Fast method for testing the smoothness of polynomials

Jean-François Biasse
Mike Jacobson

University of Calgary

October 2013
Presentation of the problem

- Let $K$ be a finite field.
- Let $B > 0$ a bound.

We want to test if a given $P \in K[X]$ is $B$-smooth, that is if

$$P = P_1^{e_1} \cdots P_n^{e_n}, \quad \text{with } \forall i \leq k \deg(P_i) \leq B.$$
Presentation of the problem

- Let $K$ be a finite field.
- Let $B > 0$ a bound.

We want to test if a given $P \in K[X]$ is $B$-smooth, that is if

$$P = P_1^{e_1} \cdots P_n^{e_n}, \quad \text{with } \forall i \leq k \deg(P_i) \leq B.$$ 

This occurs in the resolution of the discrete logarithm problem (DLP):

- Function field sieve in $(\mathbb{F}_p^m)^*$.
Presentation of the problem

- Let $K$ be a finite field.
- Let $B > 0$ a bound.

We want to test if a given $P \in K[X]$ is $B$-smooth, that is if

$$P = P_1^{e_1} \cdots P_n^{e_n}, \quad \text{with } \forall i \leq k \deg(P_i) \leq B.$$ 

This occurs in the resolution of the discrete logarithm problem (DLP):

- Function field sieve in $(\mathbb{F}_{p^m})^*$.
- Random walk method in $\mathcal{J}(C)$.
Presentation of the problem

- Let $K$ be a finite field.
- Let $B > 0$ a bound.

We want to test if a given $P \in K[X]$ is $B$-smooth, that is if

$$P = P_1^{e_1} \cdots P_n^{e_n}, \quad \text{with } \forall i \leq k \deg(P_i) \leq B.$$ 

This occurs in the resolution of the discrete logarithm problem (DLP):

- Function field sieve in $(\mathbb{F}_{p^m})^\ast$.
- Random walk method in $\mathcal{J}(C)$.
- Quadratic sieve method in the Jacobian of $\mathcal{J}(C)$.

where $\mathcal{J}(C)$ is the Jacobian of a hyperelliptic curve $C$ over a finite field.
1 Motivation

2 Bernstein’s approach

3 Complexity analysis

4 Practical examples
The jacobian of a hyperelliptic curve

Let $K$ be a finite field, a hyperelliptic curve $C$ of genus $g$ is defined by

$$Y^2 + h(X)Y + f(X) = 0,$$

where $h, f \in K[X]$, $\deg(h) \leq g$ and $\deg(f) = 2g + 1$ or $2g + 2$. 
The jacobian of a hyperelliptic curve

Let $K$ be a finite field, a hyperelliptic curve $C$ of genus $g$ is defined by

$$Y^2 + h(X)Y + f(X) = 0,$$

where $h, f \in K[X]$, $\deg(h) \leq g$ and $\deg(f) = 2g + 1$ or $2g + 2$.

The Jacobian variety

A hyperelliptic curve is associated to a group $\mathcal{J}(C)$ with

- $|\mathcal{J}(C)| \approx q^g$ where $K = \mathbb{F}_q$.
- Solving the DLP at fixed $g$ is exponential in $\log(q)$. 
The jacobian of a hyperelliptic curve

Let $K$ be a finite field, a hyperelliptic curve $C$ of genus $g$ is defined by

$$Y^2 + h(X)Y + f(X) = 0,$$

where $h, f \in K[X]$, $\deg(h) \leq g$ and $\deg(f) = 2g + 1$ or $2g + 2$.

The Jacobian variety

A hyperelliptic curve is associated to a group $\mathcal{J}(C)$ with

- $|\mathcal{J}(C)| \approx q^g$ where $K = \mathbb{F}_q$.
- Solving the DLP at fixed $g$ is exponential in $\log(q)$.

- The DLP in $|\mathcal{J}(C)|$ is an essential topic in cryptography.
- Elliptic curves are the special case $g = 1$. 
Smoothness in $\mathcal{J}(\mathcal{C})$

Elements of $\mathcal{J}(\mathcal{C})$ can be represented by $(u(X), v(X))$ where

- $\deg(u) \leq g$ is the degree of $(u(X), v(X))$.
- $\deg(v) < \deg(\nu)$.
Smoothness in $\mathcal{J}(\mathcal{C})$

Elements of $\mathcal{J}(\mathcal{C})$ can be represented by $(u(X), v(X))$ where
- $\deg(u) \leq g$ is the degree of $(u(X), v(X))$.
- $\deg(v) < \deg(v)$.

Smoothness of divisors

We say that $a \in \mathcal{J}(\mathcal{C})$ is $B$-smooth if

$$a = p_1 \cdots p_n \text{ for some } n > 0, \text{ with } \forall i, \deg(p_i) \leq B.$$
Smoothness in $\mathcal{J}(C)$

Elements of $\mathcal{J}(C)$ can be represented by $(u(X), v(X))$ where
- $\deg(u) \leq g$ is the degree of $(u(X), v(X))$.
- $\deg(v) < \deg(\nu)$.

Smoothness of divisors

We say that $a \in \mathcal{J}(C)$ is $B$-smooth if

$$a = p_1 \cdots p_n \text{ for some } n > 0, \text{ with } \forall i, \deg(p_i) \leq B.$$ 

If $u(X)$ is $B$-smooth for $B \leq g$, then $(u(X), v(X))$ is $B$-smooth in $\mathcal{J}(C)$. 

Biasse-Jacobson (U of C) Fast smoothness test October 2013 4 / 24
Solving the DLP in $\mathcal{J}(C)$ from relations

- Let $a, b \in \mathcal{J}(C)$, we want to find $x \in \mathbb{Z}$ such that $b = a^x$.
- Let $p_1, \ldots, p_n$ generating $\mathcal{J}(C)$. 
Solving the DLP in $\mathcal{J}(C)$ from relations

- Let $a, b \in \mathcal{J}(C)$, we want to find $x \in \mathbb{Z}$ such that $b = a^x$.
- Let $p_1, \cdots, p_n$ generating $\mathcal{J}(C)$.

$$M = \begin{pmatrix}
m_{1,1} & \cdots & m_{1,n} & 0 & 0 \\
\vdots & \ddots & \vdots & \vdots & \vdots \\
m_{l,1} & \cdots & m_{l,n} & 0 & 0 \\
m_{l+1,1} & m_{l+1,n} & 1 & 0 \\
m_{l+2,1} & m_{l+2,n} & 0 & 1
\end{pmatrix}$$

**A** : $l + 2$ rows $n + 1$ columns

$$p_1^{m_{1,1}} \cdots p_n^{m_{1,n}} = 1$$
$$\vdots$$
$$p_1^{m_{k,1}} \cdots p_n^{m_{k,n}} = 1$$
$$p_1^{m_{k+1,1}} \cdots p_n^{m_{k+1,n}} b = 1$$
$$p_1^{m_{k+2,1}} \cdots p_n^{m_{k+2,n}} a = 1$$
Solving the DLP in $\mathcal{J}(C)$ from relations

- Let $a, b \in \mathcal{J}(C)$, we want to find $x \in \mathbb{Z}$ such that $b = a^x$.
- Let $p_1, \cdots, p_n$ generating $\mathcal{J}(C)$.

\[
M = \begin{pmatrix}
    m_{1,1} & \cdots & m_{1,n} & 0 & 0 \\
    \vdots & \ddots & \vdots & \vdots & \vdots \\
    m_{l,1} & \cdots & m_{l,n} & 0 & 0 \\
    m_{l+1,1} & m_{l+1,n} & 1 & 0 \\
    m_{l+2,1} & m_{l+2,n} & 0 & 1 \\
\end{pmatrix}
\]

\[
\begin{array}{l}
p_1^{m_{1,1}} \cdots p_n^{m_{1,n}} = 1 \\
p_1^{m_{k,1}} \cdots p_n^{m_{k,n}} = 1 \\
p_1^{m_{k+1,1}} \cdots p_n^{m_{k+1,n}} b = 1 \\
p_1^{m_{k+2,1}} \cdots p_n^{m_{k+2,n}} a = 1 \\
\end{array}
\]

\[A : l + 2 \text{ rows} \ n + 1 \text{ columns}\]

- If $XA = (0, \cdots, 0, 1)$, then $\exists y \in \mathbb{Z}$ such that $XM = (0, \cdots, 0, 1, y)$.
- This means $b a^y = 1$, so $x = -y$ is a solution.
Relations in $\mathcal{J}(C)$ from random walk

We can solve the DLP in $\mathcal{J}(C)$ from relations $p_1 \cdots p_n = 1$ where

- $\mathcal{B} := \{p_1 \cdots p_n\}$ generates $\mathcal{J}(C)$.
- $\mathcal{B} = \{p = (u, v) \in \mathcal{J}(C) \mid u \text{ prime}, \deg(u) \leq B\}$. 

Random walk strategy

We repeat the following steps.

1. Draw $p_{e1} \cdots p_{en} = (u, v)$ at random.
2. Test if $u \in \mathbb{F}_q[X]$ is $B$-smooth.

Each time $u$ is $B$-smooth, we have a relation $i_{p_{e_i}} = j_q$. 

The two main contributions to the cost are

- Arithmetic in $\mathcal{J}(C)$.
- Smoothness test of $u$.

Biasse-Jacobson (U of C)

Fast smoothness test

October 2013
Relations in $\mathcal{J}(C)$ from random walk

We can solve the DLP in $\mathcal{J}(C)$ from relations $p_1 \cdots p_n = 1$ where

- $B := \{p_1 \cdots p_n\}$ generates $\mathcal{J}(C)$.
- $B = \{p = (u, v) \in \mathcal{J}(C) \mid u \text{ prime }, \deg(u) \leq B\}$.

Random walk strategy

We repeat the following steps.

- Draw $p_1^{e_1} \cdots p_n^{e_n} = (u, v)$ at random.
- Test if $u \in \mathbb{F}_q[X]$ is $B$-smooth.

Each time $u$ is $B$-smooth, we have a relation $\prod_i p_i^{e_i} = \prod_j q_j$. 
 Relations in $\mathcal{J}(C)$ from random walk

We can solve the DLP in $\mathcal{J}(C)$ from relations $p_1 \cdots p_n = 1$ where

- $B := \{p_1 \cdots p_n\}$ generates $\mathcal{J}(C)$.
- $B = \{p = (u, v) \in \mathcal{J}(C) | u \text{ prime, } \deg(u) \leq B\}$.

Random walk strategy

We repeat the following steps.

- Draw $p_1^{e_1} \cdots p_n^{e_n} = (u, v)$ at random.
- Test if $u \in \mathbb{F}_q[X]$ is $B$-smooth.

Each time $u$ is $B$-smooth, we have a relation $\prod_i p_i^{e_i} = \prod_j q_j$.

The two main contribution to the cost are

- Arithmetic in $\mathcal{J}(C)$.
- Smoothness test of $u$. 
Sieving in a fonction field

- Let $P \in K[x][y]$ of degree $g$.
- Let $B > 0$ and $S \subset K[x]^{g+1}$.

We want to find $(a_i(x)) \in S$ such that $P(a_0(x), \ldots, a_g(x))$ is $B$—smooth.
Sieving in a function field

- Let \( P \in K[x][y] \) of degree \( g \).
- Let \( B > 0 \) and \( S \subset K[x]^{g+1} \).

We want to find \((a_i(x)) \in S\) such that \( P(a_0(x), \cdots, a_g(x)) \) is \( B \)-smooth.

**Sieving methods**

Using roots of \( P \mod p_i \) where \( \deg(p_i) \leq B \), we
- Preselect rapidly candidates \( Q_1(x), \cdots, Q_l(x) \) where \( Q_j \in P(S) \).
- Then we test the \((Q_i(x)))_{i \leq l} \) for smoothness.
Sieving in a fonction field

- Let $P \in K[x][y]$ of degree $g$.
- Let $B > 0$ and $S \subset K[x]^{g+1}$.

We want to find $(a_i(x)) \in S$ such that $P(a_0(x), \cdots, a_g(x))$ is $B -$ smooth.

Sieving methods

Using roots of $P$ mod $p_i$ where $\text{deg}(p_i) \leq B$, we

- Preselect rapidly candidates $Q_1(x), \cdots, Q_l(x)$ where $Q_j \in P(S)$.
- Then we test the $(Q_i(x))_{i \leq l}$ for smoothness.

- Sieving is faster than testing $P(a_0(x), \cdots, a_g(x))$ for all $(a_i(x)) \in S$.
- It still involves smoothness tests of elements in $K[x]$. 
Relations in $\mathcal{J}(C)$ from sieving

Let $C : Y^2 + h(X)Y + f(X) = F(X, Y) = 0$ with $\deg(f) = 2g + 1$.  
- Let $\mathcal{O} := \mathbb{F}_q[X][Y]/F(X, Y)$ be the equation order.  
- $\text{Cl}(\mathcal{O}) := \{\text{ideals of } \mathcal{O}\}/\{\text{principal ideals}\} \simeq \mathcal{J}(C)$.  

We derive relations from $\mathcal{B}$-smooth values of $\psi(x, y)$ obtained by sieving.
Relations in $\mathcal{J}(\mathcal{C})$ from sieving

Let $\mathcal{C} : Y^2 + h(X)Y + f(X) = F(X, Y) = 0$ with $\text{deg}(f) = 2g + 1$.

- Let $\mathcal{O} := \mathbb{F}_q[X][Y]/F(X, Y)$ be the equation order.
- $\text{Cl}(\mathcal{O}) := \{\text{ideals of } \mathcal{O}\}/\{\text{principal ideals}\} \simeq \mathcal{J}(\mathcal{C})$.

Relations in $\text{Cl}(\mathcal{O})$

Relations in $\mathcal{J}(\mathcal{C})$ correspond to identities $p_1 \cdots p_n = (\alpha)$

- Where the $p_i$ are ideals of $\mathcal{O}$ and $\alpha \in \mathbb{F}_q[X]$.
- If $\mathcal{N}(\alpha) \in \mathbb{F}_q[X]$ is $B$-smooth, then the relation in $\mathcal{J}(\mathcal{C})$ is too.
Relations in $\mathcal{J}(C)$ from sieving

Let $C : Y^2 + h(X)Y + f(X) = F(X, Y) = 0$ with $\deg(f) = 2g + 1$.

- Let $\mathcal{O} := \mathbb{F}_q[X][Y]/F(X, Y)$ be the equation order.
- $\text{Cl}(\mathcal{O}) := \{\text{ideals of } \mathcal{O}\}/\{\text{principal ideals}\} \simeq \mathcal{J}(C)$.

Relations in $\text{Cl}(\mathcal{O})$

Relations in $\mathcal{J}(C)$ correspond to identities $p_1 \cdots p_n = (\alpha)$

- Where the $p_i$ are ideals of $\mathcal{O}$ and $\alpha \in \mathbb{F}_q[X]$.
- If $\mathcal{N}(\alpha) \in \mathbb{F}_q[X]$ is $B$-smooth, then the relation in $\mathcal{J}(C)$ is too.

Let $[a, \omega]$ be an integral basis of $\mathcal{O}$. We have

$$\mathcal{N}(xa + y\omega) = a^2x^2 + a\text{Tr}(\omega)xy + \mathcal{N}(\omega)y^2 := \psi(x, y).$$
Relations in $\mathcal{J}(\mathcal{C})$ from sieving

Let $\mathcal{C} : Y^2 + h(X)Y + f(X) = F(X, Y) = 0$ with $\deg(f) = 2g + 1$.

- Let $\mathcal{O} := \mathbb{F}_q[X][Y]/F(X, Y)$ be the equation order.
- $\text{Cl}(\mathcal{O}) := \{\text{ideals of } \mathcal{O}\}/\{\text{principal ideals}\} \simeq \mathcal{J}(\mathcal{C})$.

Relations in $\text{Cl}(\mathcal{O})$

Relations in $\mathcal{J}(\mathcal{C})$ correpond to identities $p_1 \cdots p_n = (\alpha)$

- Where the $p_i$ are ideals of $\mathcal{O}$ and $\alpha \in \mathbb{F}_q[X]$.
- If $\mathcal{N}(\alpha) \in \mathbb{F}_q[X]$ is $B$-smooth, then the relation in $\mathcal{J}(\mathcal{C})$ is too.

Let $[a, \omega]$ be an integral basis of $\mathcal{O}$. We have

\[ \mathcal{N}(xa + y\omega) = a^2x^2 + a\text{Tr}(\omega)xy + \mathcal{N}(\omega)y^2 := \psi(x, y). \]

We derive relations from $B$-smooth values of $\psi(x, y)$ obtained by sieving.
The function field sieve

- We want to solve the DLP in $K = \mathbb{F}_p^m$. We construct relations in $\mathbb{F}_p^m$.
- Let $f, g \in \mathbb{F}_p[x][y]$ with $\varphi(x) \mid \text{Res}(f, g)$, $\deg(\varphi) = m$. 

Biasse-Jacobson (U of C)  
Fast smoothness test  
October 2013
The function field sieve

- We want to solve the DLP in $K = \mathbb{F}_{p^m}$. We construct relations in $\mathbb{F}_{p^m}$.
- Let $f, g \in \mathbb{F}_p[x][y]$ with $\varphi(x) \mid \text{Res}(f, g)$, $\deg(\varphi) = m$.

We have the commutative diagramm

\[
\begin{array}{ccc}
\mathbb{F}_p[x][y] & \longrightarrow & \mathbb{F}_p[x][y]/g(x, y) \\
\downarrow & & \downarrow \\
\mathbb{F}_p[x][y]/f(x, y) & \longrightarrow & \mathbb{F}_p[x]/\varphi(x) = K
\end{array}
\]
The function field sieve

- We want to solve the DLP in $K = \mathbb{F}_{p^m}$. We construct relations in $\mathbb{F}_{p^m}$.
- Let $f, g \in \mathbb{F}_p[x][y]$ with $\varphi(x) \mid \text{Res}(f, g)$, $\deg(\varphi) = m$.

We have the commutative diagramm

\[ \begin{array}{ccc}
\mathbb{F}_p[x][y] & \longrightarrow & \mathbb{F}_p[x][y]/g(x, y) \\
\downarrow & & \downarrow \\
\mathbb{F}_p[x][y]/f(x, y) & \longrightarrow & \mathbb{F}_p[x]/\varphi(x) = K
\end{array} \]

- Let $\mathcal{N}(a(x) + b(x)y)$ $B$-smooth in $\mathbb{F}_p[x][y]/g$ and $\mathbb{F}_p[x][y]/f$.
- We obtain a relation between small elements in $K$. 
The function field sieve

- We want to solve the DLP in $K = \mathbb{F}_{p^m}$. We construct relations in $\mathbb{F}_{p^m}$.
- Let $f, g \in \mathbb{F}_p[x][y]$ with $\varphi(x) \mid \text{Res}(f, g)$, $\deg(\varphi) = m$.

We have the commutative diagramm

$$\begin{array}{c}
\mathbb{F}_p[x][y] \\ \downarrow \quad \downarrow \\
\mathbb{F}_p[x][y]/f(x, y) \\ \mathbb{F}_p[x][y]/g(x, y)
\end{array}$$

- Let $\mathcal{N}(a(x) + b(x)y)$ $B$-smooth in $\mathbb{F}_p[x][y]/g$ and $\mathbb{F}_p[x][y]/f$.
- We obtain a relation between small elements in $K$.

We recombine the relations in $K$ to get the DLP of all the small elements.
1 Motivation

2 Bernstein’s approach

3 Complexity analysis

4 Practical examples
Smoothness test over the integers

Bernstein described a smoothness test for integers.

- Runs in $O(b(\log(b))^2 \log \log(b))$ where $b$ is the total size of the input.
- To be compared to ECM: $O(b \cdot L_b(1/2, 2 + o(1)))$.

Applications

Bernstein's method was successfully used for:

- Directly testing smoothness of integers.
- Testing the smoothness of cofactors in sieving algorithms.

It is straightforward to adapt this method to $\mathbb{F}_q[X]$ but:

- Unlike in $\mathbb{Z}$, factorization in $\mathbb{F}_q[X]$ takes polynomial.
- It requires efficient implementation of fast multiplication algorithms.
Smoothness test over the integers

Bernstein described a smoothness test for integers.
- Runs in $O(b(\log(b))^2 \log \log(b))$ where $b$ is the total size of the input.
- To be compared to ECM: $O(b \cdot L_b(1/2, 2 + o(1)))$.

Applications

Bernstein’s method was successfully used for
- Directly testing smoothness of integers.
- Testing the smoothness of cofactors in sieving algorithms.
Smoothness test over the integers

Bernstein described a smoothness test for integers.
- Runs in $O(b(\log(b))^2 \log \log(b))$ where $b$ is the total size of the input.
- To be compared to ECM: $O(b \cdot L_b(1/2, 2 + o(1)))$.

Applications

Bernstein’s method was successfully used for
- Directly testing smoothness of integers.
- Testing the smoothness of cofactors in sieving algorithms.

It is straightforward to adapt this method to $\mathbb{F}_q[X]$ but:
- Unlike in $\mathbb{Z}$, factorization in $\mathbb{F}_q[X]$ takes polynomial.
- It requires efficient implementation of fast multiplication algorithms.
Product tree

- **Input**: $b_1, \ldots, b_n, \ n = 2^N$.
- **Output**: $\prod_i b_i$. 
Product tree

- **Input**: $b_1, \ldots, b_n$, $n = 2^N$.
- **Output**: $\prod_i b_i$. 
Product tree

- **Input**: $b_1, \cdots, b_n$, $n = 2^N$.
- **Output**: $\prod_i b_i$. 
Product tree

- **Input**: \( b_1, \cdots, b_n, \ n = 2^N \).
- **Output**: \( \prod_i b_i \).
Product tree

- **Input**: $b_1, \cdots, b_n$, $n = 2^N$.
- **Output**: $\prod_i b_i$. 

\[
\begin{align*}
\frac{b_1 \cdots b_n}{b_1 \cdots b_{n/2}} & \quad b_{n/2+1} \cdots b_n \\
\frac{b_1 b_2}{b_1} & \quad b_2 & \quad \cdots & \quad b_{n-1} & \quad b_{n-1} b_n \\
& \quad b_n
\end{align*}
\]
Remainder tree

- **Input**: $P, b_1, \cdots, b_n$, $n = 2^N$, product tree of $(b_1, \cdots, b_n)$.
- **Output**: $P \mod b_1, \cdots, P \mod b_n$. 
Remainder tree

- **Input**: $P, b_1, \cdots, b_n$, $n = 2^N$, product tree of $(b_1, \cdots, b_n)$.
- **Output**: $P \mod b_1, \cdots, P \mod b_n$. 
Remainder tree

- **Input**: $P, b_1, \cdots, b_n$, $n = 2^N$, product tree of $(b_1, \cdots, b_n)$.
- **Output**: $P \mod b_1, \cdots, P \mod b_n$.

\[
P \mod b_1 \cdots b_n
\]
Remainder tree

- **Input**: $P$, $b_1, \cdots, b_n$, $n = 2^N$, product tree of $(b_1, \cdots, b_n)$.
- **Output**: $P \mod b_1, \cdots, P \mod b_n$. 

![Remainder Tree Diagram]

Biasse-Jacobson (U of C)
Remainder tree

- **Input**: $P, b_1, \cdots, b_n$, $n = 2^N$, product tree of $(b_1, \cdots, b_n)$.
- **Output**: $P \mod b_1, \cdots, P \mod b_n$. 

![Diagram of the remainder tree]

Let $P$ be an integer. The remainder tree algorithm computes the remainders of $P$ modulo each $b_i$ efficiently by recursively computing remainders modulo halves of the product. This is particularly useful for large integers and can significantly reduce the computational cost compared to computing each modulo operation separately.
Remainder tree

- **Input**: $P, b_1, \cdots, b_n$, $n = 2^N$, product tree of $(b_1, \cdots, b_n)$.
- **Output**: $P \mod b_1, \cdots, P \mod b_n$. 

![Remainder tree diagram]

$P \mod b_1 \cdots b_n$

$P \mod b_1 \cdots b_{n/2} \quad P \mod b_{n/2+1} \cdots b_n$

$P \mod b_1 b_2$

$P \mod b_2 \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots

Biasse-Jacobson (U of C)

Fast smoothness test

October 2013 12 / 24
Batch smoothness test

Algorithm

- **Input**: $B > 0$, $b_1, \ldots, b_n$.
- **Output**: $B$-smooth part of each $b_i$.}

We start with the construction of the factor base $B = \{p_i \mid \deg(p_i) \leq B\}$. Calculate $P = \prod_i p_i$ with a product tree. Calculate the product tree of $b_1, \ldots, b_n$. Then, the remainder tree gives us $P \mod b_i$ for each $i \leq n$. Calculate $c_i := P^{2^e} \mod b_i$ with $e$ such that $2^e > \deg(b_i)$. If $c_i = 0$, $b_i$ is $B$-smooth. We compute $P^{2^e}$ to account for possible powers in the decomposition of $b_i$. 

Biasse-Jacobson (U of C)  
Fast smoothness test  
October 2013  
13 / 24
Batch smoothness test

Algorithm

- **Input**: $B > 0$, $b_1, \cdots, b_n$.
- **Output**: $B$-smooth part of each $b_i$.

We start with the construction of the factor base $\mathcal{B} = \{p_i | \deg(p_i) \leq B\}$.
- Calculate $P = \prod_i p_i$ with a product tree.
- Calculate the product tree of $b_1, \cdots, b_n$. 

Biasse-Jacobson (U of C)
Batch smoothness test

**Algorithm**

- **Input**: \( B > 0, b_1, \cdots, b_n \).
- **Output**: \( B \)-smooth part of each \( b_i \).

We start with the construction of the factor base \( \mathcal{B} = \{ p_i \mid \deg(p_i) \leq B \} \).

- Calculate \( P = \prod_i p_i \) with a product tree.
- Calculate the product tree of \( b_1, \cdots, b_n \).

Then, the remainder tree gives us \( P \mod b_i \) for each \( i \leq n \).

- Calculate \( c_i := P^{2^e} \mod b_i \) with \( e \) such that \( 2^e > \deg(b_i) \).
- If \( c_i = 0 \), \( b_i \) is \( B \)-smooth.
Batch smoothness test

**Algorithm**

- **Input**: $B > 0, b_1, \cdots, b_n$.
- **Output**: $B$-smooth part of each $b_i$.

We start with the construction of the factor base $\mathcal{B} = \{p_i \mid \deg(p_i) \leq B\}$.

- Calculate $P = \prod p_i$ with a product tree.
- Calculate the product tree of $b_1, \cdots, b_n$.

Then, the remainder tree gives us $P \mod b_i$ for each $i \leq n$.

- Calculate $c_i := P^{2^e} \mod b_i$ with $e$ such that $2^e > \deg(b_i)$.
- If $c_i = 0$, $b_i$ is $B$-smooth.

We compute $P^{2^e}$ to account for possible powers in the decomposition of $b_i$. 
1 Motivation

2 Bernstein's approach

3 Complexity analysis

4 Practical examples
Standard smoothness test

- Smoothness test in $\mathbb{F}_q[X]$ is more efficient than in $\mathbb{Z}$.
- Let $B > 0$ and $N \in \mathbb{F}_q[X]$ to be tested for $B$-smoothness.
Standard smoothness test

- Smoothness test in $\mathbb{F}_q[X]$ is more efficient than in $\mathbb{Z}$.
- Let $B > 0$ and $N \in \mathbb{F}_q[X]$ to be tested for $B$-smoothness.

**The standard algorithm**

Let $l' = \lfloor B/2 \rfloor + 1$ and $i = \lceil \deg(N)/q \rceil$.

- Compute $H = (X^{q^{l'}} + X)(X^{q^{l'}+1} + X) \cdots (X^{q^B} + X) \mod N$.
- $H \leftarrow H^{q^i} \mod N$.

If $H = 0$, then $N$ is $B$-smooth.
Standard smoothness test

- Smoothness test in $\mathbb{F}_q[X]$ is more efficient than in $\mathbb{Z}$.
- Let $B > 0$ and $N \in \mathbb{F}_q[X]$ to be tested for $B$-smoothness.

The standard algorithm

Let $l' = \lfloor B/2 \rfloor + 1$ and $i = \lceil \deg(N)/q \rceil$.

- Compute $H = (X^{q^{l'}} + X)(X^{q^{l'}+1} + X) \cdots (X^{q^B} + X) \mod N$.
- $H \leftarrow H^{q^i} \mod N$.

If $H = 0$, then $N$ is $B$-smooth.

The cost in operations in $\mathbb{F}_q$ is $O(\deg(N)^3 + B \deg(N)^2)$. 
Product tree with quadratic multiplication

- We assume that $\forall i \leq n, \deg(b_i) = g$.
- We assume that the multiplication has quadratic complexity.
Product tree with quadratic multiplication

- We assume that \( \forall i \leq n, \deg(b_i) = g \).
- We assume that the multiplication has quadratic complexity.

![Diagram of a product tree with quadratic multiplication]

Leaves: \( \frac{n}{2} \) multiplications of degree \( g \) polynomials

Root: 1 multiplication of degree \( ng/2 \) polynomials

Biasse-Jacobson (U of C)

Fast smoothness test

October 2013
Product tree with quadratic multiplication

- We assume that $\forall i \leq n$, $\deg(b_i) = g$.
- We assume that the multiplication has quadratic complexity.

```
<table>
<thead>
<tr>
<th>$b_1 \cdots b_n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b_1 \cdots b_{n/2}$</td>
</tr>
<tr>
<td>$b_1 b_2$</td>
</tr>
<tr>
<td>$b_1$</td>
</tr>
</tbody>
</table>

Leaves: $n/2$ multiplications of degree $g$ polynomials
```
Product tree with quadratic multiplication

- We assume that $\forall i \leq n, \deg(b_i) = g$.
- We assume that the multiplication has quadratic complexity.

Leaves: $n/2$ multiplications of degree $g$ polynomials

Leaves: amortized complexity $O(g^2)$
Product tree with quadratic multiplication

- We assume that $\forall i \leq n, \deg(b_i) = g$.
- We assume that the multiplication has quadratic complexity.

\[ b_1 \cdots b_n \]

\[ b_1 \cdots b_{n/2} \quad b_{n/2+1} \cdots b_n \]

\[ b_1b_2 \quad \ldots \quad b_{n-1}b_n \]

- **Root**: 1 multiplication of degree $ng/2$ polynomials
- **Leaves**: amortized complexity $O(g^2)$
We assume that $\forall i \leq n, \deg(b_i) = g$.

We assume that the multiplication has quadratic complexity.

- Leaves: $\frac{n}{2}$ multiplications of degree $g$ polynomials

Root: 1 multiplication of degree $\frac{ng}{2}$ polynomials

Leaves: amortized complexity $O(g^2)$

Root: amortized complexity $O(ng^2)$
Polynomial time multiplication

The complexity of multiplying degree-$g$ polynomials is in operations in $\mathbb{F}_q$. 

Naive multiplication

Direct application of the formula

$$c_i = \sum_{j+k=i} a_j b_k.$$

Complexity $O(g^2)$.

Karatsuba multiplication

Let $a = a_0 + x g/2 a_1$ and $b = b_0 + x g/2 b_1$. Then

$$ab = a_1 b_1 x g + (a_1 b_0 + a_0 b_1) x + a_0 b_0.$$

Complexity $O(g^{1.58})$. 

Biasse-Jacobson (U of C)

Fast smoothness test

October 2013 16 / 24
Polynomial time multiplication

The complexity of multiplying degree-$g$ polynomials is in operations in $\mathbb{F}_q$.

**Naive multiplication**
- Direct application of the formula $c_i = \sum_{j+k=i} a_j b_k$.
- Complexity $O(g^2)$. 
Polynomial time multiplication

The complexity of multiplying degree-\(g\) polynomials is in operations in \(\mathbb{F}_q\).

**Naive multiplication**

- Direct application of the formula \(c_i = \sum_{j+k=i} a_j b_k\).
- Complexity \(O(g^2)\).

**Karatsuba multiplication**

- Let \(a = a_0 + x^{g/2}a_1\) and \(b = b_0 + x^{g/2}b_1\). then

\[
ab = a_1b_1x^g + (a_1b_0 + a_0b_1)x + a_0b_0.
\]

- Complexity \(O(g^{1.58})\).
Schönhage-Strassen quasi-linear time multiplication

Let $R$ be a ring with an $g$-th root of unity $\omega$. We have the correspondence

$$P \in R[x] \text{ with } \deg(P) \leq g \iff (P(1), P(\omega), \ldots, P(\omega^{g-1})) =: \text{DFT}_\omega(P).$$
Schönhage-Strassen quasi-linear time multiplication

Let $R$ be a ring with an $g$-th root of unity $\omega$. We have the correspondence

$P \in R[x]$ with $\deg(P) \leq g \leftrightarrow (P(1), P(\omega), \cdots, P(\omega^{g-1})) =: \text{DFT}_\omega(P)$. 

**Interpolation of product**

Let $P, Q \in R[x]$, and $\omega$ a $g$-th root of unity then

$$\text{DFT}_\omega(PQ) = (P(1)Q(1), P(\omega)Q(\omega), \cdots, P(\omega^{g-1})Q(\omega^{g-1})).$$
Schönhage-Strassen quasi-linear time multiplication

Let $R$ be a ring with an $g$-th root of unity $\omega$. We have the correspondence

$P \in R[x]$ with $\deg(P) \leq g \leftrightarrow (P(1), P(\omega), \ldots, P(\omega^{g-1})) =: \text{DFT}_\omega(P)$.

Interpolation of product

Let $P, Q \in R[x]$, and $\omega$ a $g$-th root of unity then

$$\text{DFT}_\omega(PQ) = (P(1)Q(1), P(\omega)Q(\omega), \ldots, P(\omega^{g-1})Q(\omega^{g-1})).$$

- Multiplying polynomials boils down to point-wise multiplication in $R^g$.
- Complexity in $O(g \log(g))$ operations in $R$. 
Schönhage-Strassen quasi-linear time multiplication

Let $R$ be a ring with an $g$-th root of unity $\omega$. We have the correspondance

$P \in R[x]$ with $\deg(P) \leq g \longleftrightarrow (P(1), P(\omega), \cdots, P(\omega^{g-1})) =: \text{DFT}_\omega(P)$.

Interpolation of product

Let $P, Q \in R[x]$, and $\omega$ a $g$-th root of unity then

$\text{DFT}_\omega(PQ) = (P(1)Q(1), P(\omega)Q(\omega), \cdots, P(\omega^{g-1})Q(\omega^{g-1})).$

- Multiplying polynomials boils down to point-wise multiplication in $R^g$.
- Complexity in $O(g \log(g))$ operations in $R$.

There is a version for rings with unity and a specific one for $\text{char}(K) = 2$. 
Product tree with fast multiplication

\[ b_1 \quad \quad \quad \quad \quad \quad \quad \quad b_2 \quad \cdots \quad \cdots \quad \cdots \quad \cdots \quad \cdots \quad b_{n-1} \quad \quad \quad \quad \quad \quad \quad \quad b_n \]
Product tree with fast multiplication

Leaves: \( \frac{n}{2} \) multiplications of degree \( g \) polynomials

Root: 1 multiplication of degree \( \frac{ng}{2} \) polynomials

Biasse-Jacobson (U of C)
Product tree with fast multiplication

Leaves: $n/2$ multiplications of degree $g$ polynomials
Product tree with fast multiplication

\[ b_1 \cdots b_n \]

\[ b_1 \cdots b_{n/2} \quad \quad \quad b_{n/2+1} \cdots b_n \]

\[ b_1 b_2 \quad \quad \quad \quad \quad \quad b_{n-1} b_n \]

Leaves: \( n/2 \) multiplications of degree \( g \) polynomials

Leaves: amortized complexity \( O(g^2) \)
Product tree with fast multiplication

Root: 1 multiplication of degree $ng/2$ polynomials

Leaves: $n/2$ multiplications of degree $g$ polynomials

Leaves: amortized complexity $O(g^2)$

Biasse-Jacobson (U of C)
Product tree with fast multiplication

Root: 1 multiplication of degree $ng/2$ polynomials

Leaves: $n/2$ multiplications of degree $g$ polynomials

Leaves: amortized complexity $O(g^2)$

Root: amortized complexity $O((\log(n) + \log(g))g)$
Optimal size of batch

- Constraint 1: minimizing $\log(n)$.
- Constraint 2: Ensuring $\deg(P) \leq \deg(b_1 \cdots b_n)$.

If $\deg(P) > \deg(b_1 \cdots b_n)$, the cost of the remainder tree is not amortized.
Optimal size of batch

- Constraint 1: minimizing \( \log(n) \).
- Constraint 2: Ensuring \( \deg(P) \leq \deg(b_1 \cdots b_n) \).

If \( \deg(P) > \deg(b_1 \cdots b_n) \), the cost of the remainder tree is not amortized.
Optimal size of batch

- Constraint 1: minimizing $\log(n)$.
- Constraint 2: Ensuring $\deg(P) \leq \deg(b_1 \cdots b_n)$.

If $\deg(P) > \deg(b_1 \cdots b_n)$, the cost of the remainder tree is not amortized.

The optimal solution is $\deg(P) = \deg(b_1 \cdots b_n)$
Overall complexity

- We are given $n$ degree-$g$ polynomials.
- Let $P = p_1 \cdots p_k$. 

Product and remainder tree take $O(\log(n)g^2)$. The exponentiations take $O(g^2 \log(g))$. The overall complexity is in $O(g^2(\log(g) + \deg(P)g))$. 

Biasse-Jacobson (U of C) Fast smoothness test October 2013 20 / 24
Overall complexity

- We are given $n$ degree-$g$ polynomials.
- Let $P = p_1 \cdots p_k$.

We choose a size of batch $n$ such that $ng = \deg(P)$.
- Product and remainder tree take $O(\log(n)g^2)$.
- The exponentiations take $O(g^2 \log(g))$. 
Overall complexity

- We are given \( n \) degree-\( g \) polynomials.
- Let \( P = p_1 \cdots p_k \).

We choose a size of batch \( n \) such that \( ng = \deg(P) \).

- Product and remainder tree take \( O(\log(n)g^2) \).
- The exponentiations take \( O(g^2 \log(g)) \).

The overall complexity is in

\[
O \left( g^2 \left( \log(g) + \frac{\deg(P)}{g} \right) \right).
\]
1 Motivation

2 Bernstein's approach

3 Complexity analysis

4 Practical examples
Comparison between multiplication methods

- We compare multiplication methods in $\mathbb{F}_{2^5}[X]$.
- We use the C++ library Mathemagix.
### Comparison between multiplication methods

- We compare multiplication methods in $\mathbb{F}_{25}[X]$.
- We use the C++ library Mathemagix.

<table>
<thead>
<tr>
<th>Degree</th>
<th>Karatsuba</th>
<th>generic FFT</th>
<th>triadic FFT</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0</td>
<td>110</td>
<td>60</td>
</tr>
<tr>
<td>50</td>
<td>60</td>
<td>580</td>
<td>250</td>
</tr>
<tr>
<td>100</td>
<td>260</td>
<td>2280</td>
<td>1620</td>
</tr>
<tr>
<td>500</td>
<td>6530</td>
<td>10650</td>
<td>5130</td>
</tr>
<tr>
<td>1000</td>
<td>26110</td>
<td>27680</td>
<td>16500</td>
</tr>
<tr>
<td>2000</td>
<td>105970</td>
<td>41260</td>
<td>17360</td>
</tr>
<tr>
<td>3000</td>
<td>245220</td>
<td>100280</td>
<td>53480</td>
</tr>
</tbody>
</table>
Comparison between multiplication methods

- We compare multiplication methods in $\mathbb{F}_{25}[X]$.
- We use the C++ library Mathemagix.

<table>
<thead>
<tr>
<th>Degree</th>
<th>Karatsuba</th>
<th>generic FFT</th>
<th>triadic FFT</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0</td>
<td>110</td>
<td>60</td>
</tr>
<tr>
<td>50</td>
<td>60</td>
<td>580</td>
<td>250</td>
</tr>
<tr>
<td>100</td>
<td>260</td>
<td>2280</td>
<td>1620</td>
</tr>
<tr>
<td>500</td>
<td>6530</td>
<td>10650</td>
<td>5130</td>
</tr>
<tr>
<td>1000</td>
<td>26110</td>
<td>27680</td>
<td>16500</td>
</tr>
<tr>
<td>2000</td>
<td>105970</td>
<td>41260</td>
<td>17360</td>
</tr>
<tr>
<td>3000</td>
<td>245220</td>
<td>100280</td>
<td>53480</td>
</tr>
</tbody>
</table>

The times are in CPU msec.
Test in $\mathbb{F}_{2^k}[X]$

We use elliptic curves over $E_x$ defined over $\mathbb{F}_x$ defining hyperelliptic curves via a Weil descent\(^1\).

- $C_{124}$: genus 31 hyperelliptic curve over $\mathbb{F}_{2^4}$ arizing from $E_{124}$.
- $C_{155}$: genus 31 hyperelliptic curve over $\mathbb{F}_{2^5}$ arizing from $E_{155}$.

---

1. Jacobson,Menezes,Stein-2001
Test in $\mathbb{F}_2^k[X]$

We use elliptic curves over $E_x$ defined over $\mathbb{F}_x$ defining hyperelliptic curves via a Weil descent $^1$.

- $C_{124}$: genus 31 hyperelliptic curve over $\mathbb{F}_{2^4}$ arising from $E_{124}$.
- $C_{155}$: genus 31 hyperelliptic curve over $\mathbb{F}_{2^5}$ arising from $E_{155}$.

<table>
<thead>
<tr>
<th>Curve</th>
<th>B</th>
<th>Batch size</th>
<th>$\deg(P)$</th>
<th>Time standard</th>
<th>Time Batch</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_{124}$</td>
<td>4</td>
<td>1933</td>
<td>69616</td>
<td>24m 32s</td>
<td>13n 38s</td>
</tr>
<tr>
<td>$C_{155}$</td>
<td>4</td>
<td>30036</td>
<td>1081312</td>
<td>27m 33s</td>
<td>21m 21s</td>
</tr>
</tbody>
</table>

1. Jacobson, Menezes, Stein-2001
Test in $\mathbb{F}_{2^k}[X]$ 

We use elliptic curves over $E_x$ defined over $\mathbb{F}_x$ defining hyperelliptic curves via a Weil descent $^1$.

- $C_{124}$: genus 31 hyperelliptic curve over $\mathbb{F}_{2^4}$ arising from $E_{124}$.
- $C_{155}$: genus 31 hyperelliptic curve over $\mathbb{F}_{2^5}$ arising from $E_{155}$.

<table>
<thead>
<tr>
<th>Curve</th>
<th>B</th>
<th>Batch size</th>
<th>$\deg(P)$</th>
<th>Time standard</th>
<th>Time Batch</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_{124}$</td>
<td>4</td>
<td>1933</td>
<td>69616</td>
<td>24m 32s</td>
<td>13n 38s</td>
</tr>
<tr>
<td>$C_{155}$</td>
<td>4</td>
<td>30036</td>
<td>1081312</td>
<td>27m 33s</td>
<td>21m 21s</td>
</tr>
</tbody>
</table>

Times correspond to the test of $\approx 1000000$ polynomials.

- Times are in CPU sec.
- Multiplication is Karatsuba from the NTL library.

---

1. Jacobson, Menezes, Stein-2001
Test in $\mathbb{F}_2[X]$

NFS is used\(^2\) to solve the DLP in $\mathbb{F}_{2^{1039}}$. The sieve selects cofactors.

- The smoothness bound is 25.
- The large prime bound is 33.
- Cofactors have degree 99 in $\mathbb{F}_2[X]$.

---

2. Detrey, Gaudry, Videau-2013
Test in $\mathbb{F}_2[X]$

NFS is used to solve the DLP in $\mathbb{F}_{2^{1039}}$. The sieve selects cofactors.

- The smoothness bound is 25.
- The large prime bound is 33.
- Cofactors have degree 99 in $\mathbb{F}_2[X]$.

Parameters

We have the following parameters:

- $\text{deg}(P) = 67100116$.
- Batch size is 677778.

2. Detrey, Gaudry, Videau-2013
Test in $\mathbb{F}_2[X]$

NFS is used to solve the DLP in $\mathbb{F}_{2^{1039}}$. The sieve selects cofactors.

- The smoothness bound is 25.
- The large prime bound is 33.
- Cofactors have degree 99 in $\mathbb{F}_2[X]$.

Parameters

We have the following parameters:

- $\deg(P) = 67100116$.
- Batch size is 677778.

We test 1355556 polynomials using the library gf2x which includes FFT.

- Standard takes 5 m 8 s.
- Batch test takes 4 m 13 s.

---

2. Detrey, Gaudry, Videau-2013
Conclusion

This is work in progress. We have achieved the following:

- Design a theoretical model showing the improvement of the batch test.
- Show that the corresponding values are within practical range for the use of fast multiplication.
- Show that we can achieve a speed-up without fast multiplication.

We still have to incorporate the fast multiplication in $F_2^m$. 

Refine the model to illustrate the optimal batch size.
Conclusion

This is work in progress. We have achieved the following:

- Design a theoretical model showing the improvement of the batch test.
- Show that the corresponding values are within practical range for the use of fast multiplication.
- Show that we can achieve a speed-up without fast multiplication.

We still have to

- Incorporate the fast multiplication in $\mathbb{F}_{2^m}[X]$.
- Refine the model to illustrate the optimal batch size.