presentation of the different algorithms
   - The Function Field Sieve
   - The BGJT algorithm
   - GKZ descent

3 Results
The Function Field Sieve

The algorithm can be divided in 2 independent phases:

1. The factor basis resolution
2. The individual logarithm descent
The Function Field Sieve

The algorithm can be divided in 2 independent phases:

1. The factor basis resolution
2. The individual logarithm descent

Example:

\[ P: \text{irreducible degree } n \text{ polynomial} \]
The Function Field Sieve

The algorithm can be divided in 2 independent phases:

1. The factor basis resolution
2. The individual logarithm descent

Example:

\[ P: \text{irreducible degree } n \text{ polynomial} \]

\[ \text{Factor basis} \rightarrow \text{Individual log descent} \rightarrow \text{logarithm of all degree 1 polynomials} \]
The Function Field Sieve

The algorithm can be divided in 2 independent phases:

1. The factor basis resolution
2. The individual logarithm descent

Example:

$P$: irreducible degree $n$ polynomial

- Factor basis
- Individual log descent
- Linear combination between $\log P$ and $\log$ of linear factors
- Logarithm of all degree 1 polynomials
The algorithm can be divided in 2 independent phases:

1. The factor basis resolution
2. The individual logarithm descent

Example:

\[ P: \text{irreducible degree } n \text{ polynomial} \]

\[ \text{Factor basis} \leadsto \text{Individual log descent} \]

\[ \text{logarithm of all degree 1 polynomials} \]

\[ \text{linear combination between } \log P \text{ and } \log \text{ of linear factors} \]
<table>
<thead>
<tr>
<th>Context</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finite field $\mathbb{F}_{q^n} = \mathbb{F}_q[X]/(f(X)) \equiv \text{polynomial in } \mathbb{F}_q[X] \text{ modulo } f(X)$</td>
</tr>
<tr>
<td>$f(X)</td>
</tr>
<tr>
<td>$h_0, h_1$ of degree at most $\Delta$</td>
</tr>
</tbody>
</table>
Context

Finite field $\mathbb{F}_{q^n} = \mathbb{F}_q[X]/(f(X)) \equiv \text{polynomial in } \mathbb{F}_q[X] \text{ modulo } f(X)$

$f(X)|h_1(X)X^q - h_0(X)$ irreducible of degree $n$

$h_0, h_1$ of degree at most $\Delta$

Systematic equation

$$X^q - X = \prod_{\alpha \in \mathbb{F}_q} (X - \alpha)$$
Barbulescu, Gaudry, Joux, Thomé

Context

Finite field $\mathbb{F}_{q^\delta} = \mathbb{F}_{q^\delta}[X]/(f(X)) \equiv$ polynomial in $\mathbb{F}_{q^\delta}[X]$ modulo $f(X)$

$f(X)|h_1(X)X^q - h_0(X)$ irreducible of degree $n$

$h_0, h_1$ of degree at most $\Delta$

Systematic equation

$$(aP+b)^q(cP+d) - (aP+b)(cP+b)^q = (cP+d) \prod_{\alpha \in \mathbb{F}_q} (aP+b - \alpha(cP+d))$$
Barbulescu, Gaudry, Joux, Thomé

Context

Finite field $\mathbb{F}_{q^\delta} = \mathbb{F}_q[\mathcal{X}] / (f(\mathcal{X})) \equiv \text{polynomial in } \mathbb{F}_q[\mathcal{X}] \text{ modulo } f(\mathcal{X})$

Let $f(\mathcal{X}) | h_1(\mathcal{X}) \mathcal{X}^q - h_0(\mathcal{X})$ be irreducible of degree $n$

$h_0, h_1$ of degree at most $\Delta$

Systematic equation

$$\frac{1}{h_1^{\deg P} (\deg \deg(P)(\Delta + 1) \text{ polynomial})} = \lambda \prod_{i=0}^{q} \text{linear polynomials in } P$$
BGJT: Sum up

**Heuristic**

For $q^\delta$ big enough, the matrix $\mathcal{H}_P$ where columns correspond to elements of $\mathbb{F}_{q^\delta}$ and rows correspond to a relation derived from the systematic equation is full rank.
BGJT: Sum up

**Heuristic**

For $q^\delta$ big enough, the matrix $\mathcal{H}_P$ where columns correspond to elements of $\mathbb{F}_{q^\delta}$ and rows correspond to a relation derived from the systematic equation is full rank.

**Consequences**

We can linearly relate the logarithm of $P + 0 = P$ with logarithm of degree $\left\lceil \frac{\deg P}{2} \right\rceil$ polynomials.
BGJT: Sum up

Heuristic

For $q^\delta$ big enough, the matrix $H_P$ where columns correspond to elements of $\mathbb{F}_{q^\delta}$ and rows correspond to a relation derived from the systematic equation is full rank.

Consequences

We can linearly relate the logarithm of $P + 0 = P$ with logarithm of degree $\left\lceil \frac{\deg P}{2} \right\rceil$ polynomials.

Trade-offs

- We are limited by matrix algorithms over $H_P$
- We relates $\log P$ with at least $q^2$ logarithms of smaller polynomials $\rightarrow$ many recursions
Basic Idea:

- We know how to express an irreducible degree 2 polynomial in term of degree 1 polynomials
Granger, Kleinjung, Zumbrägel

Basic Idea:
- We know how to express an irreducible degree 2 polynomial in term of degree 1 polynomials
- An irreducible degree $2^m$ polynomial in $\mathbb{F}_q$ is an irreducible degree 2 polynomial in $\mathbb{F}_{q^{2^m-1}}$
Granger, Kleinjung, Zumbrägel

Basic Idea:

- We know how to express an irreducible degree 2 polynomial in terms of degree 1 polynomials.
- An irreducible degree $2^m$ polynomial in $\mathbb{F}_q$ is an irreducible degree 2 polynomial in $\mathbb{F}_{q^{2^{m-1}}}$.

$\Rightarrow$ We know how to express an irreducible degree $2^m$ polynomials in terms of irreducible degree $2^{m-1}$ polynomials.
Degree $2^m$ Descent

\[
\begin{array}{c}
\mathbb{F}_{q^{\delta 2^m-1}} \\
\vdots \\
\mathbb{F}_{q^{\delta 2^m-2}} \\
\vdots \\
\mathbb{F}_{q^{\delta 4}} \\
\vdots \\
\mathbb{F}_{q^{\delta 2}} \\
\vdots \\
\mathbb{F}_{q^{\delta}} \\
\end{array}
\]
Degree $2^m$ Descent

$$\mathbb{F}_{q^{\delta 2^m-1}} \quad \downarrow$$

$$\mathbb{F}_{q^{\delta 2^m-2}} \quad \downarrow$$

$$\vdots$$

$$\vdots$$

$$\mathbb{F}_{q^{\delta 4}} \quad \downarrow$$

$$\mathbb{F}_{q^{\delta 2}} \quad \downarrow$$

$$\mathbb{F}_{q^{\delta}}$$

$2^m$
Degree $2^m$ Descent

\[ F_{q^{\delta 2^m - 1}} \]
\[ F_{q^{\delta 2^m - 2}} \]
\[ \vdots \]
\[ F_{q^{\delta 4}} \]
\[ \vdots \]
\[ F_{q^{\delta 2}} \]
\[ F_{q^{\delta}} \]
Degree $2^m$ Descent

\[ \mathbb{F}_{q^{\delta 2^m - 1}} \quad 1 \quad 2 \]

\[ \mathbb{F}_{q^{\delta 2^m - 2}} \]

\[ \ldots \]

\[ \mathbb{F}_{q^{\delta 4}} \]

\[ \mathbb{F}_{q^{\delta 2}} \]

\[ \mathbb{F}_{q^{\delta}} \]

\[ 2^m \]
Degree $2^m$ Descent
On the Discrete Logarithm Problem

Presentation of the different algorithms

GKZ descent

Analyse of the Method

- No heavy linear algebra during descent phase
- Restrictive conditions to be used along with other descent methods
  \( P \) irreducible of degree \( k \cdot 2^m \) to do \( m \) descent steps
- Relation between \( P \) and \( (q + 2)^m \) degree 1 polynomials for a \( 2^m \)-descent
- Can be enhanced to be space-efficient
Outline

1 Introduction
   - Presentation of cryptology
   - Presentation of the discrete logarithm
   - State of the art

2 Presentation of the different algorithms
   - The Function Field Sieve
   - The BGJT algorithm
   - GKZ descent

3 Results
Overview of the Results and Implementation Notes

- 1073 lines of Magma code
- The BGJT algorithm is suitable only for very big examples that cannot be implemented yet
- The GKZ algorithm is on average 40× faster than Coppersmith algorithm on a 204 bits field
Overview of the Results and Implementation Notes

- 1073 lines of Magma code
- The BGJT algorithm is suitable only for very big examples that cannot be implemented yet
- The GKZ algorithm is on average $40 \times$ faster than Coppersmith algorithm on a 204 bits field

* Demo *
The new **GKZ** algorithm change the **FFS** ecosystem by putting the bottleneck on the factor basis solving instead of the descent.

We started to work on this purpose improving this phase using different relations for the linear phase.

The descent tree quality can still be improved.

There is still many way not explored to improve the algorithm both theoretically and in practice.
References

Razvan Barbulescu, Pierrick Gaudry, Antoine Joux, and Emmanuel Thomé.
A quasi-polynomial algorithm for discrete logarithm in finite fields of small characteristic.
http://eprint.iacr.org/.

Robert Granger, Thorsten Kleinjung, and Jens Zumbrägel.
On the powers of 2.
http://eprint.iacr.org/.
Thank you for your attention,
Feel free to ask questions
How to improve the memory efficiency of GKZ

We make the computations in a depth-first way.
Future works?

- Exploit the liberty that the choice of $h_1$ and $h_0$ gives us
- Try to make a bounded expansion descent ($1 \rightarrow k$ instead of $1 \rightarrow q$ descent)
- Automatize the choice of the polynomial expansions like Kleinjung did for the NFS
- Evaluate the efficiency of the different descent to build a portfolio algorithm
About Sparse Matrix Algorithms

A sparse matrix algorithm takes into account the fact that the matrix is sparse and keeps this property during the operations. Usually, they use matrix-vector product operations as they are cheap in this configuration (for a $n \times m$ matrix with at most $p$ elements per row, the cost of one matrix-vector product is $p \times n$).

Some matrix algorithms we used:

- Block Wiedemann Algorithm implemented in `gpulinsolve` from Loria
- Block Lanczos Algorithm implemented in `Magma`
- Gaussian Elimination (not sparse matrix algorithm) when the matrix is small enough