Complexity Bound on Semaev's Naive Index Calculus Method for ECDLP

Kazuhiro Yokoyama [1] Masaya Yasuda[2] Yasushi Takahashi[3]
Jun Kogure[3]

[1] Rikkyo University, Tokyo, Japan

[2] Kyushu University, Fukuoka, Japan

[3] FUJITSU Laboratories LTD, Kawasaki, Japan

24/06/2019

Aim, Goal and Strategy

Aim, Goal and Strategy

- Aim:

Aim, Goal and Strategy

- Aim: Give a precise analysis on the complexity of Index Calculus Method (ICM) for ECDLP.

Aim, Goal and Strategy

- Aim: Give a precise analysis on the complexity of Index Calculus Method (ICM) for ECDLP. As the first step, we consider a naive ICM.

Aim, Goal and Strategy

- Aim: Give a precise analysis on the complexity of Index Calculus Method (ICM) for ECDLP. As the first step, we consider a naive ICM.
- Goal:

Aim, Goal and Strategy

- Aim: Give a precise analysis on the complexity of Index Calculus Method (ICM) for ECDLP. As the first step, we consider a naive ICM.
- Goal: Under statistical assumption based on linear algebra related to Subresultant Theory, we show that the naive ICM cannot be more efficient than brute-force methods for ECDLP.

Aim, Goal and Strategy

- Aim: Give a precise analysis on the complexity of Index Calculus Method (ICM) for ECDLP. As the first step, we consider a naive ICM.
- Goal: Under statistical assumption based on linear algebra related to Subresultant Theory, we show that the naive ICM cannot be more efficient than brute-force methods for ECDLP.
- Strategy:

Aim, Goal and Strategy

- Aim: Give a precise analysis on the complexity of Index Calculus Method (ICM) for ECDLP. As the first step, we consider a naive ICM.
- Goal: Under statistical assumption based on linear algebra related to Subresultant Theory, we show that the naive ICM cannot be more efficient than brute-force methods for ECDLP.
- Strategy: Estimate an average lower bound on the complexity of Gröbner basis computation.

Aim, Goal and Strategy

- Aim: Give a precise analysis on the complexity of Index Calculus Method (ICM) for ECDLP. As the first step, we consider a naive ICM.
- Goal: Under statistical assumption based on linear algebra related to Subresultant Theory, we show that the naive ICM cannot be more efficient than brute-force methods for ECDLP.
- Strategy: Estimate an average lower bound on the complexity of Gröbner basis computation.
  - The dominant part is solving the Point Decomposition Problem, to which Gröbner basis computation is applied.

Aim, Goal and Strategy

- Aim: Give a precise analysis on the complexity of Index Calculus Method (ICM) for ECDLP. As the first step, we consider a naive ICM.
- Goal: Under statistical assumption based on linear algebra related to Subresultant Theory, we show that the naive ICM cannot be more efficient than brute-force methods for ECDLP.
- Strategy: Estimate an average lower bound on the complexity of Gröbner basis computation.
  - The dominant part is solving the Point Decomposition Problem, to which Gröbner basis computation is applied.
  - To understand the computational behavior of Gröbner basis computation very precisely, we focus on S-polynomial computation, the main step of Gröbner basis computation.

Aim, Goal and Strategy

- Aim: Give a precise analysis on the complexity of Index Calculus Method (ICM) for ECDLP. As the first step, we consider a naive ICM.
- Goal: Under statistical assumption based on linear algebra related to Subresultant Theory, we show that the naive ICM cannot be more efficient than brute-force methods for ECDLP.
- Strategy: Estimate an average lower bound on the complexity of Gröbner basis computation.
  - The dominant part is solving the Point Decomposition Problem, to which Gröbner basis computation is applied.
  - To understand the computational behavior of Gröbner basis computation very precisely, we focus on S-polynomial computation, the main step of Gröbner basis computation.
  - Introducing the notion signature, we find that the generatd sequence of S-polynomials is very similar to regular polynomial remainder sequence in the Euclidean algorithm for GCD computation.

ECDLP and ICM

ECDLP and ICM

- DLP: Let G be a finite group generated by α, and its order n. For a given element β in G, find a positive integer ℓ such that $\beta = \alpha^{\ell}$.

ECDLP and ICM

- DLP: Let G be a finite group generated by α, and its order n. For a given element β in G, find a positive integer ℓ such that $\beta = \alpha^{\ell}$.
- ECDLP: DLP defined over the additive group $E(K)$ of the rational points of an elliptic curve E over a finite field K.

ECDLP and ICM

- DLP: Let G be a finite group generated by α, and its order n. For a given element β in G, find a positive integer ℓ such that $\beta = \alpha^{\ell}$.
- ECDLP: DLP defined over the additive group $E(K)$ of the rational points of an elliptic curve E over a finite field K.
  For a give points $P, Q \in E(K)$, find a positive integer ℓ such that $Q = \ell P$. (We assume that $Q \in \langle P \rangle$.) We denote ℓ by $\log_P(Q)$.

ECDLP and ICM

- DLP: Let G be a finite group generated by α, and its order n. For a given element β in G, find a positive integer ℓ such that $\beta = \alpha^{\ell}$.
- ECDLP: DLP defined over the additive group $E(K)$ of the rational points of an elliptic curve E over a finite field K.
  For a give points $P, Q \in E(K)$, find a positive integer ℓ such that $Q = \ell P$. (We assume that $Q \in \langle P \rangle$.) We denote ℓ by $\log_P(Q)$.
- As a generic method for DLP, the ICM and its variants are proposed.

ECDLP and ICM

- DLP: Let G be a finite group generated by α, and its order n. For a given element β in G, find a positive integer ℓ such that $\beta = \alpha^{\ell}$.
- ECDLP: DLP defined over the additive group $E(K)$ of the rational points of an elliptic curve E over a finite field K.
  For a give points $P, Q \in E(K)$, find a positive integer ℓ such that $Q = \ell P$. (We assume that $Q \in \langle P \rangle$.) We denote ℓ by $\log_P(Q)$.
- As a generic method for DLP, the ICM and its variants are proposed. For the DLP over $(\mathbb{Z}/p\mathbb{Z})^*$, it attains Sub-Exponential time-complexity.

ECDLP and ICM

- DLP: Let G be a finite group generated by α, and its order n. For a given element β in G, find a positive integer ℓ such that $\beta = \alpha^{\ell}$.
- ECDLP: DLP defined over the additive group $E(K)$ of the rational points of an elliptic curve E over a finite field K.
  For a give points $P, Q \in E(K)$, find a positive integer ℓ such that $Q = \ell P$. (We assume that $Q \in \langle P \rangle$.) We denote ℓ by $\log_P(Q)$.
- As a generic method for DLP, the ICM and its variants are proposed. For the DLP over $(\mathbb{Z}/p\mathbb{Z})^*$, it attains Sub-Exponential time-complexity.
- There are a number of advanced (improved) ICM methods for ECDLP. Especially, ICMs with the Weil descent technique are studied as the fastest ones.

ECDLP and ICM

- DLP: Let G be a finite group generated by α, and its order n. For a given element β in G, find a positive integer ℓ such that $\beta = \alpha^{\ell}$.
- ECDLP: DLP defined over the additive group $E(K)$ of the rational points of an elliptic curve E over a finite field K.
  For a give points $P, Q \in E(K)$, find a positive integer ℓ such that $Q = \ell P$. (We assume that $Q \in \langle P \rangle$.) We denote ℓ by $\log_P(Q)$.
- As a generic method for DLP, the ICM and its variants are proposed. For the DLP over $(\mathbb{Z}/p\mathbb{Z})^*$, it attains Sub-Exponential time-complexity.
- There are a number of advanced (improved) ICM methods for ECDLP. Especially, ICMs with the Weil descent technique are studied as the fastest ones. Also, there are several conjectures about upper bounds on the complexity of Gröbner basis computation used in ICMs.

Generic Form of Index Calculus Method for ECDLP

Generic Form of Index Calculus Method for ECDLP

Generic Form of Index Calculus Method for ECDLP

(Step 1) Take a subset $\mathcal{B}$ of $E(K)$. We number elements of $\mathcal{B}$ as $\mathcal{B} = \{B_1, \ldots, B_t\}$.

Generic Form of Index Calculus Method for ECDLP

(Step 1) Take a subset $\mathcal{B}$ of $E(K)$. We number elements of $\mathcal{B}$ as $\mathcal{B} = \{B_1, \ldots, B_t\}$.

(Step 2)

Generic Form of Index Calculus Method for ECDLP

(Step 1) Take a subset $\mathcal{B}$ of $E(K)$. We number elements of $\mathcal{B}$ as $\mathcal{B} = \{B_1, \ldots, B_t\}$.
(Step 2) For randomly chosen two integers $0 \leq a, b < n = |E(K)|$, find the following decomposition:

$$aP + bQ = B_{s_1} + \cdots + B_{s_m}, \quad m \in \{1, \ldots, t\},$$

where we fix m in advance.

Generic Form of Index Calculus Method for ECDLP

(Step 1) Take a subset $\mathcal{B}$ of $E(K)$. We number elements of $\mathcal{B}$ as $\mathcal{B} = \{B_1, \ldots, B_t\}$.
(Step 2) For randomly chosen two integers $0 \leq a, b < n = |E(K)|$, find the following decomposition:

$$aP + bQ = B_{s_1} + \cdots + B_{s_m}, \quad m \in \{1, \ldots, t\},$$

where we fix m in advance. When it succeeds, one has the following linear congruence:

$$a + b \log_P(Q) \equiv \sum_{i=1}^{m} \log_P(B_{s_i}) \pmod{n}.$$

Generic Form of Index Calculus Method for ECDLP

(Step 1) Take a subset $\mathcal{B}$ of $E(K)$. We number elements of $\mathcal{B}$ as $\mathcal{B} = \{B_1, \ldots, B_t\}$.
(Step 2) For randomly chosen two integers $0 \leq a, b < n = |E(K)|$, find the following decomposition:

$$aP + bQ = B_{s_1} + \cdots + B_{s_m}, \quad m \in \{1, \ldots, t\},$$

where we fix m in advance. When it succeeds, one has the following linear congruence:

$$a + b \log_P(Q) \equiv \sum_{i=1}^{m} \log_P(B_{s_i}) \pmod{n}.$$

We call this the point decomposition problem (PDP).

Generic Form of Index Calculus Method for ECDLP

(Step 1) Take a subset $\mathcal{B}$ of $E(K)$. We number elements of $\mathcal{B}$ as $\mathcal{B} = \{B_1, \ldots, B_t\}$.
(Step 2) For randomly chosen two integers $0 \leq a, b < n = |E(K)|$, find the following decomposition:

$$aP + bQ = B_{s_1} + \cdots + B_{s_m}, \quad m \in \{1, \ldots, t\},$$

where we fix m in advance. When it succeeds, one has the following linear congruence:

$$a + b \log_P(Q) \equiv \sum_{i=1}^{m} \log_P(B_{s_i}) \pmod{n}.$$

We call this the point decomposition problem (PDP).
(Step 3) After collecting enough number of linear congruences, we compute $\log_P(Q)$ by linear algebra.

Generic Form of Index Calculus Method for ECDLP

(Step 1) Take a subset $\mathcal{B}$ of $E(K)$. We number elements of $\mathcal{B}$ as $\mathcal{B} = \{B_1, \ldots, B_t\}$.
(Step 2) For randomly chosen two integers $0 \leq a, b < n = |E(K)|$, find the following decomposition:

$$aP + bQ = B_{s_1} + \cdots + B_{s_m}, \quad m \in \{1, \ldots, t\},$$

where we fix m in advance. When it succeeds, one has the following linear congruence:

$$a + b \log_P(Q) \equiv \sum_{i=1}^{m} \log_P(B_{s_i}) \pmod{n}.$$

We call this the point decomposition problem (PDP).
(Step 3) After collecting enough number of linear congruences, we compute $\log_P(Q)$ by linear algebra. In fact, after collecting $t+1$ linearly independent congruences, $\log_p(Q)$ can be determined.

Solving PDP by Algebraic Computation

Solving PDP by Algebraic Computation

Solving PDP by Algebraic Computation

(Detailed Step 2)

Solving PDP by Algebraic Computation

(Detailed Step 2) For randomly chosen a, b, finding points $B_1, \ldots, B_m$ in $\mathcal{B}$ such that

$$aP + bQ = B_1 + \cdots + B_m$$

Solving PDP by Algebraic Computation

(Detailed Step 2) For randomly chosen a, b, finding points $B_1, \ldots, B_m$ in $\mathcal{B}$ such that

$$aP + bQ = B_1 + \cdots + B_m$$

can be reduced to finding a solution $\alpha_1, \ldots, \alpha_m \in V = \{x(P) \mid P \in \mathcal{B}\}$ such that

$$S_{m+1}(\alpha_1, \ldots, \alpha_m, x(aP + bQ)) = 0,$$

for the Semaev's summation polynomial S_{m+1}.

Solving PDP by Algebraic Computation

(Detailed Step 2) For randomly chosen a, b, finding points $B_1, \ldots, B_m$ in $\mathcal{B}$ such that

$$aP + bQ = B_1 + \cdots + B_m$$

can be reduced to finding a solution $\alpha_1, \ldots, \alpha_m \in V = \{x(P) \mid P \in \mathcal{B}\}$ such that

$$S_{m+1}(\alpha_1, \ldots, \alpha_m, x(aP + bQ)) = 0,$$

for the Semaev's summation polynomial S_{m+1}. Letting $\beta = x(aP + bQ)$ and

$$F(x) = \prod_{\alpha \in V} (x - \alpha),$$

Solving PDP by Algebraic Computation

(Detailed Step 2) For randomly chosen a, b, finding points $B_1, \ldots, B_m$ in $\mathcal{B}$ such that

$$aP + bQ = B_1 + \cdots + B_m$$

can be reduced to finding a solution $\alpha_1, \ldots, \alpha_m \in V = \{x(P) \mid P \in \mathcal{B}\}$ such that

$$S_{m+1}(\alpha_1, \ldots, \alpha_m, x(aP + bQ)) = 0,$$

for the Semaev's summation polynomial S_{m+1}. Letting $\beta = x(aP + bQ)$ and

$$F(x) = \prod_{\alpha \in V} (x - \alpha),$$

the problem is reduced to solving the following system:

$$\mathcal{S}ys(\beta) : \begin{cases} S_{m+1}(x_1, \ldots, x_m, \beta) = 0 \\ F(x_1) = 0 \\ \vdots \quad \vdots \quad \vdots \\ F(x_m) = 0 \end{cases}$$

Semaev's Summation Polynomial

Semaev's Summation Polynomial

Semaev's Summation Polynomial

Let r be a positive integer greater than 1. There is a polynomial $S_r(x_1, \ldots, x_r)$ in r variables over K, called the (Semaev's) summation polynomial of order r, which satisfies the following:

For each $b_1, \ldots, b_r \in \overline{K}$ with $S_r(b_1, \ldots, b_r) = 0$, there are points $P_1, \ldots, P_r$ in $E(\overline{K})$ such that

$$P_1 + \cdots + P_r = O,$$

and $x(P_i) = b_i$ for $1 \leq i \leq r$.

Semaev's Summation Polynomial

Semaev's Summation Polynomial

Let r be a positive integer greater than 1. There is a polynomial $S_r(x_1, \ldots, x_r)$ in r variables over K, called the (Semaev's) summation polynomial of order r, which satisfies the following:

For each $b_1, \ldots, b_r \in \overline{K}$ with $S_r(b_1, \ldots, b_r) = 0$, there are points $P_1, \ldots, P_r$ in $E(\overline{K})$ such that

$$P_1 + \cdots + P_r = O,$$

and $x(P_i) = b_i$ for $1 \leq i \leq r$. ($\overline{K}$ denotes the algebraic closure of K.)

The Semaev's summation polynomial of order r can be computed from that of order $r-1$ and that of order 3 by elimination technique:

Semaev's Summation Polynomial

Semaev's Summation Polynomial

Let r be a positive integer greater than 1. There is a polynomial $S_r(x_1, \ldots, x_r)$ in r variables over K, called the (Semaev's) summation polynomial of order r, which satisfies the following:

For each $b_1, \ldots, b_r \in \overline{K}$ with $S_r(b_1, \ldots, b_r) = 0$, there are points $P_1, \ldots, P_r$ in $E(\overline{K})$ such that

$$P_1 + \cdots + P_r = O,$$

and $x(P_i) = b_i$ for $1 \leq i \leq r$. ($\overline{K}$ denotes the algebraic closure of K.)

The Semaev's summation polynomial of order r can be computed from that of order $r-1$ and that of order 3 by elimination technique:

$$S_r = \mathrm{Res}_y(S_{r-1}(x_1, \ldots, x_{r-2}, y), S_3(x_{r-1}, x_r, y)),$$

where Res_y denotes the resultant with respect to y.

Complexity of Naive ICM

Complexity of Naive ICM

Complexity of Naive ICM

- Let $\mathcal{P}rob_{\mathrm{PDP}}$ be the probability (ratio) of success of solving PDP for randomly chosen $aP + bQ$.
- Let C_{dec} be the average cost for solving $\mathcal{S}ys(x(aP + bQ))$.

Complexity of Naive ICM

- Let $\mathcal{P}rob_{\mathrm{PDP}}$ be the probability (ratio) of success of solving PDP for randomly chosen $aP + bQ$.
- Let C_{dec} be the average cost for solving $\mathcal{S}ys(x(aP + bQ))$.
- Then the total (average) cost is, at least,

$$\frac{\#V \times C_{\mathrm{dec}}}{\mathcal{P}rob_{\mathrm{PDP}} \times \gamma},$$

where γ denotes the average number of distinct solutions which $\mathcal{S}ys(x(aP + bQ))$ has.

Complexity of Naive ICM

- Let $\mathcal{P}rob_{\mathrm{PDP}}$ be the probability (ratio) of success of solving PDP for randomly chosen $aP + bQ$.
- Let C_{dec} be the average cost for solving $\mathcal{S}ys(x(aP + bQ))$.
- Then the total (average) cost is, at least,

$$\frac{\#V \times C_{\mathrm{dec}}}{\mathcal{P}rob_{\mathrm{PDP}} \times \gamma},$$

  where γ denotes the average number of distinct solutions which $\mathcal{S}ys(x(aP + bQ))$ has.
- In our setting, $\#V \approx t$ and $\mathcal{P}rob_{\mathrm{PDP}} \approx \frac{t^m}{\gamma q}$, where $q = \#K$, that is, $K = \mathbb{F}_q$.

Gröbner Basis Computation for Ideals of Special Form

Gröbner Basis Computation for Ideals of Special Form

Estimating the cost of Gröbner basis computation for PDP, we can obtain the total estimation of our Naive ICM.

Gröbner Basis Computation for Ideals of Special Form

Estimating the cost of Gröbner basis computation for PDP, we can obtain the total estimation of our Naive ICM.

Let $I_m(\beta)$ be the ideal associated to the system $\mathcal{S}ys(\beta)$ generated by

$$\mathcal{F}(\beta) = \{F(x_1), F(x_2), \ldots, F(x_m), S_{m+1}(x_1, \ldots, x_m, \beta)\}.$$

Gröbner Basis Computation for Ideals of Special Form

Estimating the cost of Gröbner basis computation for PDP, we can obtain the total estimation of our Naive ICM.
Let $I_m(\beta)$ be the ideal associated to the system $\mathcal{S}ys(\beta)$ generated by

$$\mathcal{F}(\beta) = \{F(x_1), F(x_2), \ldots, F(x_m), S_{m+1}(x_1, \ldots, x_m, \beta)\}.$$

To compute its Gröbner basis, we set a rev-lex monomial ordering as the most efficient one.

Gröbner Basis Computation for Ideals of Special Form

Estimating the cost of Gröbner basis computation for PDP, we can obtain the total estimation of our Naive ICM.
Let $I_m(\beta)$ be the ideal associated to the system $\mathcal{S}ys(\beta)$ generated by

$$\mathcal{F}(\beta) = \{F(x_1), F(x_2), \ldots, F(x_m), S_{m+1}(x_1, \ldots, x_m, \beta)\}.$$

To compute its Gröbner basis, we set a rev-lex monomial ordering as the most efficient one.
We say that the generator $\mathcal{F}(\beta)$ is of special form and consider the ideal in general setting.

Gröbner Basis Computation for Ideals of Special Form

Estimating the cost of Gröbner basis computation for PDP, we can obtain the total estimation of our Naive ICM.
Let $I_m(\beta)$ be the ideal associated to the system $\mathcal{S}ys(\beta)$ generated by

$$\mathcal{F}(\beta) = \{F(x_1), F(x_2), \ldots, F(x_m), S_{m+1}(x_1, \ldots, x_m, \beta)\}.$$

To compute its Gröbner basis, we set a rev-lex monomial ordering as the most efficient one.
We say that the generator $\mathcal{F}(\beta)$ is of special form and consider the ideal in general setting.

Let I be the ideal generated by

$$\mathcal{F} = \{F(x_1), F(x_2), \ldots, F(x_m), S(x_1, \ldots, x_m))\},$$

and J the ideal generated by $\mathcal{G}_0 = \{F(x_1), \ldots, F(x_m)\}$.

Gröbner Basis Computation for Ideals of Special Form

Estimating the cost of Gröbner basis computation for PDP, we can obtain the total estimation of our Naive ICM.
Let $I_m(\beta)$ be the ideal associated to the system $\mathcal{S}ys(\beta)$ generated by

$$\mathcal{F}(\beta) = \{F(x_1), F(x_2), \ldots, F(x_m), S_{m+1}(x_1, \ldots, x_m, \beta)\}.$$

To compute its Gröbner basis, we set a rev-lex monomial ordering as the most efficient one.
We say that the generator $\mathcal{F}(\beta)$ is of special form and consider the ideal in general setting.

Let I be the ideal generated by

$$\mathcal{F} = \{F(x_1), F(x_2), \ldots, F(x_m), S(x_1, \ldots, x_m))\},$$

and J the ideal generated by $\mathcal{G}_0 = \{F(x_1), \ldots, F(x_m)\}$. Then J is 0 dimensional and radical. Also let $d = \deg(F)$.

Main Statement

Main Statement

Under Assumption related Subresultant Theory,

Main Statement

Under Assumption related Subresultant Theory, for computation of a Gröbner basis of $I_m(\beta)$, it computes at least dm^{d-1} S-polynomials.

Main Statement

Under Assumption related Subresultant Theory, for computation of a Gröbner basis of $I_m(\beta)$, it computes at least dm^{d-1} S-polynomials.

$$\Downarrow$$

Our naive ICM cannot be more efficient than brute-force methods.

Main Statement

Under Assumption related Subresultant Theory, for computation of a Gröbner basis of $I_m(\beta)$, it computes at least dm^{d-1} S-polynomials.

$$\Downarrow$$

Our naive ICM cannot be more efficient than brute-force methods.

We are considering $S_{m+1}(x_1, \ldots, x_m, \beta)$ for a number of β, most of which become dense polynomials. Also they are symmetic in $x_1, \ldots, x_m$.

Main Statement

Under Assumption related Subresultant Theory, for computation of a Gröbner basis of $I_m(\beta)$, it computes at least dm^{d-1} S-polynomials.

$$\Downarrow$$

Our naive ICM cannot be more efficient than brute-force methods.

We are considering $S_{m+1}(x_1, \ldots, x_m, \beta)$ for a number of β, most of which become dense polynomials. Also they are symmetic in $x_1, \ldots, x_m$.

Considering the ideal $I_m(\beta)$ over $\mathbb{Q}$, we may say

Main Statement

Under Assumption related Subresultant Theory, for computation of a Gröbner basis of $I_m(\beta)$, it computes at least dm^{d-1} S-polynomials.

$$\Downarrow$$

Our naive ICM cannot be more efficient than brute-force methods.

We are considering $S_{m+1}(x_1, \ldots, x_m, \beta)$ for a number of β, most of which become dense polynomials. Also they are symmetic in $x_1, \ldots, x_m$.

Considering the ideal $I_m(\beta)$ over $\mathbb{Q}$, we may say

if the lower bound holds for some values β and modulus q, it also holds for almost every β and q.

Key Arguments for Counting Necessary S-Polynomials

Key Arguments for Counting Necessary S-Polynomials

- Signature-Based Algorithm:

Key Arguments for Counting Necessary S-Polynomials

- Signature-Based Algorithm: Signature-based algorithms, such as F_5 or its variants, are designed to discard unnecessary S-polynomials as much as possible, by using the notion signature.

Key Arguments for Counting Necessary S-Polynomials

- Signature-Based Algorithm: Signature-based algorithms, such as F_5 or its variants, are designed to discard unnecessary S-polynomials as much as possible, by using the notion signature. Since reduction operations are restricted to keep the signature, it is not proven rigidly that signature-based algorithms always produce lesser number of S-polynomials.

Key Arguments for Counting Necessary S-Polynomials

- Signature-Based Algorithm: Signature-based algorithms, such as F_5 or its variants, are designed to discard unnecessary S-polynomials as much as possible, by using the notion signature. Since reduction operations are restricted to keep the signature, it is not proven rigidly that signature-based algorithms always produce lesser number of S-polynomials.
- Other Efficient Methods:

Key Arguments for Counting Necessary S-Polynomials

- Signature-Based Algorithm: Signature-based algorithms, such as F_5 or its variants, are designed to discard unnecessary S-polynomials as much as possible, by using the notion signature. Since reduction operations are restricted to keep the signature, it is not proven rigidly that signature-based algorithms always produce lesser number of S-polynomials.
- Other Efficient Methods: However, due to the special structure of the ideal, we can see that other algorithms with normal strategy have similar computational behavior.

Key Arguments for Counting Necessary S-Polynomials

- Signature-Based Algorithm: Signature-based algorithms, such as F_5 or its variants, are designed to discard unnecessary S-polynomials as much as possible, by using the notion signature. Since reduction operations are restricted to keep the signature, it is not proven rigidly that signature-based algorithms always produce lesser number of S-polynomials.
- Other Efficient Methods: However, due to the special structure of the ideal, we can see that other algorithms with normal strategy have similar computational behavior. They compute S-polynomials almost in ascending order in signature.

Key Arguments for Counting Necessary S-Polynomials

- Signature-Based Algorithm: Signature-based algorithms, such as F_5 or its variants, are designed to discard unnecessary S-polynomials as much as possible, by using the notion signature. Since reduction operations are restricted to keep the signature, it is not proven rigidly that signature-based algorithms always produce lesser number of S-polynomials.
- Other Efficient Methods: However, due to the special structure of the ideal, we can see that other algorithms with normal strategy have similar computational behavior. They compute S-polynomials almost in ascending order in signature.
- Our Case:

Key Arguments for Counting Necessary S-Polynomials

- Signature-Based Algorithm: Signature-based algorithms, such as F_5 or its variants, are designed to discard unnecessary S-polynomials as much as possible, by using the notion signature. Since reduction operations are restricted to keep the signature, it is not proven rigidly that signature-based algorithms always produce lesser number of S-polynomials.
- Other Efficient Methods: However, due to the special structure of the ideal, we can see that other algorithms with normal strategy have similar computational behavior. They compute S-polynomials almost in ascending order in signature.
- Our Case: As the number of possible signatures for S-polynomials is finite,

Key Arguments for Counting Necessary S-Polynomials

- Signature-Based Algorithm: Signature-based algorithms, such as F_5 or its variants, are designed to discard unnecessary S-polynomials as much as possible, by using the notion signature. Since reduction operations are restricted to keep the signature, it is not proven rigidly that signature-based algorithms always produce lesser number of S-polynomials.
- Other Efficient Methods: However, due to the special structure of the ideal, we can see that other algorithms with normal strategy have similar computational behavior. They compute S-polynomials almost in ascending order in signature.
- Our Case: As the number of possible signatures for S-polynomials is finite, with help of linear algebra related to Subresultant Theory,

Key Arguments for Counting Necessary S-Polynomials

- Signature-Based Algorithm: Signature-based algorithms, such as F_5 or its variants, are designed to discard unnecessary S-polynomials as much as possible, by using the notion signature. Since reduction operations are restricted to keep the signature, it is not proven rigidly that signature-based algorithms always produce lesser number of S-polynomials.
- Other Efficient Methods: However, due to the special structure of the ideal, we can see that other algorithms with normal strategy have similar computational behavior. They compute S-polynomials almost in ascending order in signature.
- Our Case: As the number of possible signatures for S-polynomials is finite, with help of linear algebra related to Subresultant Theory, we can construct an approximate set of signatures, for which necessary S-polynomials appear.

Key Arguments for Counting Necessary S-Polynomials

- Signature-Based Algorithm: Signature-based algorithms, such as F_5 or its variants, are designed to discard unnecessary S-polynomials as much as possible, by using the notion signature. Since reduction operations are restricted to keep the signature, it is not proven rigidly that signature-based algorithms always produce lesser number of S-polynomials.
- Other Efficient Methods: However, due to the special structure of the ideal, we can see that other algorithms with normal strategy have similar computational behavior. They compute S-polynomials almost in ascending order in signature.
- Our Case: As the number of possible signatures for S-polynomials is finite, with help of linear algebra related to Subresultant Theory, we can construct an approximate set of signatures, for which necessary S-polynomials appear. $\Leftrightarrow$ Our Assumption

Key Arguments for Counting Necessary S-Polynomials

- Signature-Based Algorithm: Signature-based algorithms, such as F_5 or its variants, are designed to discard unnecessary S-polynomials as much as possible, by using the notion signature. Since reduction operations are restricted to keep the signature, it is not proven rigidly that signature-based algorithms always produce lesser number of S-polynomials.
- Other Efficient Methods: However, due to the special structure of the ideal, we can see that other algorithms with normal strategy have similar computational behavior. They compute S-polynomials almost in ascending order in signature.
- Our Case: As the number of possible signatures for S-polynomials is finite, with help of linear algebra related to Subresultant Theory, we can construct an approximate set of signatures, for which necessary S-polynomials appear. $\Leftrightarrow$ Our Assumption

Our approximation is roughly examined by computational experience.

Computational Experiments

Computational Experiments

We set finite prime fields $\mathbb{F}_q$ with $q = 2^B + c, (2^B \gg c)$ such that $\mathbb{F}_p^\times$ has a subgroup of order d. (Thus $F(x) = x^d - 1$ and $\delta = \deg_{x_i} S_{m+1} = 2^{m-2}$.)

Computational Experiments

We set finite prime fields $\mathbb{F}_q$ with $q = 2^B + c, (2^B \gg c)$ such that $\mathbb{F}_p^\times$ has a subgroup of order d. (Thus $F(x) = x^d - 1$ and $\delta = \deg_{x_i} S_{m+1} = 2^{m-2}$.) We also generated elliptic curves with prime order over $\mathbb{F}_q$ and considered the PDP for randomly generated points $aP + bQ$.

Computational Experiments

We set finite prime fields $\mathbb{F}_q$ with $q = 2^B + c, (2^B \gg c)$ such that $\mathbb{F}_p^\times$ has a subgroup of order d. (Thus $F(x) = x^d - 1$ and $\delta = \deg_{x_i} S_{m+1} = 2^{m-2}$.) We also generated elliptic curves with prime order over $\mathbb{F}_q$ and considered the PDP for randomly generated points $aP + bQ$. We computed 5 examples for each parameter by Risa/Asir CA system (without using signature).

Computational Experiments

We set finite prime fields $\mathbb{F}_q$ with $q = 2^B + c, (2^B \gg c)$ such that $\mathbb{F}_p^\times$ has a subgroup of order d. (Thus $F(x) = x^d - 1$ and $\delta = \deg_{x_i} S_{m+1} = 2^{m-2}$.) We also generated elliptic curves with prime order over $\mathbb{F}_q$ and considered the PDP for randomly generated points $aP + bQ$. We computed 5 examples for each parameter by Risa/Asir CA system (without using signature).

Computational Experiments

We set finite prime fields $\mathbb{F}_q$ with $q = 2^B + c, (2^B \gg c)$ such that $\mathbb{F}_p^\times$ has a subgroup of order d. (Thus $F(x) = x^d - 1$ and $\delta = \deg_{x_i} S_{m+1} = 2^{m-2}$.) We also generated elliptic curves with prime order over $\mathbb{F}_q$ and considered the PDP for randomly generated points $aP + bQ$. We computed 5 examples for each parameter by Risa/Asir CA system (without using signature).

- $\mathcal{G}^*$: the computed Gröbner basis, that is, the set of non-zero elements obtained from S-polynomials.

Computational Experiments

We set finite prime fields $\mathbb{F}_q$ with $q = 2^B + c, (2^B \gg c)$ such that $\mathbb{F}_p^\times$ has a subgroup of order d. (Thus $F(x) = x^d - 1$ and $\delta = \deg_{x_i} S_{m+1} = 2^{m-2}$.) We also generated elliptic curves with prime order over $\mathbb{F}_q$ and considered the PDP for randomly generated points $aP + bQ$. We computed 5 examples for each parameter by Risa/Asir CA system (without using signature).

- $\mathcal{G}^*$: the computed Gröbner basis, that is, the set of non-zero elements obtained from S-polynomials.
- $NS = \{x_1^{e_1} \cdots x_m^{e_m} \mid 0 \leq e_i < d\}$: the set of possible signatures for which S-polynomials appear,

Computational Experiments

We set finite prime fields $\mathbb{F}_q$ with $q = 2^B + c, (2^B \gg c)$ such that $\mathbb{F}_p^\times$ has a subgroup of order d. (Thus $F(x) = x^d - 1$ and $\delta = \deg_{x_i} S_{m+1} = 2^{m-2}$.) We also generated elliptic curves with prime order over $\mathbb{F}_q$ and considered the PDP for randomly generated points $aP + bQ$. We computed 5 examples for each parameter by Risa/Asir CA system (without using signature).

- $\mathcal{G}^*$: the computed Gröbner basis, that is, the set of non-zero elements obtained from S-polynomials.
- $NS = \{x_1^{e_1} \cdots x_m^{e_m} \mid 0 \leq e_i < d\}$: the set of possible signatures for which S-polynomials appear,
- $\overline{NS} = NS \setminus \{x_1^{e_1} \cdots x_m^{e_m} \mid 0 \leq e_i < \delta\} \cup \{1\}$: Refined set of possible signatures.

Computational Experiments

We set finite prime fields $\mathbb{F}_q$ with $q = 2^B + c, (2^B \gg c)$ such that $\mathbb{F}_p^\times$ has a subgroup of order d. (Thus $F(x) = x^d - 1$ and $\delta = \deg_{x_i} S_{m+1} = 2^{m-2}$.) We also generated elliptic curves with prime order over $\mathbb{F}_q$ and considered the PDP for randomly generated points $aP + bQ$. We computed 5 examples for each parameter by Risa/Asir CA system (without using signature).

- $\mathcal{G}^*$: the computed Gröbner basis, that is, the set of non-zero elements obtained from S-polynomials.
- $NS = \{x_1^{e_1} \cdots x_m^{e_m} \mid 0 \leq e_i < d\}$: the set of possible signatures for which S-polynomials appear,
- $\overline{NS} = NS \setminus \{x_1^{e_1} \cdots x_m^{e_m} \mid 0 \leq e_i < \delta\} \cup \{1\}$: Refined set of possible signatures. $\Leftarrow$ Our Assumption

Computational Experiments

We set finite prime fields $\mathbb{F}_q$ with $q = 2^B + c, (2^B \gg c)$ such that $\mathbb{F}_p^\times$ has a subgroup of order d. (Thus $F(x) = x^d - 1$ and $\delta = \deg_{x_i} S_{m+1} = 2^{m-2}$.) We also generated elliptic curves with prime order over $\mathbb{F}_q$ and considered the PDP for randomly generated points $aP + bQ$. We computed 5 examples for each parameter by Risa/Asir CA system (without using signature).

- $\mathcal{G}^*$: the computed Gröbner basis, that is, the set of non-zero elements obtained from S-polynomials.
- $NS = \{x_1^{e_1} \cdots x_m^{e_m} \mid 0 \leq e_i < d\}$: the set of possible signatures for which S-polynomials appear,
- $\overline{NS} = NS \setminus \{x_1^{e_1} \cdots x_m^{e_m} \mid 0 \leq e_i < \delta\} \cup \{1\}$: Refined set of possible signatures. $\Leftarrow$ Our Assumption
- $[**]$ denotes the average of $**$, and $Ratio$ denotes the ratio of $[\#\mathcal{G}^*]$ and $\#\overline{NS}$, PA denotes the total number of multiplications of some polynomials and monomials.

Computational Experiments:$m = 4$ and $\delta = 8$

Computational Experiments:$m = 4$ and $\delta = 8$

Computational Experiments:$m = 4$ and $\delta = 8$

Trivial-Ideal Case:

B	10	11	12	13	14	15
d	5	7	8	8	11	12
d^m	625	2401	4096	4096	14641	20736
$\#NS$	625	2401	4096	4096	14641	20736
$\#\overline{NS}$	625	2401	4096	4096	14560	20480
$[\#\mathcal{G}^*]$	625	2401	4096	4096	14301	19770
$Ratio$	1	1	1	1	0.98	0.97
$[PA]$	122162	1175808	2902403	2902748	26852120.2	46097906.4

Computational Experiments:$m = 4$ and $\delta = 8$

Trivial-Ideal Case:

B	10	11	12	13	14	15
d	5	7	8	8	11	12
d^m	625	2401	4096	4096	14641	20736
$\#NS$	625	2401	4096	4096	14641	20736
$\#\overline{NS}$	625	2401	4096	4096	14560	20480
$[\#\mathcal{G}^*]$	625	2401	4096	4096	14301	19770
$Ratio$	1	1	1	1	0.98	0.97
$[PA]$	122162	1175808	2902403	2902748	26852120.2	46097906.4

Non-Trivial-Ideal Case: $V(I)$ denotes the number of zeros.

B	12	13	15
d	5	6	7
d^m	625	1296	2401
$[\#NS]$	613.6	1287.8	2382.8
$[\#\overline{NS}]$	613.6	1287.8	2382.8
$[\#\mathcal{G}^*]$	613.6	1287.8	2382.8
$Ratio$	1	1	1
$[\#V(I)]$	11.4	8.2	18.2
$[PA]$	121820	408965	1173939

Computational Experiments:$m = 3$ and $\delta = 4$

Computational Experiments:$m = 3$ and $\delta = 4$

We set primes q as $2^{15} + c, (2^15 \gg c)$ and consider the problem over $\mathbb{F}_q$. $Ratio2$ denotes the ratio of $[\#\mathcal{G}^*]$ and md^{m-1}.

Computational Experiments:$m = 3$ and $\delta = 4$

We set primes q as $2^{15} + c, (2^15 \gg c)$ and consider the problem over $\mathbb{F}_q$. $Ratio2$ denotes the ratio of $[\#\mathcal{G}^*]$ and md^{m-1}.

Trivial-Ideal Case:

d	12	20	30
$\#NS$	1728	8000	27000
$\#\overline{NS}$	1216	3904	9424
md^{m-1}	432	1200	2700
$[\#\mathcal{G}^*]$	947	2599	5744
$Ratio$	0.78	0.67	0.61
$Ratio1$	2.19	2.17	2.13
$[PA]$	147488	862113.6	3472845.4

Computational Experiments:$m = 3$ and $\delta = 4$

We set primes q as $2^{15} + c, (2^15 \gg c)$ and consider the problem over $\mathbb{F}_q$. $Ratio2$ denotes the ratio of $[\#\mathcal{G}^*]$ and md^{m-1}.

Trivial-Ideal Case:

d	12	20	30
$\#NS$	1728	8000	27000
$\#\overline{NS}$	1216	3904	9424
md^{m-1}	432	1200	2700
$[\#\mathcal{G}^*]$	947	2599	5744
$Ratio$	0.78	0.67	0.61
$Ratio1$	2.19	2.17	2.13
$[PA]$	147488	862113.6	3472845.4

Non-Trivial Ideal Case:

d	12	20	30
$[\#NS]$	1722.6	7992.8	26995.2
$[\#\overline{NS}]$	1211.6	3897.8	9420.8
md^{m-1}	432	1200	2700
$[\#\mathcal{G}^*]$	941.6	2591.8	5739.8
$Ratio$	0.78	0.66	0.61
$Ratio1$	2.17	2.16	2.13
$[\#V(I)]$	5.4	7.2	4.2
$[PA]$	147417.8	861979	3472685.4

Computational Experiments: $m = 3$ and $\delta = 4$

We set primes q as $2^{15} + c, (2^15 \gg c)$ and consider the problem over $\mathbb{F}_q$. $Ratio2$ denotes the ratio of $[\#\mathcal{G}^*]$ and md^{m-1}.

Trivial-Ideal Case:

d	12	20	30
$\#NS$	1728	8000	27000
$\#\overline{NS}$	1216	3904	9424
md^{m-1}	432	1200	2700
$[\#\mathcal{G}^*]$	947	2599	5744
$Ratio$	0.78	0.67	0.61
$Ratio1$	2.19	2.17	2.13
$[PA]$	147488	862113.6	3472845.4

Non-Trivial Ideal Case:

d	12	20	30
$[\#NS]$	1722.6	7992.8	26995.2
$[\#\overline{NS}]$	1211.6	3897.8	9420.8
md^{m-1}	432	1200	2700
$[\#\mathcal{G}^*]$	941.6	2591.8	5739.8
$Ratio$	0.78	0.66	0.61
$Ratio1$	2.17	2.16	2.13
$[\#V(I)]$	5.4	7.2	4.2
$[PA]$	147417.8	861979	3472685.4

$$d^m - (d-\delta)^m + 1 > \#\mathcal{G}^* > md^{m-1}$$

Euclidean Algorithm and Gröbner Basis Computation

Euclidean Algorithm and Gröbner Basis Computation

As the simplest case, we consider $m = 1$ and $\mathcal{F} = \{f = S(x_1), g = F(x_1)\}$.

Euclidean Algorithm and Gröbner Basis Computation

As the simplest case, we consider $m = 1$ and $\mathcal{F} = \{f = S(x_1), g = F(x_1)\}$.

Euclidean Algorithm and Gröbner Basis Computation

As the simplest case, we consider $m = 1$ and $\mathcal{F} = \{f = S(x_1), g = F(x_1)\}$.

Let $f_0 = f$, $f_1 = g$ and suppose $\deg(f) = d > \deg(g) = d - 1$ and the polynomial remainder sequence by Euclidean algorithm is regular:

Euclidean Algorithm and Gröbner Basis Computation

As the simplest case, we consider $m = 1$ and $\mathcal{F} = \{f = S(x_1), g = F(x_1)\}$.

Let $f_0 = f$, $f_1 = g$ and suppose $\deg(f) = d > \deg(g) = d - 1$ and the polynomial remainder sequence by Euclidean algorithm is regular:

$$
\begin{aligned}
f_2 &= f_0 - (a_1 x + b_1) f_1 \\
f_3 &= f_1 - (a_2 x + b_2) f_2 \\
&\vdots \\
f_k &= f_{k-2} - (a_{k-1} x + b_{k-1}) f_{k-1}
\end{aligned}
$$

Euclidean Algorithm and Gröbner Basis Computation

As the simplest case, we consider $m = 1$ and $\mathcal{F} = \{f = S(x_1), g = F(x_1)\}$.

Let $f_0 = f$, $f_1 = g$ and suppose $\deg(f) = d > \deg(g) = d - 1$ and the polynomial remainder sequence by Euclidean algorithm is regular:

$$
\begin{array}{rcl@{\qquad}rcl}
f_2 & = & f_0 - (a_1 x + b_1) f_1 & \mathrm{Spol}(f_0, f_1) & = & f_0 - a_1 x f_1 \xrightarrow{f_1} f_2 \\
f_3 & = & f_1 - (a_2 x + b_2) f_2 & \mathrm{Spol}(f_1, f_2) & = & f_1 - a_2 x f_2 \xrightarrow{f_2} f_3 \\
 & \vdots & & & \vdots & \\
f_k & = & f_{k-2} - (a_{k-1} x + b_{k-1}) f_{k-1} & \mathrm{Spol}(f_{k-2}, f_{k-1}) & = & f_{k-2} - a_{k-1} x f_{k-1} \xrightarrow{f_{k-1}} f_k
\end{array}
$$

Euclidean Algorithm and Gröbner Basis Computation

As the simplest case, we consider $m = 1$ and $\mathcal{F} = \{f = S(x_1), g = F(x_1)\}$.

Let $f_0 = f$, $f_1 = g$ and suppose $\deg(f) = d > \deg(g) = d - 1$ and the polynomial remainder sequence by Euclidean algorithm is regular:

$$
\begin{array}{rcl@{\qquad}rcl}
f_2 & = & f_0 - (a_1 x + b_1) f_1 & \mathrm{Spol}(f_0, f_1) & = & f_0 - a_1 x f_1 \xrightarrow{f_1} f_2 \\
f_3 & = & f_1 - (a_2 x + b_2) f_2 & \mathrm{Spol}(f_1, f_2) & = & f_1 - a_2 x f_2 \xrightarrow{f_2} f_3 \\
 & \vdots & & & \vdots & \\
f_k & = & f_{k-2} - (a_{k-1} x + b_{k-1}) f_{k-1} & \mathrm{Spol}(f_{k-2}, f_{k-1}) & = & f_{k-2} - a_{k-1} x f_{k-1} \xrightarrow{f_{k-1}} f_k
\end{array}
$$

$\mathrm{Spol}(f_i, f_j)$ denotes the S-polynomial of f_i, f_j and $A \xrightarrow{C} B$ implies that A is reduced to B by C.

Euclidean Algorithm and Gröbner Basis Computation

As the simplest case, we consider $m = 1$ and $\mathcal{F} = \{f = S(x_1), g = F(x_1)\}$.

Let $f_0 = f$, $f_1 = g$ and suppose $\deg(f) = d > \deg(g) = d - 1$ and the polynomial remainder sequence by Euclidean algorithm is regular:

$$
\begin{array}{rcl@{\qquad}rcl}
f_2 & = & f_0 - (a_1 x + b_1) f_1 & \mathrm{Spol}(f_0, f_1) & = & f_0 - a_1 x f_1 \xrightarrow{f_1} f_2 \\
f_3 & = & f_1 - (a_2 x + b_2) f_2 & \mathrm{Spol}(f_1, f_2) & = & f_1 - a_2 x f_2 \xrightarrow{f_2} f_3 \\
 & \vdots & & & \vdots & \\
f_k & = & f_{k-2} - (a_{k-1} x + b_{k-1}) f_{k-1} & \mathrm{Spol}(f_{k-2}, f_{k-1}) & = & f_{k-2} - a_{k-1} x f_{k-1} \xrightarrow{f_{k-1}} f_k
\end{array}
$$

$\mathrm{Spol}(f_i, f_j)$ denotes the S-polynomial of f_i, f_j and $A \xrightarrow{C} B$ implies that A is reduced to B by C.

In Buchbeger algorithm, we have to compute the S-polynomial for every distinct pair (f_i, f_j).

Euclidean Algorithm and Gröbner Basis Computation

As the simplest case, we consider $m = 1$ and $\mathcal{F} = \{f = S(x_1), g = F(x_1)\}$.

Let $f_0 = f$, $f_1 = g$ and suppose $\deg(f) = d > \deg(g) = d - 1$ and the polynomial remainder sequence by Euclidean algorithm is regular:

$$
\begin{array}{rcl@{\qquad}rcl}
f_2 & = & f_0 - (a_1 x + b_1) f_1 & \mathrm{Spol}(f_0, f_1) & = & f_0 - a_1 x f_1 \xrightarrow{f_1} f_2 \\
f_3 & = & f_1 - (a_2 x + b_2) f_2 & \mathrm{Spol}(f_1, f_2) & = & f_1 - a_2 x f_2 \xrightarrow{f_2} f_3 \\
 & \vdots & & & \vdots & \\
f_k & = & f_{k-2} - (a_{k-1} x + b_{k-1}) f_{k-1} & \mathrm{Spol}(f_{k-2}, f_{k-1}) & = & f_{k-2} - a_{k-1} x f_{k-1} \xrightarrow{f_{k-1}} f_k
\end{array}
$$

$\mathrm{Spol}(f_i, f_j)$ denotes the S-polynomial of f_i, f_j and $A \xrightarrow{C} B$ implies that A is reduced to B by C.

In Buchbeger algorithm, we have to compute the S-polynomial for every distinct pair (f_i, f_j). Introducing signature, we may discard unnecessary S-polynomials.

Regular PRS as the Worst Complexity Case

Regular PRS as the Worst Complexity Case

Regular PRS as the Worst Complexity Case

By Extended Euclidean Algorithm, for each f_i, there are polynomials A_i, B_i such that

Regular PRS as the Worst Complexity Case

By Extended Euclidean Algorithm, for each f_i, there are polynomials A_i, B_i such that

$$\begin{aligned} f_i &= A_i f + B_i g, \\ \deg(A_i) &< \deg(g) - \deg(f_i) = d - 1 - \deg(f_i), \\ \deg(B_i) &< \deg(f) - \deg(f_i) = d - \deg(f_i). \end{aligned}$$

Regular PRS as the Worst Complexity Case

By Extended Euclidean Algorithm, for each f_i, there are polynomials A_i, B_i such that

$$\begin{aligned} f_i &= A_i f + B_i g, \\ \deg(A_i) &< \deg(g) - \deg(f_i) = d - 1 - \deg(f_i), \\ \deg(B_i) &< \deg(f) - \deg(f_i) = d - \deg(f_i). \end{aligned}$$

As the PRS is regular,

Regular PRS as the Worst Complexity Case

By Extended Euclidean Algorithm, for each f_i, there are polynomials A_i, B_i such that

$$\begin{aligned} f_i &= A_i f + B_i g, \\ \deg(A_i) &< \deg(g) - \deg(f_i) = d - 1 - \deg(f_i), \\ \deg(B_i) &< \deg(f) - \deg(f_i) = d - \deg(f_i). \end{aligned}$$

As the PRS is regular, $\deg(f_i) = d - i$ and $\deg(A_i) = d - (d - i) - 2 = i - 2$.

Regular PRS as the Worst Complexity Case

By Extended Euclidean Algorithm, for each f_i, there are polynomials A_i, B_i such that

$$\begin{aligned} f_i &= A_i f + B_i g, \\ \deg(A_i) &< \deg(g) - \deg(f_i) = d - 1 - \deg(f_i), \\ \deg(B_i) &< \deg(f) - \deg(f_i) = d - \deg(f_i). \end{aligned}$$

As the PRS is regular, $\deg(f_i) = d - i$ and $\deg(A_i) = d - (d - i) - 2 = i - 2$.

By Subresultant Theory, PRS is regular if and only if its subresultants are non-zero.

Regular PRS as the Worst Complexity Case

By Extended Euclidean Algorithm, for each f_i, there are polynomials A_i, B_i such that

$$\begin{aligned} f_i &= A_i f + B_i g, \\ \deg(A_i) &< \deg(g) - \deg(f_i) = d - 1 - \deg(f_i), \\ \deg(B_i) &< \deg(f) - \deg(f_i) = d - \deg(f_i). \end{aligned}$$

As the PRS is regular, $\deg(f_i) = d - i$ and $\deg(A_i) = d - (d - i) - 2 = i - 2$.

By Subresultant Theory, PRS is regular if and only if its subresultants are non-zero.

$$\Downarrow$$

If f is given as $f = f(x, \beta)$, the condition is an algebraic condition on β.

Signature and S-polynomial in GCD

Signature and S-polynomial in GCD

Signature in GCD

Signature and S-polynomial in GCD

Signature in GCD

Each element h of the ideal $\langle f, g \rangle$ can be expressed as $h = Af + Bg$.

Signature and S-polynomial in GCD

Signature in GCD

Each element h of the ideal $\langle f, g \rangle$ can be expressed as $h = Af + Bg$. Among such expressions, we define the signature of h by x^m for the smallest degree $m = \deg(A)$, and denote it by $\mathrm{sig}(h)$.

Signature and S-polynomial in GCD

Signature in GCD

Each element h of the ideal $\langle f, g \rangle$ can be expressed as $h = Af + Bg$. Among such expressions, we define the signature of h by x^m for the smallest degree $m = \deg(A)$, and denote it by $\mathrm{sig}(h)$.

For each f_i in the PRS, A_i is a polynomial with the smallest degree and $\mathrm{sig}(f_i) = x^{i-2}$.

Signature and S-polynomial in GCD

Signature in GCD

Each element h of the ideal $\langle f, g \rangle$ can be expressed as $h = Af + Bg$. Among such expressions, we define the signature of h by x^m for the smallest degree $m = \deg(A)$, and denote it by $\mathrm{sig}(h)$.

For each f_i in the PRS, A_i is a polynomial with the smallest degree and $\mathrm{sig}(f_i) = x^{i-2}$.
Then, the signature of the S-polynomial $\mathrm{Spol}(f_i, f_{i+1})$ coincides with that of $a_{i+1} x f_{i+1}$ and hence,

Signature and S-polynomial in GCD

Signature in GCD

Each element h of the ideal $\langle f, g \rangle$ can be expressed as $h = Af + Bg$. Among such expressions, we define the signature of h by x^m for the smallest degree $m = \deg(A)$, and denote it by $\mathrm{sig}(h)$.

For each f_i in the PRS, A_i is a polynomial with the smallest degree and $\mathrm{sig}(f_i) = x^{i-2}$.
Then, the signature of the S-polynomial $\mathrm{Spol}(f_i, f_{i+1})$ coincides with that of $a_{i+1} x f_{i+1}$ and hence,

$$\mathrm{sig}(f_{i+2}) = x \times \mathrm{sig}(f_{i+1}).$$

Signature and S-polynomial in GCD

Signature in GCD

Each element h of the ideal $\langle f, g \rangle$ can be expressed as $h = Af + Bg$. Among such expressions, we define the signature of h by x^m for the smallest degree $m = \deg(A)$, and denote it by $\mathrm{sig}(h)$.

For each f_i in the PRS, A_i is a polynomial with the smallest degree and $\mathrm{sig}(f_i) = x^{i-2}$.
Then, the signature of the S-polynomial $\mathrm{Spol}(f_i, f_{i+1})$ coincides with that of $a_{i+1} x f_{i+1}$ and hence,

$$\mathrm{sig}(f_{i+2}) = x \times \mathrm{sig}(f_{i+1}).$$

In signature-based algorithms, S-polynomials are dealt with in ascending order, and for each signature, at most one S-polynomial is computed.

Signature and S-polynomial in GCD

Signature in GCD

Each element h of the ideal $\langle f, g \rangle$ can be expressed as $h = Af + Bg$. Among such expressions, we define the signature of h by x^m for the smallest degree $m = \deg(A)$, and denote it by $\mathrm{sig}(h)$.

For each f_i in the PRS, A_i is a polynomial with the smallest degree and $\mathrm{sig}(f_i) = x^{i-2}$.
Then, the signature of the S-polynomial $\mathrm{Spol}(f_i, f_{i+1})$ coincides with that of $a_{i+1} x f_{i+1}$ and hence,

$$\mathrm{sig}(f_{i+2}) = x \times \mathrm{sig}(f_{i+1}).$$

In signature-based algorithms, S-polynomials are dealt with in ascending order, and for each signature, at most one S-polynomial is computed. Thus, we have only to compute $\mathrm{Spol}(f_i, f_{i+1})$ for the signature x^{i-1}.

Signature and S-polynomial in GCD

Signature in GCD

Each element h of the ideal $\langle f, g \rangle$ can be expressed as $h = Af + Bg$. Among such expressions, we define the signature of h by x^m for the smallest degree $m = \deg(A)$, and denote it by $\mathrm{sig}(h)$.

For each f_i in the PRS, A_i is a polynomial with the smallest degree and $\mathrm{sig}(f_i) = x^{i-2}$.
Then, the signature of the S-polynomial $\mathrm{Spol}(f_i, f_{i+1})$ coincides with that of $a_{i+1} x f_{i+1}$ and hence,

$$\mathrm{sig}(f_{i+2}) = x \times \mathrm{sig}(f_{i+1}).$$

In signature-based algorithms, S-polynomials are dealt with in ascending order, and for each signature, at most one S-polynomial is computed. Thus, we have only to compute $\mathrm{Spol}(f_i, f_{i+1})$ for the signature x^{i-1}.

Signature is a useful tool for analysing S-polynomials.

Definition of Signature for I

Definition of Signature for I

Definition of Signature for I

- Consider the ideal quotient of $J = \langle F(x_1), \ldots, F(x_m) \rangle$ by S and call it the Syzygy ideal;

Definition of Signature for I

- Consider the ideal quotient of $J = \langle F(x_1), \ldots, F(x_m) \rangle$ by S and call it the Syzygy ideal;

$$(J : S) = \{f \in R \mid fS \in J\}.$$

Definition of Signature for I

- Consider the ideal quotient of $J = \langle F(x_1), \ldots, F(x_m) \rangle$ by S and call it the Syzygy ideal;

$$(J : S) = \{f \in R \mid fS \in J\}.$$

- Let its Gröbner basis be $\overline{\mathcal{G}}_0$, and NS the set of monomials reduced with respect to $\overline{\mathcal{G}}_0$. Here $LM(g)$ denotes the leading monomial of g.

$$NS = \{t \in \mathcal{M} \mid LM(g) \nmid t \text{ for any } g \in \tilde{\mathcal{G}}_0\}.$$

Definition of Signature for I

- Consider the ideal quotient of $J = \langle F(x_1), \ldots, F(x_m) \rangle$ by S and call it the Syzygy ideal;

$$(J : S) = \{f \in R \mid fS \in J\}.$$

- Let its Gröbner basis be $\overline{\mathcal{G}}_0$, and NS the set of monomials reduced with respect to $\overline{\mathcal{G}}_0$. Here $LM(g)$ denotes the leading monomial of g.

$$NS = \{t \in \mathcal{M} \mid LM(g) \nmid t \text{ for any } g \in \tilde{\mathcal{G}}_0\}.$$

- For each $f \in I \setminus J$, there are $T, A_i, \ldots, A_m$ such that T consists of monomials in NS and

Definition of Signature for I

- Consider the ideal quotient of $J = \langle F(x_1), \ldots, F(x_m) \rangle$ by S and call it the Syzygy ideal;

$$(J : S) = \{f \in R \mid fS \in J\}.$$

- Let its Gröbner basis be $\overline{\mathcal{G}}_0$, and NS the set of monomials reduced with respect to $\overline{\mathcal{G}}_0$. Here $LM(g)$ denotes the leading monomial of g.

$$NS = \{t \in \mathcal{M} \mid LM(g) \nmid t \text{ for any } g \in \tilde{\mathcal{G}}_0\}.$$

- For each $f \in I \setminus J$, there are $T, A_i, \ldots, A_m$ such that T consists of monomials in NS and

$$f = TS + A_1 F(x_1) + \cdots + A_m F(x_m).$$

Definition of Signature for I

- Consider the ideal quotient of $J = \langle F(x_1), \ldots, F(x_m) \rangle$ by S and call it the Syzygy ideal;
$$(J : S) = \{f \in R \mid fS \in J\}.$$
- Let its Gröbner basis be $\overline{\mathcal{G}}_0$, and NS the set of monomials reduced with respect to $\overline{\mathcal{G}}_0$. Here $LM(g)$ denotes the leading monomial of g.
$$NS = \{t \in \mathcal{M} \mid LM(g) \nmid t \text{ for any } g \in \tilde{\mathcal{G}}_0\}.$$
- For each $f \in I \setminus J$, there are $T, A_i, \ldots, A_m$ such that T consists of monomials in NS and
$$f = TS + A_1 F(x_1) + \cdots + A_m F(x_m).$$
- Then T is determined uniquely. We call the leading monomial $LM(T)$ the signature of f, and denote it by $\mathrm{sig}(f)$.

Definition of Signature for I

- Consider the ideal quotient of $J = \langle F(x_1), \ldots, F(x_m) \rangle$ by S and call it the Syzygy ideal;

$$(J : S) = \{f \in R \mid fS \in J\}.$$

- Let its Gröbner basis be $\overline{\mathcal{G}}_0$, and NS the set of monomials reduced with respect to $\overline{\mathcal{G}}_0$. Here $LM(g)$ denotes the leading monomial of g.

$$NS = \{t \in \mathcal{M} \mid LM(g) \nmid t \text{ for any } g \in \tilde{\mathcal{G}}_0\}.$$

- For each $f \in I \setminus J$, there are $T, A_i, \ldots, A_m$ such that T consists of monomials in NS and

$$f = TS + A_1 F(x_1) + \cdots + A_m F(x_m).$$

- Then T is determined uniquely. We call the leading monomial $LM(T)$ the signature of f, and denote it by $\mathrm{sig}(f)$. For each $f \in J$, its signature is defined by 0.

Fundamentals in Signature

Fundamentals in Signature

$\mathfrak{S}$-reduction

$\mathfrak{S}$-reduction

Let $f, g, h \in I$. We say that f is $\mathfrak{S}$-reduced to g by h, if there are $t \in \mathcal{M}$ and $a \in K \setminus \{0\}$ such that $g = f - a \cdot t \cdot h$, $\mathrm{sig}(th) \prec \mathrm{sig}(f)$ and $LM(g) \prec LM(f)$.

$\mathfrak{S}$-reduction

Let $f, g, h \in I$. We say that f is $\mathfrak{S}$-reduced to g by h, if there are $t \in \mathcal{M}$ and $a \in K \setminus \{0\}$ such that $g = f - a \cdot t \cdot h$, $\mathrm{sig}(th) \prec \mathrm{sig}(f)$ and $LM(g) \prec LM(f)$. We also call h (or $t \cdot h$) a $\mathfrak{S}$-reducer of f. This reduction don't change the signature.

$\mathfrak{S}$-reduction

Let $f, g, h \in I$. We say that f is $\mathfrak{S}$-reduced to g by h, if there are $t \in \mathcal{M}$ and $a \in K \setminus \{0\}$ such that $g = f - a \cdot t \cdot h$, $\mathrm{sig}(th) \prec \mathrm{sig}(f)$ and $LM(g) \prec LM(f)$. We also call h (or $t \cdot h$) a $\mathfrak{S}$-reducer of f. This reduction don't change the signature. If $f \in I$ has no $\mathfrak{S}$-reducer, we say that it is $\mathfrak{S}$-irreducible.

Fundamentals in Signature

$\mathfrak{S}$-reduction

Let $f, g, h \in I$. We say that f is $\mathfrak{S}$-reduced to g by h, if there are $t \in \mathcal{M}$ and $a \in K \setminus \{0\}$ such that $g = f - a \cdot t \cdot h$, $\mathrm{sig}(th) \prec \mathrm{sig}(f)$ and $LM(g) \prec LM(f)$. We also call h (or $t \cdot h$) a $\mathfrak{S}$-reducer of f. This reduction don't change the signature. If $f \in I$ has no $\mathfrak{S}$-reducer, we say that it is $\mathfrak{S}$-irreducible.

Property of $\mathfrak{S}$-Irreducible Element

Fundamentals in Signature

$\mathfrak{S}$-reduction

Let $f, g, h \in I$. We say that f is $\mathfrak{S}$-reduced to g by h, if there are $t \in \mathcal{M}$ and $a \in K \setminus \{0\}$ such that $g = f - a \cdot t \cdot h$, $\mathrm{sig}(th) \prec \mathrm{sig}(f)$ and $LM(g) \prec LM(f)$. We also call h (or $t \cdot h$) a $\mathfrak{S}$-reducer of f. This reduction don't change the signature. If $f \in I$ has no $\mathfrak{S}$-reducer, we say that it is $\mathfrak{S}$-irreducible.

Property of $\mathfrak{S}$-Irreducible Element

For each $s \in NS$, there is an element of I with signature s.

Fundamentals in Signature

$\mathfrak{S}$-reduction

Let $f, g, h \in I$. We say that f is $\mathfrak{S}$-reduced to g by h, if there are $t \in \mathcal{M}$ and $a \in K \setminus \{0\}$ such that $g = f - a \cdot t \cdot h$, $\mathrm{sig}(th) \prec \mathrm{sig}(f)$ and $LM(g) \prec LM(f)$. We also call h (or $t \cdot h$) a $\mathfrak{S}$-reducer of f. This reduction don't change the signature. If $f \in I$ has no $\mathfrak{S}$-reducer, we say that it is $\mathfrak{S}$-irreducible.

Property of $\mathfrak{S}$-Irreducible Element

For each $s \in NS$, there is an element of I with signature s. Among such elements, one which has the smallest leading monomial is $\mathfrak{S}$-irreducible and its leading monomial is determined uniquely by s.

Fundamentals in Signature

$\mathfrak{S}$-reduction

Let $f, g, h \in I$. We say that f is $\mathfrak{S}$-reduced to g by h, if there are $t \in \mathcal{M}$ and $a \in K \setminus \{0\}$ such that $g = f - a \cdot t \cdot h$, $\mathrm{sig}(th) \prec \mathrm{sig}(f)$ and $LM(g) \prec LM(f)$. We also call h (or $t \cdot h$) a $\mathfrak{S}$-reducer of f. This reduction don't change the signature. If $f \in I$ has no $\mathfrak{S}$-reducer, we say that it is $\mathfrak{S}$-irreducible.

Property of $\mathfrak{S}$-Irreducible Element

For each $s \in NS$, there is an element of I with signature s. Among such elements, one which has the smallest leading monomial is $\mathfrak{S}$-irreducible and its leading monomial is determined uniquely by s.

For $s \in NS$, we define the following;

$$\Phi(s) = \min_{\prec}\{\mathrm{LM}(f) \mid f \in I, \mathrm{sig}(f) = s\}.$$

Fundamentals in Signature

$\mathfrak{S}$-reduction

Let $f, g, h \in I$. We say that f is $\mathfrak{S}$-reduced to g by h, if there are $t \in \mathcal{M}$ and $a \in K \setminus \{0\}$ such that $g = f - a \cdot t \cdot h$, $\mathrm{sig}(th) \prec \mathrm{sig}(f)$ and $LM(g) \prec LM(f)$. We also call h (or $t \cdot h$) a $\mathfrak{S}$-reducer of f. This reduction don't change the signature. If $f \in I$ has no $\mathfrak{S}$-reducer, we say that it is $\mathfrak{S}$-irreducible.

Property of $\mathfrak{S}$-Irreducible Element

For each $s \in NS$, there is an element of I with signature s. Among such elements, one which has the smallest leading monomial is $\mathfrak{S}$-irreducible and its leading monomial is determined uniquely by s.

For $s \in NS$, we define the following;

$$\Phi(s) = \min_{\prec}\{\mathrm{LM}(f) \mid f \in I, \mathrm{sig}(f) = s\}.$$

$\#NS = \#V(J) - \#V(I) = d^m - \#V(I)$. When $\#V(I) \ll d^m$, $\#NS \approx d^m$.

$\mathfrak{S}$-Gröbner Basis

$\mathfrak{S}$-Gröbner Basis

$\mathfrak{S}$-Gröbner Basis

$\mathfrak{S}$-Gröbner Basis

A finite subset $\mathcal{H}$ is called a $\mathfrak{S}$-Gröbner basis of I if it satisfies the following:

$\mathfrak{S}$-Gröbner Basis

$\mathfrak{S}$-Gröbner Basis

A finite subset $\mathcal{H}$ is called a $\mathfrak{S}$-Gröbner basis of I if it satisfies the following:

(1) $\mathcal{H}$ contains a Gröbner basis of J.

$\mathfrak{S}$-Gröbner Basis

$\mathfrak{S}$-Gröbner Basis

A finite subset $\mathcal{H}$ is called a $\mathfrak{S}$-Gröbner basis of I if it satisfies the following:

(1) $\mathcal{H}$ contains a Gröbner basis of J.

(2) For each $s \in NS$, there are $t \in \mathcal{M}$ and $g \in \mathcal{H}$ such that $t \times \mathrm{sig}(g) = s$ and tg is $\mathfrak{S}$-irreducible.

$\mathfrak{S}$-Gröbner Basis

A finite subset $\mathcal{H}$ is called a $\mathfrak{S}$-Gröbner basis of I if it satisfies the following:

(1) $\mathcal{H}$ contains a Gröbner basis of J.

(2) For each $s \in NS$, there are $t \in \mathcal{M}$ and $g \in \mathcal{H}$ such that $t \times \mathrm{sig}(g) = s$ and tg is $\mathfrak{S}$-irreducible.

When the condition (2) holds only for $t = 1$, that is, there is no pair (t, g) such that $t \neq 1$, $t \times \mathrm{sig}(g) = s$ and tg is $\mathfrak{S}$-irreducible,

$\mathfrak{S}$-Gröbner Basis

$\mathfrak{S}$-Gröbner Basis

A finite subset $\mathcal{H}$ is called a $\mathfrak{S}$-Gröbner basis of I if it satisfies the following:

(1) $\mathcal{H}$ contains a Gröbner basis of J.

(2) For each $s \in NS$, there are $t \in \mathcal{M}$ and $g \in \mathcal{H}$ such that $t \times \mathrm{sig}(g) = s$ and tg is $\mathfrak{S}$-irreducible.

When the condition (2) holds only for $t = 1$, that is, there is no pair (t, g) such that $t \neq 1$, $t \times \mathrm{sig}(g) = s$ and tg is $\mathfrak{S}$-irreducible, an S-polynomial of signature s appears, from which the element of $\mathcal{H}$ of signature s is computed by $\mathfrak{S}$-reduction.

$\mathfrak{S}$-Gröbner Basis

$\mathfrak{S}$-Gröbner Basis

A finite subset $\mathcal{H}$ is called a $\mathfrak{S}$-Gröbner basis of I if it satisfies the following:

(1) $\mathcal{H}$ contains a Gröbner basis of J.

(2) For each $s \in NS$, there are $t \in \mathcal{M}$ and $g \in \mathcal{H}$ such that $t \times \mathrm{sig}(g) = s$ and tg is $\mathfrak{S}$-irreducible.

When the condition (2) holds only for $t = 1$, that is, there is no pair (t, g) such that $t \neq 1$, $t \times \mathrm{sig}(g) = s$ and tg is $\mathfrak{S}$-irreducible, an S-polynomial of signature s appears, from which the element of $\mathcal{H}$ of signature s is computed by $\mathfrak{S}$-reduction.

By this condition, we can check signatures for which S-polynomials appear.

Condition for Necessary S-Polynomial

Condition for Necessary S-Polynomial

If $s = t \times \mathrm{sig}(g)$ $(t \neq 1)$,

Condition for Necessary S-Polynomial

If $s = t \times \mathrm{sig}(g)$ $(t \neq 1)$, then $s \succ \mathrm{sig}(g)$ and it is highly expected

Condition for Necessary S-Polynomial

If $s = t \times \mathrm{sig}(g)$ $(t \neq 1)$, then $s \succ \mathrm{sig}(g)$ and it is highly expected

$$\Phi(\mathrm{sig}(g)) \succ \Phi(s),$$

Condition for Necessary S-Polynomial

If $s = t \times \mathrm{sig}(g)$ $(t \neq 1)$, then $s \succ \mathrm{sig}(g)$ and it is highly expected

$$\Phi(\mathrm{sig}(g)) \succ \Phi(s),$$

which means $t \times \Phi(\mathrm{sig}(g)) \neq \Phi(s)$ and we need to compute an S-polynomial with signature s.

Condition for Necessary S-Polynomial

If $s = t \times \mathrm{sig}(g)$ $(t \neq 1)$, then $s \succ \mathrm{sig}(g)$ and it is highly expected

$$\Phi(\mathrm{sig}(g)) \succ \Phi(s),$$

which means $t \times \Phi(\mathrm{sig}(g)) \neq \Phi(s)$ and we need to compute an S-polynomial with signature s. Because, it seems that the larger s becomes, the smaller $\Phi(s)$ becomes *in general* by looking the following:

Condition for Necessary S-Polynomial

If $s = t \times \mathrm{sig}(g)$ $(t \neq 1)$, then $s \succ \mathrm{sig}(g)$ and it is highly expected

$$\Phi(\mathrm{sig}(g)) \succ \Phi(s),$$

which means $t \times \Phi(\mathrm{sig}(g)) \neq \Phi(s)$ and we need to compute an S-polynomial with signature s. Because, it seems that the larger s becomes, the smaller $\Phi(s)$ becomes *in general* by looking the following:

$$\Phi(s) = \min_{\prec}\{LM(f) \mid f = \left(\sum_{\substack{t \in NS \\ t \preceq s}} c_t t\right) S + A_1 F(x_1) + \cdots + A_m F(x_m)\}$$

Condition for Necessary S-Polynomial

If $s = t \times \mathrm{sig}(g)$ $(t \neq 1)$, then $s \succ \mathrm{sig}(g)$ and it is highly expected

$$\Phi(\mathrm{sig}(g)) \succ \Phi(s),$$

which means $t \times \Phi(\mathrm{sig}(g)) \neq \Phi(s)$ and we need to compute an S-polynomial with signature s. Because, it seems that the larger s becomes, the smaller $\Phi(s)$ becomes *in general* by looking the following:

$$\Phi(s) = \min_{\prec}\{LM(f) \mid f = \left(\sum_{\substack{t \in NS \\ t \preceq s}} c_t t\right) S + A_1 F(x_1) + \cdots + A_m F(x_m)\}$$

$\Phi(s)$ can be determined by solving several systems of linear equations by considering c_t as indeterminates.

Condition for Necessary S-Polynomial

If $s = t \times \mathrm{sig}(g)$ $(t \neq 1)$, then $s \succ \mathrm{sig}(g)$ and it is highly expected

$$\Phi(\mathrm{sig}(g)) \succ \Phi(s),$$

which means $t \times \Phi(\mathrm{sig}(g)) \neq \Phi(s)$ and we need to compute an S-polynomial with signature s. Because, it seems that the larger s becomes, the smaller $\Phi(s)$ becomes *in general* by looking the following:

$$\Phi(s) = \min_{\prec}\{LM(f) \mid f = \left(\sum_{\substack{t \in NS \\ t \preceq s}} c_t t\right) S + A_1 F(x_1) + \cdots + A_m F(x_m)\}$$

$\Phi(s)$ can be determined by solving several systems of linear equations by considering c_t as indeterminates.

$$\Downarrow$$

As NS is a finite set, this relates to Subresultant Theory.

Regularity and Assumption

Regularity and Assumption

Regular and Semi-Regular

We say that the ideal I is regular with respect to the signature, if

Regularity and Assumption

Regular and Semi-Regular

We say that the ideal I is regular with respect to the signature, if

$(*)$ $\Phi(s) \not| \Phi(s')$ holds for any distinct pair s, s' in $NS(\mathrm{Syz})$ with $s \mid s'$.

Regularity and Assumption

Regular and Semi-Regular

We say that the ideal I is regular with respect to the signature, if

$$(*) \quad \Phi(s) \nmid \Phi(s') \text{ holds for any distinct pair } s, s' \text{ in } NS(\mathrm{Syz}) \text{ with } s \mid s'.$$

For a subset A of NS, we say that I is A-semi-regular with respect to the signature, if the condition $(*)$ holds for almost every $s, s' \in A$.

Regularity and Assumption

Regular and Semi-Regular

We say that the ideal I is regular with respect to the signature, if

$$(*) \quad \Phi(s) \nmid \Phi(s') \text{ holds for any distinct pair } s, s' \text{ in } NS(\mathrm{Syz}) \text{ with } s \mid s'.$$

For a subset A of NS, we say that I is A-semi-regular with respect to the signature, if the condition $(*)$ holds for almost every $s, s' \in A$.

We set

$$\overline{NS} = NS \setminus \{x_1^{e_1} \cdots x_m^{e_m} \mid 0 \leq e_i < d - \delta\} \cup \{1\}.$$

Regularity and Assumption

Regular and Semi-Regular

We say that the ideal I is regular with respect to the signature, if

$$(*) \quad \Phi(s) \nmid \Phi(s') \text{ holds for any distinct pair } s, s' \text{ in } NS(\mathrm{Syz}) \text{ with } s \mid s'.$$

For a subset A of NS, we say that I is A-semi-regular with respect to the signature, if the condition $(*)$ holds for almost every $s, s' \in A$.

We set

$$\overline{NS} = NS \setminus \{x_1^{e_1} \cdots x_m^{e_m} \mid 0 \leq e_i < d - \delta\} \cup \{1\}.$$

Assumption:

Regularity and Assumption

Regular and Semi-Regular

We say that the ideal I is regular with respect to the signature, if

$$(*) \quad \Phi(s) \nmid \Phi(s') \text{ holds for any distinct pair } s, s' \text{ in } NS(\mathrm{Syz}) \text{ with } s \mid s'.$$

For a subset A of NS, we say that I is A-semi-regular with respect to the signature, if the condition $(*)$ holds for almost every $s, s' \in A$.

We set

$$\overline{NS} = NS \setminus \{x_1^{e_1} \cdots x_m^{e_m} \mid 0 \leq e_i < d - \delta\} \cup \{1\}.$$

Assumption: For almost every $\beta = x(aP + bQ)$, $I_m(\beta)$ is $\overline{NS}$-semi-regular.

Complexity of Naive ICM

Complexity of Naive ICM

Under Assumption, the number of necessary S-polynomials is estimated as follows:

Complexity of Naive ICM

Under Assumption, the number of necessary S-polynomials is estimated as follows:

$$\#\overline{NS} \approx d^m - (d-\delta)^m + 1 > md^{m-1}.$$

Complexity of Naive ICM

Under Assumption, the number of necessary S-polynomials is estimated as follows:

$$\#\overline{NS} \approx d^m - (d-\delta)^m + 1 > md^{m-1}.$$

$$\Downarrow$$

Our naive ICM cannot be more efficient than brute force methods.

Complexity of Naive ICM

Under Assumption, the number of necessary S-polynomials is estimated as follows:

$$\#\overline{NS} \approx d^m - (d-\delta)^m + 1 > md^{m-1}.$$

$$\Downarrow$$

Our naive ICM cannot be more efficient than brute force methods.

Considering the ideal $I_{m+1}(\beta)$ over $\mathbb{Q}$, we may conclude the following, since $\Phi(s)$ can be determined by checking (semi) algebraic conditions on β:

Complexity of Naive ICM

Under Assumption, the number of necessary S-polynomials is estimated as follows:

$$\#\overline{NS} \approx d^m - (d-\delta)^m + 1 > md^{m-1}.$$

$$\Downarrow$$

Our naive ICM cannot be more efficient than brute force methods.

Considering the ideal $I_{m+1}(\beta)$ over $\mathbb{Q}$, we may conclude the following, since $\Phi(s)$ can be determined by checking (semi) algebraic conditions on β:

For some value $\beta(= x(aP + bQ))$ and modulus p, if the ideal is $\overline{NS}$-semi-regular, then it is so for almost every value and modulus.

Future Works

Future Works

- Make our arguments rigid by deeper analysis on the behavior of signature-based algorithms.

Future Works

- Make our arguments rigid by deeper analysis on the behavior of signature-based algorithms.
  - Examine whether signature-based algorithms always produce lesser number of S-polynomials (at least in our case).

Future Works

- Make our arguments rigid by deeper analysis on the behavior of signature-based algorithms.
  - Examine whether signature-based algorithms always produce lesser number of S-polynomials (at least in our case).
  - Make further computational experiments for larger m and p.

Future Works

- Make our arguments rigid by deeper analysis on the behavior of signature-based algorithms.
  - Examine whether signature-based algorithms always produce lesser number of S-polynomials (at least in our case).
  - Make further computational experiments for larger m and p.
- Apply our arguments to ICMs with Weil descent technique, where we define the signature on the $K[X]$ module.

Future Works

- Make our arguments rigid by deeper analysis on the behavior of signature-based algorithms.
  - Examine whether signature-based algorithms always produce lesser number of S-polynomials (at least in our case).
  - Make further computational experiments for larger m and p.
- Apply our arguments to ICMs with Weil descent technique, where we define the signature on the $K[X]$ module.
- Our arguments may find some weak keys.

Future Works

- Make our arguments rigid by deeper analysis on the behavior of signature-based algorithms.
  - Examine whether signature-based algorithms always produce lesser number of S-polynomials (at least in our case).
  - Make further computational experiments for larger m and p.
- Apply our arguments to ICMs with Weil descent technique, where we define the signature on the $K[X]$ module.
- Our arguments may find some weak keys.

Thank you for your attention.