Root finding over finite fields using Graeffe transforms

Bruno Grenet
LIRMM
U. Montpellier

Joris van der Hoeven & Grégoire Lecerf
CNRS – LIX
École polytechnique

LJK — Grenoble — March 23., 2017

Root finding over finite fields

Given $f \in \mathbb{F}_q[X]$, compute its roots, that is $\{\alpha \in \mathbb{F}_q : f(\alpha) = 0\}$.

Root finding over finite fields

Given $f \in \mathbb{F}_q[X]$, compute its roots, that is $\{\alpha \in \mathbb{F}_q : f(\alpha) = 0\}$.

- Building block for many algorithms in computer algebra: root finding over $\mathbb{Z}$, factorization, sparse interpolation, ...
- Applications in cryptography, error correcting codes, ...
- Derandomization
- Sparse interpolation: bottleneck in practice

[van der Hoeven & Lecerf, 2014]

Theorem

The roots of $f \in \mathbb{F}_q[X]$ can be computed in **deterministic time** $\mathrm{poly}(q, d)$, where $d = \deg(f)$.

- **Algo:** Test each $\alpha \in \mathbb{F}_q$, sequentially.

Theorem

The roots of $f \in \mathbb{F}_q[X]$ can be computed in **deterministic time** $\mathrm{poly}(q, d)$, where $d = \deg(f)$.

- **Algo:** Test each $\alpha \in \mathbb{F}_q$, sequentially.

⚠ Input size $(1 + d) \log q$: exponential time!

Theorem

The roots of $f \in \mathbb{F}_q[X]$ can be computed in **deterministic time** $\mathrm{poly}(q, d)$, where $d = \deg(f)$.

- **Algo:** Test each $\alpha \in \mathbb{F}_q$, sequentially.

⚠ Input size $(1 + d) \log q$: exponential time!

- Randomization: expected time $(1 - \frac{1}{d+1})q$.

- **No deterministic polytime algorithm is known**

- **No deterministic polytime algorithm is known**
  - $\sqrt{S(p-1)}(d \log p)^{O(1)}$ under ERH [Shoup (1991a)]

- **No deterministic polytime algorithm is known**
  - $\sqrt{S(p-1)}(d \log p)^{O(1)}$ under ERH [Shoup (1991a)]
  - $\sqrt{p}(d \log p)^{O(1)}$ unconditionally [Shoup (1991b)]

- **No deterministic polytime algorithm is known**
  - $\sqrt{S(p-1)}(d \log p)^{O(1)}$ under ERH [Shoup (1991a)]
  - $\sqrt{p}(d \log p)^{O(1)}$ unconditionally [Shoup (1991b)]
  - Both require the factorization of $(p-1)$

- **No deterministic polytime algorithm is known**
  - $\sqrt{S(p-1)}(d \log p)^{O(1)}$ under ERH [Shoup (1991a)]
  - $\sqrt{p}(d \log p)^{O(1)}$ unconditionally [Shoup (1991b)]
  - Both require the factorization of $(p-1)$
  - ERH useless if a primitive element is given

- **No deterministic polytime algorithm is known**
  - $\sqrt{S(p-1)}(d \log p)^{O(1)}$ under ERH [Shoup (1991a)]
  - $\sqrt{p}(d \log p)^{O(1)}$ unconditionally [Shoup (1991b)]
  - Both require the factorization of $(p-1)$
  - ERH useless if a primitive element is given
  - Root finding $\iff$ primitive element [von zur Gathen (1987)]

- **No deterministic polytime algorithm is known**
  - $\sqrt{S(p-1)}(d \log p)^{O(1)}$ under ERH [Shoup (1991a)]
  - $\sqrt{p}(d \log p)^{O(1)}$ unconditionally [Shoup (1991b)]
  - Both require the factorization of $(p-1)$
  - ERH useless if a primitive element is given
  - Root finding $\Longleftrightarrow$ primitive element [von zur Gathen (1987)]
- Randomized algorithm in **expected time** $\tilde{O}(d \log^2 q)$

  [Legendre (1785), Berlekamp (1970), Rabin (1980)]

- **No deterministic polytime algorithm is known**
  - $\sqrt{S(p-1)}(d \log p)^{O(1)}$ under ERH [Shoup (1991a)]
  - $\sqrt{p}(d \log p)^{O(1)}$ unconditionally [Shoup (1991b)]
  - Both require the factorization of $(p-1)$
  - ERH useless if a primitive element is given
  - Root finding $\Longleftrightarrow$ primitive element [von zur Gathen (1987)]
- Randomized algorithm in **expected time** $\tilde{O}(d \log^2 q)$

  [Legendre (1785), Berlekamp (1970), Rabin (1980)]
- Factorization algorithms $\rightsquigarrow$ no improvement for root finding

  [Cantor-Zassenhaus (1981), Kaltofen-Shoup (1998), Kedlaya-Umans (2011)]

- **No deterministic polytime algorithm is known**
  - $\sqrt{S(p-1)}(d \log p)^{O(1)}$ under ERH [Shoup (1991a)]
  - $\sqrt{p}(d \log p)^{O(1)}$ unconditionally [Shoup (1991b)]
  - Both require the factorization of $(p-1)$
  - ERH useless if a primitive element is given
  - Root finding $\iff$ primitive element [von zur Gathen (1987)]
- Randomized algorithm in **expected time** $\tilde{O}(d \log^2 q)$

  [Legendre (1785), Berlekamp (1970), Rabin (1980)]
- Factorization algorithms $\rightsquigarrow$ no improvement for root finding

  [Cantor-Zassenhaus (1981), Kaltofen-Shoup (1998), Kedlaya-Umans (2011)]
- Better complexity bounds when $q-1$ is *smooth*

  [Moenck (1977), von zur Gathen (1987), Mignotte-Schnorr (1988), Rónyai (1989), Shoup (1992), Źrałek (2010), ...]

Goal

Obtain **fast** algorithms for polynomial root finding in finite fields.

- Deterministic, probabilistic, heuristic; in practice or in theory.

Goal

Obtain **fast** algorithms for polynomial root finding in finite fields.

- Deterministic, probabilistic, heuristic; in practice or in theory.
- **Assumption:** f is **monic**, **separable**, **splits** over $\mathbb{F}_q$, $f(0) \neq 0$:

$$f(X) = \prod_{i=1}^{d} (X - \alpha_i), \quad \alpha_i \in \mathbb{F}_q^*, \quad \alpha_i \neq \alpha_j \qquad \text{(A)}$$

(easy reduction: $f \leftarrow \gcd(f, X^{q-1} - 1)$)

Goal

Obtain **fast** algorithms for polynomial root finding in finite fields.

- Deterministic, probabilistic, heuristic; in practice or in theory.
- **Assumption:** f is **monic**, **separable**, **splits** over $\mathbb{F}_q$, $f(0) \neq 0$:

$$f(X) = \prod_{i=1}^{d} (X - \alpha_i), \quad \alpha_i \in \mathbb{F}_q^*, \quad \alpha_i \neq \alpha_j \qquad \text{(A)}$$

(easy reduction: $f \leftarrow \gcd(f, X^{q-1} - 1)$)

- **Extra input:** A primitive element ζ of $\mathbb{F}_q^*$

Goal

Obtain **fast** algorithms for polynomial root finding in finite fields.

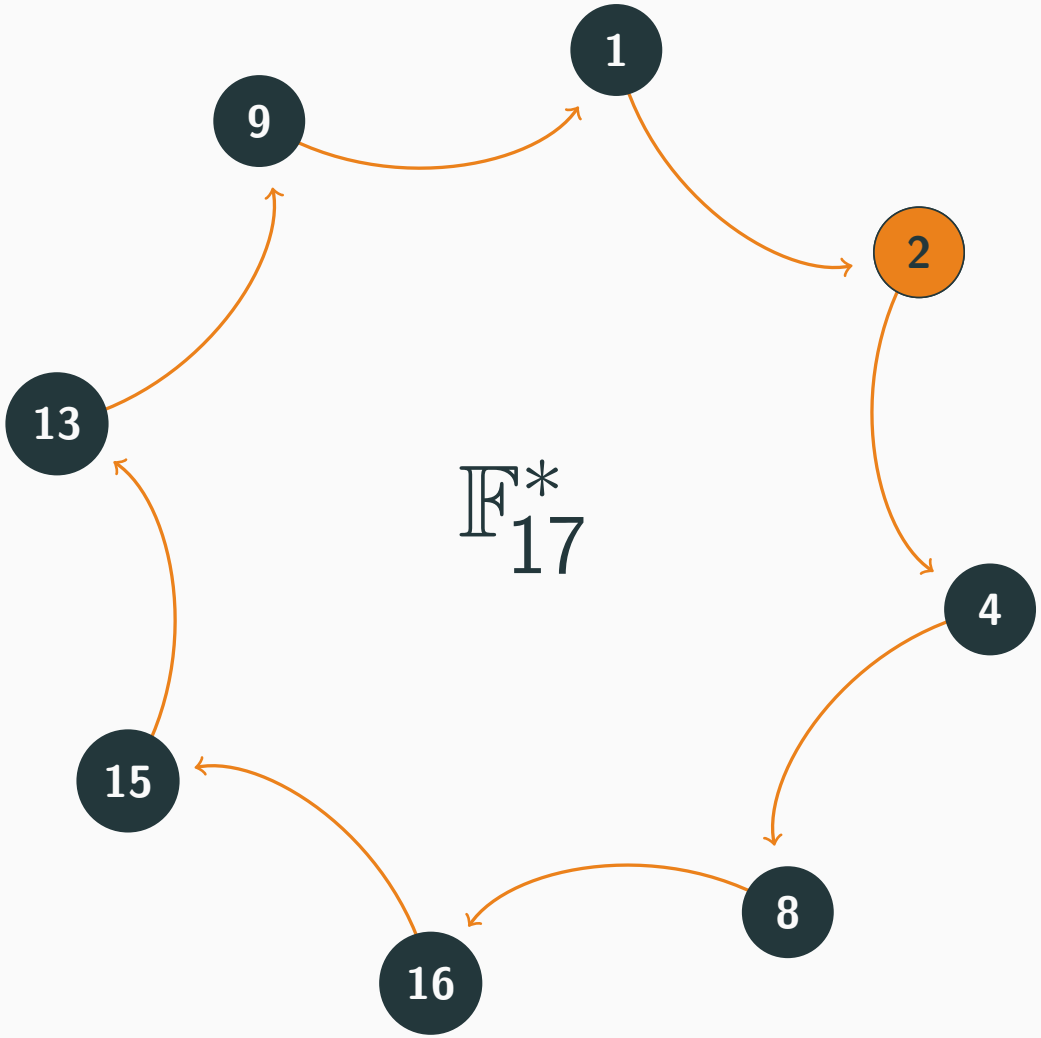
- Deterministic, probabilistic, heuristic; in practice or in theory.
- **Assumption:** f is **monic**, **separable**, **splits** over $\mathbb{F}_q$, $f(0) \neq 0$:

$$f(X) = \prod_{i=1}^{d} (X - \alpha_i), \quad \alpha_i \in \mathbb{F}_q^*, \quad \alpha_i \neq \alpha_j \qquad \text{(A)}$$

(easy reduction: $f \leftarrow \gcd(f, X^{q-1} - 1)$)

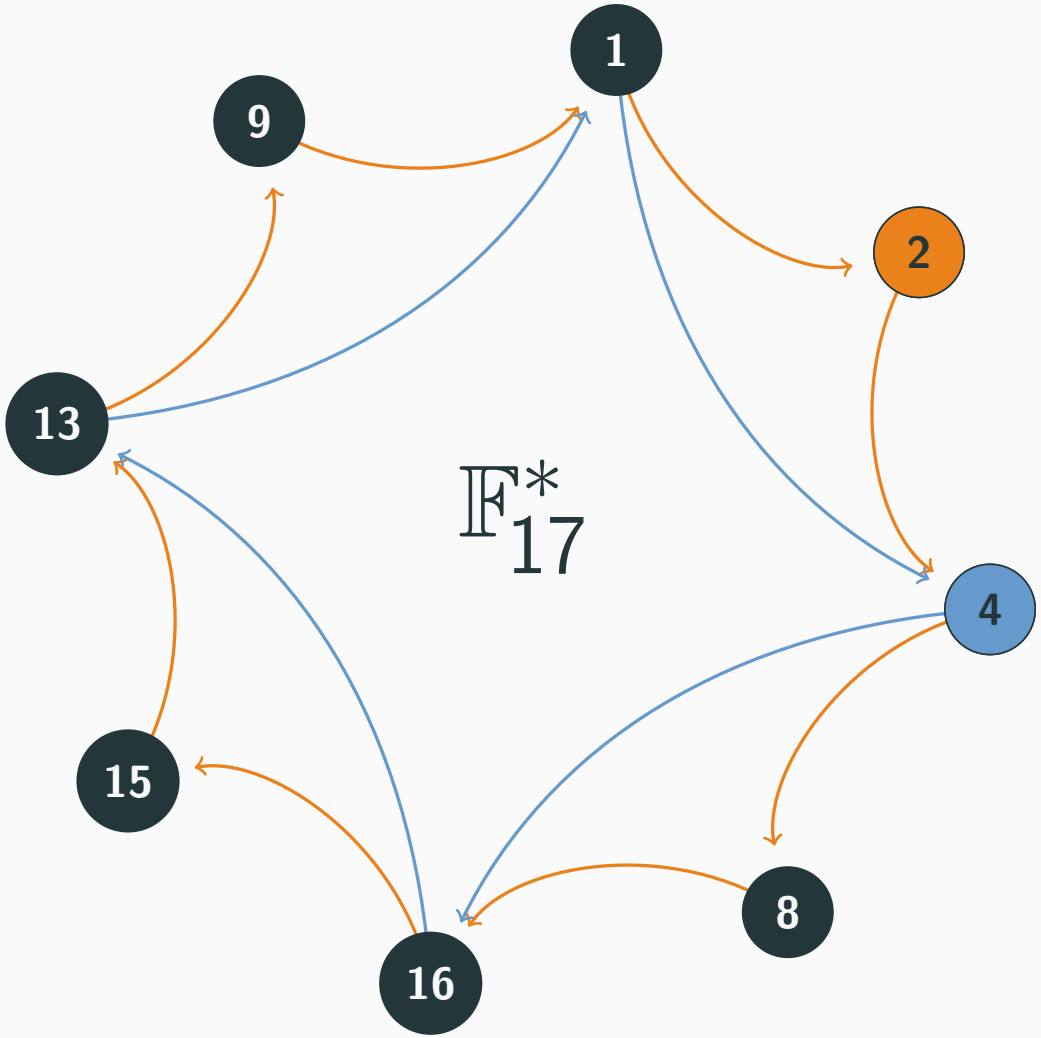
- **Extra input:** A primitive element ζ of $\mathbb{F}_q^*$
- **FFT finite field**: $\mathbb{F}_p$ with $p = M \cdot 2^m + 1$ and $M = O(\log p)$
  - Old algorithms revisited
  - New technique based on **Graeffe transforms**
  - Fast implementations

Multiplicative structure of finite fields



2 is a primitive root of unity of order 8

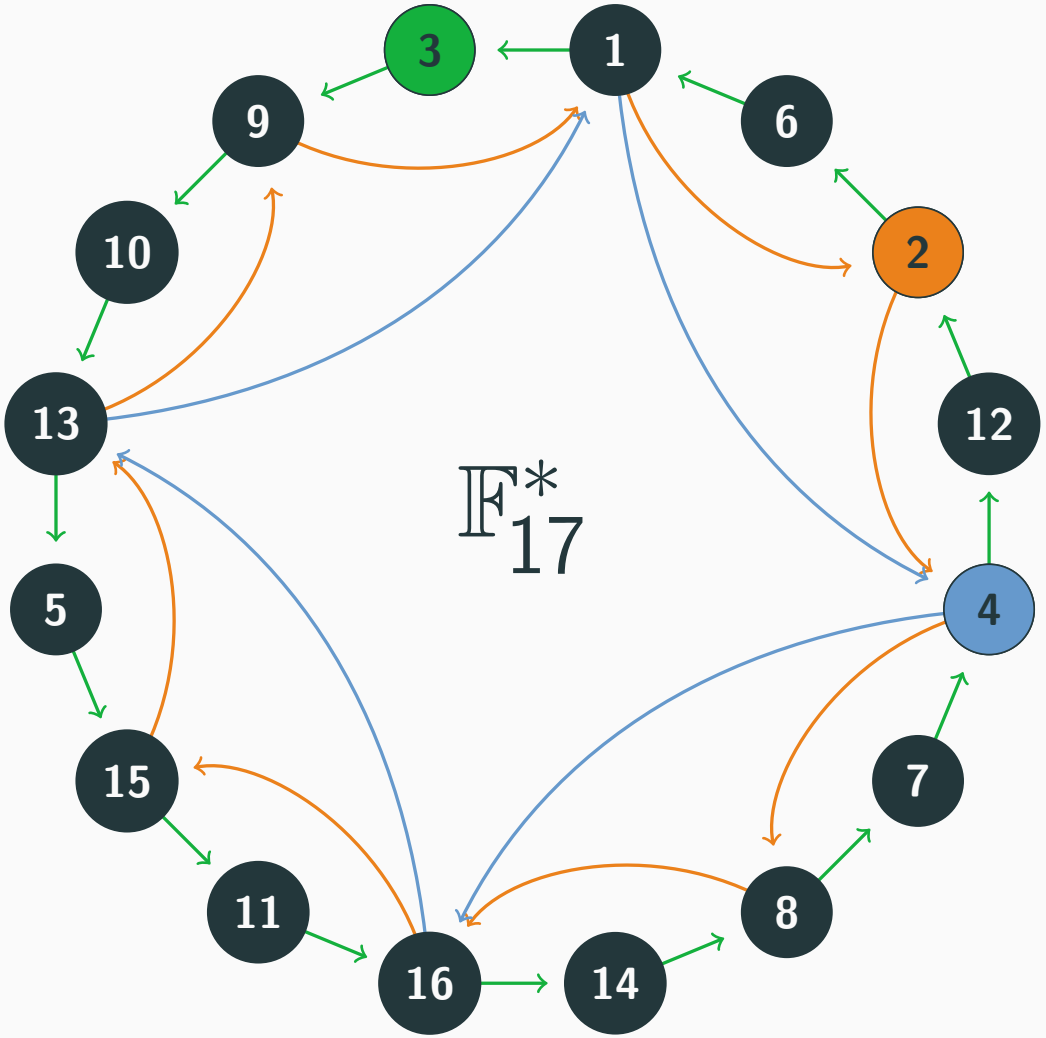
Multiplicative structure of finite fields



2 is a primitive root of unity of order 8

4 is a primitive root of unity of order 4

Multiplicative structure of finite fields



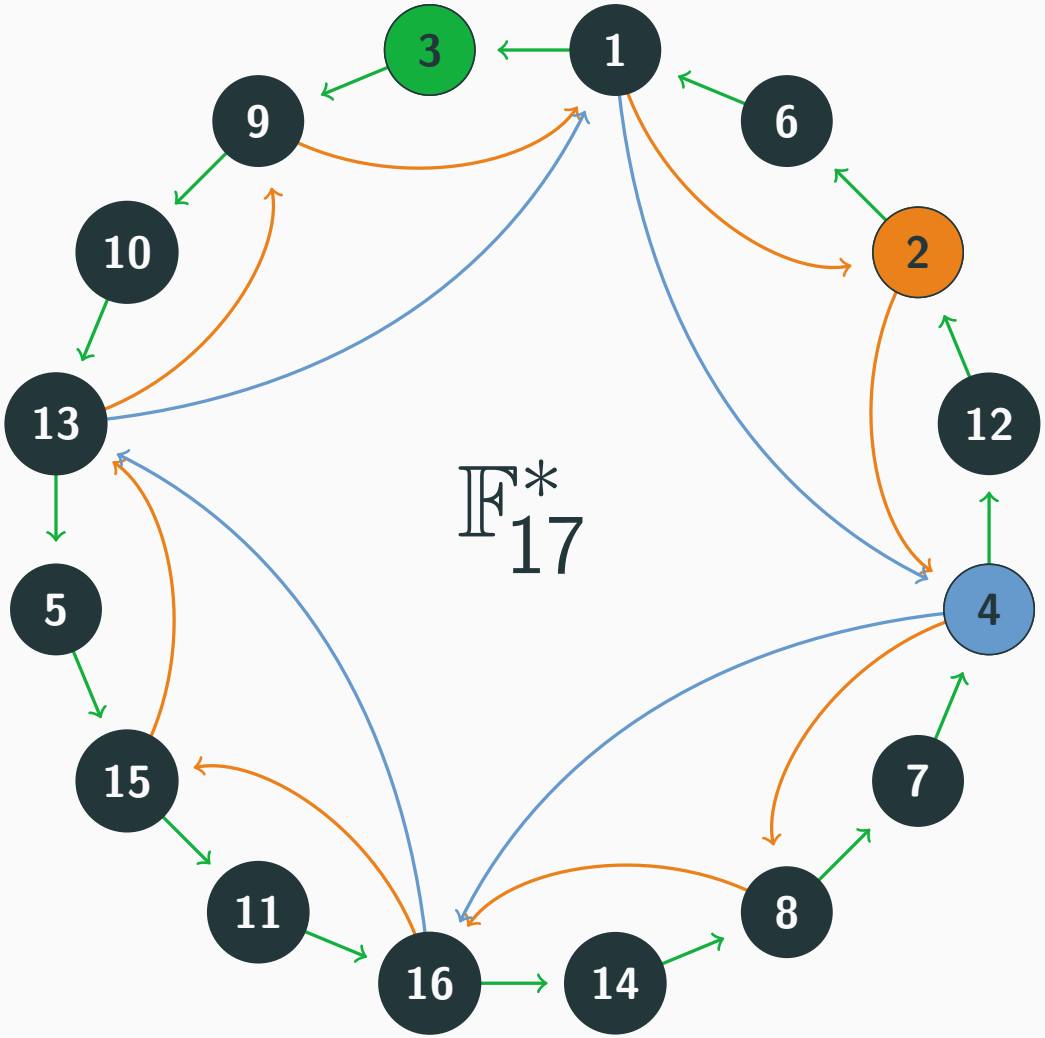
2 is a primitive root of unity of order 8

4 is a primitive root of unity of order 4

3 is a primitive element of $\mathbb{F}_{17}^*$:

$\mathbb{F}_{17}^* = \{3^i : 0 \leq i < 16\}$

Multiplicative structure of finite fields



2 is a primitive root of unity of order 8

4 is a primitive root of unity of order 4

3 is a primitive element of $\mathbb{F}_{17}^*$:
$\mathbb{F}_{17}^* = \{3^i : 0 \leq i < 16\}$

$$X^{q-1} - 1 = \prod_{\alpha \in \mathbb{F}_q^*} (X - \alpha)$$

Old algorithms revisited

Goal

- Many known randomized algorithms
  - for the general case
  - for special cases such as smooth $(q-1)$

Goal

- Many known randomized algorithms
  - for the general case
  - for special cases such as smooth $(q-1)$
- Idea: Adapt these algorithms for FFT finite fields

Goal

- Many known randomized algorithms
  - for the general case
  - for special cases such as smooth $(q-1)$
- Idea: Adapt these algorithms for FFT finite fields
- Algorithms:
  - Rabin's algorithm (general case)
  - Mignotte-Schnorr's algorithm (smooth case)
  - Moenck's algorithm (FFT finite fields)

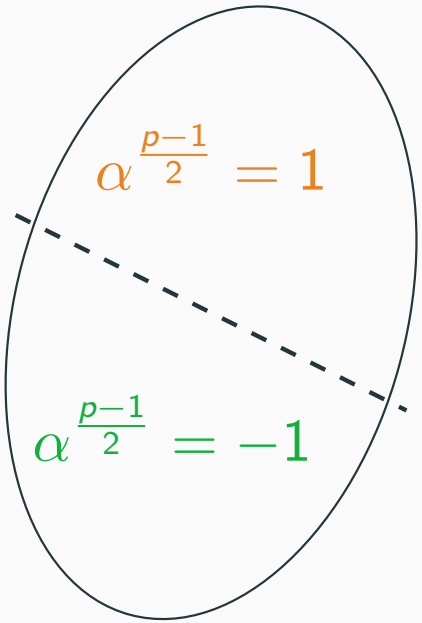
Goal

- Many known randomized algorithms
  - for the general case
  - for special cases such as smooth $(q-1)$
- Idea: Adapt these algorithms for FFT finite fields
- Algorithms:
  - Rabin's algorithm (general case)
  - Mignotte-Schnorr's algorithm (smooth case)
  - Moenck's algorithm (FFT finite fields)

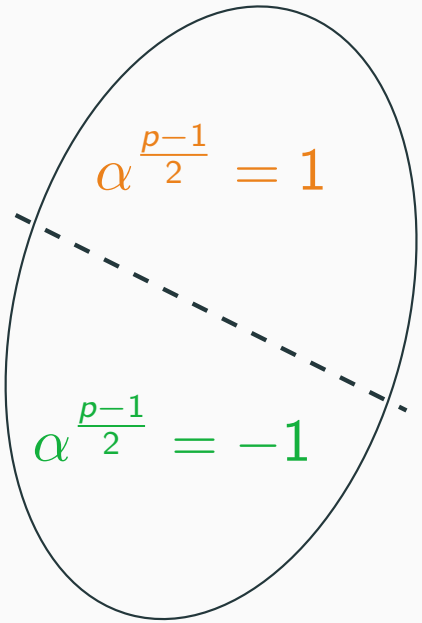
- $\prod_{\alpha \in \mathbb{F}_p^*} (X - \alpha) = X^{p-1} - 1$

- $\prod_{\alpha \in \mathbb{F}_p^*} (X - \alpha) = X^{p-1} - 1 = (X^{\frac{p-1}{2}} - 1)(X^{\frac{p-1}{2}} + 1)$

- $\prod_{\alpha \in \mathbb{F}_p^*} (X - \alpha) = X^{p-1} - 1 = (X^{\frac{p-1}{2}} - 1)(X^{\frac{p-1}{2}} + 1)$

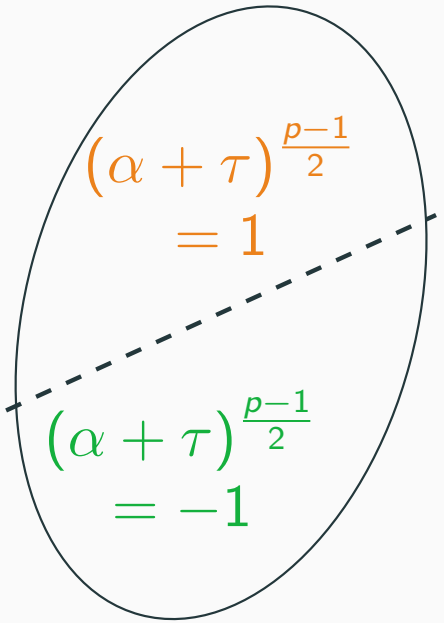


- $\prod_{\alpha \in \mathbb{F}_p^*} (X - \alpha) = X^{p-1} - 1 = (X^{\frac{p-1}{2}} - 1)(X^{\frac{p-1}{2}} + 1)$



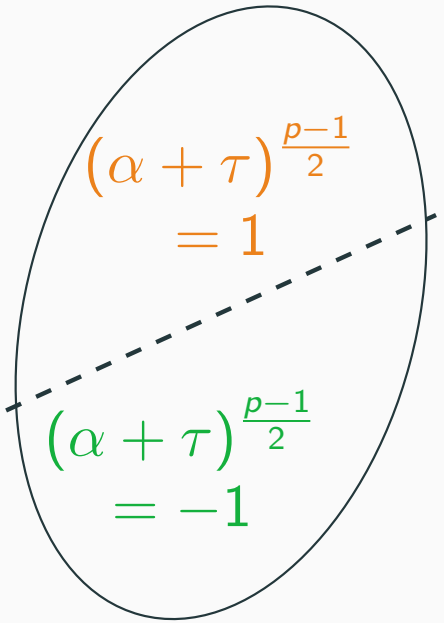
  - With some luck, $\gcd(f, X^{\frac{p-1}{2}} - 1) \notin \{1, f\}$.

- $\prod_{\alpha \in \mathbb{F}_p^*} (X - \alpha) = X^{p-1} - 1 = (X^{\frac{p-1}{2}} - 1)(X^{\frac{p-1}{2}} + 1)$



- With some luck, $\gcd(f, X^{\frac{p-1}{2}} - 1) \notin \{1, f\}$.
- Push your luck: $\gcd(f, (X + \tau)^{\frac{p-1}{2}} - 1)$ for some **random** $\tau \in \mathbb{F}_p$

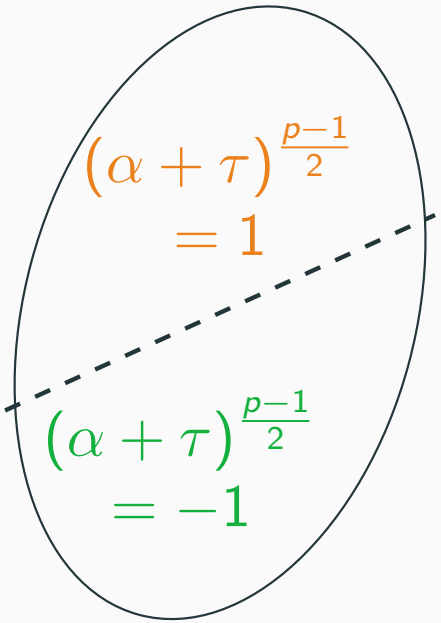
- $\prod_{\alpha \in \mathbb{F}_p^*} (X - \alpha) = X^{p-1} - 1 = (X^{\frac{p-1}{2}} - 1)(X^{\frac{p-1}{2}} + 1)$



- With some luck, $\gcd(f, X^{\frac{p-1}{2}} - 1) \notin \{1, f\}$.
- Push your luck: $\gcd(f, (X + \tau)^{\frac{p-1}{2}} - 1)$ for some **random** $\tau \in \mathbb{F}_p$

$$\deg\left(\gcd(f, (X + \tau)^{\frac{p-1}{2}} - 1)\right) \simeq d/2$$

- $\prod_{\alpha \in \mathbb{F}_p^*} (X - \alpha) = X^{p-1} - 1 = (X^{\frac{p-1}{2}} - 1)(X^{\frac{p-1}{2}} + 1)$



- With some luck, $\gcd(f, X^{\frac{p-1}{2}} - 1) \notin \{1, f\}$.
- Push your luck: $\gcd(f, (X + \tau)^{\frac{p-1}{2}} - 1)$ for some **random** $\tau \in \mathbb{F}_p$

$$\deg\left(\gcd(f, (X + \tau)^{\frac{p-1}{2}} - 1)\right) \simeq d/2$$

Randomized algorithm

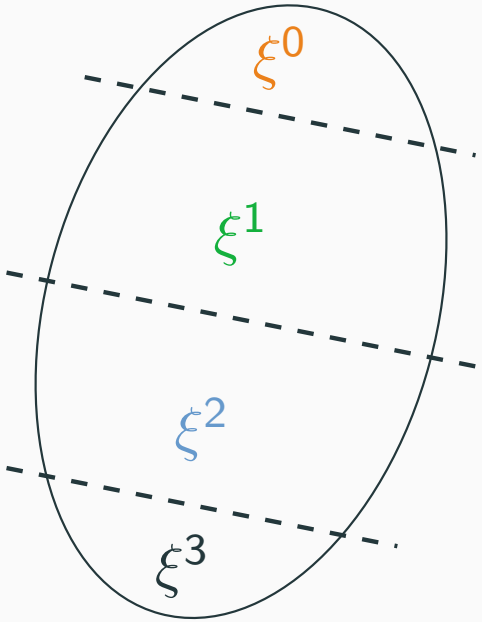
The roots of $f \in \mathbb{F}_p[X]$ can be computed in expected time $\tilde{O}(d \log^2 p)$.

Modified Rabin's algorithm (for FFT finite fields)

$$X^{p-1} - 1 = \prod_{i=0}^{2^\ell - 1} (X^{M2^{m-\ell}} - \xi^i), \text{ where } \xi \text{ is primitive of order } 2^\ell.$$

Modified Rabin's algorithm (for FFT finite fields)

$$X^{p-1} - 1 = \prod_{i=0}^{2^\ell - 1} (X^{M2^{m-\ell}} - \xi^i), \text{ where } \xi \text{ is primitive of order } 2^\ell.$$



$\gcd(f, (X+\tau)^{M2^{m-\ell}} - \xi^0)$

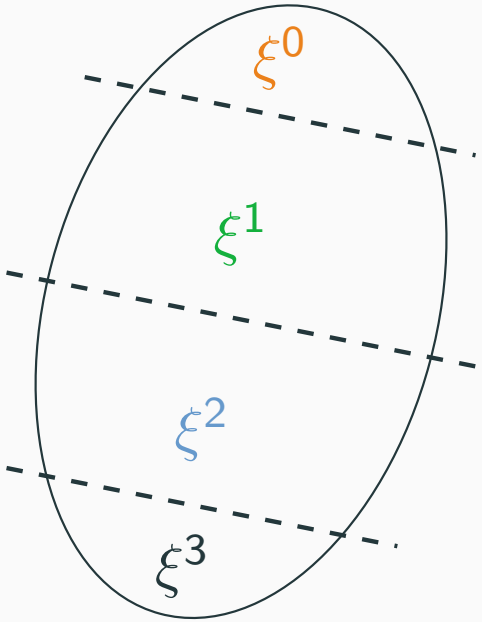
$\gcd(f, (X+\tau)^{M2^{m-\ell}} - \xi^1)$

$\gcd(f, (X+\tau)^{M2^{m-\ell}} - \xi^2)$

$\gcd(f, (X+\tau)^{M2^{m-\ell}} - \xi^3)$

Modified Rabin's algorithm (for FFT finite fields)

$$X^{p-1} - 1 = \prod_{i=0}^{2^\ell - 1} (X^{M2^{m-\ell}} - \xi^i), \text{ where } \xi \text{ is primitive of order } 2^\ell.$$



$\gcd(f, (X+\tau)^{M2^{m-\ell}} - \xi^0)$

$\gcd(f, (X+\tau)^{M2^{m-\ell}} - \xi^1)$

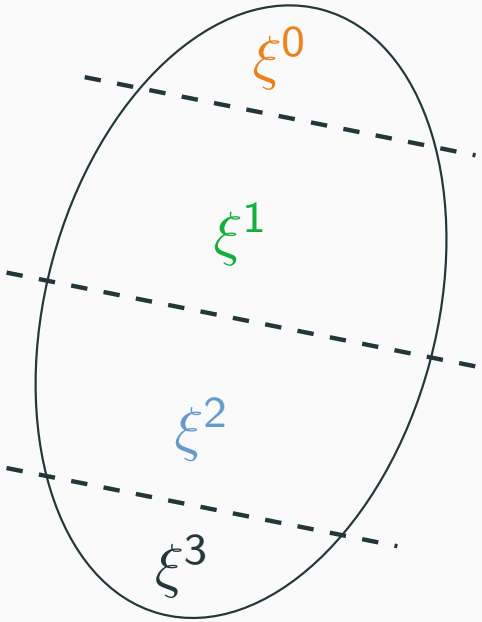
$\gcd(f, (X+\tau)^{M2^{m-\ell}} - \xi^2)$

$\gcd(f, (X+\tau)^{M2^{m-\ell}} - \xi^3)$

degrees
$\simeq d/2^\ell$

Modified Rabin's algorithm (for FFT finite fields)

$$X^{p-1} - 1 = \prod_{i=0}^{2^\ell - 1} (X^{M2^{m-\ell}} - \xi^i), \text{ where } \xi \text{ is primitive of order } 2^\ell.$$



$\gcd(f, (X+\tau)^{M2^{m-\ell}} - \xi^0)$

$\gcd(f, (X+\tau)^{M2^{m-\ell}} - \xi^1)$

$\gcd(f, (X+\tau)^{M2^{m-\ell}} - \xi^2)$

$\gcd(f, (X+\tau)^{M2^{m-\ell}} - \xi^3)$

degrees $\simeq d/2^\ell$

Worthwhile in practice for **small** $\ell = 2, 3, \ldots$

New technique: Graeffe transform

Let $f(X) = \prod_i (X - \alpha_i)$

$$f(X)f(-X) = \prod_i (X - \alpha_i)(-X - \alpha_i) = (-1)^d \prod_i (X^2 - \alpha_i^2)$$

Let $f(X) = \prod_i (X - \alpha_i)$

$$f(X)f(-X) = \prod_i (X - \alpha_i)(-X - \alpha_i) = (-1)^d \prod_i (X^2 - \alpha_i^2)$$

Definition

$G_2(f)(X) = \prod_i (X - \alpha_i^2)$ is the **Graeffe transform** of f.

Let $f(X) = \prod_i (X - \alpha_i)$

$$f(X)f(-X) = \prod_i (X - \alpha_i)(-X - \alpha_i) = (-1)^d \prod_i (X^2 - \alpha_i^2)$$

Definition

$G_2(f)(X) = \prod_i (X - \alpha_i^2)$ is the **Graeffe transform** of f.

$G_\rho(f)(X) = \prod_i (X - \alpha_i^\rho)$ is the **Graeffe transform of order** ρ of f.

Let $f(X) = \prod_i (X - \alpha_i)$

$$f(X)f(-X) = \prod_i (X - \alpha_i)(-X - \alpha_i) = (-1)^d \prod_i (X^2 - \alpha_i^2)$$

Definition

$G_2(f)(X) = \prod_i (X - \alpha_i^2)$ is the **Graeffe transform** of f.

$G_\rho(f)(X) = \prod_i (X - \alpha_i^\rho)$ is the **Graeffe transform of order** ρ of f.

Remarks:

- $G_{\rho_1\rho_2} = G_{\rho_1} \circ G_{\rho_2}$
- $G_{q-1}(f)(X) = \prod_i (X - \alpha_i^{q-1}) = (X-1)^d$

Lemma

Let ξ be a primitive root of unity of order π. Then

$$G_\pi(f)(X^\pi) = f(X)f(\xi X)\cdots f(\xi^{\pi-1}X).$$

$\rightsquigarrow$ Given ξ, $G_\pi(f)$ can be computed in time $\tilde{O}(\pi d \log q)$

Lemma

Let ξ be a primitive root of unity of order π. Then

$$G_\pi(f)(X^\pi) = f(X)f(\xi X)\cdots f(\xi^{\pi-1}X).$$

$\rightsquigarrow$ Given ξ, $G_\pi(f)$ can be computed in time $\tilde{O}(\pi d \log q)$

Theorem

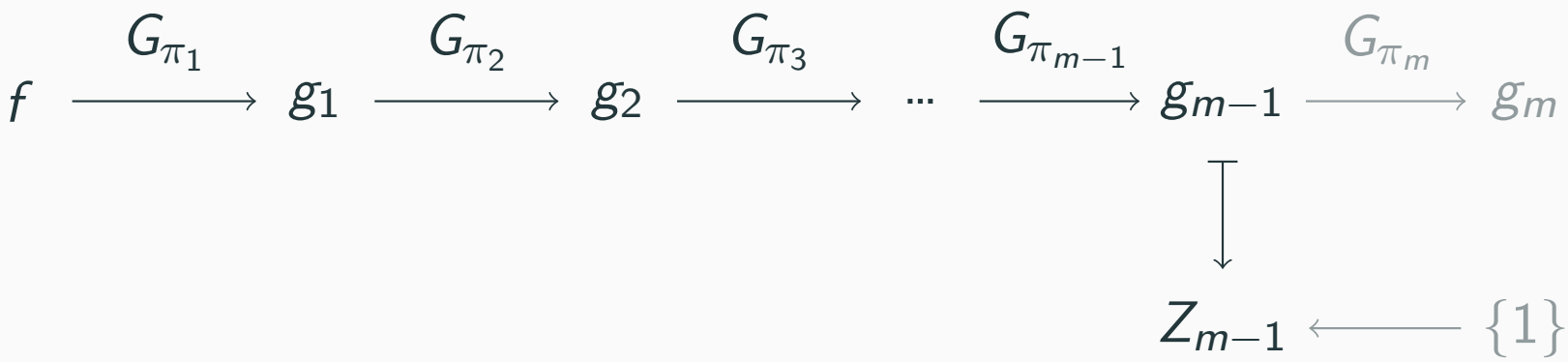
Let $f \in \mathbb{F}_q[X]$ and $q-1 = \pi_1 \cdots \pi_m$. The Graeffe transforms $G_{\pi_1}(f)$, $G_{\pi_1\pi_2}(f)$, ..., $G_{\pi_1\cdots\pi_{m-1}}(f)$ can be computed in time $(d\log^2 q)^{1+\delta}$ for all $\delta > 0$.

Based on modular composition [Kedlaya-Umans (2011)].

Using Graeffe transforms

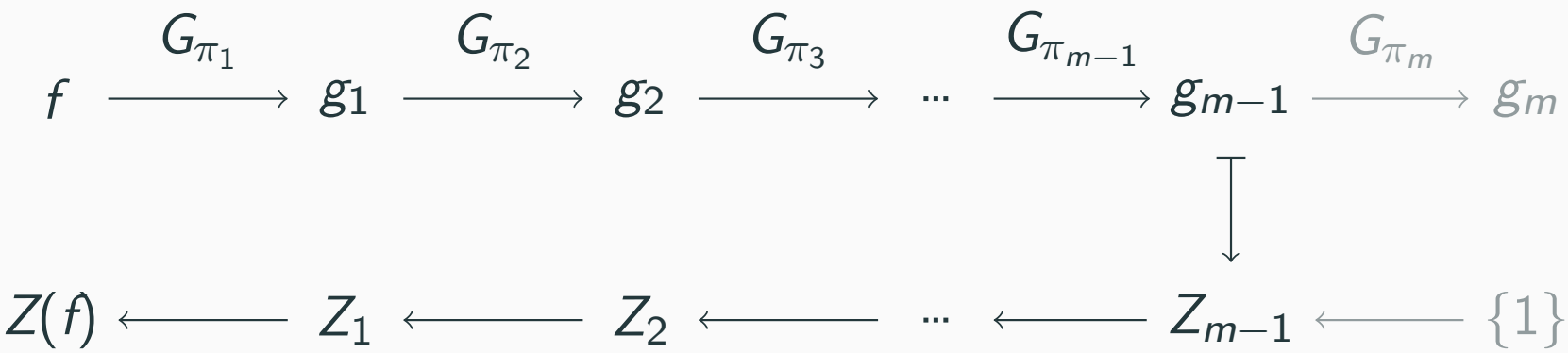
$$f \xrightarrow{G_{\pi_1}} g_1 \xrightarrow{G_{\pi_2}} g_2 \xrightarrow{G_{\pi_3}} \cdots \xrightarrow{G_{\pi_{m-1}}} g_{m-1} \xrightarrow{G_{\pi_m}} g_m$$

Using Graeffe transforms

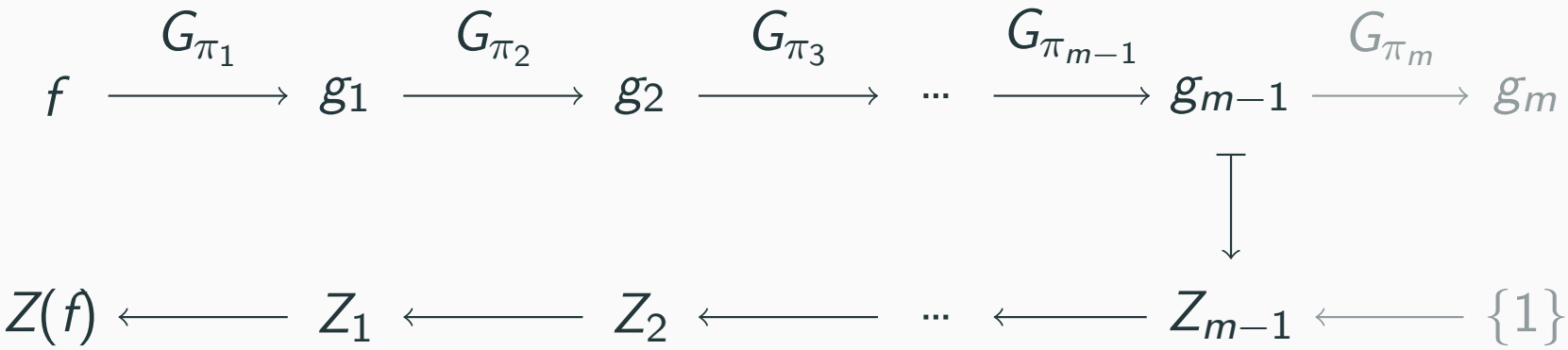


- $Z_{m-1} \subseteq \{\xi^i : 0 \leq i < \pi_m\}$ (ξ: primitive root of order π_m)
  $= \{\zeta^{(q-1)i/\pi_m} : 0 \leq i < \pi_m\}$ (ζ: primitive element of $\mathbb{F}_q^*$)

Using Graeffe transforms

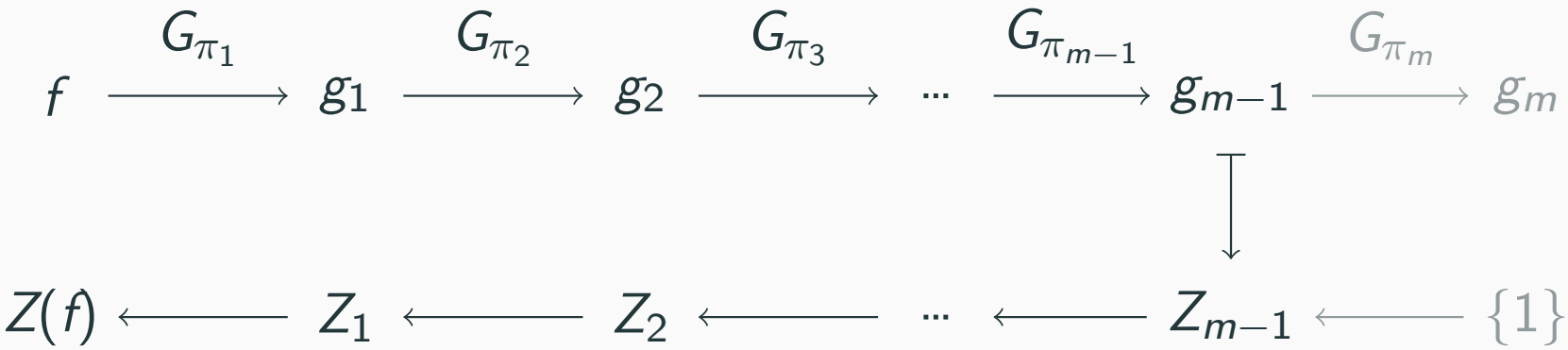


- $Z_{m-1} \subseteq \{\xi^i : 0 \le i < \pi_m\}$ (ξ: primitive root of order π_m)
  $= \{\zeta^{(q-1)i/\pi_m} : 0 \le i < \pi_m\}$ (ζ: primitive element of $\mathbb{F}_q^*$)



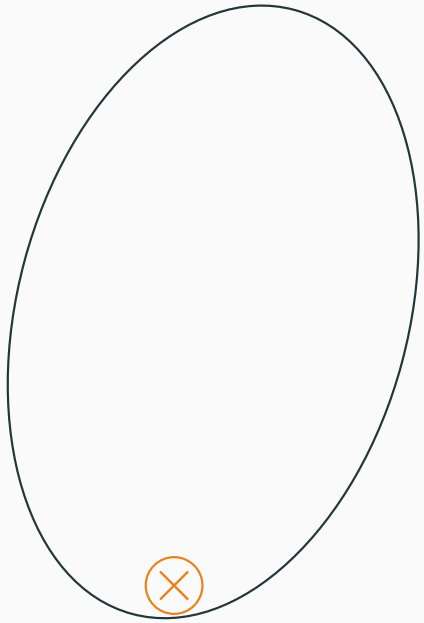
- $Z_{m-1} \subseteq \{\xi^i : 0 \leq i < \pi_m\}$ (ξ: primitive root of order π_m)
  $= \{\zeta^{(q-1)i/\pi_m} : 0 \leq i < \pi_m\}$ (ζ: primitive element of $\mathbb{F}_q^*$)
- If $\beta \in Z_k$, there exists $\alpha \in Z_{k-1}$ s.t. $\alpha^{\pi_k} = \beta$

Using Graeffe transforms



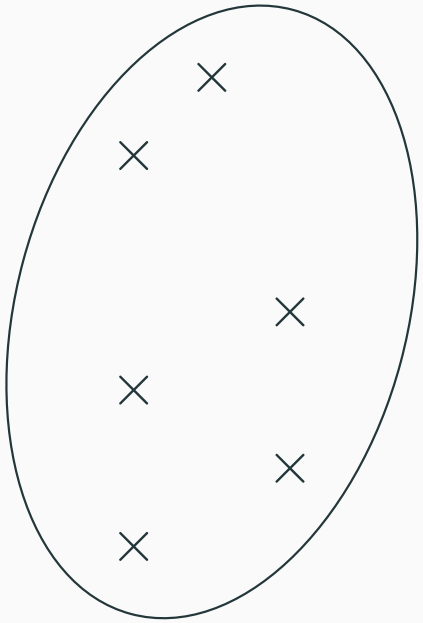
- $Z_{m-1} \subseteq \{\xi^i : 0 \leq i < \pi_m\}$ (ξ: primitive root of order π_m)
  $= \{\zeta^{(q-1)i/\pi_m} : 0 \leq i < \pi_m\}$ (ζ: primitive element of $\mathbb{F}_q^*$)
- If $\beta \in Z_k$, there exists $\alpha \in Z_{k-1}$ s.t. $\alpha^{\pi_k} = \beta$
- If $\beta = \zeta^e$, $\alpha \in \left\{\zeta^{\frac{e+(q-1)i}{\pi_k}} : 0 \leq i < \pi_k\right\}$

A deterministic algorithm

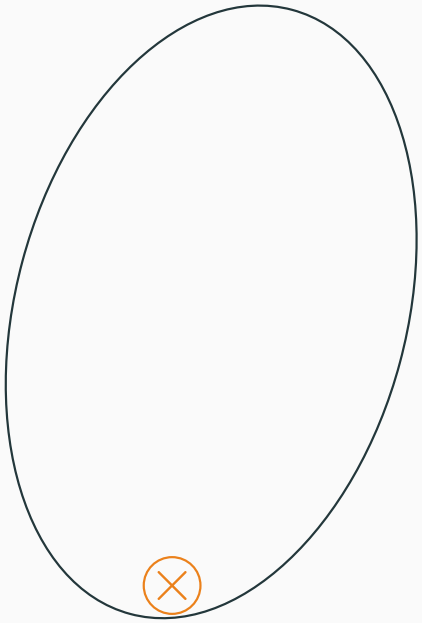


$$Z_m = \{\zeta^0\}$$

A deterministic algorithm

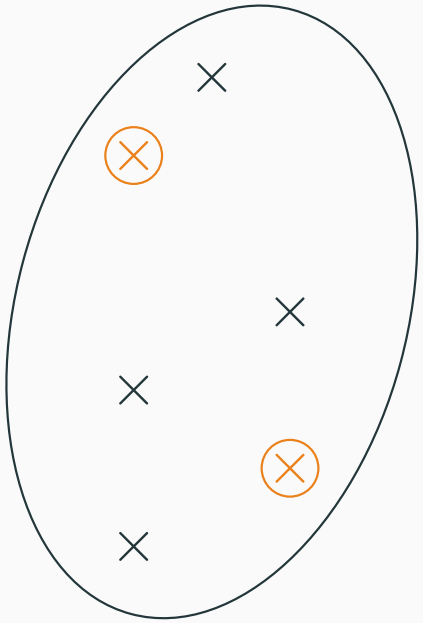


$$\{\zeta^{\frac{(q-1)i}{\pi_m}} : 0 \leq i < \pi_m\}$$



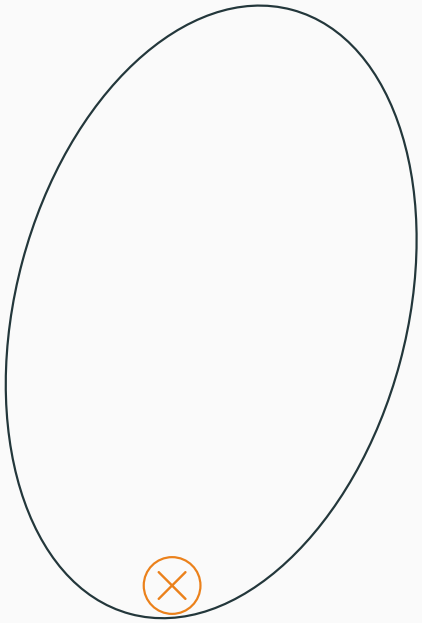
$$Z_m = \{\zeta^0\}$$

A deterministic algorithm



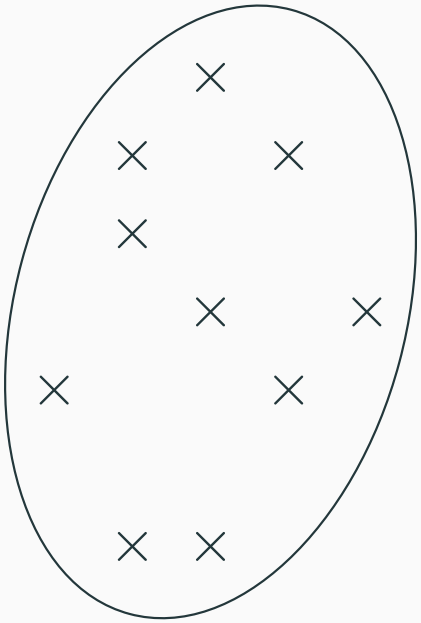
$$\{\zeta^{\frac{(q-1)i}{\pi_m}} : 0 \leq i < \pi_m\}$$

$$g_{m-1}(\zeta^e) = 0$$

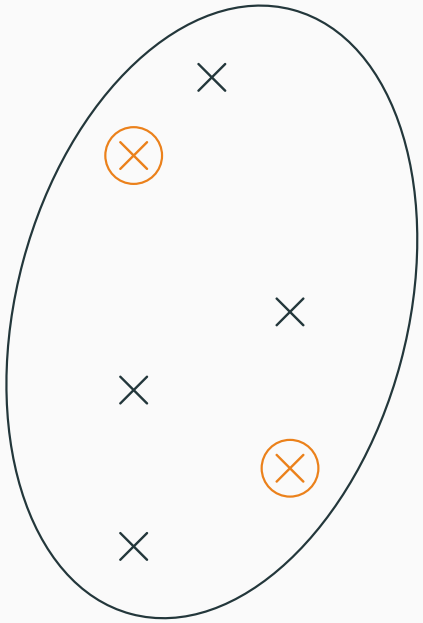


$$Z_m = \{\zeta^0\}$$

A deterministic algorithm

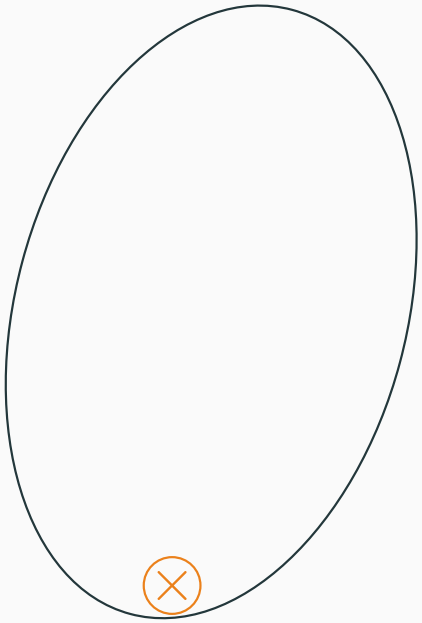


$\{\zeta^{\frac{e+(q-1)i}{\pi_{m-1}}} : 0 \leq i < \pi_{m-1}\}$



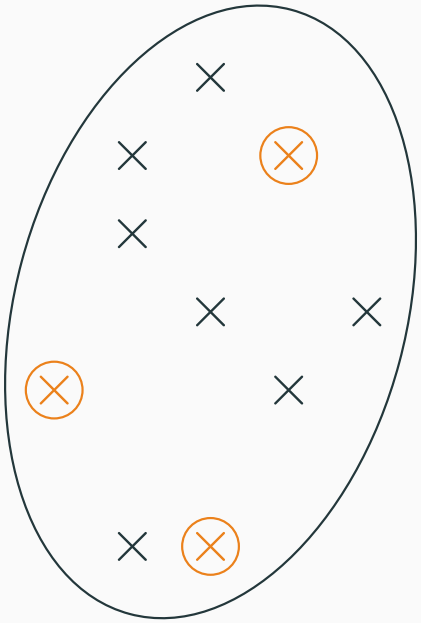
$\{\zeta^{\frac{(q-1)i}{\pi_m}} : 0 \leq i < \pi_m\}$

$g_{m-1}(\zeta^e) = 0$



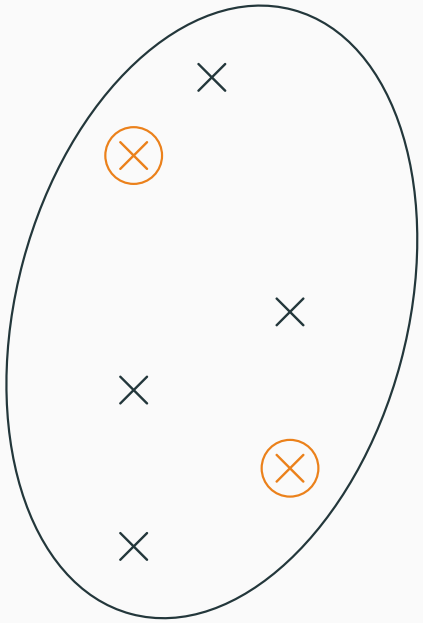
$Z_m = \{\zeta^0\}$

A deterministic algorithm



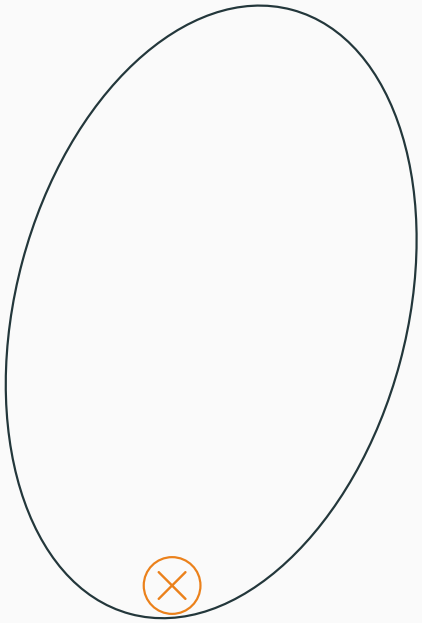
$\{\zeta^{\frac{e+(q-1)i}{\pi_{m-1}}} : 0 \leq i < \pi_{m-1}\}$

$g_{m-2}(\zeta^e) = 0$



$\{\zeta^{\frac{(q-1)i}{\pi_m}} : 0 \leq i < \pi_m\}$

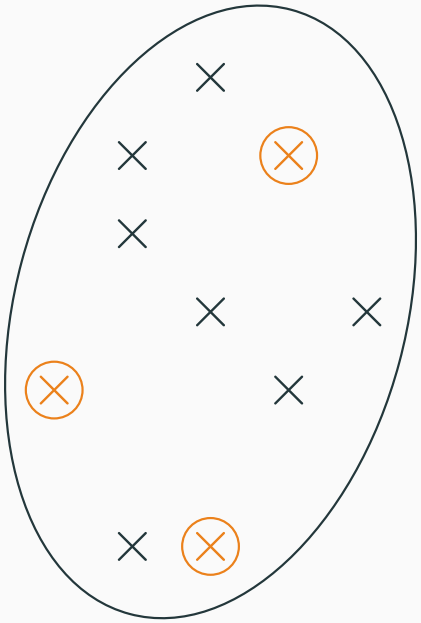
$g_{m-1}(\zeta^e) = 0$



$Z_m = \{\zeta^0\}$

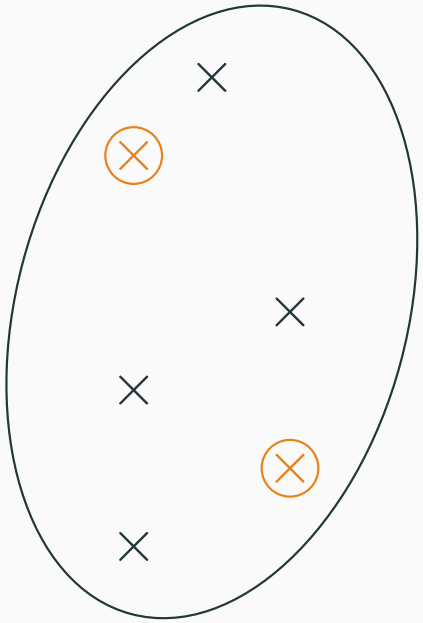
A deterministic algorithm

...



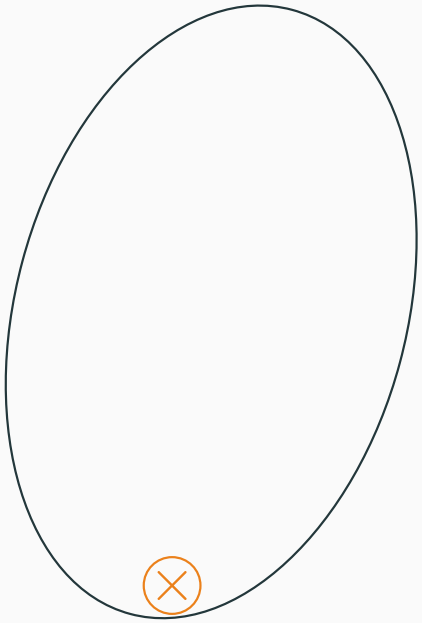
$\{\zeta^{\frac{e+(q-1)i}{\pi_{m-1}}} : 0 \leq i < \pi_{m-1}\}$

$g_{m-2}(\zeta^e) = 0$



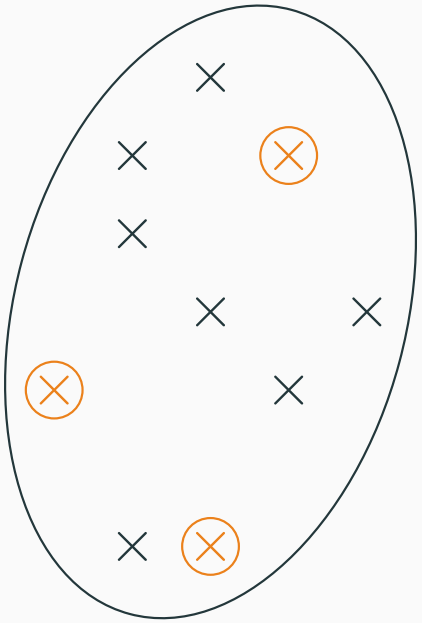
$\{\zeta^{\frac{(q-1)i}{\pi_m}} : 0 \leq i < \pi_m\}$

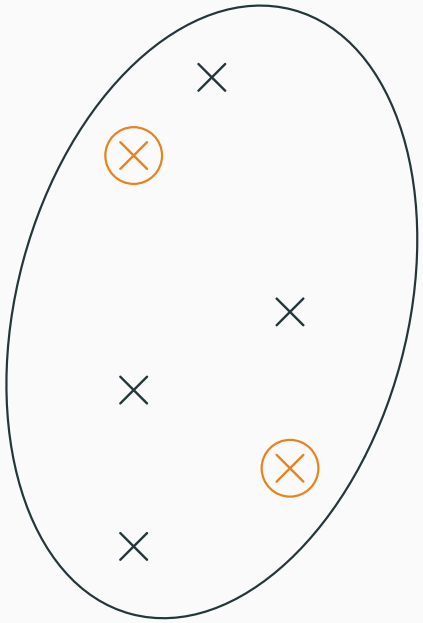
$g_{m-1}(\zeta^e) = 0$

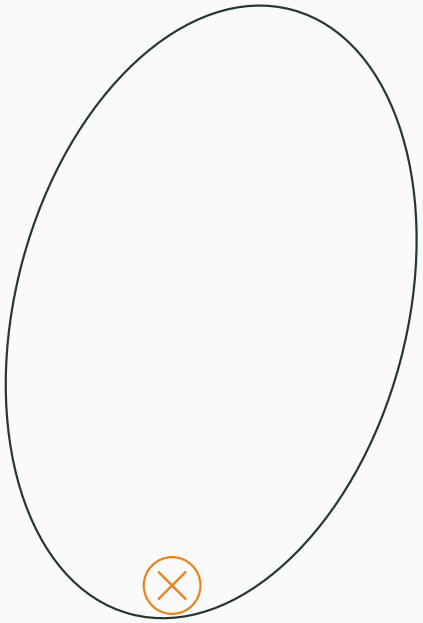


$Z_m = \{\zeta^0\}$

...







$\{\zeta^{\frac{e+(q-1)i}{\pi_{m-1}}} : 0 \le i < \pi_{m-1}\}$ $\{\zeta^{\frac{(q-1)i}{\pi_m}} : 0 \le i < \pi_m\}$ $Z_m = \{\zeta^0\}$

$g_{m-2}(\zeta^e) = 0$ $g_{m-1}(\zeta^e) = 0$

Theorem

If $\max_i \pi_i = O(\log q)$, the algorithm runs in time $\tilde{O}(d \log^3 q)$.

Deterministic complexity

Lemma

Given $g = G_\pi(f)$ and the roots $\{\beta_1, \ldots, \beta_k\}$ of g, one can compute the roots of f in time $\tilde{O}(\sqrt{\pi} d \log q) + (d \log q)^{1+\delta}$ for all $\delta > 0$.

Lemma

Given $g = G_\pi(f)$ and the roots $\{\beta_1, \ldots, \beta_k\}$ of g, one can compute the roots of f in time $\tilde{O}(\sqrt{\pi} d \log q) + (d \log q)^{1+\delta}$ for all $\delta > 0$.

Ingredients:

1. Factor $f = \prod_i f_i$ s.t. for all i, $G_\pi(f_i) = (X - \beta_i)^{m_i}$

   divide & conquer + modular composition [Kedlaya-Umans (2011)]

Lemma

Given $g = G_\pi(f)$ and the roots $\{\beta_1, \ldots, \beta_k\}$ of g, one can compute the roots of f in time $\tilde{O}(\sqrt{\pi} d \log q) + (d \log q)^{1+\delta}$ for all $\delta > 0$.

Ingredients:

1. Factor $f = \prod_i f_i$ s.t. for all i, $G_\pi(f_i) = (X - \beta_i)^{m_i}$

   divide & conquer + modular composition [Kedlaya-Umans (2011)]

2. Compute the ζ-logarithms of all β_i s.t. $\deg(f_i) > 1$

   discrete logarithms [Pohlig-Hellman (1978)]

Lemma

Given $g = G_\pi(f)$ and the roots $\{\beta_1, \ldots, \beta_k\}$ of g, one can compute the roots of f in time $\tilde{O}(\sqrt{\pi} d \log q) + (d \log q)^{1+\delta}$ for all $\delta > 0$.

Ingredients:

1. Factor $f = \prod_i f_i$ s.t. for all i, $G_\pi(f_i) = (X - \beta_i)^{m_i}$

   divide & conquer + modular composition [Kedlaya-Umans (2011)]

2. Compute the ζ-logarithms of all β_i s.t. $\deg(f_i) > 1$

   discrete logarithms [Pohlig-Hellman (1978)]

3. Deduce the roots of each f_i by multipoint evaluation

Theorem

Given $f \in \mathbb{F}_q[X]$ satisfying (A), the factorization of $q-1$ and a primitive element of $\mathbb{F}_q^*$, the roots of f can be computed in time

$$\tilde{O}(\sqrt{S(q-1)}d\log^2 q) + (d\log^2 q)^{1+\delta}$$

where $S(q-1)$ is the largest factor of $q-1$.

Theorem

Given $f \in \mathbb{F}_q[X]$ satisfying (A), the factorization of $q-1$ and a primitive element of $\mathbb{F}_q^*$, the roots of f can be computed in time

$$\tilde{O}(\sqrt{S(q-1)}d\log^2 q) + (d\log^2 q)^{1+\delta}$$

where $S(q-1)$ is the largest factor of $q-1$.

- Refine Shoup's complexity bounds

Randomization

$\mathbb{F}_p$ **with** $p = M \cdot 2^m + 1$

Definition

The **tangent Graeffe transform of order** π of $f \in \mathbb{F}_p[X]$ is

$$G_\pi(f + \varepsilon f') \in (\mathbb{F}_p[\varepsilon]/\langle \varepsilon^2 \rangle)[X].$$

Definition

The **tangent Graeffe transform of order** π of $f \in \mathbb{F}_p[X]$ is

$$G_\pi(f + \varepsilon f') \in (\mathbb{F}_p[\varepsilon]/\langle \varepsilon^2 \rangle)[X].$$

Remarks:

- $(f + \varepsilon f')(X) = f(X + \varepsilon)$
- $G_2(f + \varepsilon f') = G_2(f) + \varepsilon \overline{g}$ with $\overline{g}(X^2) = f(X)f'(-X) + f(-X)f'(X)$

Definition

The **tangent Graeffe transform of order** π of $f \in \mathbb{F}_p[X]$ is

$$G_\pi(f + \varepsilon f') \in (\mathbb{F}_p[\varepsilon]/\langle \varepsilon^2 \rangle)[X].$$

Remarks:

- $(f + \varepsilon f')(X) = f(X + \varepsilon)$
- $G_2(f + \varepsilon f') = G_2(f) + \varepsilon \overline{g}$ with $\overline{g}(X^2) = f(X)f'(-X) + f(-X)f'(X)$

Lemma

Let $g + \varepsilon \overline{g} = G_{2^\ell}(f + \varepsilon f')$. **A root β of g is simple iff** $\overline{g}(\beta) \neq 0$.
The corresponding root of f is $\alpha = 2^\ell \beta g'(\beta)/\overline{g}(\beta)$.

Goal: Ensure many simple roots

- Replace f by $f_\tau(X) = f(X + \tau)$ for a random $\tau \in \mathbb{F}_p$.

Goal: Ensure many simple roots

- Replace f by $f_\tau(X) = f(X+\tau)$ for a random $\tau \in \mathbb{F}_p$.

Lemma

If $2^\ell \leq \frac{p-1}{d(d-1)}$, $G_{2^\ell}(f_\tau)$ has **no multiple root** with prob. $\geq 1/2$.

Goal: Ensure many simple roots

- Replace f by $f_\tau(X) = f(X+\tau)$ for a random $\tau \in \mathbb{F}_p$.

Lemma

If $2^\ell \leq \frac{p-1}{d(d-1)}$, $G_{2^\ell}(f_\tau)$ has **no multiple root** with prob. $\geq 1/2$.

$$f(X+\tau+\varepsilon) \xrightarrow{G_2} \cdots \xrightarrow{G_2} g_\ell + \varepsilon\overline{g}_\ell \xrightarrow{G_2} \cdots \xrightarrow{G_2} g_m + \varepsilon\overline{g}_m$$

Goal: Ensure many simple roots

- Replace f by $f_\tau(X) = f(X+\tau)$ for a random $\tau \in \mathbb{F}_p$.

Lemma

If $2^\ell \leq \frac{p-1}{d(d-1)}$, $G_{2^\ell}(f_\tau)$ has **no multiple root** with prob. $\geq 1/2$.

$$f(X+\tau+\varepsilon) \xrightarrow{G_2} \cdots \xrightarrow{G_2} g_\ell + \varepsilon\overline{g}_\ell \xrightarrow{G_2} \cdots \xrightarrow{G_2} g_m + \varepsilon\overline{g}_m$$

$$\downarrow$$

$$Z_m$$

$$\cap$$

$$\{\xi^e : 0 \leq e < M\}$$

Goal: Ensure many simple roots

- Replace f by $f_\tau(X) = f(X+\tau)$ for a random $\tau \in \mathbb{F}_p$.

Lemma

If $2^\ell \leq \frac{p-1}{d(d-1)}$, $G_{2^\ell}(f_\tau)$ has **no multiple root** with prob. $\geq 1/2$.

$$f(X+\tau+\varepsilon) \xrightarrow{G_2} \cdots \xrightarrow{G_2} g_\ell + \varepsilon\overline{g}_\ell \xrightarrow{G_2} \cdots \xrightarrow{G_2} g_m + \varepsilon\overline{g}_m$$

$$\downarrow$$

$$Z_\ell \longleftarrow \cdots \longleftarrow Z_m$$

$$\cap$$

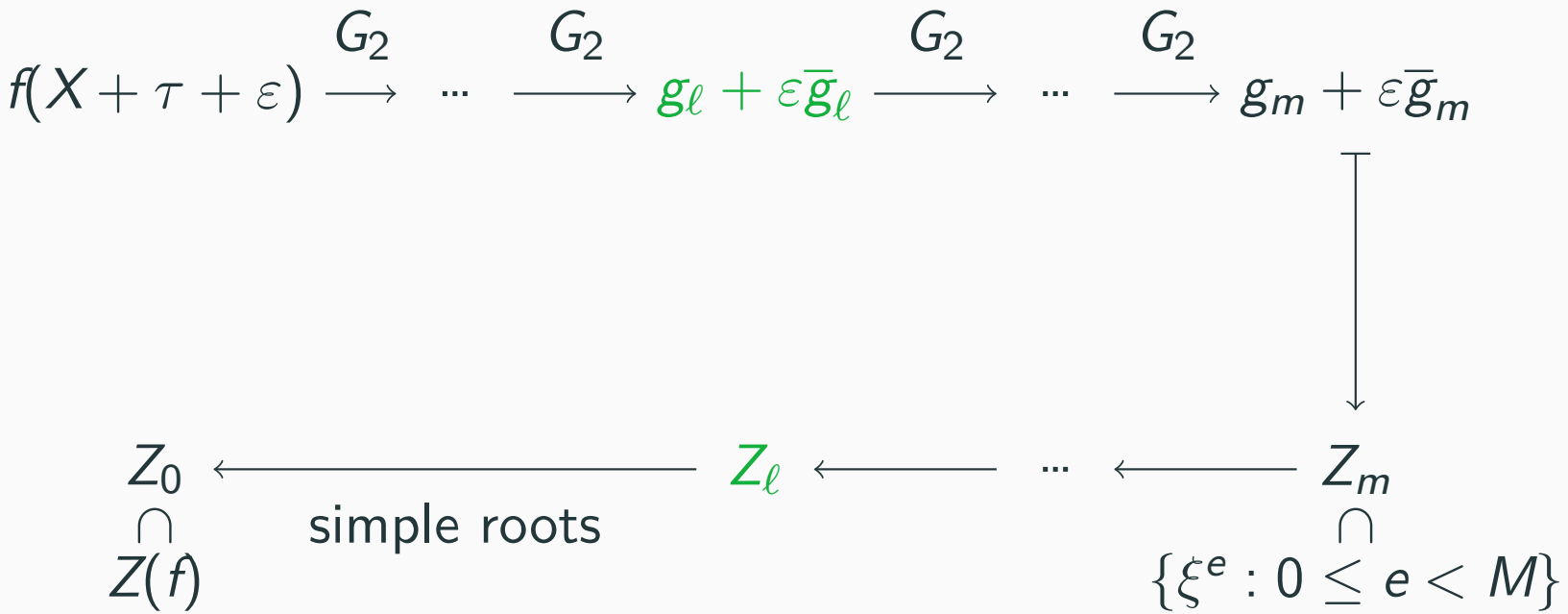
$$\{\xi^e : 0 \leq e < M\}$$

Goal: Ensure many simple roots

- Replace f by $f_\tau(X) = f(X+\tau)$ for a random $\tau \in \mathbb{F}_p$.

Lemma

If $2^\ell \leq \frac{p-1}{d(d-1)}$, $G_{2^\ell}(f_\tau)$ has **no multiple root** with prob. $\geq 1/2$.

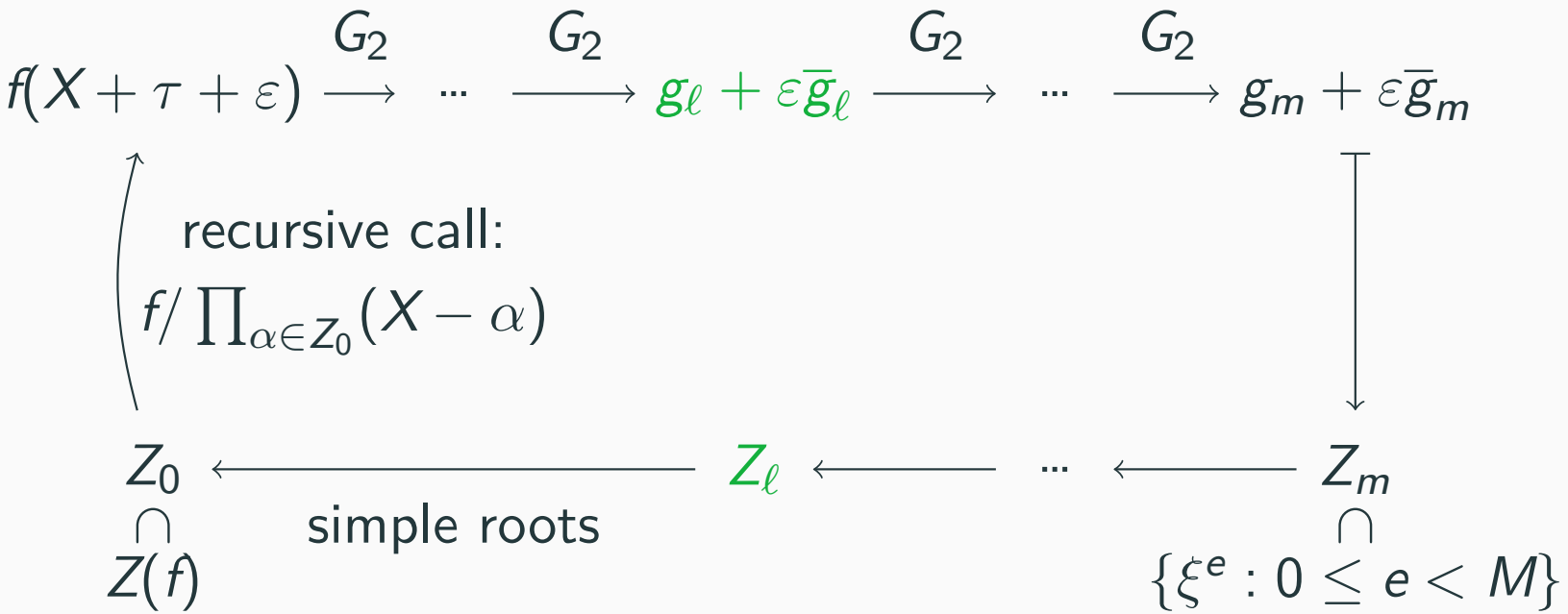


Goal: Ensure many simple roots

- Replace f by $f_\tau(X) = f(X+\tau)$ for a random $\tau \in \mathbb{F}_p$.

Lemma

If $2^\ell \leq \frac{p-1}{d(d-1)}$, $G_{2^\ell}(f_\tau)$ has **no multiple root** with prob. $\geq 1/2$.



Theorem

Given $f \in \mathbb{F}_p[X]$ satisfying (A) and a primitive element of $\mathbb{F}_p^*$, the randomized algorithm runs in **expected time** $\tilde{O}(d \log^2 p)$, for $p = M \cdot 2^m + 1$ with $M = O(\log p)$.

Theorem

Given $f \in \mathbb{F}_p[X]$ satisfying (A) and a primitive element of $\mathbb{F}_p^*$, the randomized algorithm runs in **expected time** $\tilde{O}(d \log^2 p)$, for $p = M \cdot 2^m + 1$ with $M = O(\log p)$.

- Same asymptotic as Rabin's algorithm
- Better efficiency in practice (gcd $\rightsquigarrow$ multipoint evaluation)
- Primitive elements easy to compute in practice

Heuristic version

$\mathbb{F}_p$ **with** $p = M \cdot 2^m + 1$

Heuristic

If $2^\ell \simeq p/d$, $G_{2^\ell}(f(X+\tau))$ has $\Omega(d)$ **simple roots** with probability $\geq 1/2$, for a random $\tau \in \mathbb{F}_p$.

Justification: holds for a random f rather than $f(X+\tau)$.

Heuristic

If $2^\ell \simeq p/d$, $G_{2^\ell}(f(X+\tau))$ has $\Omega(d)$ **simple roots** with probability $\geq 1/2$, for a random $\tau \in \mathbb{F}_p$.

Justification: holds for a random f rather than $f(X+\tau)$.

$$f(X+\tau+\varepsilon) \xrightarrow{\quad G_{2^\ell} \quad} g_\ell + \varepsilon \bar{g}_\ell$$

Heuristic

If $2^\ell \simeq p/d$, $G_{2^\ell}(f(X+\tau))$ has $\Omega(d)$ **simple roots** with probability $\geq 1/2$, for a random $\tau \in \mathbb{F}_p$.

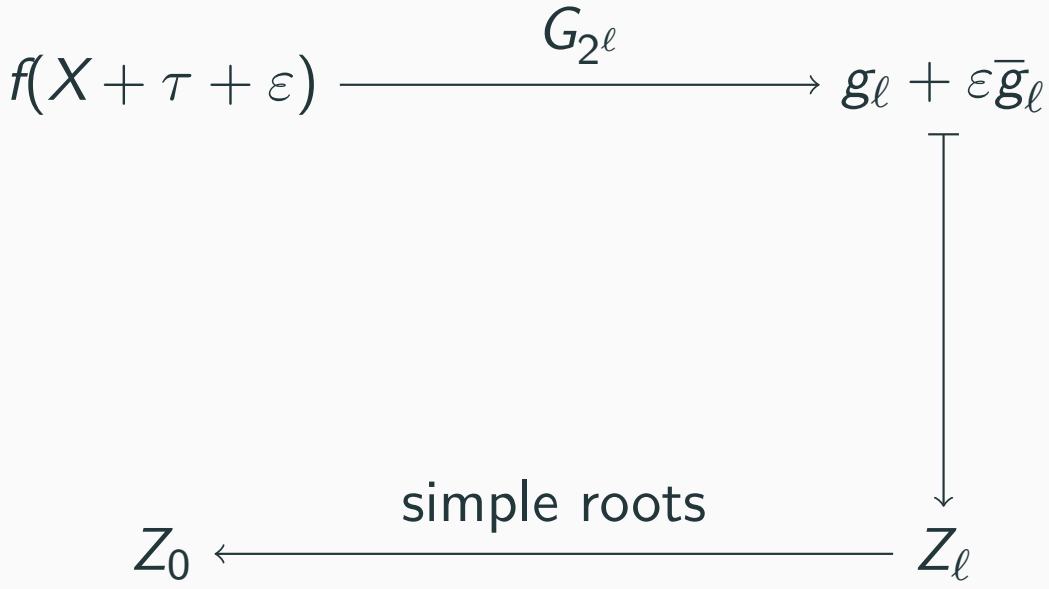
Justification: holds for a random f rather than $f(X+\tau)$.

$$f(X+\tau+\varepsilon) \xrightarrow{\quad G_{2^\ell} \quad} g_\ell + \varepsilon \bar{g}_\ell \longrightarrow Z_\ell$$

Heuristic

If $2^\ell \simeq p/d$, $G_{2^\ell}(f(X+\tau))$ has $\Omega(d)$ **simple roots** with probability $\geq 1/2$, for a random $\tau \in \mathbb{F}_p$.

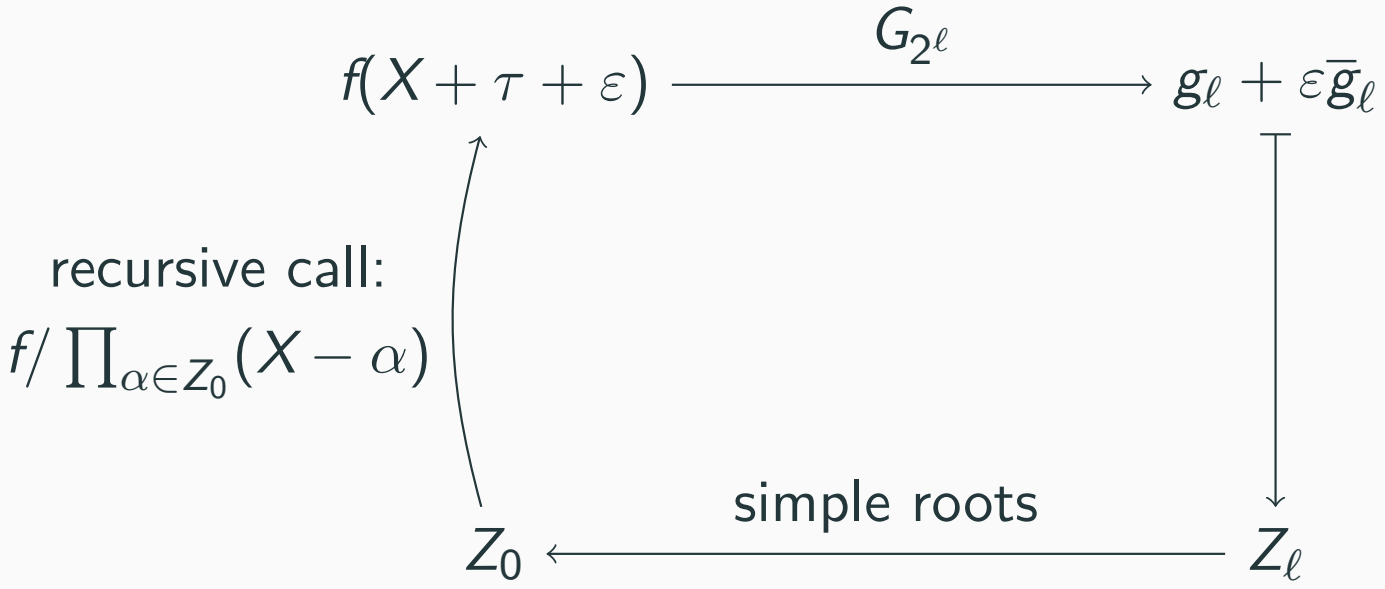
Justification: holds for a random f rather than $f(X+\tau)$.



Heuristic

If $2^\ell \simeq p/d$, $G_{2^\ell}(f(X+\tau))$ has $\Omega(d)$ **simple roots** with probability $\geq 1/2$, for a random $\tau \in \mathbb{F}_p$.

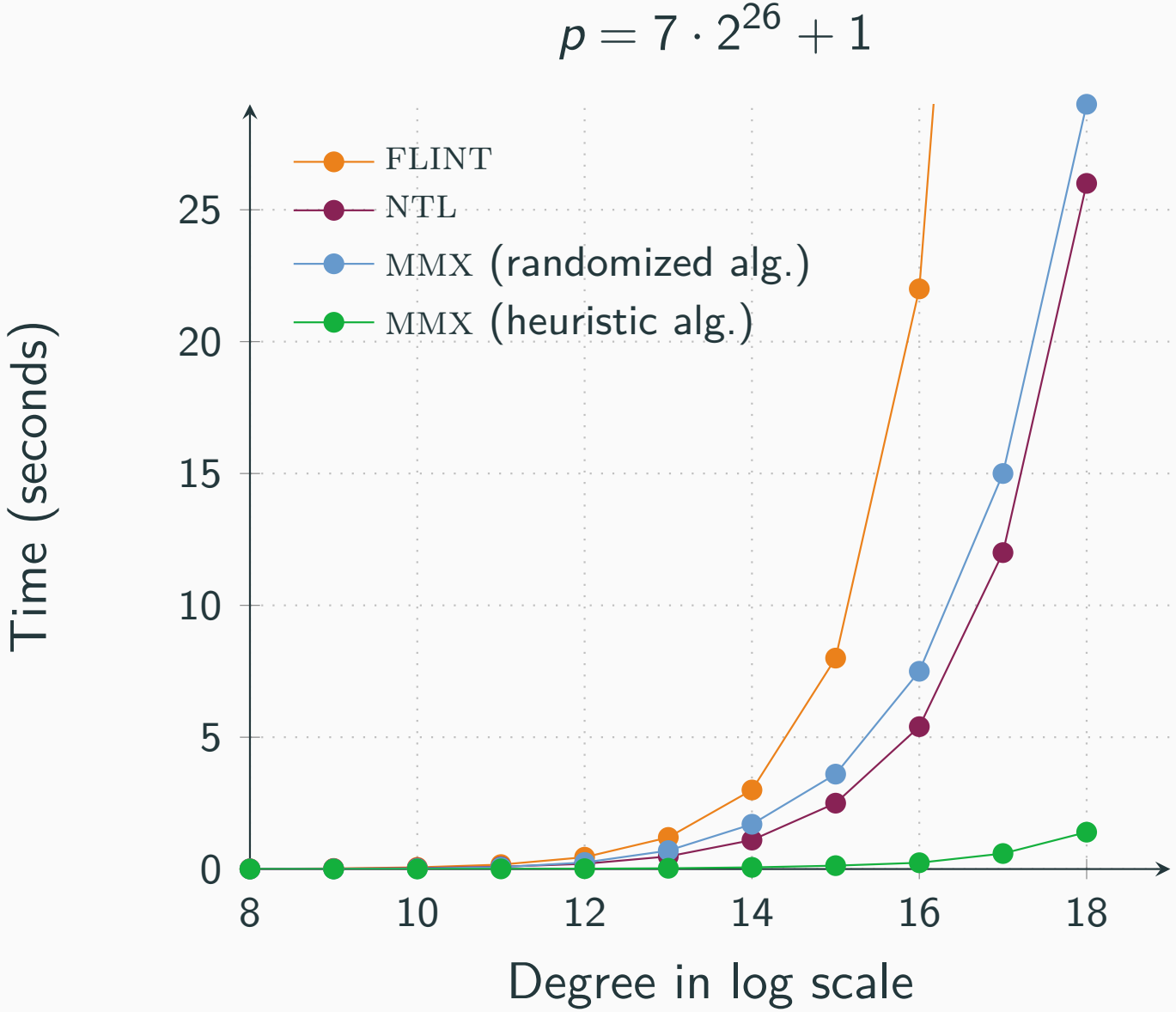
Justification: holds for a random f rather than $f(X+\tau)$.



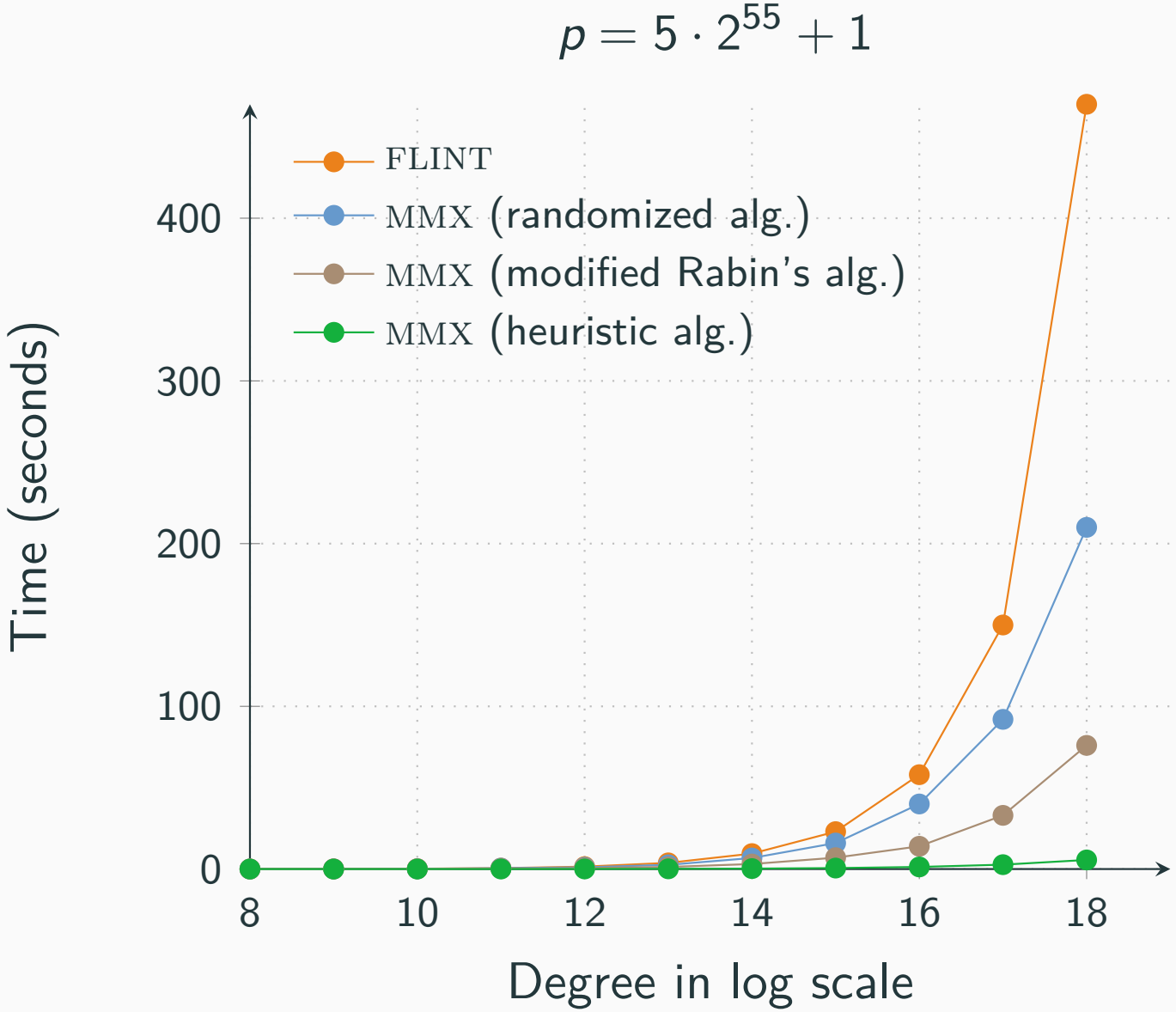
Theorem

Suppose that f is chosen at random in $\mathbb{F}_p[X]$ or that the heuristic holds. Given a primitive element of $\mathbb{F}_p^*$, the heuristic algorithm runs in **expected time** $\tilde{O}(d \log^2 p)$, for $p = M \cdot 2^m + 1$ with $M = O(\log p)$.

Running times


$p = 7 \cdot 2^{26} + 1$
FLINT
NTL
MMX (randomized alg.)
MMX (heuristic alg.)
Time (seconds)
0
5
10
15
20
25
8
10
12
14
16
18
Degree in log scale

Timings


$p = 5 \cdot 2^{55} + 1$
FLINT
MMX (randomized alg.)
MMX (modified Rabin's alg.)
MMX (heuristic alg.)
Time (seconds)
0
100
200
300
400
8
10
12
14
16
18
Degree in log scale

Conclusion

- Revisit classical algorithms for FFT finite fields

Conclusion

- Revisit classical algorithms for FFT finite fields
- New approach using Graeffe transforms
  - ✓ deterministic complexity bounds
  - ✓ probabilistic complexity bounds
  - ✓ running times

Conclusion

- Revisit classical algorithms for FFT finite fields
- New approach using Graeffe transforms
  - ✓ deterministic complexity bounds
  - ✓ probabilistic complexity bounds
  - ✓ running times
- Source code in C++ within MATHEMAGIX

Conclusion

- Revisit classical algorithms for FFT finite fields
- New approach using Graeffe transforms
  - ✓ deterministic complexity bounds
  - ✓ probabilistic complexity bounds
  - ✓ running times
- Source code in C++ within MATHEMAGIX
- Not the bottleneck anymore for sparse interpolation

Conclusion

- Revisit classical algorithms for FFT finite fields
- New approach using Graeffe transforms
  - ✓ deterministic complexity bounds
  - ✓ probabilistic complexity bounds
  - ✓ running times
- Source code in C++ within MATHEMAGIX
- Not the bottleneck anymore for sparse interpolation
- Open questions:
  - ? Deterministic alg.: use of tangent Graeffe transforms
  - ? Heuristic alg.: Graeffe transform of order 2^ℓ
  - ? Prove the heuristic

Conclusion

- Revisit classical algorithms for FFT finite fields
- New approach using Graeffe transforms
  - ✓ deterministic complexity bounds
  - ✓ probabilistic complexity bounds
  - ✓ running times
- Source code in C++ within MATHEMAGIX
- Not the bottleneck anymore for sparse interpolation
- Open questions:
  - ? Deterministic alg.: use of tangent Graeffe transforms
  - ? Heuristic alg.: Graeffe transform of order 2^ℓ
  - ? Prove the heuristic

Merci de votre attention !