

# Fast arithmetics in Artin-Schreier towers over finite fields

Luca De Feo<sup>1</sup>  
joint work with É. Schost<sup>2</sup>

<sup>1</sup>École Polytechnique and INRIA, France

<sup>2</sup>ORCCA and CSD, The University of Western Ontario, London, ON

October 10, 2009  
RAIM, École Normale Supérieure, Lyon

# Doing arithmetics in towers of extensions

$$\begin{array}{c} \mathbb{U}_k \\ \left| \begin{array}{c} p \\ \vdots \\ p \end{array} \right. \\ \mathbb{U}_{k-1} \\ \vdots \\ \mathbb{U}_2 \\ \left| \begin{array}{c} p \\ \vdots \\ p \end{array} \right. \\ \mathbb{U}_1 \\ \left| \begin{array}{c} p \\ \vdots \\ p \end{array} \right. \\ \mathbb{F}_q \end{array}$$

Standard arithmetics

$$+, -, \times, / : \begin{cases} \mathbb{U}_i \times \mathbb{U}_i & \rightarrow \mathbb{U}_i \\ (u, v) & \mapsto u \operatorname{op} v \end{cases}$$

# Doing arithmetics in towers of extensions

$$\begin{array}{c} \mathbb{U}_k \\ \left| \begin{array}{c} p \\ \vdots \\ \vdots \\ \vdots \end{array} \right. \\ \mathbb{U}_{k-1} \\ \vdots \\ \mathbb{U}_2 \\ \left| \begin{array}{c} p \\ \vdots \\ \vdots \\ \vdots \end{array} \right. \\ \mathbb{U}_1 \\ \left| \begin{array}{c} p \\ \vdots \\ \vdots \\ \vdots \end{array} \right. \\ \mathbb{F}_q \end{array}$$

Inclusion

$$\iota : \begin{cases} \mathbb{U}_i & \subset \mathbb{U}_{i+1} \\ v & \mapsto \bar{v} \end{cases}$$

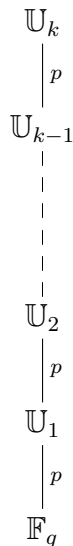
# Doing arithmetics in towers of extensions

$$\begin{array}{c} \mathbb{U}_k \\ \left| \begin{array}{c} p \\ \vdots \\ p \end{array} \right. \\ \mathbb{U}_{k-1} \\ \vdots \\ \mathbb{U}_2 \\ \left| \begin{array}{c} p \\ \vdots \\ p \end{array} \right. \\ \mathbb{U}_1 \\ \left| \begin{array}{c} p \\ \vdots \\ p \end{array} \right. \\ \mathbb{F}_q \end{array}$$

Membership

$$\iota^{-1} : \begin{cases} \mathbb{U}_{i+1} & \supset \mathbb{U}_i \\ \iota(v) & \mapsto v \end{cases}$$

# Doing arithmetics in towers of extensions



Projection

$$\pi : \begin{cases} \mathbb{U}_{i+1} & \xrightarrow{\sim} \mathbb{U}_i^p \simeq \mathbb{U}_i[\gamma] \\ v & \mapsto (v_0, \dots, v_{p-1}) \end{cases}$$

$$\pi^{-1} : \begin{cases} \mathbb{U}_i^p \simeq \mathbb{U}_i[\gamma] & \xrightarrow{\sim} \mathbb{U}_{i+1} \\ (v_0, \dots, v_{p-1}) & \mapsto \sum_j v_j \gamma^j \end{cases}$$

# Doing arithmetics in towers of extensions

$$\begin{array}{c} \mathbb{U}_k \\ \left| \begin{array}{c} p \\ \vdots \\ p \end{array} \right. \\ \mathbb{U}_{k-1} \\ \vdots \\ \mathbb{U}_2 \\ \left| \begin{array}{c} p \\ \vdots \\ p \end{array} \right. \\ \mathbb{U}_1 \\ \left| \begin{array}{c} p \\ \vdots \\ p \end{array} \right. \\ \mathbb{F}_q \end{array}$$

Traces

$$\mathrm{Tr} : \begin{cases} \mathbb{U}_{i+1} & \rightarrow \mathbb{U}_i \\ v & \mapsto \mathrm{Tr}(v) \end{cases}$$

# Doing arithmetics in towers of extensions

$$\begin{array}{c} \mathbb{U}_k \\ \left| \begin{array}{c} p \\ \vdots \\ \vdots \\ \vdots \end{array} \right. \\ \mathbb{U}_{k-1} \\ \vdots \\ \vdots \\ \vdots \\ \mathbb{U}_2 \\ \left| \begin{array}{c} p \\ \vdots \\ \vdots \\ \vdots \end{array} \right. \\ \mathbb{U}_1 \\ \left| \begin{array}{c} p \\ \vdots \\ \vdots \\ \vdots \end{array} \right. \\ \mathbb{F}_q \end{array}$$

Galois action

$$\varphi : \begin{cases} G \times \mathbb{U}_i & \rightarrow \mathbb{U}_i \\ (\sigma, v) & \mapsto \sigma(v) \end{cases}$$

$$G := \text{Gal}(\mathbb{U}_{i+1}/\mathbb{U}_i) \simeq \mathbb{Z}/p\mathbb{Z}$$

# Crypto application : Isogeny computation

$$\begin{array}{l} \mathbb{U}_{16} \leftarrow E[2^{18}] \\ | \\ 2 \\ \mathbb{U}_{15} \leftarrow E[2^{17}] \\ | \\ \vdots \\ \mathbb{U}_2 \leftarrow E[16] \\ | \\ 2 \\ \mathbb{U}_1 \leftarrow E[8] \\ | \\ 2 \\ \mathbb{F}_q \leftarrow E[4] \end{array}$$

$E, E'$  elliptic curves  
with  $\#E(\mathbb{F}_q) = \#E'(\mathbb{F}_q)$

## Theorem/Algorithm

Knowing  $E[2^{k+3}]$  and  $E'[2^{k+3}]$

$\Rightarrow$  all isogenies of degree  $< 2^k$

## Example

- $\mathbb{F}_q = \mathbb{F}_{2^{163}}$ ,
- $E[4] \subset E(\mathbb{F}_q)$ ,  $E[2^{i+2}] \subset E(\mathbb{U}_i)$ ,
- Isogeny degree  $< 2^{15} \Rightarrow 16$  levels !!
- One element of  $\mathbb{U}_{16} \sim 1.5\text{MB}$  !!

# Our context

$$\mathbb{U}_k = \frac{\mathbb{U}_{k-1}[X_k]}{P_{k-1}(X_k)}$$

$\left| \begin{array}{c} p \\ \hline \end{array} \right.$

$$\mathbb{U}_{k-1}$$

$\vdots$

$$\mathbb{U}_1 = \frac{\mathbb{U}_0[X_1]}{P_0(X_1)}$$

$\left| \begin{array}{c} p \\ \hline \end{array} \right.$

$$\mathbb{F}_q = \frac{\mathbb{F}_p[X_0]}{Q(X_0)}$$

Tower over finite fields

$P_i$  irreducible polynomial in  $\mathbb{U}_i[X]$

# Our context

$$\mathbb{U}_k = \frac{\mathbb{U}_{k-1}[X_k]}{P_{k-1}(X_k)}$$

$\left| \begin{array}{c} p \\ \vdots \\ p \end{array} \right.$

$$\mathbb{U}_{k-1}$$

$\vdots$

$$\mathbb{U}_1 = \frac{\mathbb{U}_0[X_1]}{P_0(X_1)}$$

$\left| \begin{array}{c} p \\ \vdots \\ p \end{array} \right.$

$$\mathbb{F}_q = \frac{\mathbb{F}_p[X_0]}{Q(X_0)}$$

Tower over finite fields

$P_i$  irreducible polynomial in  $\mathbb{U}_i[X]$

But this is too hard.

# Artin-Schreier

## Definition (Artin-Schreier polynomial)

$\mathbb{K}$  a field of characteristic  $p$ ,  $\alpha \in \mathbb{K}$

$$X^p - X - \alpha$$

is an Artin-Schreier polynomial.

## Theorem

$\mathbb{K}$  finite.  $X^p - X - \alpha$  irreducible  $\Leftrightarrow \text{Tr}_{\mathbb{K}/\mathbb{F}_p}(\alpha) \neq 0$ .

If  $\eta \in \mathbb{K}$  is a root, then  $\eta + 1, \dots, \eta + (p-1)$  are roots.

## Definition (Artin-Schreier extension)

$\mathcal{P}$  an irreducible Artin-Schreier polynomial.

$$\mathbb{L} = \mathbb{K}[X]/\mathcal{P}(X).$$

$\mathbb{L}/\mathbb{K}$  is called an Artin-Schreier extension.

# Our context

$$\mathbb{U}_k = \frac{\mathbb{U}_{k-1}[X_k]}{P_{k-1}(X_k)}$$

$\left| \begin{array}{c} p \\ \hline \end{array} \right.$

$$\mathbb{U}_{k-1}$$

$\vdots$

$$\mathbb{U}_1 = \frac{\mathbb{U}_0[X_1]}{P_0(X_1)}$$

$\left| \begin{array}{c} p \\ \hline \end{array} \right.$

$$\mathbb{U}_0 = \mathbb{F}_{p^d} = \frac{\mathbb{F}_p[X_0]}{Q(X_0)}$$

## Towers over finite fields

$$P_i = X^p - X - \alpha_i$$

We say that  $(\mathbb{U}_0, \dots, \mathbb{U}_k)$  is defined by  $(\alpha_0, \dots, \alpha_{k-1})$  over  $\mathbb{U}_0$ .

**ANY** separable extension of degree  $p$  can be expressed this way

# Size, complexities

$$\#\mathbb{U}_i = p^{p^i d}$$

 $\mathbb{U}_k$ 

## Optimal representation

All common representations achieve it:  $O(p^i d)$

 $\mathbb{U}_{k-1}$ 

## Complexities

|                 |                                  |                      |
|-----------------|----------------------------------|----------------------|
| optimal:        | $O(p^i d)$                       | addition             |
| quasi-optimal:  | $\tilde{O}(i^a p^i d)$           | FFT multiplication   |
| almost-optimal: | $\tilde{O}(i^a p^{i+b} d)$       |                      |
| suboptimal:     | $\tilde{O}(i^a p^{i+b} d^c)$     |                      |
| too bad:        | $\tilde{O}(i^a (p^{i+b})^e d^c)$ | naive multiplication |

 $\mathbb{U}_1$  $\mathbb{U}_0$ 

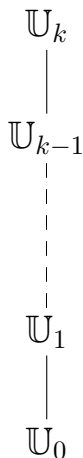
## Multiplication function $M(n)$

FFT:  $M(n) = O(n \log n \log \log n)$ ,      Naive:  $M(n) = O(n^2)$ .

# Outline

- 1 Representation
- 2 More arithmetics
- 3 Implementation and benchmarks

# Representation matters!



## Multivariate representation of $v \in U_i$

$$v = X_0^{d-1} X_1^{p-1} \cdots X_i^{p-1} + 2X_0^{d-1} X_1^{p-1} \cdots X_i^{p-2} + \cdots$$

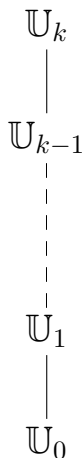
## Univariate representation of $v \in U_i$

- $U_i = \mathbb{F}_p[x_i]$ ,
- $v = c_0 + c_1 x_i + c_2 x_i^2 + \cdots + c_{p^i d-1} x_i^{p^i d-1}$  with  $c_i \in \mathbb{F}_p$ .

## How much does it cost to...

- Multiply?
- Express the embedding  $U_{i-1} \subset U_i$  ?
- Express the vector space isomorphism  $U_i = U_{i-1}^p$  ?
- Switch between the representations?

# A primitive tower



## Definition (Primitive tower)

A tower is primitive if  $\mathbb{U}_i = \mathbb{F}_p[X_i]$ .

In general this is not the case. Think of  $P_0 = X^p - X - 1$ .

## Theorem (extends a result in [Cantor '89])

Let  $x_0 = X_0$  such that  $\text{Tr}_{\mathbb{U}_0/\mathbb{F}_p}(x_0) \neq 0$ , let

$$P_0 = X^p - X - x_0$$

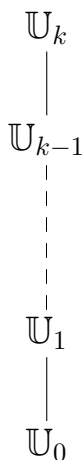
$$P_i = X^p - X - x_i^{2^{p-1}}$$

with  $x_{i+1}$  a root of  $P_i$  in  $\mathbb{U}_{i+1}$ .

Then, the tower defined by  $(P_0, \dots, P_{k-1})$  is primitive.

Some tricks to play when  $p = 2$ .

# Computing the minimal polynomials



We look for  $Q_i$ , the minimal polynomial of  $x_i$  over  $\mathbb{F}_p$

## Algorithm [Cantor '89]

- $Q_0 = Q$  easy,
- $Q_1 = Q_0(X^p - X)$  easy,

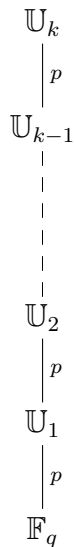
Let  $\omega$  be a  $2p - 1$ -th root of unity,

- $q_{i+1}(X^{2p-1}) = \prod_{j=0}^{2p-2} Q_i(\omega^j X)$  not too hard,
- $Q_{i+1} = q_{i+1}(X^p - X)$  easy.

## Complexity

$$O(M(p^{i+2}d) \log p)$$

Yes, we can multiply !



Standard arithmetics

$$+, -, \times, / : \begin{cases} \mathbb{U}_i \times \mathbb{U}_i & \rightarrow \mathbb{U}_i \\ (u, v) & \mapsto u \text{ op } v \end{cases}$$

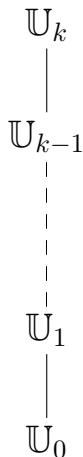
# Outline

1 Representation

2 More arithmetics

3 Implementation and benchmarks

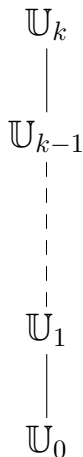
# Level embedding



$$\pi : \begin{cases} \mathbb{U}_{i+1} & \xrightarrow{\sim} \mathbb{U}_i^p \simeq \mathbb{U}_i[\gamma] \\ v & \mapsto (v_0, \dots, v_{p-1}) \end{cases}$$

$$\pi^{-1} : \begin{cases} \mathbb{U}_i^p \simeq \mathbb{U}_i[\gamma] & \xrightarrow{\sim} \mathbb{U}_{i+1} \\ (v_0, \dots, v_{p-1}) & \mapsto \sum_j v_j \gamma^j \end{cases}$$

# Level embedding



## Push-down

**Input**  $v \in U_i$ ,

**Output**  $v_0, \dots, v_{p-1} \in U_{i-1}$  such that  $v = v_0 + \dots + v_{p-1}x_i^{p-1}$ .

## Lift-up

**Input**  $v_0, \dots, v_{p-1} \in U_{i-1}$ ,

**Output**  $v \in U_i$  such that  $v = v_0 + \dots + v_{p-1}x_i^{p-1}$ .

## Complexity function $L(i)$

It turns out that the two operations lie in the same complexity class, we note  $L(i)$  for it:

$$L(i) = O(pM(p^i d) + p^{i+1}d \log_p(p^i d)^2)$$

---

## Push-down

---

**Input**  $v \in \mathbb{U}_i$ ,

**Output**  $v_0, \dots, v_{p-1} \in \mathbb{U}_{i-1}$  s.t.  $v = v_0 + \dots + v_{p-1}x_i^{p-1}$ .

- 1 Reduce  $v$  modulo  $x_i^p - x_i - x_{i-1}^{2p-1}$  by a divide-and-conquer approach,
  - 2 each of the coefficients of  $x_i$  has degree in  $x_{i-1}$  less than  $2 \deg_{x_i}(v)$ ,
  - 3 reduce each of the coefficients.
-

## Theorem

Up to some simple formulae:

$$\begin{pmatrix} \pi^{-1} \end{pmatrix} \begin{pmatrix} v \end{pmatrix} \sim \begin{pmatrix} \pi^T \end{pmatrix} \begin{pmatrix} M_v^T \end{pmatrix} \begin{pmatrix} \text{Tr}^T \end{pmatrix}$$

## Transposed algorithms (see [Bürgisser, Clausen and Shokrollahi '97])

- $\text{Tr}$  can be easily computed through the *residue formula*.
- *Linear algorithms* can be *transposed* much like linear applications;
- computing  $v \cdot \text{Tr} := (M_v)(\text{Tr}^T)$  is *transposed multiplication*.
- Computing  $\pi^T$  is *transposed push-down*.

## Theorem

Up to some simple formulae:

$$\begin{pmatrix} \pi^{-1} \end{pmatrix} \begin{pmatrix} v \end{pmatrix} \sim \begin{pmatrix} \pi^T \end{pmatrix} \begin{pmatrix} M_v^T \end{pmatrix} \begin{pmatrix} \text{Tr}^T \end{pmatrix}$$

Transposed algorithms (see [Bürgisser, Clausen and Shokrollahi '97])

- $\text{Tr}$  can be easily computed through the *residue formula*.
- *Linear algorithms* can be *transposed* much like linear applications;
- computing  $v \cdot \text{Tr} := (M_v)(\text{Tr}^T)$  is *transposed multiplication*.
- Computing  $\pi^T$  is *transposed push-down*.

## Theorem

Up to some simple formulae:

$$\begin{pmatrix} \pi^{-1} \end{pmatrix} \begin{pmatrix} v \end{pmatrix} \sim \begin{pmatrix} \pi^T \end{pmatrix} \begin{pmatrix} M_v^T \end{pmatrix} \begin{pmatrix} \text{Tr}^T \end{pmatrix}$$

## Transposed algorithms (see [Bürgisser, Clausen and Shokrollahi '97])

- $\text{Tr}$  can be easily computed through the *residue formula*.
- *Linear algorithms* can be *transposed* much like linear applications;
- **computing  $v \cdot \text{Tr} := (M_v)(\text{Tr}^T)$  is *transposed multiplication*.**
- Computing  $\pi^T$  is *transposed push-down*.

## Theorem

Up to some simple formulae:

$$\begin{pmatrix} \pi^{-1} \end{pmatrix} \begin{pmatrix} v \end{pmatrix} \sim \begin{pmatrix} \pi^T \end{pmatrix} \begin{pmatrix} M_v^T \end{pmatrix} \begin{pmatrix} \text{Tr}^T \end{pmatrix}$$

## Transposed algorithms (see [Bürgisser, Clausen and Shokrollahi '97])

- $\text{Tr}$  can be easily computed through the *residue formula*.
- *Linear algorithms* can be *transposed* much like linear applications;
- computing  $v \cdot \text{Tr} := (M_v)(\text{Tr}^T)$  is *transposed multiplication*.
- **Computing  $\pi^T$  is transposed push-down.**

---

## Lift-up

---

**Input**  $v_0, \dots, v_{p-1} \in \mathbb{U}_{i-1}$

**Output**  $v \in \mathbb{U}_i$  s.t.  $v = v_0 + \dots + v_{p-1}x_i^{p-1}$

- 1 Compute the linear form  $\text{Tr} \in \mathbb{U}_i^{D^*}$ ,
  - 2 compute  $\ell = (v_0 + \dots + v_{p-1}x_i^{p-1}) \cdot \text{Tr}$ ,
  - 3 compute  $P_v = \text{Push-down}^T(\ell)$ ,
  - 4 compute  $N_v(Z) = P_v(Z) \cdot \text{rev}(Q_i)(Z) \pmod{Z^{p^i d-1}}$ ,
  - 5 return  $\text{rev}(N_v)/Q'_i \pmod{Q_i}$ .
-

# Speeding up some arithmetics

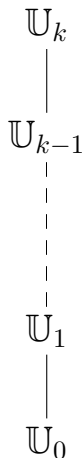
$$\begin{array}{c} \mathbb{U}_k \\ | \\ p \\ \mathbb{U}_{k-1} \\ \vdots \\ \vdots \\ \mathbb{U}_2 \\ | \\ p \\ \mathbb{U}_1 \\ | \\ p \\ \mathbb{F}_q \end{array}$$

Galois action

$$\varphi : \begin{cases} G \times \mathbb{U}_i & \rightarrow \mathbb{U}_i \\ (\sigma, v) & \mapsto \sigma(v) \end{cases}$$

$$G := \text{Gal}(\mathbb{U}_{i+1}/\mathbb{U}_i) \simeq \mathbb{Z}/p\mathbb{Z}$$

# Speeding up some arithmetics



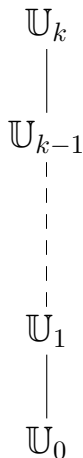
## Divide and conquer

We improve some operations in  $U_i$   $\text{op}(v)$

## Where it works

- traces,
- $p$ -th roots,
- pseudotraces,
- inversion,
- Galois action,
- ...

# Speeding up some arithmetics



## Divide and conquer

We improve some operations in  $\mathbb{U}_i$

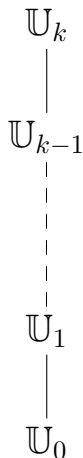
- push-down the operands;

$$\begin{array}{c} \text{op}(v) \\ \downarrow \\ v_0, \dots, v_{p-1} \end{array}$$

## Where it works

- traces,
- $p$ -th roots,
- pseudotraces,
- inversion,
- Galois action,
- ...

# Speeding up some arithmetics



## Divide and conquer

We improve some operations in  $U_i$

- push-down the operands;
- recursively solve  $p$  instances in  $U_{i-1}$ ;

$$\text{op}(v_0), \dots, \text{op}(v_{p-1})$$

$\text{op}(v)$   
↓

## Where it works

- traces,
- $p$ -th roots,
- pseudotraces,
- inversion,
- Galois action,
- ...

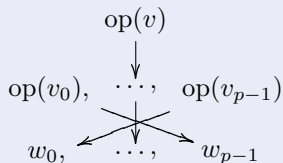
# Speeding up some arithmetics



## Divide and conquer

We improve some operations in  $\mathbb{U}_i$

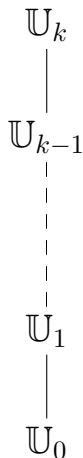
- push-down the operands;
- recursively solve  $p$  instances in  $\mathbb{U}_{i-1}$ ;
- combine the results;



## Where it works

- traces,
- $p$ -th roots,
- pseudotraces,
- inversion,
- Galois action,
- ...

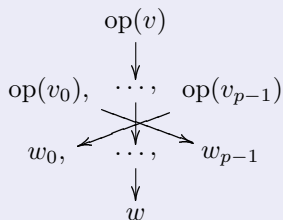
# Speeding up some arithmetics



## Divide and conquer

We improve some operations in  $\mathbb{U}_i$

- push-down the operands;
- recursively solve  $p$  instances in  $\mathbb{U}_{i-1}$ ;
- combine the results;
- lift-up.



## Where it works

- traces,
- $p$ -th roots,
- pseudotraces,
- inversion,
- Galois action,
- ...

# Important application : Isomorphisms with generic towers

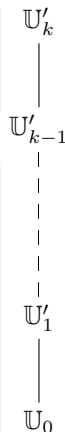
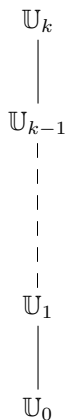
## Generic towers

- Let  $(\alpha_0, \dots, \alpha_{k-1})$  define a generic tower over  $\mathbb{U}_0$ ,
- if we find an isomorphism we can bring fast arithmetics to it.

## Computing the isomorphism [Couveignes '00]

**Goal:** factor  $X^p - X - \alpha_i$  in  $U_{i+1}$ .

- Change of variables  $X' = X - \mu$  s.t.
- $X'^p - X' - \alpha_i$  has a root in  $\mathbb{U}_i$ ,
- Push-down, solve recursively, result is  $\Delta$ ,
- Lift-up  $\Delta$ ,
- return  $\Delta + \mu$ .



# Outline

- 1 Representation
- 2 More arithmetics
- 3 Implementation and benchmarks**

# Implementation

## Implementation in NTL + gf2x

Three types

- GF2:  $p = 2$ , FFT, bit optimisation,
- zz\_p:  $p < 2^{|\text{long}|}$ , FFT, no bit-tricks,
- ZZ\_p: generic  $p$ , like zz\_p but slower.

## Comparison to Magma

Three ways of handling field extensions

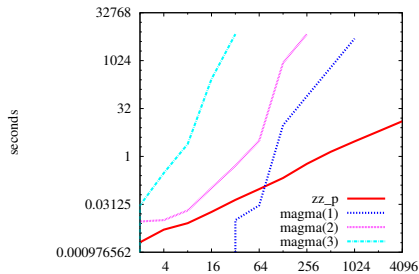
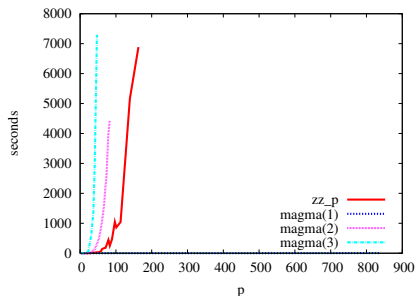
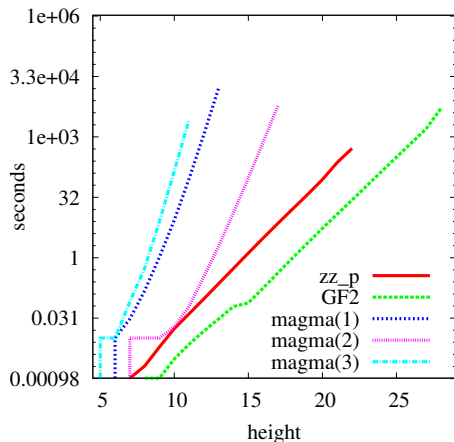
- ①  $\text{quo}\langle U|P \rangle$ : quotient of multivariate polynomial ring + Gröbner bases
- ②  $\text{ext}\langle k|P \rangle$ : field extension by  $X^p - X - \alpha$ , precomputed bases + multivariate
- ③  $\text{ext}\langle k|p \rangle$ : field extension of degree  $p$ , precomputed bases + multivariate

## Benchmarks (on 14 AMD Opteron 2500)

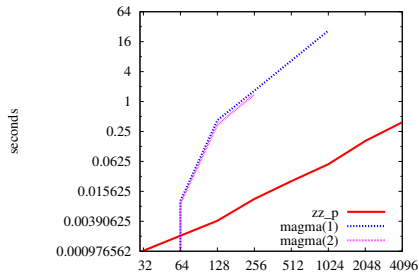
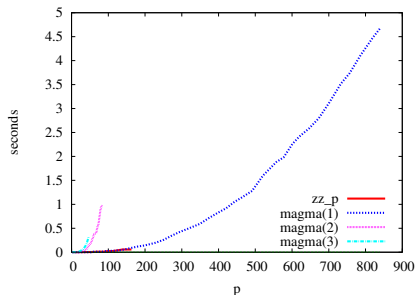
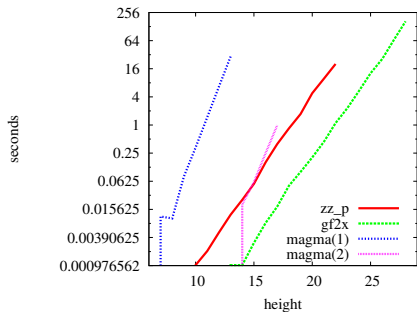
Three modes

- $p = 2$ ,  $d = 1$ , height varying,
- $p$  varying,  $d = 1$ , height = 2,
- $p = 5$ ,  $d$  varying, height = 2.

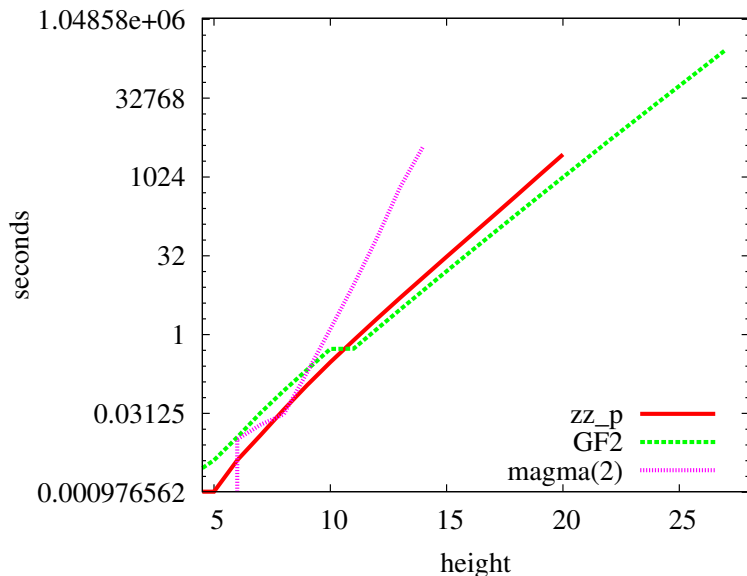
# Construction of the tower + precomputations



# Multiplication

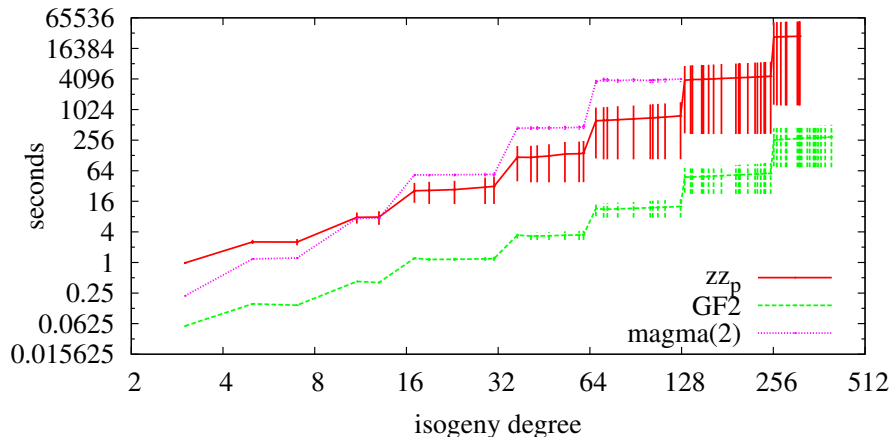


# Isomorphism ([Couveignes '00] vs Magma)



# Benchmarks on isogenies ([Couveignes '96])

Over  $\mathbb{F}_{2^{101}}$ , on an Intel Xeon E5430 Quad Core Processor 2.66GHz, 64GB ram



These algorithms are packaged in a library

Download FAAST at

<http://www.lix.polytechnique.fr/Labo/Luca.De-Feo/FAAST>

We are currently writing an `spkg` for Sage.