Discrete Logarithms in Medium Characteristic Finite Fields

Cécile Pierrot [1,2]

[1]Funded by CNRS and DGA
[2]Laboratoire d'Informatique de Paris 6
UPMC, Sorbonne-Universités, France

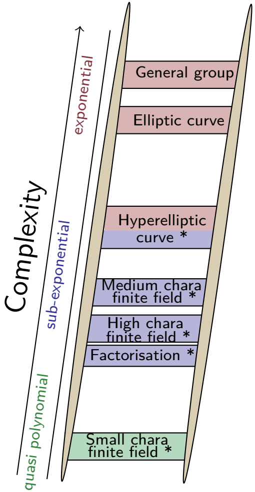
September 28th, 2015
ECC 2015, Bordeaux

The Discrete Logarithm Problem (DLP)

- Multiplicative group G generated by g: solving the DLP in G is inverting the map: $x \mapsto g^x$
- A hard problem in general, and used as such in cryptography.

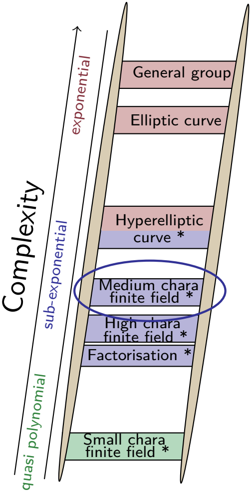
The Discrete Logarithm Problem (DLP)

- Multiplicative group G generated by g: solving the DLP in G is inverting the map: $x \mapsto g^x$
- A hard problem in general, and used as such in cryptography.
- Several groups in practice:



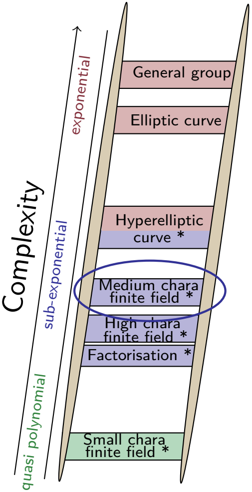
The Discrete Logarithm Problem (DLP)

- Multiplicative group G generated by g: solving the DLP in G is inverting the map: $x \mapsto g^x$
- A hard problem in general, and used as such in cryptography.
- Several groups in practice:



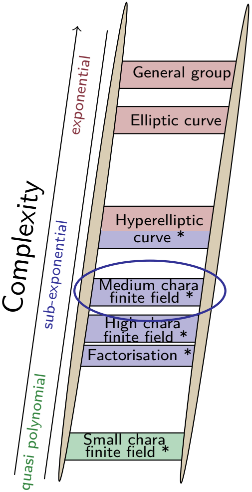
The Discrete Logarithm Problem (DLP)

- Multiplicative group G generated by g: solving the DLP in G is inverting the map: $x \mapsto g^x$
- A hard problem in general, and used as such in cryptography.
- Several groups in practice:
- Two families of algorithms :
  - Generic algorithms (Pollard's Rho, Pohlig-Hellman...)
  - Specific algorithms (Index Calculus *)



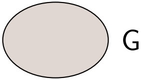
The Discrete Logarithm Problem (DLP)

- Multiplicative group G generated by g: solving the DLP in G is inverting the map: $x \mapsto g^x$
- A hard problem in general, and used as such in cryptography.
- Several groups in practice:
- Two families of algorithms :
  - Generic algorithms (Pollard's Rho, Pohlig-Hellman...)
  - Specific algorithms (Index Calculus *)



Index Calculus Algorithms

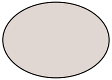
If you want to compute Discrete Logs in G:



1. Relation Collection (or Sieving) Phase

Index Calculus Algorithms

If you want to compute Discrete Logs in G:

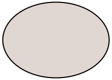

G

1. Relation Collection (or Sieving) Phase
   $\rightarrow$ Create a lot of sparse multiplicative relations between some (small) specific elements = the factor base

$$\prod g_i^{e_i} = \prod g_i^{e_i'}$$

Index Calculus Algorithms

If you want to compute Discrete Logs in G:



G

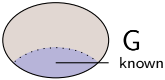
1. Relation Collection (or Sieving) Phase
   $\rightarrow$ Create a lot of sparse multiplicative relations between some (small) specific elements = the factor base

$$\prod g_i^{e_i} = \prod g_i^{e_i'} \quad \Rightarrow \quad \sum (e_i - e_i') \log(g_i) = 0$$

   $\rightarrow$ So a lot of sparse linear equations

Index Calculus Algorithms

If you want to compute Discrete Logs in G:



1. Relation Collection (or Sieving) Phase
   $\rightarrow$ Create a lot of sparse multiplicative relations between some (small) specific elements = the factor base

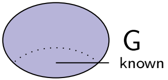
$$\prod g_i^{e_i} = \prod g_i^{e_i'} \quad \Rightarrow \quad \sum (e_i - e_i') \log(g_i) = 0$$

   $\rightarrow$ So a lot of sparse linear equations

2. Linear Algebra
   $\rightarrow$ Recover the Discrete Logs of the factor base

Index Calculus Algorithms

If you want to compute Discrete Logs in G:



1. Relation Collection (or Sieving) Phase
   $\rightarrow$ Create a lot of sparse multiplicative relations between some (small) specific elements = the factor base

$$\prod g_i^{e_i} = \prod g_i^{e_i'} \quad \Rightarrow \quad \sum (e_i - e_i') \log(g_i) = 0$$

   $\rightarrow$ So a lot of sparse linear equations
2. Linear Algebra
   $\rightarrow$ Recover the Discrete Logs of the factor base
3. Individual Logarithm Phase
   $\rightarrow$ Recover the Discrete Log of an arbitrary element

Sieving Phase and Commutative Diagram

- How to obtain relations?

$$\begin{tikzcd} & E \arrow[dl, "u_1"'] \arrow[dr, "u_2"] & \\ E_1 \arrow[dr, "v_1"] & & E_2 \arrow[dl, "v_2"] \\ & G & \end{tikzcd}$$

Sieving Phase and Commutative Diagram

- How to obtain relations?

$$\begin{tikzcd} & E \arrow[dl, "u_1"'] \arrow[dr, "u_2"] & \\ E_1 \arrow[dr, "v_1"] & & E_2 \arrow[dl, "v_2"] \\ & G & \end{tikzcd}$$

$\forall x \in E, v_1(u_1(x)) = v_2(u_2(x))$ thanks to commutativity.

Sieving Phase and Commutative Diagram

- How to obtain relations?

$$\begin{tikzcd} & E \arrow[dl, "u_1"'] \arrow[dr, "u_2"] & \\ B_1 \subset E_1 \arrow[dr, "v_1"] & & E_2 \supset B_2 \arrow[dl, "v_2"'] \\ & G & \end{tikzcd}$$

$\forall x \in E, v_1(u_1(x)) = v_2(u_2(x))$ thanks to commutativity.

- How to obtain "good" relations ?
  - Define B_1 and B_2 two small sets.
    Factor base $:= v_1(B_1) \bigcup v_2(B_2)$

Sieving Phase and Commutative Diagram

- How to obtain relations?

$$\begin{tikzcd}
 & & E \arrow[dl, "u_1"'] \arrow[dr, "u_2"] & & \\
B_1 \subset & E_1 \arrow[dr, "v_1"'] & & E_2 \arrow[dl, "v_2"] & \supset B_2 \\
 & & G & &
\end{tikzcd}$$

$\forall x \in E, v_1(u_1(x)) = v_2(u_2(x))$ thanks to commutativity.

- How to obtain "good" relations ?
  - Define B_1 and B_2 two small sets.
    Factor base $:= v_1(B_1) \bigcup v_2(B_2)$
  - Keep only x such that $u_i(x) = \prod_{b_i \in B_i} b_i$ and get:

$$v_1(u_1(x)) = v_2(u_2(x))$$

Sieving Phase and Commutative Diagram

- How to obtain relations?

$$\begin{tikzcd} & E \arrow[dl, "u_1"'] \arrow[dr, "u_2"] & \\ B_1 \subset E_1 \arrow[dr, "v_1"] & & E_2 \supset B_2 \arrow[dl, "v_2"] \\ & G & \end{tikzcd}$$

$\forall x \in E, v_1(u_1(x)) = v_2(u_2(x))$ thanks to commutativity.

- How to obtain "good" relations ?
  - Define B_1 and B_2 two small sets.
    Factor base $:= v_1(B_1) \bigcup v_2(B_2)$
  - Keep only x such that $u_i(x) = \prod_{b_i \in B_i} b_i$ and get:

$$v_1(\prod_{b_i \in B_2} b_i) = v_2(\prod_{b_i \in B_2} b_i)$$

Sieving Phase and Commutative Diagram

- How to obtain relations?

$$\begin{tikzcd}
& E \arrow[dl, "u_1"'] \arrow[dr, "u_2"] & \\
B_1 \subset E_1 \arrow[dr, "v_1"] & & E_2 \supset B_2 \arrow[dl, "v_2"'] \\
& G &
\end{tikzcd}$$

$\forall x \in E, v_1(u_1(x)) = v_2(u_2(x))$ thanks to commutativity.

- How to obtain "good" relations ?
  - Define B_1 and B_2 two small sets.
    Factor base $:= v_1(B_1) \bigcup v_2(B_2)$
  - Keep only x such that $u_i(x) = \prod_{b_i \in B_i} b_i$ and get:

$$\prod_{b_i \in B_2} v_1(b_i) = \prod_{b_i \in B_2} v_2(b_i) \quad \text{thanks to morphisms.}$$

Number Field Sieve (NFS)

- Solves the DLP for medium and high char. fields $\mathbb{F}_{p^n}$.
- Belongs to the family of Index Calculus algorithms
  $\Rightarrow$ 3 phases.

Number Field Sieve (NFS)

- Solves the DLP for medium and high char. fields $\mathbb{F}_{p^n}$.
- Belongs to the family of Index Calculus algorithms $\Rightarrow$ 3 phases.
- Commutative Diagram ?
  With $m \in \mathbb{F}_{p^n}$ a root of f_1 and f_2 :

$$\begin{tikzcd}
& \mathbb{Z}[X] \arrow[dl, "X \mapsto \theta_1"'] \arrow[dr, "X \mapsto \theta_2"] & \\
\mathbb{Q}[X]/(f_1(X)) \cong \mathbb{Q}(\theta_1) \arrow[dr, "\theta_1 \mapsto m"] & & \mathbb{Q}(\theta_2) \cong \mathbb{Q}[X]/(f_2(X)) \arrow[dl, "\theta_2 \mapsto m"] \\
& \mathbb{F}_{p^n} &
\end{tikzcd}$$

Number Field Sieve (NFS)

- Solves the DLP for medium and high char. fields $\mathbb{F}_{p^n}$.
- Belongs to the family of Index Calculus algorithms
  $\Rightarrow$ 3 phases.
- Commutative Diagram ?
  With $m \in \mathbb{F}_{p^n}$ a root of f_1 and f_2 :

$$\begin{tikzcd}
 & \mathbb{Z}[X] \arrow[dl, "X \mapsto \theta_1"'] \arrow[dr, "X \mapsto \theta_2"] & \\
\mathbb{Q}[X]/(f_1(X)) \cong \mathbb{Q}(\theta_1) \arrow[dr, "\theta_1 \mapsto m"'] & & \mathbb{Q}(\theta_2) \cong \mathbb{Q}[X]/(f_2(X)) \arrow[dl, "\theta_2 \mapsto m"] \\
 & \mathbb{F}_{p^n} &
\end{tikzcd}$$

Factor base ? B_i:= prime ideals (of the ring of integers) with a norm smaller than a certain smoothness* bound.

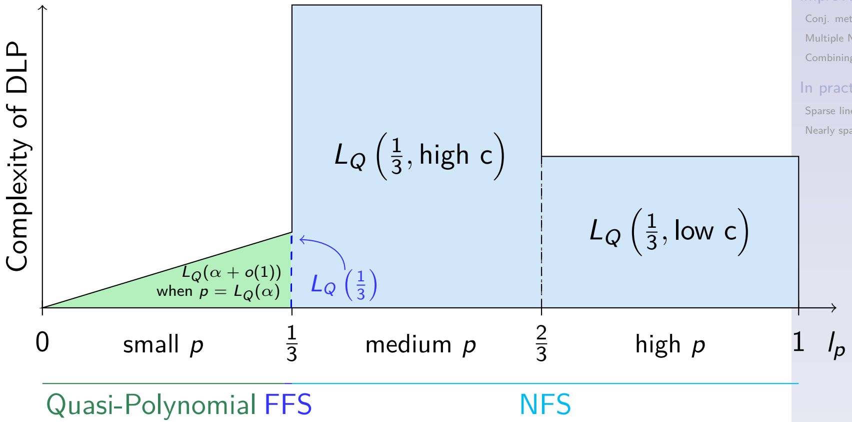
*An ideal $\mathfrak{I}$ is B-smooth if all its factors have norms lower than B.

Complexities

- Notation : $L_Q(\alpha, c) = \exp\left(c(\log Q)^{\alpha}(\log\log Q)^{1-\alpha}\right)$

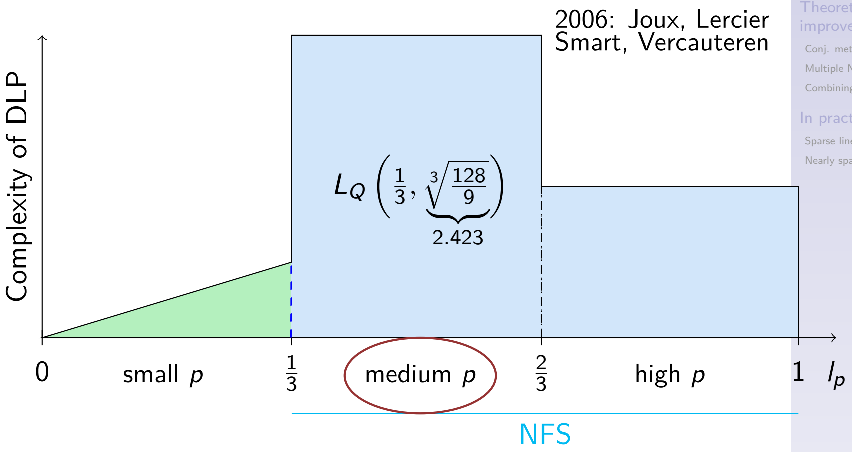
Complexities

- Notation : $L_Q(\alpha, c) = \exp\left(c(\log Q)^{\alpha}(\log\log Q)^{1-\alpha}\right)$
- In $\mathbb{F}_Q$ of characteristic $p = L_Q(l_p, c)$:



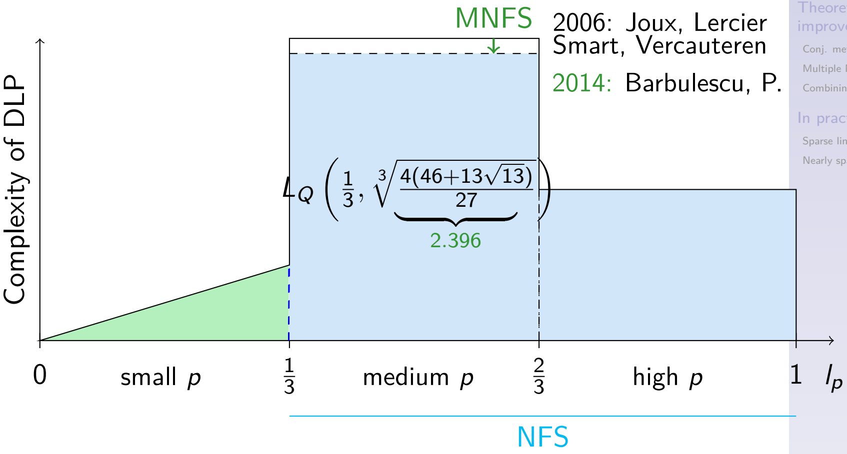
Complexities

- Notation : $L_Q(\alpha, c) = \exp\left(c(\log Q)^{\alpha}(\log\log Q)^{1-\alpha}\right)$
- In $\mathbb{F}_Q$ of characteristic $p = L_Q(l_p, c)$:



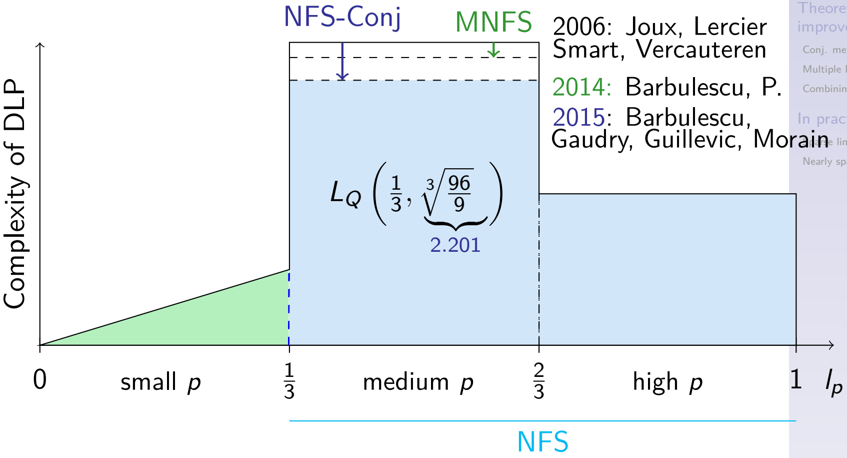
Complexities

- Notation : $L_Q(\alpha, c) = \exp\left(c(\log Q)^{\alpha}(\log\log Q)^{1-\alpha}\right)$
- In $\mathbb{F}_Q$ of characteristic $p = L_Q(l_p, c)$:



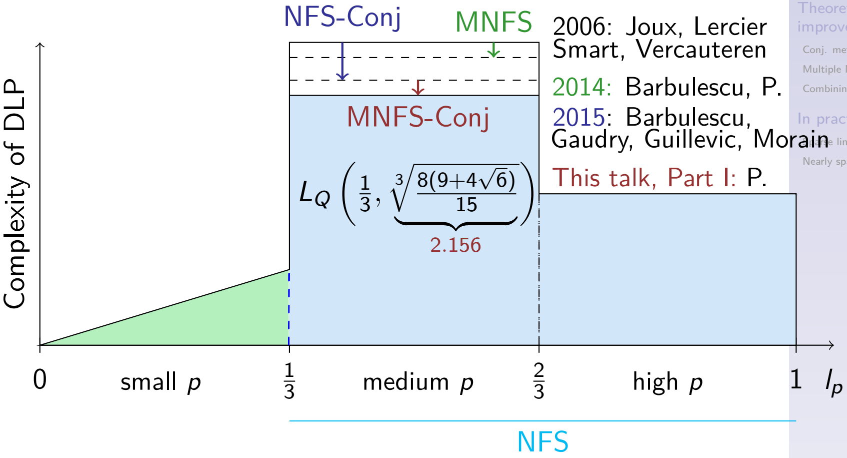
Complexities

- Notation : $L_Q(\alpha, c) = \exp\left(c(\log Q)^{\alpha}(\log\log Q)^{1-\alpha}\right)$
- In $\mathbb{F}_Q$ of characteristic $p = L_Q(l_p, c)$:



Complexities

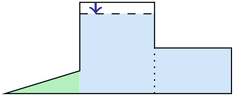
- Notation : $L_Q(\alpha, c) = \exp\left(c(\log Q)^{\alpha}(\log\log Q)^{1-\alpha}\right)$
- In $\mathbb{F}_Q$ of characteristic $p = L_Q(l_p, c)$:



Part I, Asymptotic Complexity downturn: MNFS-Conj

Polynomial Selection

NFS-Conj



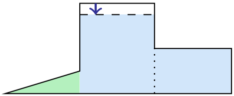
Preliminaries to the diagram:

Find two polynomials f_1 and f_2 with
an irreducible factor $\mathcal{I}$ of degree n modulo p.

- Define $\mathbb{F}_{p^n}$ as $\mathbb{F}_p[X]/(\mathcal{I})$.
- $\Rightarrow f_1$ and f_2 have a common root $m \in \mathbb{F}_{p^n}$.

† $\mathrm{Norm}_{\mathbb{Q}[X]/(f)}(\varphi) = \mathrm{Res}(\varphi, f)$ if f is monic.

Polynomial Selection



Preliminaries to the diagram:

Find two polynomials f_1 and f_2 with an irreducible factor $\mathcal{I}$ of degree n modulo p.

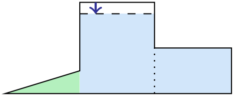
- Define $\mathbb{F}_{p^n}$ as $\mathbb{F}_p[X]/(\mathcal{I})$.
- $\Rightarrow$ f_1 and f_2 have a common root $m \in \mathbb{F}_{p^n}$.

Requirement: Good prob. to obtain a relation

† $\mathrm{Norm}_{\mathbb{Q}[X]/(f)}(\varphi) = \mathrm{Res}(\varphi, f)$ if f is monic.

Polynomial Selection

NFS-Conj



Preliminaries to the diagram:

Find two polynomials f_1 and f_2 with an irreducible factor $\mathcal{I}$ of degree n modulo p.

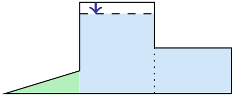
- Define $\mathbb{F}_{p^n}$ as $\mathbb{F}_p[X]/(\mathcal{I})$.
- $\Rightarrow f_1$ and f_2 have a common root $m \in \mathbb{F}_{p^n}$.

Requirement: Good prob. to obtain a relation
$\rightarrow$ Good prob. for a norm to be smooth

$^\dagger \mathrm{Norm}_{\mathbb{Q}[X]/(f)}(\varphi) = \mathrm{Res}(\varphi, f)$ if f is monic.

Polynomial Selection

NFS-Conj



Preliminaries to the diagram:

Find two polynomials f_1 and f_2 with an irreducible factor $\mathcal{I}$ of degree n modulo p.

- Define $\mathbb{F}_{p^n}$ as $\mathbb{F}_p[X]/(\mathcal{I})$.
- $\Rightarrow$ f_1 and f_2 have a common root $m \in \mathbb{F}_{p^n}$.

Requirement: Good prob. to obtain a relation

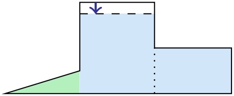
$\rightarrow$ Good prob. for a norm to be smooth

$\rightarrow$ Small norms† in the two number fields

† $\mathrm{Norm}_{\mathbb{Q}[X]/(f)}(\varphi) = \mathrm{Res}(\varphi, f)$ if f is monic.

Polynomial Selection

NFS-Conj



Preliminaries to the diagram:

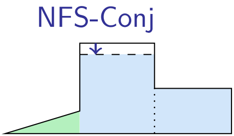
Find two polynomials f_1 and f_2 with
an irreducible factor $\mathcal{I}$ of degree n modulo p.

- Define $\mathbb{F}_{p^n}$ as $\mathbb{F}_p[X]/(\mathcal{I})$.
- $\Rightarrow$ f_1 and f_2 have a common root $m \in \mathbb{F}_{p^n}$.

Requirement: Good prob. to obtain a relation
$\rightarrow$ Good prob. for a norm to be smooth
$\rightarrow$ Small norms† in the two number fields
$\rightarrow$ f_1 and f_2 with not too high degrees and not too large coefficients.

†$\mathrm{Norm}_{\mathbb{Q}[X]/(f)}(\varphi) = \mathrm{Res}(\varphi, f)$ if f is monic.

Polynomial Selection



Preliminaries to the diagram:

Find two polynomials f_1 and f_2 with
an irreducible factor $\mathcal{I}$ of degree n modulo p.

} Polynomial selection

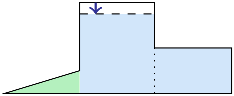
- Define $\mathbb{F}_{p^n}$ as $\mathbb{F}_p[X]/(\mathcal{I})$.
- $\Rightarrow f_1$ and f_2 have a common root $m \in \mathbb{F}_{p^n}$.

Requirement: Good prob. to obtain a relation
$\rightarrow$ Good prob. for a norm to be smooth
$\rightarrow$ Small norms† in the two number fields
$\rightarrow$ f_1 and f_2 with not too high degrees and not too large coefficients.

†$\mathrm{Norm}_{\mathbb{Q}[X]/(f)}(\varphi) = \mathrm{Res}(\varphi, f)$ if f is monic.

Polynomial Selection

NFS-Conj



Preliminaries to the diagram:

Find two polynomials f_1 and f_2 with
an irreducible factor $\mathcal{I}$ of degree n modulo p. } Polynomial selection

- Define $\mathbb{F}_{p^n}$ as $\mathbb{F}_p[X]/(\mathcal{I})$.
- $\Rightarrow f_1$ and f_2 have a common root $m \in \mathbb{F}_{p^n}$.

Requirement: Good prob. to obtain a relation
$\rightarrow$ Good prob. for a norm to be smooth
$\rightarrow$ Small norms† in the two number fields
$\rightarrow$ f_1 and f_2 with not too high degrees and not too large coefficients.

New polynomial selection proposed by Barbulescu, Gaudry, Guillevic and Morain: the Conjugation Method.

† $\mathrm{Norm}_{\mathbb{Q}[X]/(f)}(\varphi) = \mathrm{Res}(\varphi, f)$ if f is monic.

The Conjugation Method

Aim: Find two polynomials f_1 and f_2 with an irreducible factor of degree n modulo p.

The Conjugation Method

Aim: Find two polynomials f_1 and f_2 with an irreducible factor of degree n modulo p.

- **Start** with g_a and $g_b \in \mathbb{Z}[X]$

The Conjugation Method

Aim: Find two polynomials f_1 and f_2 with an irreducible factor of degree n modulo p.

- **Start** with g_a and $g_b \in \mathbb{Z}[X]$
- **Find** u and v small integers such that $X^2 + uX + v$ is:
  - irreducible over $\mathbb{Z}[X]$ but has roots λ and λ' modulo p
  - $g_a + \lambda g_b$ is irreducible modulo p

The Conjugation Method

Aim: Find two polynomials f_1 and f_2 with an irreducible factor of degree n modulo p.

- **Start** with g_a and $g_b \in \mathbb{Z}[X]$
- **Find** u and v small integers such that $X^2 + uX + v$ is:
  - irreducible over $\mathbb{Z}[X]$ but has roots λ and λ' modulo p
  - $g_a + \lambda g_b$ is irreducible modulo p
- **Set** $f_1 = {g_a}^2 - u g_a g_b + v {g_b}^2$. Note that

$$\begin{aligned} f_1 &\equiv {g_a}^2 + (\lambda + \lambda') g_a g_b + \lambda\lambda' {g_b}^2 \mod p \\ &\equiv (g_a + \lambda g_b)(g_a + \lambda' g_b) \mod p \end{aligned}$$

The Conjugation Method

Aim: Find two polynomials f_1 and f_2 with an irreducible factor of degree n modulo p.

- **Start** with g_a and $g_b \in \mathbb{Z}[X]$
- **Find** u and v small integers such that $X^2 + uX + v$ is:
  - irreducible over $\mathbb{Z}[X]$ but has roots λ and λ' modulo p
  - $g_a + \lambda g_b$ is irreducible modulo p
- **Set** $f_1 = {g_a}^2 - u g_a g_b + v {g_b}^2$. Note that
$$\begin{aligned} f_1 &\equiv {g_a}^2 + (\lambda + \lambda') g_a g_b + \lambda\lambda' {g_b}^2 \mod p \\ &\equiv (g_a + \lambda g_b)(g_a + \lambda' g_b) \mod p \end{aligned}$$
- **Rewrite** $\lambda = a/b \mod p$ with $a, b \approx \sqrt{p}$ (continued frac.)

The Conjugation Method

Aim: Find two polynomials f_1 and f_2 with an irreducible factor of degree n modulo p.

- **Start** with g_a and $g_b \in \mathbb{Z}[X]$
- **Find** u and v small integers such that $X^2 + uX + v$ is:
  - irreducible over $\mathbb{Z}[X]$ but has roots λ and λ' modulo p
  - $g_a + \lambda g_b$ is irreducible modulo p
- **Set** $f_1 = {g_a}^2 - u g_a g_b + v {g_b}^2$. Note that
  $$\begin{aligned} f_1 &\equiv {g_a}^2 + (\lambda + \lambda') g_a g_b + \lambda\lambda' {g_b}^2 \mod p \\ &\equiv (g_a + \lambda g_b)(g_a + \lambda' g_b) \mod p \end{aligned}$$
- **Rewrite** $\lambda = a/b \mod p$ with $a, b \approx \sqrt{p}$ (continued frac.)
- **Set** $f_2 = b g_a + a g_b$. Note that $f_2 \equiv g_a + \lambda g_b \mod p$.

The Conjugation Method

Aim: Find two polynomials f_1 and f_2 with an irreducible factor of degree n modulo p.

- **Start** with g_a and $g_b \in \mathbb{Z}[X]$
- **Find** u and v small integers such that $X^2 + uX + v$ is:
  - irreducible over $\mathbb{Z}[X]$ but has roots λ and λ' modulo p
  - $g_a + \lambda g_b$ is irreducible modulo p

n

- **Set** $f_1 = {g_a}^2 - u g_a g_b + v {g_b}^2$. Note that
$$\begin{aligned} f_1 &\equiv {g_a}^2 + (\lambda + \lambda') g_a g_b + \lambda\lambda' {g_b}^2 \mod p \\ &\equiv (g_a + \lambda g_b)(g_a + \lambda' g_b) \mod p \end{aligned}$$
- **Rewrite** $\lambda = a/b \mod p$ with $a, b \approx \sqrt{p}$ (continued frac.)
- **Set** $f_2 = b g_a + a g_b$. Note that $f_2 \equiv g_a + \lambda g_b \mod p$.

Degrees

The Conjugation Method

Aim: Find two polynomials f_1 and f_2 with an irreducible factor of degree n modulo p.

$< n$

- **Start** with g_a and $g_b \in \mathbb{Z}[X]$
- **Find** u and v small integers such that $X^2 + uX + v$ is:
  - irreducible over $\mathbb{Z}[X]$ but has roots λ and λ' modulo p
  - $g_a + \lambda g_b$ is irreducible modulo p

n

- **Set** $f_1 = {g_a}^2 - u g_a g_b + v {g_b}^2$. Note that
$$\begin{aligned} f_1 &\equiv {g_a}^2 + (\lambda + \lambda')g_a g_b + \lambda\lambda' {g_b}^2 \mod p \\ &\equiv (g_a + \lambda g_b)(g_a + \lambda' g_b) \mod p \end{aligned}$$
- **Rewrite** $\lambda = a/b \mod p$ with $a, b \approx \sqrt{p}$ (continued frac.)
- **Set** $f_2 = b g_a + a g_b$. Note that $f_2 \equiv g_a + \lambda g_b \mod p$.

Degrees

The Conjugation Method

Aim: Find two polynomials f_1 and f_2 with an irreducible factor of degree n modulo p.

- $< n$ **Start** with g_a and $g_b \in \mathbb{Z}[X]$
- **Find** u and v small integers such that $X^2 + uX + v$ is:
  - irreducible over $\mathbb{Z}[X]$ but has roots λ and λ' modulo p
  - n $g_a + \lambda g_b$ is irreducible modulo p
- $2n$ **Set** $f_1 = {g_a}^2 - u g_a g_b + v {g_b}^2$. Note that

$$\begin{aligned} f_1 &\equiv {g_a}^2 + (\lambda + \lambda') g_a g_b + \lambda\lambda' {g_b}^2 \mod p \\ &\equiv (g_a + \lambda g_b)(g_a + \lambda' g_b) \mod p \end{aligned}$$

- **Rewrite** $\lambda = a/b \mod p$ with $a, b \approx \sqrt{p}$ (continued frac.)
- n **Set** $f_2 = b g_a + a g_b$. Note that $f_2 \equiv g_a + \lambda g_b \mod p$.

Degrees

The Conjugation Method

Aim: Find two polynomials f_1 and f_2 with an irreducible factor of degree n modulo p.

$< n$

- **Start** with g_a and $g_b \in \mathbb{Z}[X]$
- **Find** u and v small integers such that $X^2 + uX + v$ is:
  - irreducible over $\mathbb{Z}[X]$ but has roots λ and λ' modulo p
  - $g_a + \lambda g_b$ is irreducible modulo p

n

$2n$

- **Set** $f_1 = {g_a}^2 - u g_a g_b + v {g_b}^2$. Note that

$$\begin{aligned} f_1 &\equiv {g_a}^2 + (\lambda + \lambda') g_a g_b + \lambda\lambda' {g_b}^2 \mod p \\ &\equiv (g_a + \lambda g_b)(g_a + \lambda' g_b) \mod p \end{aligned}$$

- **Rewrite** $\lambda = a/b \mod p$ with $a, b \approx \sqrt{p}$ (continued frac.)
- **Set** $f_2 = b g_a + a g_b$. Note that $f_2 \equiv g_a + \lambda g_b \mod p$.

n

Degrees

Coeffs.

The Conjugation Method

Aim: Find two polynomials f_1 and f_2 with an irreducible factor of degree n modulo p.

- $< n$ **Start** with g_a and $g_b \in \mathbb{Z}[X]$
- **Find** u and v small integers such that $X^2 + uX + v$ is:
  - irreducible over $\mathbb{Z}[X]$ but has roots λ and λ' modulo p small
  - n $g_a + \lambda g_b$ is irreducible modulo p
- $2n$ **Set** $f_1 = {g_a}^2 - u g_a g_b + v {g_b}^2$. Note that

$$\begin{aligned} f_1 &\equiv {g_a}^2 + (\lambda + \lambda') g_a g_b + \lambda\lambda' {g_b}^2 \mod p \\ &\equiv (g_a + \lambda g_b)(g_a + \lambda' g_b) \mod p \end{aligned}$$

- **Rewrite** $\lambda = a/b \mod p$ with $a, b \approx \sqrt{p}$ (continued frac.)
- n **Set** $f_2 = b g_a + a g_b$. Note that $f_2 \equiv g_a + \lambda g_b \mod p$.

Degrees

Coeffs.

The Conjugation Method

Aim: Find two polynomials f_1 and f_2 with an irreducible factor of degree n modulo p.

- **Start** with g_a and $g_b \in \mathbb{Z}[X]$ $< n$
- **Find** u and v small integers such that $X^2 + uX + v$ is:
  - irreducible over $\mathbb{Z}[X]$ but has roots λ and λ' modulo p
  - $g_a + \lambda g_b$ is irreducible modulo p n
- **Set** $f_1 = {g_a}^2 - u g_a g_b + v {g_b}^2$. Note that $2n$

$$\begin{aligned} f_1 &\equiv {g_a}^2 + (\lambda + \lambda') g_a g_b + \lambda\lambda' {g_b}^2 \mod p \\ &\equiv (g_a + \lambda g_b)(g_a + \lambda' g_b) \mod p \end{aligned}$$

- **Rewrite** $\lambda = a/b \mod p$ with $a, b \approx \sqrt{p}$ (continued frac.)
- **Set** $f_2 = b g_a + a g_b$. Note that $f_2 \equiv g_a + \lambda g_b \mod p$. n

Degrees

small

Coeffs.

The Conjugation Method

Aim: Find two polynomials f_1 and f_2 with an irreducible factor of degree n modulo p.

- $< n$ **Start** with g_a and $g_b \in \mathbb{Z}[X]$
- **Find** u and v small integers such that $X^2 + uX + v$ is:
  - irreducible over $\mathbb{Z}[X]$ but has roots λ and λ' modulo p — small
  - n — $g_a + \lambda g_b$ is irreducible modulo p
- $2n$ **Set** $f_1 = {g_a}^2 - u g_a g_b + v {g_b}^2$. Note that

$$\begin{aligned} f_1 &\equiv {g_a}^2 + (\lambda + \lambda') g_a g_b + \lambda\lambda' {g_b}^2 \mod p \\ &\equiv (g_a + \lambda g_b)(g_a + \lambda' g_b) \mod p \end{aligned}$$

- **Rewrite** $\lambda = a/b \mod p$ with $a, b \approx \sqrt{p}$ (continued frac.)
- n **Set** $f_2 = b g_a + a g_b$. Note that $f_2 \equiv g_a + \lambda g_b \mod p$. — $\sqrt{p}$

Degrees

Coeffs.

The Conjugation Method

Aim: Find two polynomials f_1 and f_2 with an irreducible factor of degree n modulo p.

- **Start** with g_a and $g_b \in \mathbb{Z}[X]$
- **Find** u and v small integers such that $X^2 + uX + v$ is:
  - irreducible over $\mathbb{Z}[X]$ but has roots λ and λ' modulo p
  - $g_a + \lambda g_b$ is irreducible modulo p
- **Set** $f_1 = {g_a}^2 - u g_a g_b + v {g_b}^2$. Note that
$$\begin{aligned} f_1 &\equiv {g_a}^2 + (\lambda + \lambda') g_a g_b + \lambda\lambda' {g_b}^2 \mod p \\ &\equiv (g_a + \lambda g_b)(g_a + \lambda' g_b) \mod p \end{aligned}$$
- **Rewrite** $\lambda = a/b \mod p$ with $a, b \approx \sqrt{p}$ (continued frac.)
- **Set** $f_2 = b g_a + a g_b$. Note that $f_2 \equiv g_a + \lambda g_b \mod p$.

$2n$

n

Degrees

small

$\sqrt{p}$

Coeffs.

The Conjugation Method

Aim: Find two polynomials f_1 and f_2 with an irreducible factor of degree n modulo p.

- **Start** with g_a and $g_b \in \mathbb{Z}[X]$
- **Find** u and v small integers such that $X^2 + uX + v$ is:
  - irreducible over $\mathbb{Z}[X]$ but has roots λ and λ' modulo p
  - $g_a + \lambda g_b$ is irreducible modulo p

small

$2n$

- **Set** $f_1 = {g_a}^2 - u g_a g_b + v {g_b}^2$. Note that

$$\begin{aligned} f_1 &\equiv {g_a}^2 + (\lambda + \lambda') g_a g_b + \lambda\lambda' {g_b}^2 \mod p \\ &\equiv (g_a + \lambda g_b)(g_a + \lambda' g_b) \mod p \end{aligned}$$

- **Rewrite** $\lambda = a/b \mod p$ with $a, b \approx \sqrt{p}$ (continued frac.)

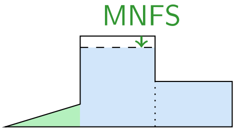
n

- **Set** $f_2 = b g_a + a g_b$. Note that $f_2 \equiv g_a + \lambda g_b \mod p$.

$\sqrt{p}$

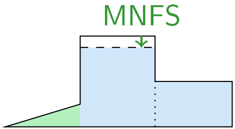
Degrees

Coeffs.



The Multiple Number Field Sieve

- Idea from integer factorization [Coppersmith 93], prime fields [Matyukhin 03], high and medium characteristic [Barbulescu, P. 14].

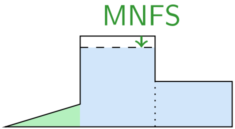


The Multiple Number Field Sieve

- Idea from integer factorization [Coppersmith 93], prime fields [Matyukhin 03], high and medium characteristic [Barbulescu, P. 14].
- With m a common root of $f_1, \ldots, f_V$ in $\mathbb{F}_{p^n}$:

$$\begin{tikzcd}
 & & \mathbb{Z}[X] \arrow[dll] \arrow[dl] \arrow[d, "X \mapsto \theta_i"] \arrow[dr] \arrow[drr] & & \\
\mathbb{Q}(\theta_1) \arrow[drr] & \mathbb{Q}(\theta_2) \arrow[dr] & \mathbb{Q}(\theta_i) \arrow[d, "\theta_i \mapsto m"] & \mathbb{Q}(\theta_{V-1}) \arrow[dl] & \mathbb{Q}(\theta_V) \arrow[dll] \\
 & & \mathbb{F}_{p^n} & &
\end{tikzcd}$$

The **Multiple** Number Field Sieve



- Idea from integer factorization [Coppersmith 93], prime fields [Matyukhin 03], high and medium characteristic [Barbulescu, P. 14].
- With m a common root of $f_1, \dots, f_V$ in $\mathbb{F}_{p^n}$:

$$
\begin{tikzcd}
 & & \mathbb{Z}[X] \arrow[dll] \arrow[dl] \arrow[d, "X \mapsto \theta_i"] \arrow[dr] \arrow[drr] & & \\
\mathbb{Q}(\theta_1) \arrow[drr] & \mathbb{Q}(\theta_2) \arrow[dr] & \mathbb{Q}(\theta_i) \arrow[d, "\theta_i \mapsto m"] & \mathbb{Q}(\theta_{V-1}) \arrow[dl] & \mathbb{Q}(\theta_V) \arrow[dll] \\
 & & \mathbb{F}_{p^n} & &
\end{tikzcd}
$$

- Choice of poly. f_1 and f_2 with a common root m in $\mathbb{F}_{p^n}$
  $\Rightarrow$ linear combination of f_1 and f_2
  $\Rightarrow$ for $i = 3, \dots, V$: $f_i = \alpha_i f_1 + \beta_i f_2$ with $\alpha_i, \beta_i \approx \sqrt{V}$.

Dissymetric MNFS in one slide

Dissymmetric = when a polynomial is better than the other.

- E.g: f_1, f_2 have same coeff. size but $\deg f_2 \geqslant \deg f_1$

Dissymetric MNFS in one slide

Dissymmetric = when a polynomial is better than the other.

- E.g: f_1, f_2 have same coeff. size but deg $f_2 \geqslant$ deg f_1
  $\Rightarrow$ Higher norms in $\mathbb{Q}(\theta_2), \ldots, \mathbb{Q}(\theta_V)$ than in $\mathbb{Q}(\theta_1)$.

Dissymetric MNFS in one slide

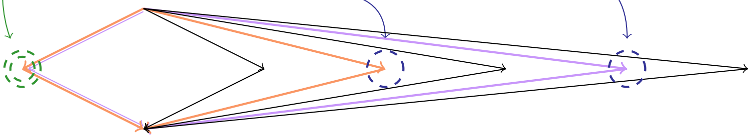
Dissymmetric = when a polynomial is better than the other.

- E.g: f_1, f_2 have same coeff. size but $\deg f_2 \geqslant \deg f_1$
  $\Rightarrow$ Higher norms in $\mathbb{Q}(\theta_2), \ldots, \mathbb{Q}(\theta_V)$ than in $\mathbb{Q}(\theta_1)$.
- Sieving: keep only polynomials that lead to a B-smooth norm in the first number field and a B'-smooth norm in (at least) one other number field.

Dissymetric MNFS in one slide

Dissymmetric = when a polynomial is better than the other.

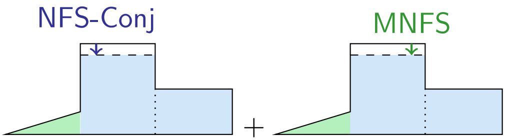
- E.g: f_1, f_2 have same coeff. size but deg $f_2 \geqslant$ deg f_1 $\Rightarrow$ Higher norms in $\mathbb{Q}(\theta_2), \ldots, \mathbb{Q}(\theta_V)$ than in $\mathbb{Q}(\theta_1)$.
- Sieving: keep only polynomials that lead to a B-smooth norm in the first number field and a B'-smooth norm in (at least) one other number field.



Our aim is to combine:

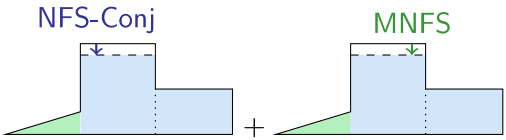
Our aim is to combine:

- the Conjugation Method
- with MNFS.

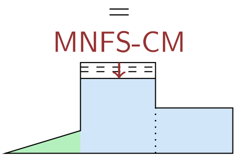


Our aim is to combine:

- the Conjugation Method
- with MNFS.



$\Rightarrow$ Best algorithm to solve the DLP in medium characteristic finite fields $\mathbb{F}_{p^n}$.



Obstruction and Dreams

Conj produces:

- f_1 with high degree, small coefficients
- f_2 with small degree, high coefficients

Obstruction and Dreams

Conj produces:

- f_1 with high degree, small coefficients
- f_2 with small degree, high coefficients
- $\Rightarrow$ Linear combinations of f_1 and f_2 would have both inconveniences: high degrees and high coefficients. 

Obstruction and Dreams

Conj produces:

- f_1 with high degree, small coefficients
- f_2 with small degree, high coefficients
- $\Rightarrow$ Linear combinations of f_1 and f_2 would have both inconveniences: high degrees and high coefficients. 

Our main idea:

- Linear combinations of f_1 and f_2

Obstruction and Dreams

Conj produces:

- f_1 with high degree, small coefficients
- f_2 with small degree, high coefficients
- $\Rightarrow$ Linear combinations of f_1 and f_2 would have both inconveniences: high degrees and high coefficients. 

Our main idea:

- Linear combinations of ~~f_1 and~~ f_2 and another poly. f_3

Obstruction and Dreams

Conj produces:

- f_1 with high degree, small coefficients
- f_2 with small degree, high coefficients
- $\Rightarrow$ Linear combinations of f_1 and f_2 would have both inconveniences: high degrees and high coefficients. 

Our main idea:

- Linear combinations of ~~f_1 and~~ f_2 and another poly. f_3
- What was the f_3 of my dreams ?

  f_3 with small degree, high coefficients

  $+$ Shares the same common root m

  $+$ Independent from f_2 over $\mathbb{Q}$

Obstruction and Dreams

Conj produces:

- f_1 with high degree, small coefficients
- f_2 with small degree, high coefficients
- $\Rightarrow$ Linear combinations of f_1 and f_2 would have both inconveniences: high degrees and high coefficients. 

Our main idea:

- Linear combinations of ~~f_1 and~~ f_2 and another poly. f_3
- What was the f_3 of my dreams ?
  f_3 with small degree, high coefficients
  $+$ Shares the same common root m
  $+$ Independent from f_2 over $\mathbb{Q}$
- $\Rightarrow$ Linear combinations of f_2 and f_3 have small degrees and high coefficients. 

Obstruction and Dreams

Conj produces:

- f_1 with high degree, small coefficients
- f_2 with small degree, high coefficients
- $\Rightarrow$ Linear combinations of f_1 and f_2 would have both inconveniences: high degrees and high coefficients. 

Our main idea:

- Linear combinations of ~~f_1 and~~ f_2 and another poly. f_3
- What was the f_3 of my dreams ?

  f_3 with small degree, high coefficients
  $+$ Shares the same common root m
  $+$ Independent from f_2 over $\mathbb{Q}$

  $\}$ How to catch it ?

- $\Rightarrow$ Linear combinations of f_2 and f_3 have small degrees and high coefficients. 

Catching f_3 in the Conjugation Method

- **Start** with g_a and $g_b \in \mathbb{Z}[X]$
- **Find** u and v small integers such that $X^2 + uX + v$ is:
  - irreducible over $\mathbb{Z}[X]$ but has roots λ and λ' modulo p
  - $g_a + \lambda g_b$ is irreducible modulo p
- **Set** $f_1 = {g_a}^2 - u g_a g_b + v {g_b}^2$. Note that

$$\begin{aligned} f_1 &\equiv {g_a}^2 + (\lambda+\lambda')g_a g_b + \lambda\lambda' {g_b}^2 \mod p \\ &\equiv (g_a + \lambda g_b)(g_a + \lambda' g_b) \mod p \end{aligned}$$

small

$2n$

- **Rewrite** $\lambda = a/b \mod p$ with $a, b \approx \sqrt{p}$ (continued frac.)
- **Set** $f_2 = b g_a + a g_b$. Note that $f_2 \equiv g_a + \lambda g_b \mod p$.

n $\sqrt{p}$

Degrees

Coeffs.

Catching f_3 in the Conjugation Method

- **Start** with g_a and $g_b \in \mathbb{Z}[X]$
- **Find** u and v small integers such that $X^2 + uX + v$ is:
  - irreducible over $\mathbb{Z}[X]$ but has roots λ and λ' modulo p
  - $g_a + \lambda g_b$ is irreducible modulo p
- **Set** $f_1 = {g_a}^2 - u g_a g_b + v {g_b}^2$. Note that
$$\begin{aligned} f_1 &\equiv {g_a}^2 + (\lambda + \lambda')g_a g_b + \lambda\lambda' {g_b}^2 \mod p \\ &\equiv (g_a + \lambda g_b)(g_a + \lambda' g_b) \mod p \end{aligned}$$
- **Rewrite** $\lambda = a/b \mod p$ with $a, b \approx \sqrt{p}$ (continued frac.)
- **Set** $f_2 = b g_a + a g_b$. Note that $f_2 \equiv g_a + \lambda g_b \mod p$.

n

Degrees

$\sqrt{p}$

Coeffs.

Catching f_3 in the Conjugation Method

- **Start** with g_a and $g_b \in \mathbb{Z}[X]$
- **Find** u and v small integers such that $X^2 + uX + v$ is:
  - irreducible over $\mathbb{Z}[X]$ but has roots λ and λ' modulo p
  - $g_a + \lambda g_b$ is irreducible modulo p
- **Set** $f_1 = {g_a}^2 - u g_a g_b + v {g_b}^2$. Note that
$$f_1 \equiv {g_a}^2 + (\lambda + \lambda') g_a g_b + \lambda\lambda' {g_b}^2 \mod p$$
$$\equiv (g_a + \lambda g_b)(g_a + \lambda' g_b) \mod p$$
- **Rewrite** $\lambda = a/b \mod p$ with $a, b \approx \sqrt{p}$ (continued frac.) **and** $\lambda = a'/b' \mod p$ with $a', b' \approx \sqrt{p}$
- **Set** $f_2 = b g_a + a g_b$. Note that $f_2 \equiv g_a + \lambda g_b \mod p$.

n

Degrees

$\sqrt{p}$

Coeffs.

Catching f_3 in the Conjugation Method

- **Start** with g_a and $g_b \in \mathbb{Z}[X]$
- **Find** u and v small integers such that $X^2 + uX + v$ is:
  - irreducible over $\mathbb{Z}[X]$ but has roots λ and λ' modulo p
  - $g_a + \lambda g_b$ is irreducible modulo p
- **Set** $f_1 = {g_a}^2 - u g_a g_b + v {g_b}^2$. Note that
$$\begin{aligned} f_1 &\equiv {g_a}^2 + (\lambda + \lambda') g_a g_b + \lambda\lambda' {g_b}^2 \mod p \\ &\equiv (g_a + \lambda g_b)(g_a + \lambda' g_b) \mod p \end{aligned}$$
- **Rewrite** $\lambda = a/b \mod p$ with $a, b \approx \sqrt{p}$ (continued frac.) **and** $\lambda = a'/b' \mod p$ with $a', b' \approx \sqrt{p}$
- **Set** $f_2 = b g_a + a g_b$. Note that $f_2 \equiv g_a + \lambda g_b \mod p$. **and** $f_3 = b' g_a + a' g_b$

n $\sqrt{p}$

n $\sqrt{p}$

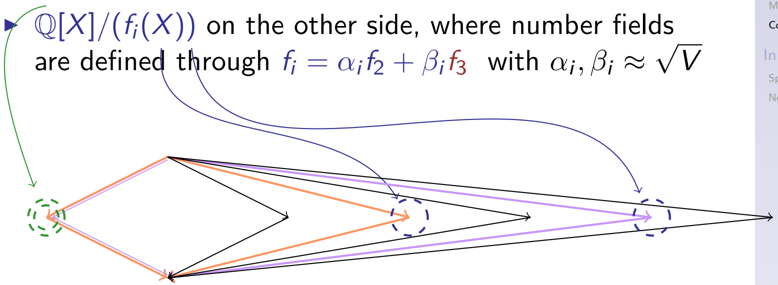
Degrees

Coeffs.

And then ?

Construct a Multiple NFS thanks to:

- $\mathbb{Q}[X]/(f_1(X))$ on one side
- $\mathbb{Q}[X]/(f_i(X))$ on the other side, where number fields are defined through $f_i = \alpha_i f_2 + \beta_i f_3$ with $\alpha_i, \beta_i \approx \sqrt{V}$



Asymptotic Complexity Analysis

The idea is classical:

1. Choose parameters of size:
   - Sieving space : $L_Q(1/3)$
   - Smoothness bounds B and B': $L_Q(1/3)$
   - Number of number fields V: $L_Q(1/3)$

Asymptotic Complexity Analysis

The idea is classical:

1. Choose parameters of size:
   - Sieving space : $L_Q(1/3)$
   - Smoothness bounds B and B': $L_Q(1/3)$
   - Number of number fields V: $L_Q(1/3)$
2. Runtime of the sieving $\approx$ cost of the linear algebra.
3. Size of the factor base $\approx$ number of equations created (i.e. the probability to obtain a good relation multiplied by the sieving space).

Asymptotic Complexity Analysis

The idea is classical:

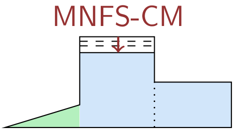
1. Choose parameters of size:
   - Sieving space : $L_Q(1/3)$
   - Smoothness bounds B and B': $L_Q(1/3)$
   - Number of number fields V: $L_Q(1/3)$
2. Runtime of the sieving $\approx$ cost of the linear algebra.
3. Size of the factor base $\approx$ number of equations created (i.e. the probability to obtain a good relation multiplied by the sieving space).
4. Optimize the total runtime under these constraints.

Asymptotic Complexity Analysis

The idea is classical:

1. Choose parameters of size:
   - Sieving space : $L_Q(1/3)$
   - Smoothness bounds B and B': $L_Q(1/3)$
   - Number of number fields V: $L_Q(1/3)$
2. Runtime of the sieving $\approx$ cost of the linear algebra.
3. Size of the factor base $\approx$ number of equations created (i.e. the probability to obtain a good relation multiplied by the sieving space).
4. Optimize the total runtime under these constraints.

$$\Rightarrow L_Q\left(\frac{1}{3}, \sqrt[3]{\frac{8(9+4\sqrt{6})}{15}}\right)$$



Concrete impact

Complexity $\searrow$ from $L_Q(1/3, 2.201)$ to $L_Q(1/3, 2.156)$.
Is it a lot?

- $\ell \leftarrow$ security level we need
  $Q \leftarrow$ order of the associated target finite field.
  With previous algorithms: $\ell = L_Q(1/3, 2.201)$.
- Now, To get Q' such that $\ell = L_{Q'}(1/3, 2.156)$ we need:
  - $(2.156)^3 \log Q' (\log\log Q')^2 = (2.201)^3 \log Q (\log\log Q)^2$
  - so $\log Q' (\log\log Q')^2 \approx 1.064 \log Q (\log\log Q)^2$
  - it yields $\log Q' \approx 1.064 \log Q$.

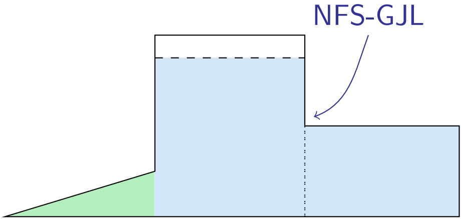
Concrete impact

Complexity $\searrow$ from $L_Q(1/3, 2.201)$ to $L_Q(1/3, 2.156)$.
Is it a lot?

- $\ell \leftarrow$ security level we need
  $Q \leftarrow$ order of the associated target finite field.
  With previous algorithms: $\ell = L_Q(1/3, 2.201)$.
- Now, To get Q' such that $\ell = L_{Q'}(1/3, 2.156)$ we need:
  - $(2.156)^3 \log Q'(\log\log Q')^2 = (2.201)^3 \log Q(\log\log Q)^2$
  - so $\log Q'(\log\log Q')^2 \approx 1.064 \log Q(\log\log Q)^2$
  - it yields $\log Q' \approx 1.064 \log Q$.

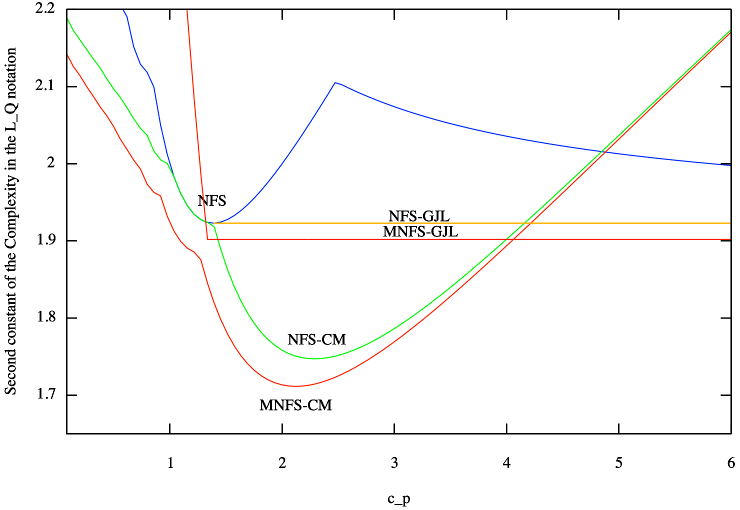
$\Rightarrow$ Increase the bitsize of the finite field by 6.4% to get the same security level.

- the Generalized Joux-Lercier Method [BGGM 15]
- with MNFS.



$$p = L_{p^n}(2/3, c_p)$$

Complexities at $p = L_{p^n}(2/3, c_p)$

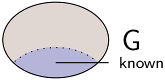


Part II, Practical improvement: Nearly Sparse Linear Algebra.

A joint work with Antoine Joux.

Index Calculus Algorithms

If you want to compute Discrete Logs in G:



1. Collection of Relations (or Sieving Phase)
   $\rightarrow$ Create a lot of sparse multiplicative relations between some (small) specific elements = the factor base

$$\prod g_i^{e_i} = \prod g_i^{e'_i} \quad \Rightarrow \quad \sum (e_i - e'_i) \log(g_i) = 0$$

   $\rightarrow$ So a lot of sparse linear equations
2. Linear Algebra
   $\rightarrow$ Recover the Discrete Logs of the factor base
3. Individual Logarithm Phase
   $\rightarrow$ Recover the Discrete Log of an arbitrary element

Linear Algebra and Index Calculus

- Matrix over finite sets.
- Sparse matrices = the major part of the entries = 0. Often: nbr of non zero coeffs per row is bounded by a constant, let us say K.

Some famous examples

- Factoring. Seek for a non trivial elt of the kernel of a matrix mod 2.
- Discrete log. Last records in small charac. for instance.

Advantages ?

- Less memory
- Specific algorithms

Sparse Linear Algebra

Sparse Linear Algebra

How to use less memory: for any non zero coeff. in a row, let memorize its column number and its value together.

Example

With $\mathbb{F}_7$ and $K = 3$.

$$M = \begin{pmatrix} 1 & 0 & 0 & 0 & 3 & 0 & 0 & 2 \\ 2 & 0 & 1 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 5 & 0 & 0 & 0 & 0 & 2 \\ 0 & 5 & 0 & 0 & 0 & 6 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 1 & 0 & 0 \\ 0 & 3 & 0 & 1 & 0 & 0 & 2 & 0 \end{pmatrix}$$

Sparse Linear Algebra

How to use less memory: for any non zero coeff. in a row, let memorize its column number and its value together.

Example

With $\mathbb{F}_7$ and $K = 3$.

$$M = \begin{pmatrix} 1 & 0 & 0 & 0 & 3 & 0 & 0 & 2 \\ 2 & 0 & 1 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 5 & 0 & 0 & 0 & 0 & 2 \\ 0 & 5 & 0 & 0 & 0 & 6 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 1 & 0 & 0 \\ 0 & 3 & 0 & 1 & 0 & 0 & 2 & 0 \end{pmatrix} \rightarrow \begin{pmatrix} [1,1] & [5,3] & [8,2] \\ [1,2] & [3,1] & [6,2] \\ [4,4] & [7,1] & [0,0] \\ [4,3] & [8,1] & [0,0] \\ [1,1] & [2,1] & [0,0] \\ [3,5] & [8,2] & [0,0] \\ [2,5] & [6,6] & [0,0] \\ [5,2] & [6,1] & [0,0] \\ [2,3] & [4,1] & [7,1] \end{pmatrix}$$

Sparse Linear Algebra, Naive method

We want to solve $Mx = 0$.
Let us manage a simple Gaussian Elimination.

$$\left(\begin{array}{cccccccc|c}
1 & 0 & 0 & 0 & 3 & 0 & 0 & 2 & 0 \\
2 & 0 & 1 & 0 & 0 & 2 & 0 & 0 & 0 \\
0 & 0 & 0 & 4 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 3 & 0 & 0 & 0 & 1 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 5 & 0 & 0 & 0 & 0 & 2 & 0 \\
0 & 5 & 0 & 0 & 0 & 6 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 2 & 1 & 0 & 0 & 0 \\
0 & 3 & 0 & 1 & 0 & 0 & 2 & 0 & 0
\end{array}\right) \rightarrow \left(\begin{array}{ccc|c}
[1,1] & [5,3] & [8,2] & 0 \\
[1,2] & [3,1] & [6,2] & 0 \\
[4,4] & [7,1] & [0,0] & 0 \\
[4,3] & [8,1] & [0,0] & 0 \\
[1,1] & [2,1] & [0,0] & 0 \\
[3,5] & [8,2] & [0,0] & 0 \\
[2,5] & [6,6] & [0,0] & 0 \\
[5,2] & [6,1] & [0,0] & 0 \\
[2,3] & [4,1] & [7,1] & 0
\end{array}\right)$$

Sparse Linear Algebra, Naive method

We want to solve $Mx = 0$.
Let us manage a simple Gaussian Elimination.

$$\left(\begin{array}{cccccccc|c} 1 & 0 & 0 & 0 & 3 & 0 & 0 & 2 & 0 \\ 0 & 0 & 1 & 0 & 1 & 2 & 0 & 2 & 4 \\ 0 & 0 & 0 & 4 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 5 & 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 5 & 0 & 0 & 0 & 6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 1 & 0 & 0 & 0 \\ 0 & 3 & 0 & 1 & 0 & 0 & 2 & 0 & 0 \end{array}\right) \begin{array}{c} l_2 - 2l_1 \\ \\ \\ \\ \rightarrow \end{array} \left(\begin{array}{ccc|c} [1,1] & [5,3] & [8,2] & 0 \\ [5,1] & [3,1] & [6,2] & 4 \\ [4,4] & [7,1] & [0,0] & 0 \\ [4,3] & [8,1] & [0,0] & 0 \\ [1,1] & [2,1] & [0,0] & 0 \\ [3,5] & [8,2] & [0,0] & 0 \\ [2,5] & [6,6] & [0,0] & 0 \\ [5,2] & [6,1] & [0,0] & 0 \\ [2,3] & [4,1] & [7,1] & 0 \end{array}\right) \quad [8,2]??$$

it overflows the available memory!
$\rightarrow$ Stupid method.

Sparse Linear Algebra, specific algorithms

- Adapted Gaussian Elimination
  = choose pivots that minimize the loss of sparsity

$$
\left(\begin{array}{cccccccc|c}
1 & 0 & 0 & 0 & 3 & 0 & 0 & 2 & 0 \\
2 & 0 & 1 & 0 & 0 & 2 & 0 & 0 & 0 \\
0 & 0 & 0 & 4 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 3 & 0 & 0 & 0 & 1 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 5 & 0 & 0 & 0 & 0 & 2 & 0 \\
0 & 5 & 0 & 0 & 0 & 6 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 2 & 1 & 0 & 0 & 0 \\
0 & 3 & 0 & 1 & 0 & 0 & 2 & 0 & 0
\end{array}\right)
\rightarrow
\left(\begin{array}{ccc|c}
[1,1] & [5,3] & [8,2] & 0 \\
[1,2] & [3,1] & [6,2] & 0 \\
[4,4] & [7,1] & [0,0] & 0 \\
[4,3] & [8,1] & [0,0] & 0 \\
[1,1] & [2,1] & [0,0] & 0 \\
[3,5] & [8,2] & [0,0] & 0 \\
[2,5] & [6,6] & [0,0] & 0 \\
[5,2] & [6,1] & [0,0] & 0 \\
[2,3] & [4,1] & [7,1] & 0
\end{array}\right)
$$

- or, without any modification of the matrix, using matrix-by-vector multiplications only:
  - Krylov Subspace methods
  - Wiedemann algorithm(s)

Wiedemann

1986

Problem

Solve:

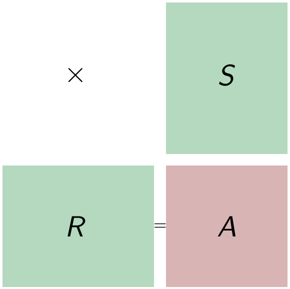
$$Sx = 0 \quad or \quad Sx = y$$

with S a sparse matrix with coefficients in a ring $\mathbb{K}$,
K non zero coeffs. per row max,
N= max(# rows, # col)

$$S = \begin{pmatrix} 1 & 0 & 0 & 0 & 3 & 0 & 0 & 2 \\ 2 & 0 & 1 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 5 & 0 & 0 & 0 & 0 & 2 \\ 0 & 5 & 0 & 0 & 0 & 6 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 1 & 0 & 0 \\ 0 & 3 & 0 & 1 & 0 & 0 & 2 & 0 \end{pmatrix} \quad \begin{matrix} K = 3 \\ N = 9 \end{matrix}$$

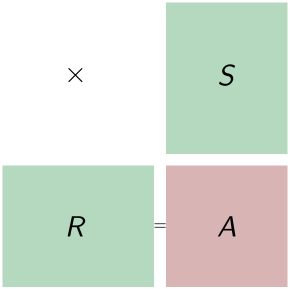
Wiedemann

1. Preconditioning step : We transform S into a square matrix A.



Wiedemann

1. Preconditioning step : We transform S into a square matrix A.



Why?

- Powers of A are well defined.
- A not sparse but multiplying $R\ S = A$ with a vector is quick: $O(KN)$
- $S.x = 0 \Rightarrow A.x = 0$ (or $S.x = y \Rightarrow A.x = y' = R.y$). The converse is true for almost all random matrices R.

Try to solve $A.x = 0$ (or $A.x = y'$).

Wiedemann

2. Computation of a scalar sequence : $({}^t w A^i v)_{i=0,\cdots,2n}$ with v, w two random vectors and $n = \#$ col. of A.
3. Reconstruction of the minimal polynomial of A.

Wiedemann

2. Computation of a scalar sequence : $({}^t w A^i v)_{i=0,\cdots,2n}$
   with v, w two random vectors and $n = \#$ col. of A.
3. Reconstruction of the minimal polynomial of $\boxed{A}$.

Why does 2 help 3 ?

Wiedemann

2. Computation of a scalar sequence : $({}^t w A^i v)_{i=0,\cdots,2n}$ with v, w two random vectors and $n = \#$ col. of A.
3. Reconstruction of the minimal polynomial of $\boxed{A}$.

Why does 2 help 3 ?

- Cayley-Hamilton theorem: the characteristic polynomial of A, of degree n, annihilates A.
- so we seek for a_i s.t. $\sum_{i=0}^{n} a_i A^i = 0$. $(\star_1)$

Wiedemann

2. Computation of a scalar sequence : $({}^t w A^i v)_{i=0,\cdots,2n}$ with v, w two random vectors and $n = \#$ col. of A.
3. Reconstruction of the minimal polynomial of A.

Why does 2 help 3 ?

- Cayley-Hamilton theorem: the characteristic polynomial of A, of degree n, annihilates A.
- so we seek for a_i s.t. $\sum_{i=0}^{n} a_i A^i = 0$. $(\star_1)$
- $\Rightarrow \forall j \in \mathbb{N}, A^j(\sum_{i=0}^{n} a_i A^i) = 0$.

Wiedemann

2. Computation of a scalar sequence : $({}^t w A^i v)_{i=0,\cdots,2n}$ with v, w two random vectors and $n = \#$ col. of A.
3. Reconstruction of the minimal polynomial of $\boxed{A}$.

Why does 2 help 3 ?

- Cayley-Hamilton theorem: the characteristic polynomial of A, of degree n, annihilates A.
- so we seek for a_i s.t. $\sum_{i=0}^{n} a_i A^i = 0$. $(\star_1)$
- $\Rightarrow \forall j \in \mathbb{N}, \sum_{i=0}^{n} a_i A^{i+j} = 0.$

Wiedemann

2. Computation of a scalar sequence : $({}^t w A^i v)_{i=0,\cdots,2n}$ with v, w two random vectors and $n = \#$ col. of A.
3. Reconstruction of the minimal polynomial of $\boxed{A}$.

Why does 2 help 3 ?

- Cayley-Hamilton theorem: the characteristic polynomial of A, of degree n, annihilates A.
- so we seek for a_i s.t. $\sum_{i=0}^{n} a_i A^i = 0$. $(\star_1)$
- $\Rightarrow \forall j \in \mathbb{N}, \forall v$ vector, $\sum_{i=0}^{n} a_i A^{i+j} v = 0$.

Wiedemann

2. Computation of a scalar sequence : $({}^t w A^i v)_{i=0,\cdots,2n}$ with v, w two random vectors and $n = \#$ col. of A.
3. Reconstruction of the minimal polynomial of $\boxed{A}$.

Why does 2 help 3 ?

- Cayley-Hamilton theorem: the characteristic polynomial of A, of degree n, annihilates A.
- so we seek for a_i s.t. $\sum_{i=0}^{n} a_i A^i = 0$. $(\star_1)$
- $\Rightarrow \forall j \in \mathbb{N}, \forall v, w$ vectors, $\sum_{i=0}^{n} a_i \, {}^t w A^{i+j} v = 0$. $(\star_2)$

Wiedemann

2. Computation of a scalar sequence : $({}^t w A^i v)_{i=0,\cdots,2n}$ with v, w two random vectors and $n = \#$ col. of A.
3. Reconstruction of the minimal polynomial of $\boxed{A}$.

Why does 2 help 3 ?

- Cayley-Hamilton theorem: the characteristic polynomial of A, of degree n, annihilates A.
- so we seek for a_i s.t. $\sum_{i=0}^{n} a_i A^i = 0$. $(\star_1)$
- $\Rightarrow \forall j \in \mathbb{N}, \forall v, w$ vectors, $\sum_{i=0}^{n} a_i \, {}^t w A^{i+j} v = 0$. $(\star_2)$
- $\Rightarrow$ There exists a linear recursive relationship between the elements of $({}^t w A^i v)_{i=0,\cdots,2n}$!

Wiedemann

2. Computation of a scalar sequence : $({}^t w A^i v)_{i=0,\cdots,2n}$ with v, w two random vectors and $n = \#$ col. of A.
3. Reconstruction of the minimal polynomial of $\boxed{A}$.

Why does 2 help 3 ?

- Cayley-Hamilton theorem: the characteristic polynomial of A, of degree n, annihilates A.
- so we seek for a_i s.t. $\sum_{i=0}^{n} a_i A^i = 0$. $(\star_1)$
- $\Rightarrow \forall j \in \mathbb{N}, \forall v, w$ vectors, $\sum_{i=0}^{n} a_i \, {}^t w A^{i+j} v = 0$. $(\star_2)$
- $\Rightarrow$ There exists a linear recursive relationship between the elements of $({}^t w A^i v)_{i=0,\cdots,2n}$!
- Berlekamp-Massey permits to recover the minimal poly. of a recursive linear sequence.

Wiedemann

2. Computation of a scalar sequence : $({}^t w A^i v)_{i=0,\cdots,2n}$ with v, w two random vectors and $n = \#$ col. of A.
3. Reconstruction of the minimal polynomial of $\boxed{A}$.

Why does 2 help 3 ?

- Cayley-Hamilton theorem: the characteristic polynomial of A, of degree n, annihilates A.
- so we seek for a_i s.t. $\sum_{i=0}^{n} a_i A^i = 0$. $(\star_1)$
- $\Rightarrow \forall j \in \mathbb{N}, \forall v, w$ vectors, $\sum_{i=0}^{n} a_i \, {}^t w A^{i+j} v = 0$. $(\star_2)$
- $\Rightarrow$ There exists a linear recursive relationship between the elements of $({}^t w A^i v)_{i=0,\cdots,2n}$!
- Berlekamp-Massey permits to recover the minimal poly. of a recursive linear sequence.
- $(\star_2)$ for some random v and w $\Rightarrow_{\text{almost always}}$ $(\star_1)$.

Wiedemann

2. Computation of a scalar sequence : $({}^t w A^i v)_{i=0,\cdots,2n}$ with v, w two random vectors and $n = \#$ col. of A.
3. Reconstruction of the minimal polynomial of $\boxed{A}$.

Why does 2 help 3 ?

- Cayley-Hamilton theorem: the characteristic polynomial of A, of degree n, annihilates A.
- so we seek for a_i s.t. $\sum_{i=0}^{n} a_i A^i = 0$. $(\star_1)$
- $\Rightarrow \forall j \in \mathbb{N}, \forall v, w$ vectors, $\sum_{i=0}^{n} a_i \, {}^t w A^{i+j} v = 0$. $(\star_2)$
- $\Rightarrow$ There exists a linear recursive relationship between the elements of $({}^t w A^i v)_{i=0,\cdots,2n}$!
- Berlekamp-Massey permits to recover the minimal poly. of a recursive linear sequence.
- $(\star_2)$ for some random v and w $\Rightarrow_{\text{almost always}}$ $(\star_1)$.

We have found a_i s.t. $\sum_{i=0}^{n} a_i A^i = 0$.

Wiedemann

4. Computation of the solution.
   - How to solve $Ax = 0$ thanks to $\sum_{i=0}^{n} a_i A^i = 0$?
     If there is a solution then $a_0 = 0$.
     So for a random vector r:
     $$\sum_{i=1}^{n} a_i A^i r = 0 \Leftrightarrow A \underbrace{\left(\sum_{i=1}^{n} a_i A^{i-1} r \right)}_{\text{Here is } x \text{ !}} = 0$$

Wiedemann

4. Computation of the solution.

- How to solve $Ax = 0$ thanks to $\sum_{i=0}^{n} a_i A^i = 0$?
  If there is a solution then $a_0 = 0$.
  So for a random vector r:
  $$\sum_{i=1}^{n} a_i A^i r = 0 \Leftrightarrow A \underbrace{\left(\sum_{i=1}^{n} a_i A^{i-1} r \right)}_{\text{Here is } x \text{ !}} = 0$$
- How to solve $Ax = y$ thanks to $\sum_{i=0}^{n} a_i A^i = 0$?
  A inversible permits to assume $a_0 \neq 0$.
  So $\displaystyle\sum_{i=0}^{n} a_i A^i x = 0 \Leftrightarrow -a_0 x = \sum_{i=1}^{n} a_i A^i x$
  $\Leftrightarrow x = -(1/a_0) \sum_{i=1}^{n} a_i A^{i-1} A x$
  $\Leftrightarrow x = -(1/a_0) \sum_{i=1}^{n} a_i A^{i-1} y$. Here is x again !

Wiedemann

1. Preconditioning step: Transformation of $\boxed{S}$ into $\boxed{A}$.
   The problem becomes:
   $$A.x = 0 \quad \text{or} \quad A.x = y'.$$
2. Computation of a scalar sequence: $({}^t wA^i v)_{i=0,\cdots,2n}$
   with v, w two random vectors and $n = \#$ col. of A.
3. Reconstruction of the minimal polynomial of $\boxed{A}$ thanks to Berlekamp-Massey algorithm.
4. Computation of the solution.

Wiedemann

1. Preconditioning step: Transformation of $\boxed{S}$ into $\boxed{A}$.
   The problem becomes:
   $$A.x = 0 \quad \text{or} \quad A.x = y'.$$
2. Computation of a scalar sequence: $({}^t wA^i v)_{i=0,\cdots,2n}$ with v, w two random vectors and $n = \#$ col. of A.

   Complexity: Cost of multiplication $\boxed{A}$-vector $\times$ length of the sequence $= O(KN^2)$
3. Reconstruction of the minimal polynomial of $\boxed{A}$ thanks to Berlekamp-Massey algorithm.
4. Computation of the solution.

Wiedemann

1. Preconditioning step: Transformation of $\boxed{S}$ into $\boxed{A}$.
   The problem becomes:
   $$A.x = 0 \quad \text{or} \quad A.x = y'.$$
2. Computation of a scalar sequence: $({}^t w A^i v)_{i=0,\cdots,2n}$ with v, w two random vectors and $n = \#$ col. of A.
   Complexity: Cost of multiplication $\boxed{A}$-vector $\times$ length of the sequence $= O(KN^2)$
3. Reconstruction of the minimal polynomial of $\boxed{A}$ thanks to Berlekamp-Massey algorithm.
   Complexity: quasi-linear in N (with fast B-M. algo).
4. Computation of the solution.

Wiedemann

1. Preconditioning step: Transformation of $\boxed{S}$ into $\boxed{A}$. The problem becomes:

$$A.x = 0 \quad \text{or} \quad A.x = y'.$$

2. Computation of a scalar sequence: $({}^t w A^i v)_{i=0,\cdots,2n}$ with v, w two random vectors and $n = \#$ col. of A.

   Complexity: Cost of multiplication $\boxed{A}$-vector $\times$ length of the sequence $= O(KN^2)$

3. Reconstruction of the minimal polynomial of $\boxed{A}$ thanks to Berlekamp-Massey algorithm.
   Complexity: quasi-linear in N (with fast B-M. algo).
4. Computation of the solution.

   Complexity: Cost of multiplication $\boxed{A}$-vector $\times$ nbr elts of the sum $= O(KN^2)$

Wiedemann

1. Preconditioning step: Transformation of $\boxed{S}$ into $\boxed{A}$.
   The problem becomes:
   $$A.x = 0 \quad \text{or} \quad A.x = y'.$$
2. Computation of a scalar sequence: $({}^t wA^i v)_{i=0,\cdots,2n}$
   with v, w two random vectors and $n = \#$ col. of A.

   Complexity: Cost of multiplication $\boxed{A}$-vector $\times$ length of the sequence $= O(KN^2)$
3. Reconstruction of the minimal polynomial of $\boxed{A}$ thanks to Berlekamp-Massey algorithm.
   Complexity: quasi-linear in N (with fast B-M. algo).
4. Computation of the solution.

   Complexity: Cost of multiplication $\boxed{A}$-vector $\times$ nbr elts of the sum $= O(KN^2)$

Final asymptotic complexity:

$$O(KN^2)$$

Let us parallelize!

- 1994. Coppersmith. Distributed computations for sparse linear algebra over $\mathbb{F}_2$.

Let us parallelize!

- 1994. Coppersmith. Distributed computations for sparse linear algebra over $\mathbb{F}_2$.
- 1995. Kaltofen. Generalized this idea to $\mathbb{F}_{p^n}$.

Let us parallelize!

- 1994. Coppersmith. Distributed computations for sparse linear algebra over $\mathbb{F}_2$.
- 1995. Kaltofen. Generalized this idea to $\mathbb{F}_{p^n}$.
- 2002. Thomé. Generalized fast Berlekamp-Massey.

From Wiedemann to Block Widemann

1. Preconditioning step: Transformation of $\boxed{S}$ into a square matrix $\boxed{A}$. The problem becomes:

$$A.x = 0 \quad \text{ou} \quad A.x = y'.$$

2. Computation of a scalar sequence: $({}^t w A^i v)_{i=0,\cdots,2n}$ with v, w two random vectors

3. Reconstruction of the minimal polynomial of $\boxed{A}$ thanks to Berlekamp-Massey algorithm.

4. Computation of the solution.

From Wiedemann to Block Widemann

1. Preconditioning step: Transformation of S into a square matrix A. The problem becomes:

$$A.x = 0 \quad \text{ou} \quad A.x = y'.$$

2. Computation of a matrix sequence: $({}^tW\ A^i V)_{i=0,\cdots,2n/c}$ with $V = (v_1, \cdots, v_c)$, W two random matrices

3. Reconstruction of the minimal polynomial of A thanks to Berlekamp-Massey algorithm.

4. Computation of the solution.

From Wiedemann to Block Widemann

1. Preconditioning step: Transformation of S into a square matrix A. The problem becomes:

$$A.x = 0 \quad \text{ou} \quad A.x = y'.$$

2. Computation of a matrix sequence: $({}^tW\ A^i V)_{i=0,\cdots,2n/c}$ with $V = (v_1, \cdots, v_c)$, W two random matrices
Parallelization over c machines :

$$1 \quad ({}^tWA^i v_1)_{i=0,\cdots,2n/c}$$

$$\cdots \qquad\qquad \cdots$$

$$c \quad ({}^tWA^i v_c)_{i=0,\cdots,2n/c}$$

3. Reconstruction of the minimal polynomial of A thanks to Berlekamp-Massey algorithm.

4. Computation of the solution.

From Wiedemann to Block Widemann

1. Preconditioning step: Transformation of $\boxed{S}$ into a square matrix $\boxed{A}$. The problem becomes:

$$A.x = 0 \quad \text{ou} \quad A.x = y'.$$

2. Computation of a matrix sequence: $({}^{t}W\ A^{i}V)_{i=0,\cdots,2n/c}$
with $V = (v_1, \cdots, v_c)$, W two random matrices
Parallelization over c machines :

$1 \quad ({}^{t}WA^{i}v_1)_{i=0,\cdots,2n/c}$

$\ldots \qquad \ldots$

$c \quad ({}^{t}WA^{i}v_c)_{i=0,\cdots,2n/c}$

Complexity : $O(KN^2)$ but distributed over c machines.

3. Reconstruction of the minimal polynomial of $\boxed{A}$ thanks to Berlekamp-Massey algorithm.

4. Computation of the solution.

From Wiedemann to Block Widemann

1. Preconditioning step: Transformation of $\boxed{S}$ into a square matrix $\boxed{A}$. The problem becomes:

$$A.x = 0 \quad \text{ou} \quad A.x = y'.$$

2. Computation of a matrix sequence: $({}^t W\ A^i V)_{i=0,\cdots,2n/c}$ with $V = (v_1, \cdots, v_c)$, W two random matrices
   Parallelization over c machines :

$$1 \quad ({}^t WA^i v_1)_{i=0,\cdots,2n/c}$$

$$\cdots \qquad\qquad \cdots$$

$$c \quad ({}^t WA^i v_c)_{i=0,\cdots,2n/c}$$

   Complexity : $O(KN^2)$ but distributed over c machines.

3. Reconstruction of coeffs. a_{ij} s.t. $\sum_{j=1}^{c} \sum_{i=0}^{n/c} a_{ij} A^i v_j = 0$ thanks to Thomé algorithm. Complexity : $\tilde{O}(c^2 N)$
4. Computation of the solution.

From Wiedemann to Block Widemann

1. Preconditioning step: Transformation of $\boxed{S}$ into a square matrix $\boxed{A}$. The problem becomes:

$$A.x = 0 \quad \text{ou} \quad A.x = y'.$$

2. Computation of a matrix sequence: $({}^tW\ A^i V)_{i=0,\cdots,2n/c}$ with $V = (v_1, \cdots, v_c)$, W two random matrices
   Parallelization over c machines :


$1 \quad ({}^tWA^i v_1)_{i=0,\cdots,2n/c}$

$\ldots \qquad \ldots$


$c \quad ({}^tWA^i v_c)_{i=0,\cdots,2n/c}$

   Complexity : $\mathrm{O}(KN^2)$ but distributed over c machines.

3. Reconstruction of coeffs. a_{ij} s.t. $\sum_{j=1}^{c}\sum_{i=0}^{n/c} a_{ij}A^i v_j = 0$ thanks to Thomé algorithm. Complexity : $\tilde{O}(c^2N)$
4. Computation of the solution. Complexity : $O(KN^2)$ distributed.

From Wiedemann to Block Widemann

1. Preconditioning step: Transformation of $\boxed{S}$ into a square matrix $\boxed{A}$. The problem becomes:

$$A.x = 0 \quad \text{ou} \quad A.x = y'.$$

2. Computation of a matrix sequence: $({}^tW\ A^iV)_{i=0,\cdots,2n/c}$ with $V = (v_1, \cdots, v_c)$, W two random matrices
Parallelization over c machines :

$$1 \quad ({}^tWA^iv_1)_{i=0,\cdots,2n/c}$$

$$\cdots \qquad \cdots$$

$$c \quad ({}^tWA^iv_c)_{i=0,\cdots,2n/c}$$

Complexity : $O(KN^2)$ but distributed over c machines.

3. Reconstruction of coeffs. a_{ij} s.t. $\sum_{j=1}^{c}\sum_{i=0}^{n/c} a_{ij}A^iv_j = 0$ thanks to Thomé algorithm. Complexity : $\tilde{O}(c^2N)$
4. Computation of the solution. Complexity : $O(KN^2)$ distributed.

Final asymptotic complexity: $O(KN^2) + \tilde{O}(c^2N)$

Dlog-NFS raises a question of identity...

... what if the matrix is not *truly* sparse?

Matrices in NFS

Computing Dlog with NFS leads to consider matrices of the form:

$$S = \begin{pmatrix} 1 & 0 & 0 & 0 & 3 & 0 & 0 & 2 & 5 & 3 \\ 2 & 0 & 1 & 0 & 0 & 2 & 0 & 0 & 6 & 2 \\ 0 & 0 & 0 & 4 & 0 & 0 & 1 & 0 & 6 & 4 \\ 0 & 0 & 0 & 3 & 0 & 0 & 0 & 1 & 5 & 2 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 1 \\ 0 & 0 & 5 & 0 & 0 & 0 & 0 & 2 & 1 & 1 \\ 0 & 5 & 0 & 0 & 0 & 6 & 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 0 & 2 & 1 & 0 & 0 & 1 & 6 \\ 0 & 3 & 0 & 1 & 0 & 0 & 2 & 0 & 5 & 6 \\ 0 & 0 & 0 & 0 & 3 & 0 & 0 & 3 & 4 & 2 \\ 0 & 2 & 0 & 3 & 0 & 0 & 2 & 0 & 5 & 1 \end{pmatrix}$$

Is it sparse? Is it dense?

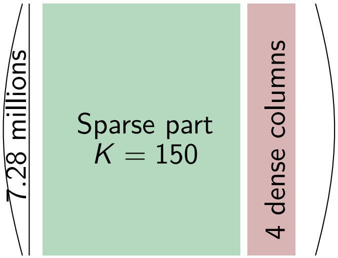
$K = 5$
$N = 11$

- If we apply a classical algo., we don't take advantage of zero coeffs.
- If we apply Block-Wiedemann, we don't take advantage of the particular distribution of non zero coeffs.

- Number fields complicate the linear algebra step: need to take into account the contribution of units in these number fields.
- $\Rightarrow$ Schirokauer maps.
- 1 unit $=$ +1 Schirokauer map $=$ +1 dense column

Example

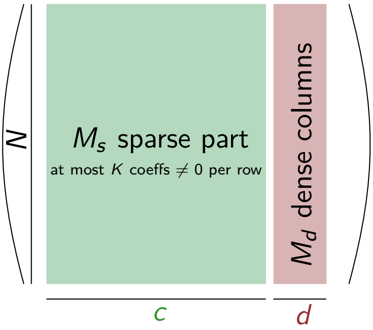
- Latest record on a prime field $\mathbb{F}_p$, ($p \approx 180$ digits)
- June 2014 by Bouvier, Gaudry, Imbert, Jeljeli, Thomé.



Nearly sparse linear algebra

Definition

M is (d-)nearly sparse if it is of the form:



Problem

Solve: $M \cdot x = 0$ *or* $M \cdot x = y$

where M is a nearly sparse matrix with coeff. in a ring $\mathbb{K}$.

Nearly sparse linear algebra

Remark

- There is no restriction on the nbr of dense columns.

Nearly sparse linear algebra

Remark

- There is no restriction on the nbr of dense columns.
- Being able to recover a non trivial elt of the kernel of a nearly sparse matrix suffices!

Let's assume we want to solve $M \cdot x = y$ with M a d-nearly sparse matrix.

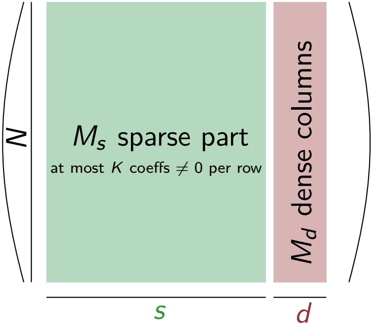
Then $\begin{pmatrix} M \end{pmatrix} \cdot \begin{pmatrix} x \end{pmatrix} = \begin{pmatrix} y \end{pmatrix} \Leftrightarrow \begin{pmatrix} M & y \end{pmatrix} \cdot \begin{pmatrix} x \\ -1 \end{pmatrix} = 0.$

Since $\begin{pmatrix} M & y \end{pmatrix}$ is $d+1$-nearly sparse, it's ok.

Nearly sparse linear algebra

Definition

M is (d-)nearly sparse if it is of the form:



Problem

Solve:

$$M \cdot x = 0$$

where M is a nearly sparse matrix with coeff. in a ring $\mathbb{K}$.

A dedicated algorithm

Since M is (also) a sparse matrix of parameters $K+d$, N, we may apply Block-Wiedemann!
Asymptotic complexity:

$$O((K+d)N^2) + \tilde{O}(c^2 N)$$

A dedicated algorithm

Since M is (also) a sparse matrix of parameters $K+d$, N, we may apply Block-Wiedemann!
Asymptotic complexity:

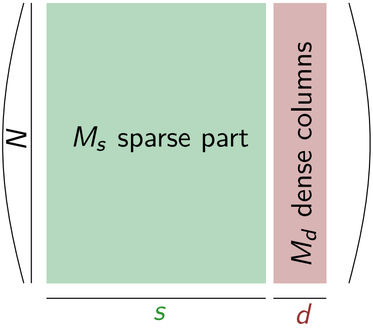
$$O((K+d)N^2)+\tilde{O}(c^2N)$$

Main result

We propose to design an algorithm with asymptotic complexity:

$$O(KN^2)+\tilde{O}(\max(c^2,d^2)N)$$

Key ideas



1. Apply Block-Wiedemann on the sparse part only.
2. Make the d dense columns contribute in the initial block V, *i.e.* set each dense col. $=$ one initial vector of the matrix sequences to construct.

Nearly sparse linear algebra algorithm

1. Preconditioning step on the RIGHT of the matrix M :

$$\begin{pmatrix} M_s & M_d \end{pmatrix} \times \begin{array}{c|cc} & s+d & d \\ \hline s & R & (0) \\ d & (0) & I_d \end{array} = \begin{pmatrix} A & M_d \end{pmatrix}$$

Why ?

- Powers of A are well defined.
- Multiplying $M_s \; R = A$ by a vector is quick enough.
- If R surj. : $(A|M_d).x = 0 \Rightarrow M.x = 0$

Try to solve $(A|M_d).x = 0$.

Nearly sparse linear algebra algorithm

For the sake of simplicity: # machines = # dense col.

2. Computation of a matrix sequence: $({}^tWA^iV)_{i=0,\cdots,2N}$
   with $V = (v_1, \cdots, v_d)$, W two rand. matrices.
   Parallelization over c machines :

1 $({}^tWA^iv_1)_{i=0,\cdots,2N/d}$

$\cdots \qquad \cdots$

d $({}^tWA^iv_d)_{i=0,\cdots,2N/d}$

.

Nearly sparse linear algebra algorithm

For the sake of simplicity: # machines = # dense col.

2. Computation of a matrix sequence: $({}^tWA^iV)_{i=0,\cdots,2N}$ with $V = (d_1, \cdots, d_d)$, W one rand. matrix and $d_1, \cdots, d_d$ the d dense col.
   Parallelization over d machines :

$1 \quad ({}^tWA^id_1)_{i=0,\cdots,2N/d}$

$\cdots \qquad \cdots$

$d \quad ({}^tWA^id_d)_{i=0,\cdots,2N/d}$

.

Nearly sparse linear algebra algorithm

For the sake of simplicity: # machines = # dense col.

2. Computation of a matrix sequence: $({}^t W A^i V)_{i=0,\cdots,2N}$
   with $V = (d_1, \cdots, d_d)$, W one rand. matrix and
   $d_1, \cdots, d_d$ the d dense col.
   Parallelization over d machines :

$1 \quad ({}^t W A^i d_1)_{i=0,\cdots,2N/d}$

$\cdots \qquad \cdots$

$d \quad ({}^t W A^i d_d)_{i=0,\cdots,2N/d}$

3. Reconstruction of coeffs. a_{ij} s.t. $\sum_{j=1}^{d} \sum_{i=0}^{N/d} a_{ij} A^i d_j = 0$
   thanks to Thomé.

Nearly sparse linear algebra algorithm

4. Computation of an elt of the kernel of $\begin{pmatrix} A & M_d \end{pmatrix}$

$$\sum_{j=1}^{d}\sum_{i=0}^{N/d} a_{ij}A^i d_j = 0 \Leftrightarrow \sum_{j=1}^{d}\sum_{i=1}^{N/d} a_{ij}A^i d_j + \sum_{j=1}^{d} a_{0j} d_j = 0$$

$$\Leftrightarrow A\cdot \underbrace{\sum_{j=1}^{d}\sum_{i=1}^{N/d} a_{ij}A^{i-1} d_j}_{\text{let us say } x'} + \sum_{j=1}^{d} a_{0j} d_j = 0$$

$$\Leftrightarrow A\cdot x' + a_{01} d_1 + a_{02} d_2 + \cdots + a_{0d} d_d = 0$$

So ${}^t(x'|a_{01}|a_{02}|\cdots|a_{0d}) \in \ker \begin{pmatrix} A & M_d \end{pmatrix}$.

Asymptotic complexity

Main result

We obtain an asymptotic complexity of:

$$O(KN^2) + \tilde{O}(\max(c^2, d^2)N) \textit{ operations,}$$

to be compared with previous $O((K + d)N^2) + \tilde{O}(c^2N)$ *complexity.*

When $d \leq c$, it becomes:

$$O(KN^2) + \tilde{O}(c^2N) \text{ operations.}$$

Remark

When we have more machines than dense columns, these columns cost NOTHING with our algorithm!

Asymptotic Complexity

And if $c < d$, how many dense col. can we still have?

- As soon as $d < N^{1-\epsilon}$ $(\epsilon > 0)$, our algorithm is better than Block-Wiedemann.
- As soon as $d < N^{\omega-2-\epsilon}$ $(\epsilon > 0)$, it is better than classical (dense) linear algebra algorithms of complexity $O(N^{\omega})$.

Asymptotic Complexity

And if $c < d$, how many dense col. can we still have?

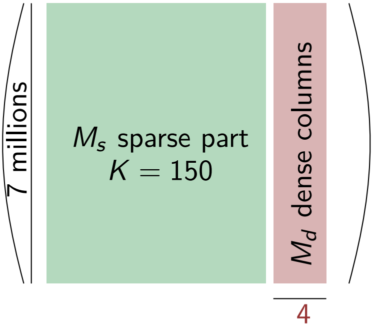
- As soon as $d < N^{1-\epsilon}$ ($\epsilon > 0$), our algorithm is better than Block-Wiedemann.
- As soon as $d < N^{\omega-2-\epsilon}$ ($\epsilon > 0$), it is better than classical (dense) linear algebra algorithms of complexity $O(N^{\omega})$.

Example

Recalling that $\omega \approx 2.37$, with $N^{1/3}$ dense columns for instance, our algorithm is still faster than any others.

Nearly Sparse Linear Algebra applied to Dlog

- Latest record on a prime field $\mathbb{F}_p$, ($p \approx 180$ digits)
- June 2014 by Bouvier, Gaudry, Imbert, Jeljeli, Thomé.
- Parameters of the matrix: $N \approx 7, 28$ millions of rows, $K = 150$ non zero coeff. per row, 4 dense columns.
- Parallelized over 16 machines.



To conclude with medium characteristic

- If your are a cryptographer:
  increase your finite fields cardinality by 6.4%
- If you are a cryptanalyst:
  do not worry about dense columns.

Merci de votre attention !