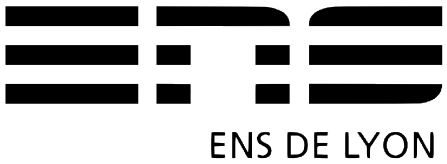
On the discrete logarithm problem

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

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

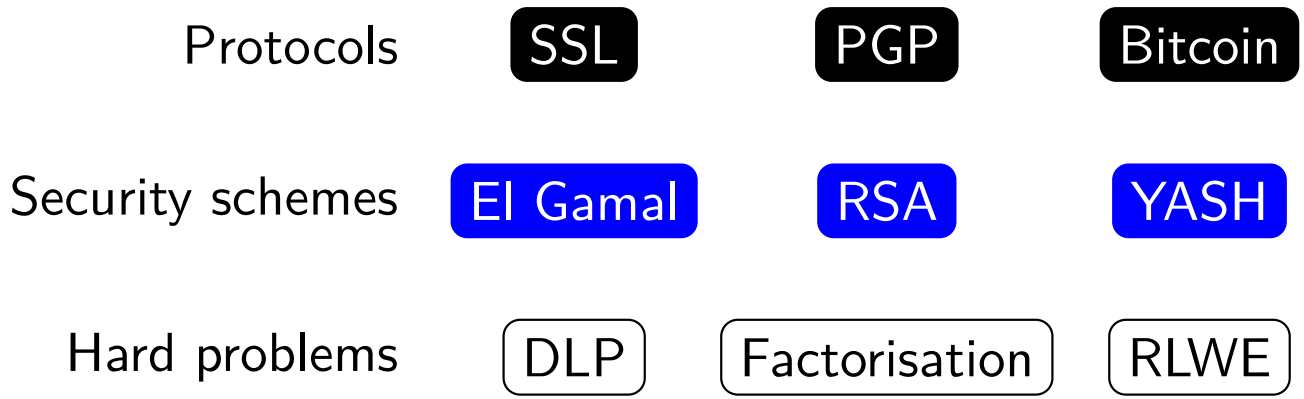
Presentation of Cryptology

Protocols SSL PGP Bitcoin

Presentation of Cryptology

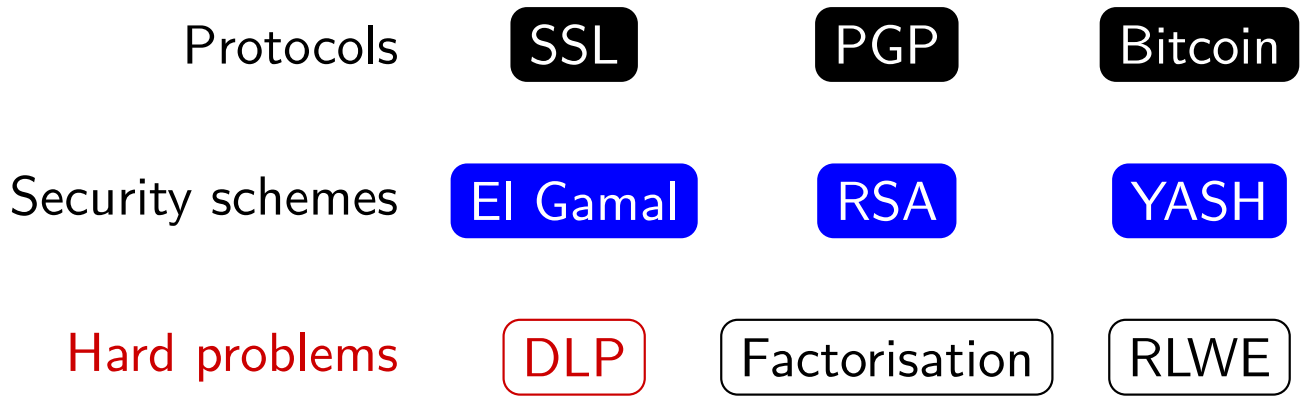
- Cryptology is based on different layers

Presentation of Cryptology



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

Presentation of Cryptology



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

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.

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, $\mathbb{F}_{2^{9234}}$ in 400,000 core hours.

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, $\mathbb{F}_{2^{9234}}$ 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:

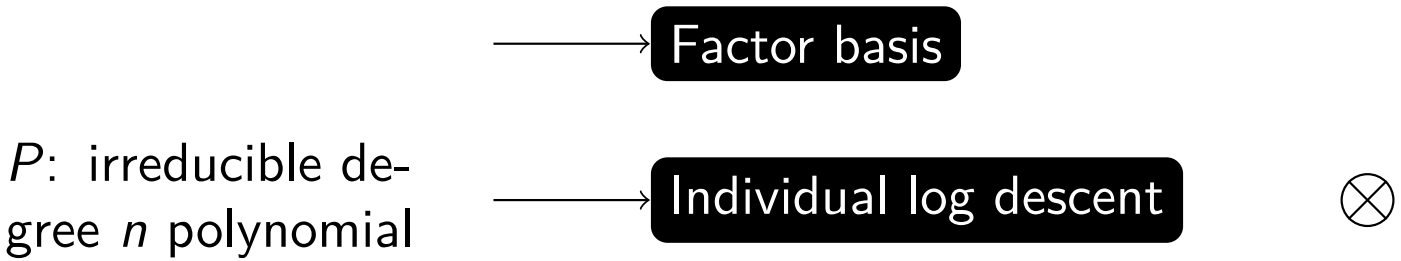
1. The factor basis resolution
2. The individual logarithm descent

The Function Field Sieve

The algorithm can be divided in 2 independent phases:

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

Example:

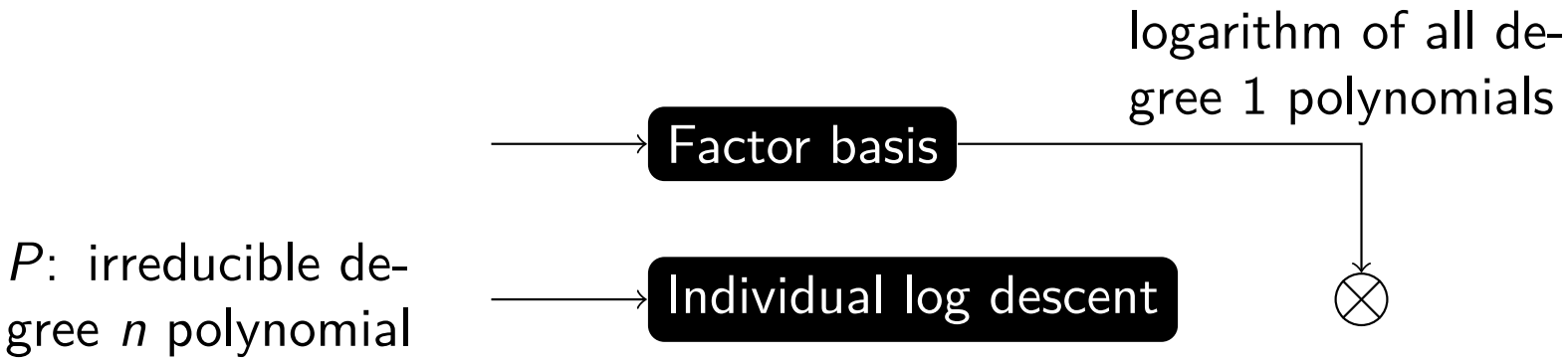


The Function Field Sieve

The algorithm can be divided in 2 independent phases:

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

Example:

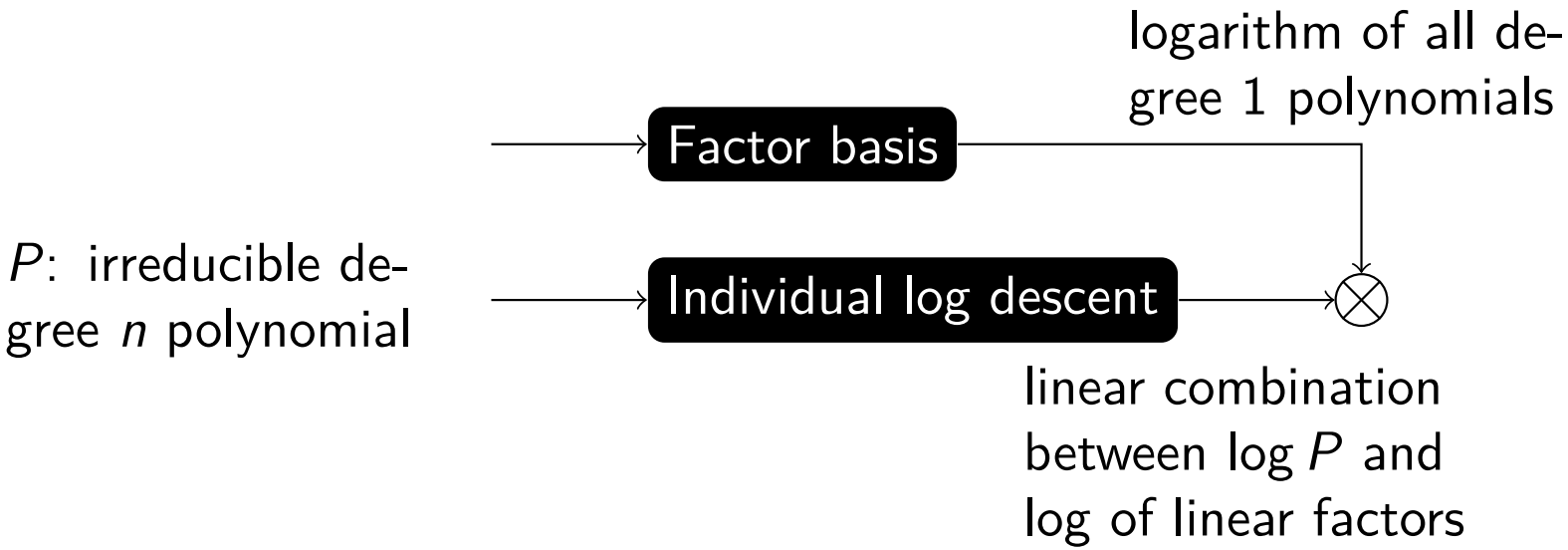


The Function Field Sieve

The algorithm can be divided in 2 independent phases:

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

Example:

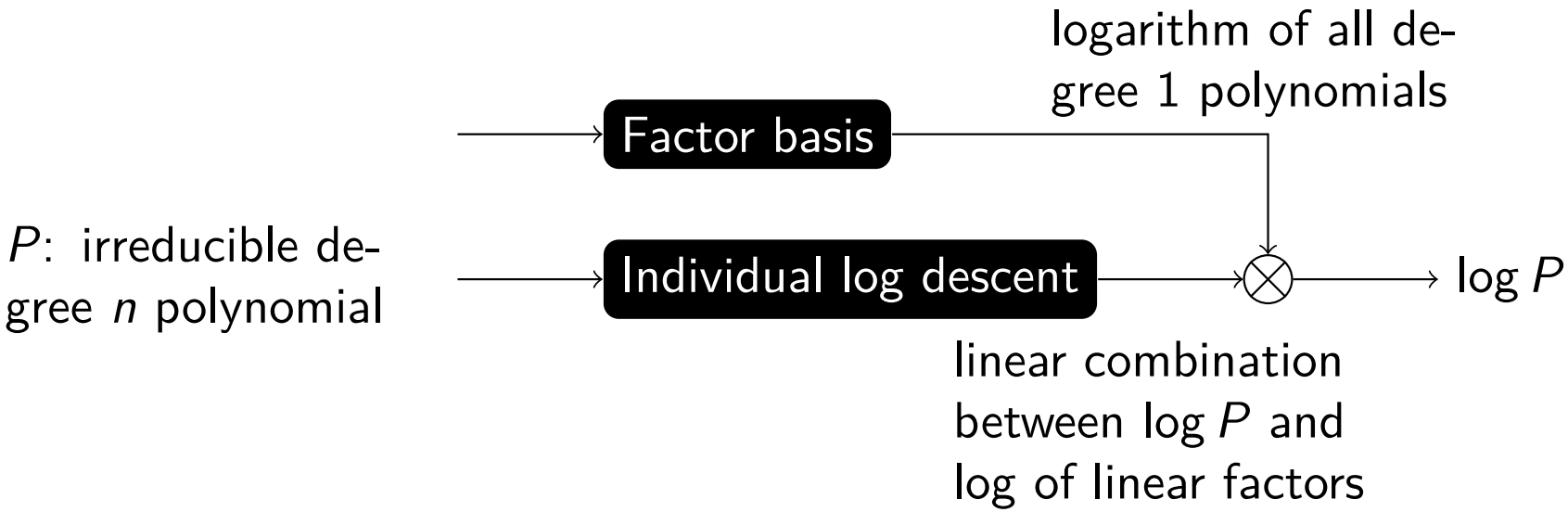


The Function Field Sieve

The algorithm can be divided in 2 independent phases:

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

Example:



Barbulescu, Gaudry, Joux, Thomé

Context

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

Barbulescu, Gaudry, Joux, Thomé

Context

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

Systematic equation

$$X^q - X = \prod_{\alpha \in \mathbb{F}_q} (X - \alpha)$$

Barbulescu, Gaudry, Joux, Thomé

Context

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

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 $\mathbb{F}_{q^{\delta n}} = \mathbb{F}_{q^\delta}[X]_{/(f(X))} \equiv$ 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 Δ

Systematic equation

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

BGJT: Sum up

Heuristic

For q^δ 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^δ 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 $\lceil \frac{\deg P}{2} \rceil$ polynomials.

BGJT: Sum up

Heuristic

For q^δ 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 $\lceil \frac{\deg P}{2} \rceil$ polynomials.

Trade-offs

- We are limited by matrix algorithms over $\mathcal{H}_P$
- We relates $\log P$ with at least q^2 logarithms of smaller polynomials $\rightarrow$ many recursions

Granger, Kleinjung, Zumbrägel

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

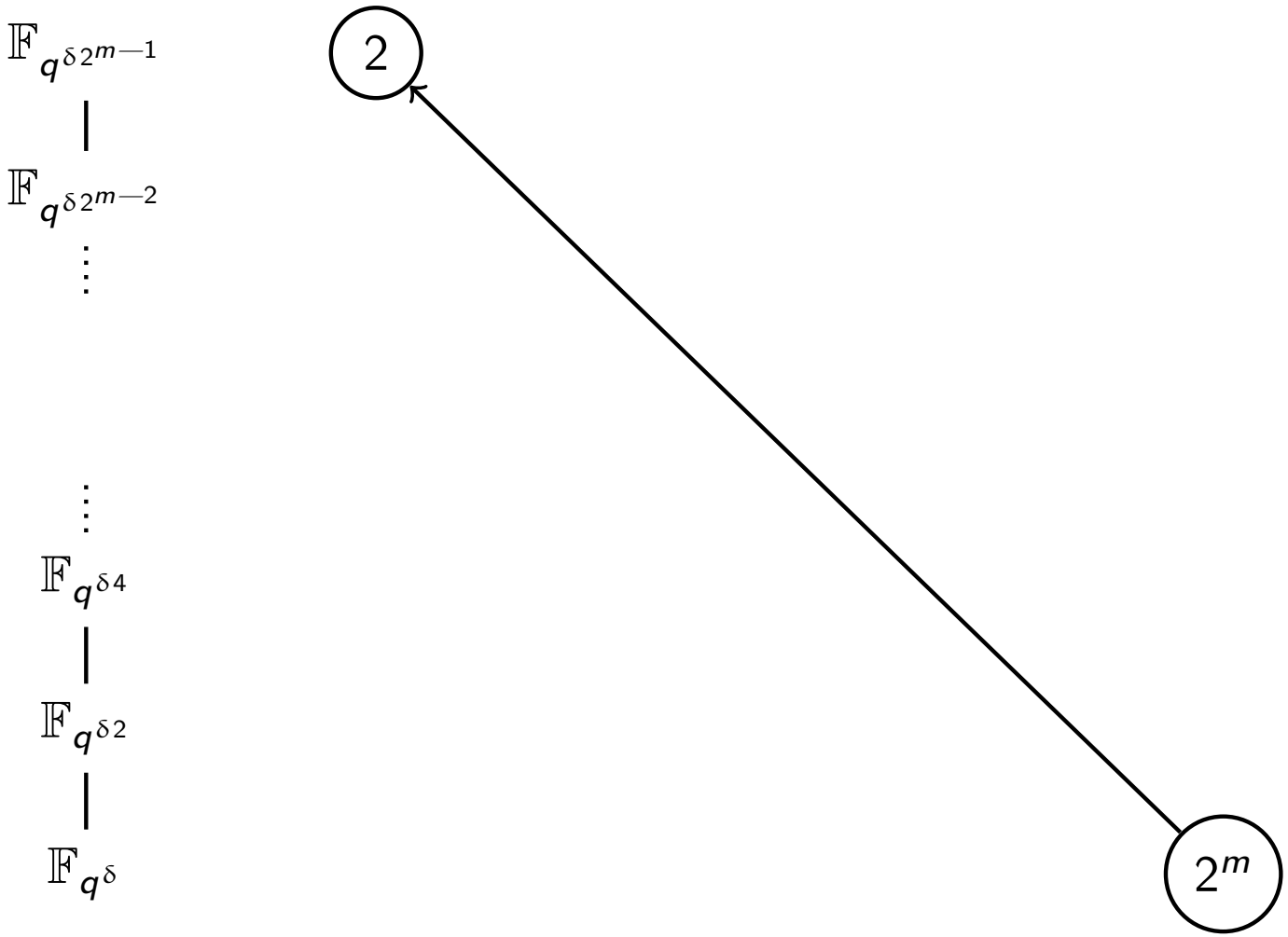
Basic Idea:

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

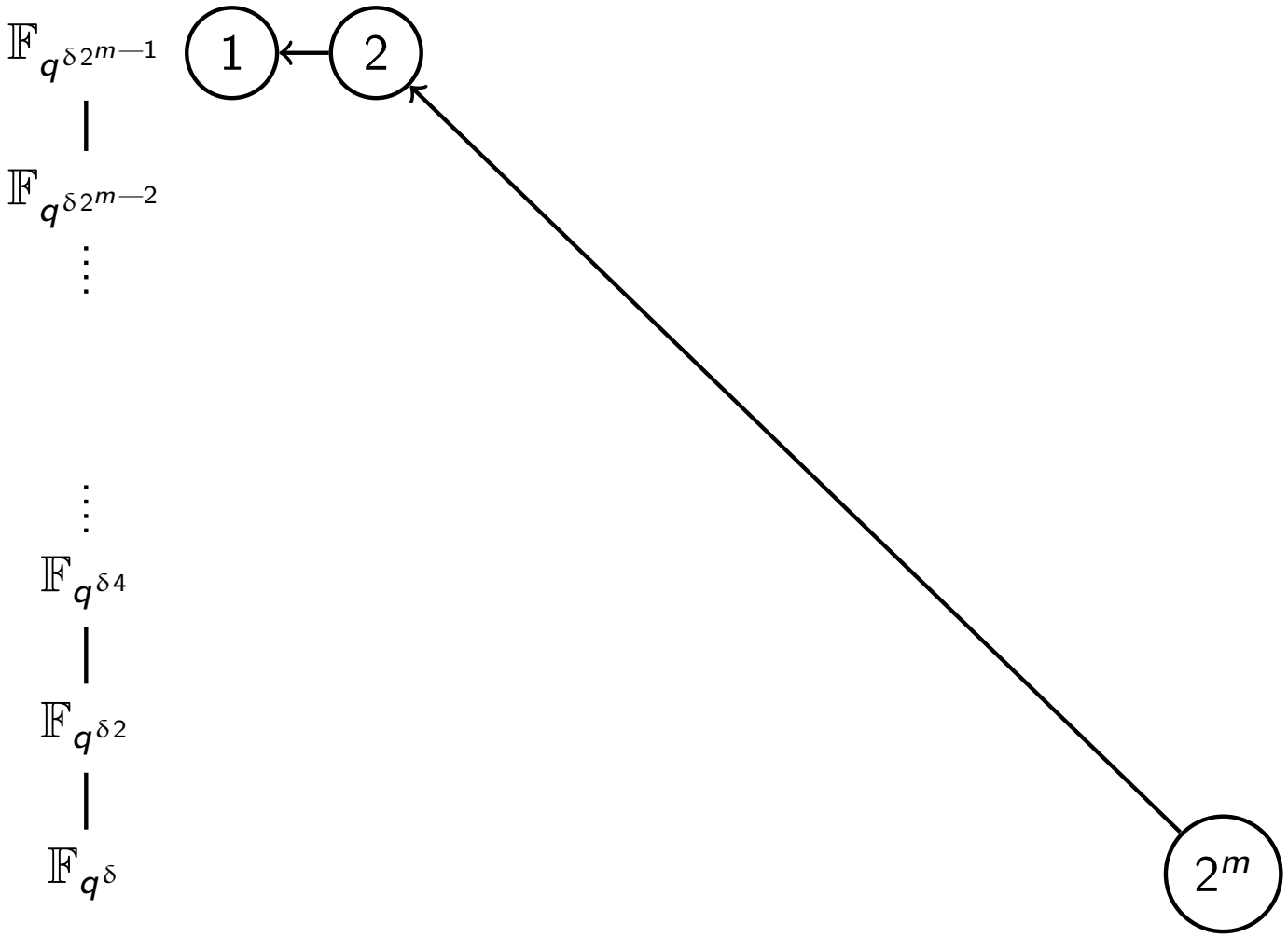
$\Rightarrow$ We know how to express an irreducible degree 2^m polynomials in term of irreducible degree 2^{m-1} polynomials

Degree 2^m Descent

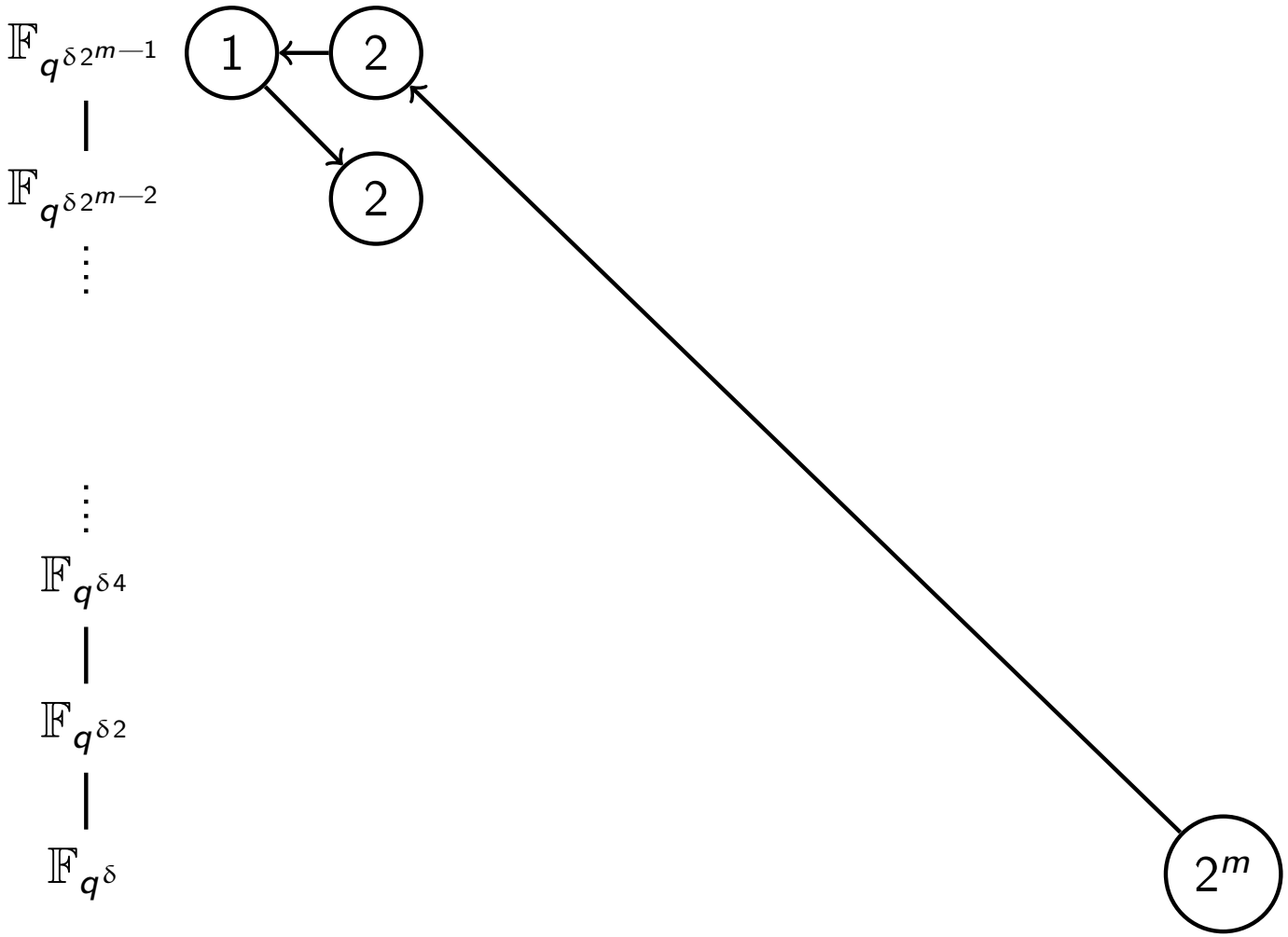
Degree 2^m Descent



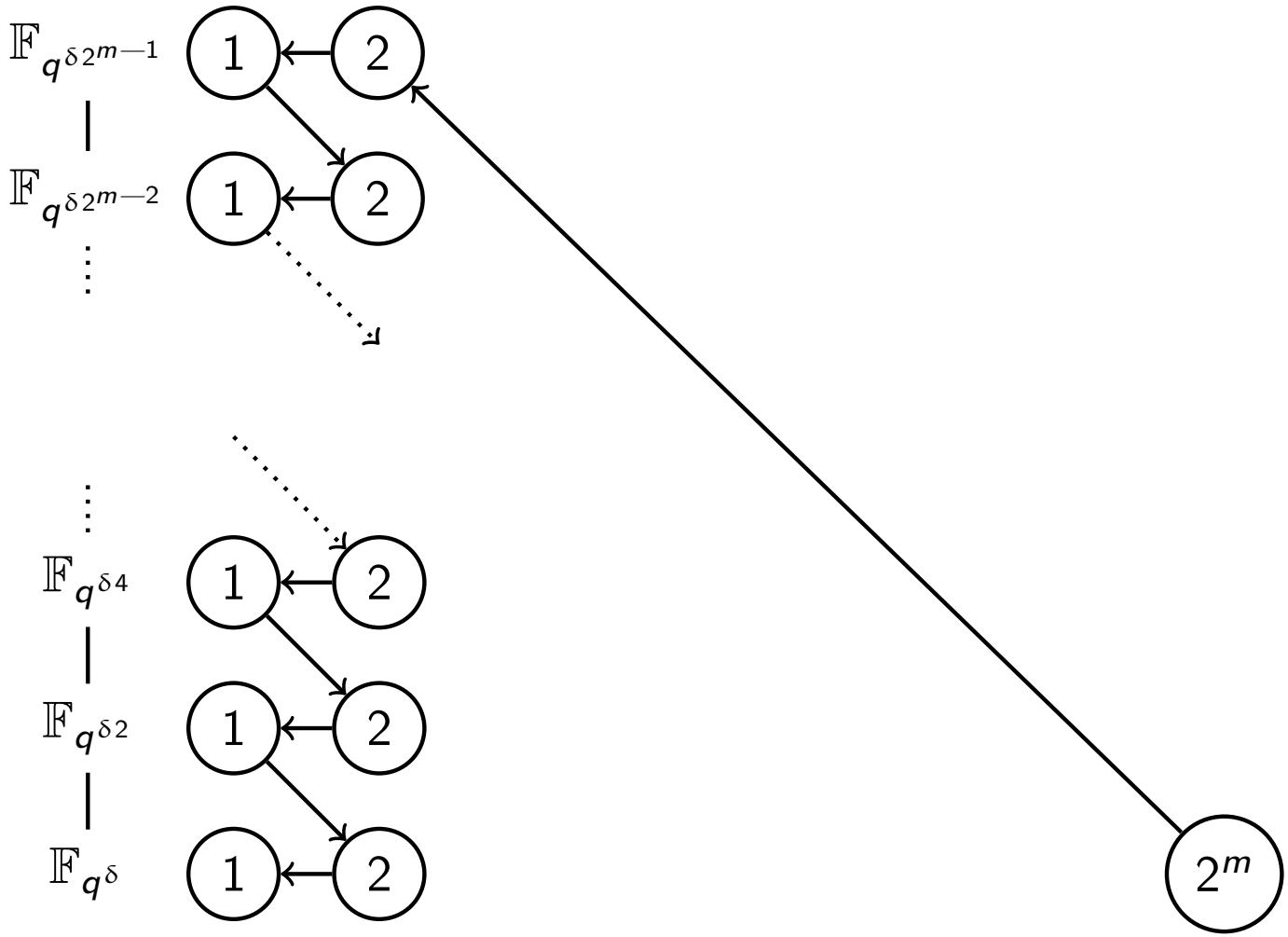
Degree 2^m Descent



Degree 2^m Descent



Degree 2^m 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$\times$ 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× 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

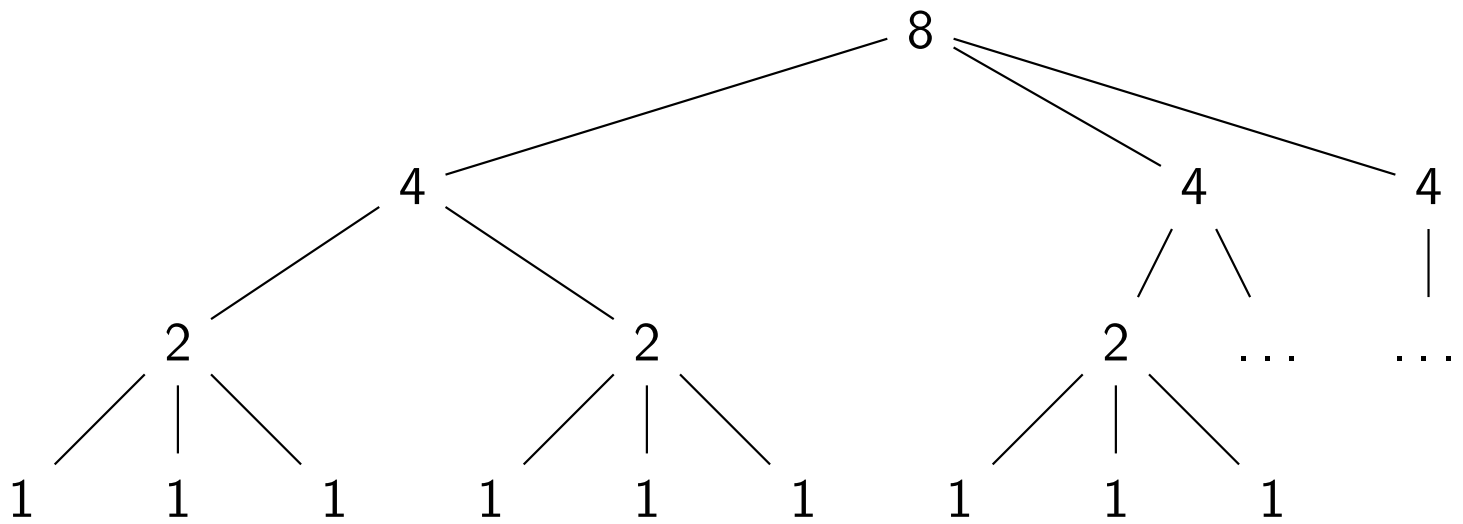

Razvan Barbulescu, Pierrick Gaudry, Antoine Joux, and Emmanuel Thomé.
A quasi-polynomial algorithm for discrete logarithm in finite fields of small characteristic.
Cryptology ePrint Archive, Report 2013/400, 2013.
http://eprint.iacr.org/.


Robert Granger, Thorsten Kleinjung, and Jens Zumbrägel.
On the powers of 2.
Cryptology ePrint Archive, Report 2014/300, 2014.
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 ($1 \to k$ instead of $1 \to q$ 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 take into account the fact that the matrix is sparse and keeps this property during the operations.
Usually they used 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