An Efficient Algorithm for Solving the MQ Problem using Hilbert Series

Kosuke Sakata[*] Tsuyoshi Takagi[*]

Abstract

The security of multivariate polynomial cryptography depends on the computational complexity of solving a multivariate quadratic polynomial system (MQ problem). One of the fastest algorithms for solving the MQ problem is F4, which computes a Gröbner basis but requires numerous calculations of zero reduction that do not affect the output. The Hilbert-driven algorithm evaluates the number of generators in the Gröbner basis of degree d using Hilbert series, then it reduces the number of zero reduction computations. In this paper, we propose a high-speed algorithm designed for randomly generated semi-regular MQ problems. Although the Hilbert-driven algorithm is limited to computing homogeneous ideals, we demonstrate its applicability to semi-regular non-homogeneous ideals in this work. Furthermore, when using the Hilbert-driven algorithm to solve non-homogeneous MQ problems with F4, we demonstrate the efficient achievement of reducing zero reduction for F4. We implemented the proposed algorithm in C++ and report successful decryption of a new record $m = 21$ Type VI equations. This was achieved using an AMD EPYC 7742 processor and 2TB RAM, and the decryption process was completed within approximately 9 h.

1 Introduction

Public key cryptography is an indispensable technology for maintaining the security of modern information communication systems, and RSA and elliptic curve cryptography are currently employed. However, with the practical realization of quantum computers, there is concern regarding the significantly decreased security of public key cryptography, as Shor's algorithm [1] may potentially break current public key encryption schemes in polynomial time. Consequently, research is underway to develop post-quantum cryptographic algorithms capable of withstanding quantum computers.

[*]Department of Mathematical Informatics, The University of Tokyo, Japan

The National Institute of Standards and Technology (NIST) is currently in the process of selecting next-generation cryptographic schemes, and evaluating the infeasibility of breaking these candidates is an important task. Currently, a selection process for digital signatures is also underway, with multivariate polynomial cryptography as a major candidate. Multivariate polynomial cryptography relies on the security of the Multivariate Quadratic (MQ) problem. Gröbner basis computation is a prominent method for computing MQ problems, and improving Gröbner basis algorithms is closely related to the evaluation of the computational complexity of MQ problems. Therefore, improving the efficiency of Gröbner basis computation is an extremely important research topic in computational algebra and cryptography.

There are several algorithms for Gröbner basis computation to solve the MQ problem, with F4 being particularly well-known[2]. One common issue in Gröbner basis computation is the excessive calculation of zero reductions, which do not affect the output, and this problem also persists in F4. The Hilbert-driven algorithm is recognized as an approach that avoids the calculation of zero reductions [3]. In the Hilbert-driven algorithm, after computing an S-polynomial of pairs of degree d, the Hilbert function is computed at each step to determine whether the subsequently calculated S-polynomials of degree d will result in zero reduction. However, practical implementation requires considerable computational effort due to the frequent computation of Hilbert functions. Furthermore, the Hilbert-driven algorithm only guarantees accuracy for homogeneous polynomial sequences, making it unsuitable for non-homogeneous polynomial sequences like the MQ problem.

In this paper, we demonstrate that when the defining field of the polynomial ring is an infinite field, the Hilbert-driven algorithm can be applied even when the input is a non-homogeneous polynomial sequence, provided that it is a semi-regular sequence. This suggests that the zero reduction decision method used in the Hilbert-driven algorithm can be applied to the MQ problem. Moreover, this approach is suitable for the non-homogeneous regular sequences defined in this paper.

Next, assuming the correctness of the Moreno-Socías conjecture, we demonstrate that when the input is a semi-regular sequence, it is possible to compute the number of elements of each degree in the ouput the Hilbert-driven algorithm without performing Gröbner basis computation. When the input is a homogeneous sequence, the number of elements of each degree is the number of the Gröbner basis of each degree. In other words, this eliminates the need to compute the Hilbert function every time S-polynomials are calculated in the Hilbert-driven algorithm. Additionally, the algorithm to calculate the number of bases for each degree is significantly more efficient compared to Gröbner basis computation. This method is also applicable to the non-homogeneous regular sequences defined in this paper.

Furthermore, we propose an algorithm for efficiently solving the MQ problem defined over a polynomial ring with a finite field using the aforementioned approach. If it is possible to select pairs of S-polynomials in a way that avoids zero reduction, then calculating the number of S-polynomials for each degree is sufficient, as it equals the number of Gröbner bases. Given a sufficiently large number of elements in the finite field, we expect that the results obtained from an infinite field can also be applied to the finite field. Therefore, through computational experiments, we demonstrate the utility of the proposed method for solving the MQ problem and its reliability in performing Gröbner basis computations by calculating slightly more S-polynomials than the number of elements of each degree.

Finally, we applied the proposed algorithm to perform computations for the Fukuoka MQ Challenge[4], which is a decryption competition for the MQ problem, using C++. As a result, we report successful decryption, achieved in approximately 9 h, of a new record problem for Type VI with $m = 21$ equations using an AMD EPYC 7742 processor and 2 TB of RAM.

2 Preliminaries

2.1 Notation

In this section, we describe the definitions and symbols used in this paper. Let $\mathbb{Z}$ denote the set of integers, $\mathbb{Z}_{\geq 0}$ denote the set of non-negative integers, k denote an arbitrary field, and $\mathbb{F}_q$ denote a finite field with q elements. Consider $R = k[x_1, \ldots, x_n]$ as an n-variable polynomial ring over the coefficient field k with variables $x_1, \ldots, x_n$. Elements of R that are composed of the product of variables with coefficients equal to 1 are referred to as monomials. Monomials in R take the form $x_1^{a_1} \cdots x_n^{a_n}$, where $(a_1, \ldots, a_n \in \mathbb{Z}_{\geq 0})$. For any monomials t and u in R, we use the notation $t \mid u$ to signify that monomial t divides monomial u. A monomial ordering , which we denote as $<$, is fixed in R. It is generally known that when computing Gröbner bases, using the degree reverse lexicographic order for monomial ordering allows for efficient calculations. In this paper, we adopt the degree reverse lexicographic order as the monomial order.

Let $f \in R$ be a polynomial. Among the monomials contained in f, we denote the monomial with the highest order according to the chosen monomial order as $\mathrm{LM}(f)$ and refer to this as the leading monomial of f. $\mathrm{LC}(f)$ represents the coefficient of the leading monomial of f, and $\mathrm{LT}(f)$ is defined as $\mathrm{LC}(f) \cdot \mathrm{LM}(f)$, which we refer to as the leading term of f. The total degree of f is denoted as $\deg(f)$.

The ideal generated by the polynomial sequence $F = \{f_1, \ldots, f_m\}$ is denoted as $\langle F \rangle$. The set of leading terms of F is denoted as $\mathrm{LT}(F)$. The ideal generated by the leading terms of F is denoted as $\langle \mathrm{LT}(F) \rangle$. Let $I \subset R$ be an ideal, and let G be a set of polynomials that generates I. For any $f \in I$, if there exists a $g \in G$ such that $\mathrm{LM}(g) \mid \mathrm{LM}(f)$, we refer to G as a Gröbner basis for I. We write f_i^{top} to denote the sum of the terms in f_i with degrees equal to $\deg(f_i)$. Likewise, we write f_i^{tail} to denote the sum of the terms in f_i with degrees less than $\deg(f_i)$. It is important to note that $f_i = f_i^{top} + f_i^{tail}$. We define $F^{top} = \{f_1^{top}, \ldots, f_m^{top}\}$.

2.2 Buchberger Algorithm

The algorithm for computing Gröbner bases was proposed by Buchberger in 1965[5], and it is commonly known as the Buchberger algorithm. The S-polynomial $\mathrm{Spoly}(f, g)$ for a pair of polynomials f and g in R is defined as

$$\mathrm{Spoly}(f, g) = \frac{\mathrm{LCM}(\mathrm{LT}(f), \mathrm{LT}(g))}{\mathrm{LT}(f)} f - \frac{\mathrm{LCM}(\mathrm{LT}(f), \mathrm{LT}(g))}{\mathrm{LT}(g)} g$$

where LCM represents a function that calculates the least common multiple of monomials. When we have polynomials f and h in R such that $\mathrm{LM}(h) \mid \mathrm{LM}(f)$, the following calculation is referred to as a reduction:

$$f - \frac{\mathrm{LT}(f)}{\mathrm{LT}(h)} h$$

Let $F = \{f_1, \ldots, f_m\} \subset R$ be a set of polynomials, and let I be the ideal generated by F. The Buchberger algorithm is computed with input F through the following steps:

(B.1) Select a pair generated by elements from F that has not yet been considered and generate its S-polynomial.

(B.2) Reduce the generated S-polynomial using the elements of F until it cannot be reduced by F.

(B.3) If the reduced S-polynomial is not zero, include it in F.

(B.4) If there are no more unconsidered pairs of elements in F, stop the algorithm and output F.

When the above algorithm terminates, F is the Gröbner basis for I.

In this paper, we refer to the computation that reduces S-polynomials to zero as "zero reduction." Zero reductions are considered unnecessary as they do not provide additional information about the Gröbner basis. Thus, the algorithm can be made more efficient by eliminating zero reductions. Several methods for avoiding zero reductions have been proposed in prior works, such as [6] and

[7]. Additionally, we will explain another technique called the Hilbert-driven algorithm in the following sections.

Theorem 1. *[2] Let F be a set of homogeneous polynomials, where $I = \langle F \rangle$, and G_d be the output of the Buchberger algorithm up to degree d. Then, the following properties hold:*

- *Any $p \in I$ with $\deg(p) \leq d$ can be reduced to zero by G_d.*
- *S-polynomials generated by f and g in G_d with $\deg(\mathrm{LCM}(\mathrm{LT}(f), \mathrm{LT}(g))) \leq d$ are reduced to zero by G_d.*

A set of G_d with the abovementioned properties is referred to as a d-Gröbner basis.S

2.3 Hilbert Series

Let k be a field and $d \geq 0$ be an integer, and consider R_d as the k-vector space generated by the monomials of degree d in the polynomial ring R. In this section, we assume that the ideal $I \subset R$ is a homogeneous ideal. For an ideal $I \subset R$, we define I_d as

$$I_d = \{f \in I \mid f \text{ is a homogeneous polynomial with } \deg(f) = d\} \cup \{0\}.$$

Definition 1. *We define the Hilbert function of the quotient ring R/I as follows:*

$$\begin{array}{rccc} h_{R/I}: & \mathbb{N} & \longrightarrow & \mathbb{N} \\ & d & \longmapsto & \dim_k(R/I)_d. \end{array}$$

Using the function $h_{R/I}$, we define the Hilbert series $H_{R/I}(z)$ of the quotient ring R/I as follows:

$$H_{R/I}(z) = \sum_{d \geq 0} h_{R/I}(d) z^d.$$

2.4 Hilbert-driven Algorithm

The Hilbert-driven algorithm [3] is an algorithm for finding Gröbner bases that reduces the number of zero reduction computations. Its input consists of a set of homogeneous polynomials F and the Hilbert series $H_{R/I}(z)$ of the ideal generated by these polynomials, denoted by $I = \langle F \rangle$. Using Hilbert series information, the calculation of unnecessary zero reductions in the algorithm can be avoided. The relationship between the Hilbert series and Gröbner bases can be characterized by the following proposition.

Theorem 2. *[8] Let I be a homogeneous ideal in a polynomial ring R, and consider a set of polynomials $F \subset I$. For all $d \in \mathbb{Z}_{\geq 0}$, the following holds:*

$$h_{R/I}(d) \leq h_{R/\langle \mathrm{LT}(F) \rangle}(d).$$

Furthermore, the equality holding for all $d \in \mathbb{Z}_{\geq 0}$ is equivalent to the set F being a Gröbner basis for the ideal I.

To explain the Hilbert-driven algorithm, we use the following definitions:

Definition 2. *The degree of the pair generating the S-polynomial* $\mathrm{Spoly}(f, g)$ *is defined as* $\deg(\mathrm{LCM}(\mathrm{LT}(f), \mathrm{LT}(g)))$.

The Hilbert-driven algorithm takes a set of homogeneous polynomials $F = \{f_1, \ldots, f_m\} \subset R$ as input, where ideal I is generated by F. Additionally, the Hilbert series $H_{R/I}(z) = \sum_{d \geq 0} H_{R/I}(d) z^d$ of the quotient ring R/I is provided as input. The algorithm transforms the Buchberger algorithm as follows:

When the algorithm stops, F serves as the Gröbner basis of I. The steps of the algorithm are summarized as follows ':

(H.1) Choose the smallest d such that $h_{R/I}(d) \neq h_{\langle \mathrm{LT}(F) \rangle}(d)$.

(H.2) Select a pair whose degree is d generated by elements from F that has not yet been considered, and generate its S-polynomial.

(H.3) Reduce the generated S-polynomial within F until it cannot be further reduced by F.

(H.4) If the reduced S-polynomial is not equal to zero, add it to the set F.

(H.5) If $H_{R/I}(z) = H_{R/\langle \mathrm{LT}(F) \rangle}$, stop the algorithm and output F.

The key difference from the Buchberger algorithm is in step (H.1). For every S-polynomial computed between a pair of a specific degree d, the algorithm calculates $h_{\langle \mathrm{LT}(F) \rangle}(d)$ using the current F and compares it with $h_{R/I}(d)$. When the condition $h_{R/\langle \mathrm{LT}(F) \rangle}(d) = h_{R/I}(d)$ is met, subsequent S-polynomials of the same degree d are not computed because they would result in only zero reductions. This minimizes unnecessary calculations and enhances the efficiency of the Hilbert-driven algorithm.

The proof that S-polynomials of degree d become zero reductions when the condition $h_{R/\langle \mathrm{LT}(F) \rangle}(d) = h_{R/I}(d)$ is met is as follows. Assume that under this condition, a case exists in which an S-polynomial of degree d do not become a zero reduction. In this case, we denote the reduced S-polynomial as f and the set of polynomials immediately before the computation of f as F. Then, at some degree $d' \leq d$, it must hold that $h_{R/I}(d') = h_{R/\langle \mathrm{LT}(F) \rangle}(d') > H_{R/\langle \mathrm{LT}(F+f) \rangle}(d')$. This leads to a contradiction with Theorem 2. In step (H.5), when $H_{R/I}(z) = H_{R/\langle \mathrm{LT}(F)\rangle}$ is satisfied, the algorithm terminates. This condition is a consequence of Theorem 2 and indicates that the algorithm has completed the computation of the Gröbner basis.

2.5 Semi-regular

In this section, let k be an infinite field, and let R be a polynomial ring over k. It is expected that "randomly" generated homogeneous polynomial sequences $f_1, \ldots, f_m \subset R$ possess the characteristics of a semi-regular sequence.

Definition 3. *[9] Let I be a homogeneous ideal in polynomial ring R, and let $f \in R$ be a homogeneous polynomial with $\deg(f) = d$. For any natural number $e > d$, if the linear map induced by multiplying by f, $(R/I)_{e-d} \to (R/I)_d$, is full rank, we call f semi-regular on R/I.*

Furthermore, for a sequence of homogeneous polynomials $f_1, \ldots, f_m \subset R$, for all $i = 1, \ldots, m$, if f_i is semi-regular on $R/\langle f_1, \ldots, f_{i-1}\rangle$, then we call $f_1, \ldots, f_m$ a semi-regular sequence.

Definition 4. *[9] Let $\sum_{i=0}^{\infty} a_i z^i$ be a series constructed with $a_i \in \mathbb{Z}$. The expression $[\sum_{i=1}^{\infty} a_i z^i]$ is defined as a series $\sum_{i=1}^{\infty} b_i z^i$ obtained from the following sequence $\{b_i\}_{i\geq 0}$: if $a_j > 0$ for all $0 \leq j \leq i$, then $b_i = a_i$; otherwise, $b_i = 0$.*

Using the Hilbert series, semi-regular sequences can be characterized as follows.

Theorem 3. *[9] A sequence of homogeneous polynomials $\{f_1, \ldots, f_m\}$ with $\deg(f_i) = d_i$ is a semi-regular sequence if and only if the following conditions are equivalent:*

$$H_{R/\langle f_1,\ldots,f_m\rangle}(z) = \left[\frac{\prod_{i=1}^{m}(1 - z^{d_i})}{(1 - z)^n}\right].$$

Definition 5. *For a semi-regular sequence $\{f_1, \ldots, f_m\}$ with $\deg(f_i) = d_i$, we define d_u, the upper degree, as the largest i for which the coefficients $a_i \geq 0$ in the following series:*

$$\sum_{i=0}^{\infty} a_i z^i = \frac{\prod_{i=1}^{m}(1 - z^{d_i})}{(1 - z)^n}$$

Definition 6. *[9] A monomial ideal $I \subset R$ is called a weakly reverse lexicographic ideal when it satisfies the following condition: for any minimal generator $m \in I$, if $m' > m$ and $\deg(m') = \deg(m)$, then $m' \in I$.*

The following conjecture is known as the Moreno-Socías conjecture.

Conjecture 1. *[10][9] Let F be a semi-regular sequence composed of homogeneous polynomials. Let $I = \langle F\rangle$, and consider $J = \langle \mathrm{LT}(I)\rangle$. Then, J forms a weakly reverse lexicographic ideal.*

The definition of a regular sequence in this paper is assumed to be the same as that in the paper [11].

2.6 F4

F4 is a variant of the Buchberger algorithm, known for its ability to efficiently compute Gröbner bases[2]. F4 operates by generating multiple S-polynomials of a given degree d and subsequently generating polynomials to reduce these S-polynomials. The generated S-polynomials are then collectively reduced.

We outline the general steps of F4 as follows:

(F.1) Select a minimum degree d among the pairs generated by F that have not yet been computed, and choose all pairs with degrees equal to d.

(F.2) Generate S-polynomials for the selected pairs.

(F.3) Generate all polynomials required to reduce the generated S-polynomials.

(F.4) Reduce the generated S-polynomials by the polynomials generated in (F.3).

(F.5) If the reduced S-polynomials are not zero, include them in F.

The key distinction between F4 and the Buchberger algorithm is in steps (F.1) and (F.2), where F4 simultaneously selects and generates S-polynomials for multiple pairs. F4 incorporates a technique introduced to reduce the number of zero reductions, as described in [6]. However, even with this technique, the number of zero reductions remains substantial.

3 Non-homogeneous Hilbert-driven Algorithm

In this section, we assume that the coefficient field of the polynomial ring is infinite. The Hilbert-driven algorithm guarantees accuracy only for the computation of Gröbner bases of homogeneous ideals. In this section, we demonstrate that the Hilbert-driven algorithm can be applied until the algorithm stops when non-homogeneous ideals form a semi-regular sequence.

Theorem 4. *Let F be a non-homogeneous semi-regular sequence. Consider G_d and G_d^{top} as the outputs obtained by running the Hilbert-driven algorithm up to degree d on F and F^{top}, respectively. Then, calculations by the Hilbert-driven algorithm up to the upper degree d_u ensure that $\mathrm{LT}(G_d) = \mathrm{LT}(G_d^{top})$ for each degree d.*

Proof. Assume that, up to degree d, $\mathrm{LT}(G_d) = \mathrm{LT}(G_d^{top})$, with the condition that $\mathrm{LT}(g_i) = \mathrm{LT}(g_i^{top})$ for all i (they are sorted accordingly). Now, we prove that $\mathrm{LT}(G_{d+1}) = \mathrm{LT}(G_{d+1}^{top})$.

First, we prove $\mathrm{LT}(G_{d+1}) \supset \mathrm{LT}(G_{d+1}^{top})$. Let g be an element of G_{d+1}^{top}. If the degree of g is d or less, then according to the assumption $\mathrm{LT}(G_d) = \mathrm{LT}(G_d^{top})$, $\mathrm{LT}(g)$ is also contained in $\mathrm{LT}(G_{d+1})$. Now, we consider the degree of g as $d+1$. Note that g is derived by reducing S-polynomials whose pair degrees are up to $d+1$. Let g be $g = \Sigma_i^t h_i g_i^{top}$, where $(g_i^{top} \in G_d^{top}, t$ is a natural number. Now, we consider $g' = \Sigma_i^t h_i g_i = \Sigma_i^t h_i g_i^{top} + \Sigma_i^t h_i g_i^{tail}, (g_i \in G_d)$. g' can be generated by reducing S-polynomials formed by pairs of elements in G_d, where the degree of each pair is $d+1$. Therefore, according to the conditions of G_{d+1}, we have $\mathrm{LT}(g) = \mathrm{LT}(g') \in \mathrm{LT}(G_{d+1})$. Hence, $\mathrm{LT}(G_{d+1}) \subset \mathrm{LT}(G_{d+1}^{top})$.

We next prove $\mathrm{LT}(G_{d+1}) \subset \mathrm{LT}(G_{d+1}^{top})$. Consider g as an element of G_{d+1}. If g is also an element of G_d, then due to the assumption $\mathrm{LT}(G_d) = \mathrm{LT}(G_d^{top})$, we have $\mathrm{LT}(g) \in \mathrm{LT}(G_d^{top}) \subset \mathrm{LT}(G_{d+1}^{top})$. Therefore, consider $g \in G_{d+1} \setminus G_d$. In this case, g is derived by reducing S-polynomials whose pair degrees are up to $d+1$. Assume that the degree of g is less than $d+1$. We prove that, in this case, $g = 0$.

Let g be $g = \Sigma_i^m h_i f_i = \Sigma_i^m h_i f_i^{top} + \Sigma_i^m h_i f_i^{tail}$, where m represents the number of elements in F. Furthermore, we define the set $h_i^{(e)} i, e \subset R$ such that $h_i = \Sigma e h_i^{(e)}$ and $\deg(h_i^{(e)} f_i) = e$. As the degree of g is less than $d+1$, we have $\Sigma_i^m h_i^{(d+1)} f_i^{top} = 0$. As F^{top} is semi-regular, $\Sigma_i^m h_i^{(d+1)} f_i^{top}$ is a trivial syzygy[11]. Hence, we can express $h_i^{(d+1)}$ using $r_{ij} i, j \subset R$ such that $rij = -r_{ji}$ and $r_{ii} = 0$ as follows: $h_i^{(d+1)} = \Sigma_j^m r_{ij} f_j^{top}$. Therefore, we can write $\Sigma_i^m h_i^{(d+1)} f_i^{tail} = \Sigma_i^m (\Sigma_j^m r_{ij} f_j^{top}) f_i^{tail}$. Here, focusing on the component of degree d in g, it can be observed that it is in $\langle F^{top} \rangle$. Because G_d^{top} serves as a d-Gröbner basis of $\langle F^{top} \rangle$, the components of degree d in g can be reduced to zero by G_d^{top}. Considering $\mathrm{LT}(G_d) = \mathrm{LT}(G_d^{top})$, it follows that the components of degree d in g can be reduced to a certain polynomial whose total degree is smaller than d by G_d. By repeating this process, we find that g is contained within $\langle F^{top} \rangle$, and it can be reduced by G_d^{top} and in turn also by G_d. However, because this does not satisfy the conditions of G_{d+1}, we can conclude that $g = 0$. Let the degree of g be $d+1$. We represent g as $g = \Sigma_i^t h_i g_i$, where t is a natural number. When considering $\Sigma_i^t h_i g_i^{top}$, this polynomial has a degree of $d+1$ and is contained within $\langle F^{top} \rangle$. Therefore, by Theorem 1, it is divisible by G_d^{top}. Hence, there exists a $g' \in G_d^{top}$ such that $\mathrm{LT}(g')$ divides $\mathrm{LT}(g)$. Consequently, $\mathrm{LT}(G_{d+1}) \subset \mathrm{LT}(G_{d+1}^{top})$.

Based on the above, we have proved that $\mathrm{LT}(G_{d+1}) = \mathrm{LT}(G_{d+1}^{top})$. $\square$

From Theorem 4, it follows that even if the computed sequence of polynomials is non-homogeneous, as long as it is a semi-regular sequence, we can use the algorithm described in Section 2.4. However, when the algorithm terminates, the output is not a Gröbner basis. To compute a Gröbner basis, it is necessary to run another Gröbner basis algorithm after the termination of the algorithm described in Section 2.4 in order to calculate the uncomputed S-polynomials.

Furthermore, the proof of Theorem 4 allows us to establish the following corollary.

Corollary 1. *Let F be a non-homogeneous regular sequence. Consider G_d and G_d^{top} as the outputs obtained by running the Hilbert-driven algorithm up to degree d on F and F^{top}, respectively. Then, calculations by the Hilbert-driven algorithm ensure that* $\mathrm{LT}(G_d) = \mathrm{LT}(G_d^{top})$ *for each degree d.*

From this corollary, it follows that the Hilbert-driven algorithm is applicable to regular sequences as well.

4 Proposal of an Algorithm to Calculate the Number of the Gröbner Basis

In this chapter, assuming the validity of Conjecture 1, we present an algorithm for computing the number of Gröbner bases at each degree for a semi-regular homogeneous polynomial sequence.

This algorithm was inspired by the Hilbert-driven algorithm and is based on the idea of utilizing the Hilbert series. When employing this algorithm, the need to compute $H_{R/\langle \mathrm{LT}(F)\rangle}(d)$ at each step of S-polynomial calculation in the Hilbert-driven algorithm is eliminated, resulting in more efficient computations. **Algorithm 1** is an algorithm to compute the number of Gröbner bases for each degree d.

Algorithm 1 Calculating the number of Gröbner bases for each degree d

```
Require: A semi-regular homogeneous sequence, $F = \{f_1, \dots, f_m\}$
Ensure: The number of a Gröbner bases for each degree $d$, $\{N_d\}_{0 \le d \le d_u}$
 1: $H_{R/\langle F\rangle}(z) \leftarrow \left[\frac{\prod_{i=1}^{m}(1-z^{d_i})}{(1-z)^n}\right], (d_i = \deg(f_i))$
 2: $d_u \leftarrow$ upper degree of $\{f_1, \dots, f_m\}$
 3: $LT_G \leftarrow \varnothing$
 4: $d \leftarrow 0$
 5: while $d \le d_u$ do
 6:     $B_d \leftarrow \{m \in M_d \mid m \notin \langle LT_G\rangle\}$
 7:     $N_d \leftarrow \#B_d - h_{R/\langle F\rangle}(d)$
 8:     $LT_G \leftarrow LT_G \cup \{$top $N_d$ elements from $B_d$ in descending order$\}$
 9:     $d \leftarrow d + 1$
10: end while
11: return $\{N_d\}_{0 \le d \le d_u}$
```

In **Algorithm 1**, the input is a semi-regular sequence $\{f_1, \dots, f_m\}$ according to Definition 3, and the output is the number of Gröbner bases for each degree $d \le d_u$, denoted as $\{N_d\}_{0 \le d \le d_u}$. In line 1, we compute $H_{R/\langle F \rangle}(z)$ using Theorem 3. In line 2, we calculate the upper degree, as defined in Definition 5. Lines 3 and 4 initialize LT_G and d. Subsequently, we proceed with the loop from lines 6 to 10 up to degree d_u. In line 6, we define the set of monomials in R of degree d as M_d and calculate the subset of M_d not contained in $\langle LT_G \rangle$ as B_d. Line 7 computes N_d as the difference between the number of elements in B_d and coefficient $h_{R/\langle F \rangle}(d)$ from the Hilbert series. In line 8, the top N_d monomials from B_d are added to the set LT_G in descending order according to the term order. Line 9 updates the degree d for the next iteration.

Here, we explain that $\{N_d\}_{0 \le d \le d_u}$ is the number of the Gröbner basis of each degree $d \le d_u$. At the beginning of the loop, we assume that $LT_G = \mathrm{LT}(G_{d-1}^{top})$ is satisfied. Then, B_d consists of monomials that are not divisible by $\mathrm{LT}(G_{d-1})$. Thus, B_d are candidates for the leading terms of Gröbner basis elements at degree d. The fact that each element in B_d cannot divide another element in B_d and the definition of the Hilbert series implies that the number of the Gröbner basis at degree d is given by $\#B_d - h_{R/\langle F \rangle}(d)$. Assuming Conjecture 1 is correct, we can conclude that the leading terms of the Gröbner basis at degree d come from the N_d monomials with the largest order in B_d. Therefore, it follows that $LT_G = \mathrm{LT}(G_d)$. Consequently, when the algorithm terminates, we conclude that $LT_G = \mathrm{LT}(G)$.

Thus far, **Algorithm 1** has been described with a focus on homogeneous semi-regular sequences as inputs. For non-homogeneous semi-regular sequences, by Theorem 4, it is possible to compute the number of elements in the output of the algorithm in Section 2.4.

We consider the computational complexity of **Algorithm 1**. It is known that the computational complexity of a Gröbner basis calculation is determined by polynomial reduction, that is, matrix operations[11]. Because the proposed algorithm does not involve matrix operations and calculates only the divisibility of monomials, its computational complexity is low. In fact, in our C++ implementation, the computation time of our proposed algorithm is less than 1% of that of Gröbner basis calculation.

Furthermore, **Algorithm 1** can be extended even when the input is a regular sequence.

5 Proposal of the Fast Algorithm for the MQ Problem

In this section, we propose an algorithm to solve the MQ problem over the finite field $\mathbb{F}_q$ quickly.

The basic concept of the proposed algorithm is as follows. Because the Buchberger algorithm calculates S-polynomials one-by-one, it is possible to check (H.1) one-by-one, then the Hilbert-driven algorithm in Section 2.4 can be used. F4 achieves high speed by simultaneously generating and reducing multiple S-polynomials of a certain degree d at once, as shown in Section 2.6. However, because all S-polynomials of pairs of degree d are calculated, (H.1) cannot be included in the algorithm. We now consider applying the idea of the Hilbert-driven algorithm to F4. As can be seen from Sections 2.2 and 2.6, the algorithm generates S-polynomials, reduces them, and obtains elements of a Gröbner basis if they do not result in zero reduction. Hence, it is necessary to compute at least as many S-polynomials as the number of elements of the Gröbner basis. Conversely, if all S-polynomials of selected pairs do not result in zero reduction, the number of S-polynomials to be calculated is equal to the number of Gröbner basis elements. The Hilbert-driven algorithm can eliminate zero reductions from the Hilbert series information, and we propose a method for calculating the number of elements $\{N_d\}_{0\leq d\leq d_u}$ in Section 4 (**Algorithm 1**) based on this idea.

By setting the number of selected pairs to $\{N_d\}_{0\leq d\leq d_u}$ in (F.1), we can utilize a method to skip the zero reduction in the Hilbert-driven algorithm. However, if the selected pairs in (F.1) include a zero reduction, the computation will fail. Hence, to select pairs with degree $d > 4$, we propose the following approach:

(#) When selecting pairs of degree d in (F.1), select pairs generated by polynomials of degree $d-1$.

This selection method is experimentally appropriate for the MQ problem. However, because the MQ problem is defined over a finite field, there is still a chance of inadvertently selecting zero reductions using this method. Therefore, the success rate of the algorithm can be improved by increasing the number of computed S-polynomials slightly beyond the number of $\{N_d\}_{0\leq d\leq d_u}$. In the proposed algorithm, as a way to calculate a slightly higher number of S-polynomials ($e \geq 0$), we compute $\{N_d\}_{0\leq d\leq d_u} + e$ elements of S-polynomials.

5.1 Proposed Algorithm

The proposed algorithm performs according to the following (F'.1) to (F'.5). The input of the algorithm is a polynomial sequence $F = \{f_1, \dots, f_m\}$, which is a semi-regular sequence for the MQ problem. First, **Algorithm 1**is employed, and the output is denoted by $\{N_d\}_{0\leq d\leq d_u}$. When the degrees of pairs to be computed are $d = 2, 3, 4$, we compute F4 as described in Section 2.6. For degrees $5 \leq d \leq d_u$, we proceed as follows:

(F'.1) Determine the minimum degree d among pairs of elements in F that have not yet been computed. Select $N_d + e$ pairs whose degrees of pairs are equal to d according to the (#) selection method.

(F'.2) Generate S-polynomials for the selected pairs.

(F'.3) Generate all polynomials that are reducible by the generated S-polynomials.

(F'.4) Reduce the generated S-polynomials by the polynomials generated in (F'.3).

(F'.5) If the reduced S-polynomials are not equal to zero, include them in F.

After completing the calculations for degree d_u, we perform calculations similar to those in F4.

The difference from F4 lies in (F'.1), and this modification significantly reduces the number of computed S-polynomials. A comparison of the number of computed S-polynomials is presented in Section 5.3.

5.2 Success Probability of the Proposed Algorithm

In this section, we confirm the effectiveness of the proposed algorithm through computational experiments. MQ($\mathbb{F}_q, n, m$) refers to the MQ problem with a field defined as $\mathbb{F}_q$, n variables, and m polynomials. Table 1 lists the numerical values representing the success rates of the algorithm in Section 5.1. We conducted 100 trials for each cell. The parameter e was varied from 0 to 3. For $e = 0$, the number of computed S-polynomials was set as the output of the algorithm proposed in Section 4. The experiments used the fields $\mathbb{F}_{31}$ and $\mathbb{F}_{11}$, and random MQ problems were generated with $m = n+1$. The value of n was varied from 8 to 14, and the corresponding upper degrees d_u for each n are listed in the table.

By individually examining each problem, we observe that the success rate of the algorithm was relatively low when $e = 0$. This is because, when computing the selected pairs with the proposed algorithm, some S-polynomials become zero reductions. Furthermore, the success rate of the proposed algorithm decreased as the number of variables n increased. As n increases, the upper degree d_u also increases, leading to an increased number of degrees to compute. This, in turn, increases the number of trial attempts for the algorithm's selected pairs, which raises the likelihood of failure. When considering the choice of finite fields, there is a trend in which problems defined over $\mathbb{F}_{11}$ have a lower success rate than those defined over $\mathbb{F}_{31}$. This difference is attributed to the fact that the problems considered are defined over finite fields, whereas the theoretical framework established in previous sections primarily applies to infinite fields.

However, for all the aforementioned problems, it is possible to achieve a success rate close to 100% with the proposed algorithm by sufficiently increasing the parameter e. As demonstrated in the following section, increasing the number of computed S-polynomials has a small impact on the computational

Table 1: Success probability of the proposed algorithm

	n	d_u	e			
			0	1	2	3
$\mathrm{MQ}(\mathbb{F}_{11}, n, n+1)$	8	5	92%	100%	100%	100%
	9	5	91%	99%	100%	100%
	10	6	80%	97%	100%	100%
	11	6	85%	97%	100%	100%
	12	7	76%	98%	98%	100%
	13	7	61%	93%	100%	100%
	14	8	59%	97%	99%	100%
$\mathrm{MQ}(\mathbb{F}_{31}, n, n+1)$	8	5	99%	100%	100%	100%
	9	5	98%	100%	100%	100%
	10	6	83%	100%	100%	100%
	11	6	96%	99%	100%	100%
	12	7	87%	100%	100%	100%
	13	7	89%	99%	100%	100%
	14	8	86%	99%	100%	100%

complexity. Therefore, it has been confirmed that the proposed algorithm can perform Gröbner basis computations with a significantly high success rate while reducing the calculation of S-polynomials that result in zero reductions.

5.3 Comparison of the Numbers of Computed S-polynomials

The results of the comparison between F4 and the proposed algorithm for the number of computed S-polynomials at the highest degree, which is degree d_u, are presented in Table 2. The number of S-polynomials computed using F4 is calculated based on the results of the V2.27-9 implementation in Magma. For the proposed algorithm, the number of S-polynomials computed is equal to the number of Gröbner bases of degree d_u, which corresponds to the output of Algorithm 1 (N_{d_u}). When considering the value of e, the number of S-polynomials computed using the proposed algorithm is represented as $N_{d_u} + e$. The ratio is obtained by dividing the "Number of S-polynomials computed at degree d_u in F4" by the "Number of Gröbner bases of degree d_u." The proposed algorithm computes significantly fewer S-polynomials than F4. Because the calculation of Gröbner bases is dominated by the reduction of S-polynomials in the algorithm, reducing the number of computed S-polynomials significantly decreases the computational complexity. Moreover, as n increases, the ratio in the table consistently increases. Thus, the proposed algorithm demonstrates that as problems become more difficult, the computational complexity significantly

Table 2: Comparison of Numbers of Computed S-Polynomials

		The number of S-polynomials computed at degree d_u in F4	The number of a Gröbner basis of degree d_u (N_{d_u})	Ratio
n	8	178	42	4.23
	9	259	63	4.11
	10	737	132	5.58
	11	1059	187	5.66
	12	3003	429	7.00
	13	4014	572	7.01
	14	11804	1430	8.25

decreases compared to F4, allowing for faster computations. Furthermore, to achieve a success rate close to 100% for the proposed algorithm, increasing the parameter e is necessary. However, if e is appropriately chosen, the impact of increased e on the computational complexity is expected to be small.

5.4 New Record in the Fukuoka MQ Challenge

To evaluate the hardness of Multivariate Quadratic (MQ) problems, the Fukuoka MQ Challenge has been held since 2015[4]. The problems are classified into Types I to VI, depending on the choice of the finite field $\mathbb{F}_q$, number of variables n, and number of equations m. In this work, we tackled the Type VI problem with $\mathbb{F}_{31}$ as the finite field and the relation $n \sim 1.5m$. The previous record for Type VI had $n = 30$ variables and $m = 20$ equations. The reported solving time was approximately 11 days using M4GB[13] on a system with 2× Intel Xeon CPU E5-2650 v3 and 256 GB of RAM (the number of trials for the hybrid approach[12] was not disclosed). In our decryption experiment, we focused on a Type VI problem with $n = 31$ variables and $m = 21$ equations. We implemented the proposed algorithm in C++ and ran it on a machine with an AMD EPYC 7742 CPU and 2 TB of RAM. To achieve a 100% success rate, as indicated in Table 1, we chose to compute a slightly larger number of S-polynomials by setting $e = 5$. The implementation targeted dense matrices over $\mathbb{F}_{31}$ and achieved substantial acceleration through parallel computing in both matrix multiplication and row echelon form. In addition, we attempted to solve the MQ problem with $n = 20$ variables and $m = 21$ equations in a Type VI instance using a hybrid approach. Due to the memory limitations of the experimental environment, we executed four simultaneous Gröbner basis computations. Consequently, we successfully obtained the solution for seed 0 in the

Type VI MQ Challenge, breaking the previous record in approximately 9 h.

$$\begin{aligned}&(4, 1, 15, 29, 9, 26, 9, 25, 1, 14, 0, 8, 12, 21, 10, 2, 1, 30,\\&\quad 21, 13, 24, 15, 15, 17, 16, 28, 18, 15, 5, 13, 24) \in \mathbb{F}_{31}^{31}\end{aligned}$$

6 Conclusion

In this study, we discuss the difficulty of the MQ problem, which underlies the security of multivariate polynomial cryptography. We proposed a method to minimize the number of S-polynomials of degree d to be computed when solving the MQ problem over finite fields. To achieve this, we provided evidence that the Hilbert-driven algorithm is applicable to both non-homogeneous semi-regular sequences up to the upper degree and non-homogeneous regular sequences at all degrees without modifications. Furthermore, we proposed an algorithm to compute the number of elements of Gröbner bases at each degree d for both sequence types under the assumption of Moreno-Socias's conjecture. Using our proposed algorithm, we successfully decrypted a Type VI problem with $m = 21$ equations, achieving a new record in the Fukuoka MQ Challenge in approximately 8 h using an AMD EPYC 7742 processor and 2 TB of RAM.

Future challenges include developing a selection method for non-zero-reduction S-polynomials for degrees $d = 3, 4$ over finite fields, establishing a method to select non-zero-reduction S-polynomials for general semi-regular problems, theoretical complexity evaluations of the MQ problem using our proposed algorithm, and further updates to Type VI and other Type problems in the MQ Challenge and others.

Acknowledgement

This research was conducted under a contract of "Research and development on new generation cryptography for secure wireless communication services" among "Research and Development for Expansion of Radio Wave Resources(JPJ000254)", which was supported by the Ministry of Internal Affairs and Communications, Japan.

References

[1] Peter W. Shor. "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer", SIAM Jounal on Computing, 26, issue 5, 1484–1509, 1997.

[2] Faugère J.-C., "A new efficient algorithm for computing Gröbner Bases (F4)", Journal of Pure and Applied Algebra, 139, 6–88, 1999.

[3] Traverso C., “Hilbert functions and the buchberger algorithm”, Journal of Symbolic Computation, 22(4), 355-376, 1996.

[4] Yasuda T., Dahan X., Huang Y-J., Takagi T., Sakurai K., “MQ Challenge: Hardness Evaluation of Solving Multivariate Quadratic Problems”, NIST Workshop on Cybersecurity in a Post-Quantum World, 2015. https://www.mqchallenge.org/

[5] Buchberger, B., “Ein Algorithmus zum Auffinden der basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal”, PhD thesis, Universität Innsbruck, 1965.

[6] Gebauer R., Möller H. M., “On an installation of Buchberger’s algorithm”, Journal of Symbolic Computation, 6, 275-286, 1988.

[7] Faugère J.-C., “A new efficient algorithm for computing Gröbner bases without reduction to zero (F5)”, In Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation, 75–83, 2002.

[8] Cox D., Little J., O’Shea D., “Ideals, Varieties, and Algorithms ”, Springer-Verlag, New York, 1997.

[9] Pardue K., “Generic sequences of polynomials”, Journal of Algebra, 324(4), 579-590, 2010.

[10] Moreno-Socías G., “Autour de la fonction de Hilbert-Samuel (escaliers d’idéaux polynomiaux)”, Thèse, École Polytechnique, 1991.

[11] Bardet M., Faugére J.-C., Salvy B., Yang B-Y., “Asymptotic behaviour of the degree of regularity of semi-regular polynomial systems”, IN Proceedings of MEGA, 8th International Symposium on Effective Method in Algebraic Geometry, 2006.

[12] Bettale L., Faugère J.C., Perret L., “Hybrid approach for solving multivariate systems over finite fields”, Journal of Mathematical Cryptology, 3, issue 3, 177–197, 2009.

[13] Makarim R. H., Stevens M. , “M4GB: An efficient Grb̈ner-basis algorithm”, In Proceedings of the 2017 International Symposium on Symbolic and Algebraic Computation, 293–300, 2017.

[14] Ito T., Shinohara N., Uchiyama S., “An efficient F4-style based algorithm to solve MQ problems”, in: Proc. of 14th International Workshop on Security 2019, 37–52, 2019.

A Improving the Efficiency of Algorithm 1

Here, we describe an efficient computation method for **Algorithm 1**. B_d is a set of monomials that cannot be divided by $\mathrm{LT}(G_{d-1}^{top})$; therefore, it is not necessary to calculate all M_d. Efficient computation is possible using the following approach:

$$M'_d \leftarrow \{r \cdot m \mid r \in M_1, m \in B_{d-1}\}$$
$$B_d \leftarrow \{m \in M'_d \mid m \notin \langle LT_G \rangle\}$$

where M'_d is smaller than M_d, which relaxes the divisibility checks. In addition, the elements in $M_d \setminus M'_d$ are included in $\langle LT_G \rangle$.

B Algorithm 1 for a Regular Sequence

When we compute a regular sequence, it may not have upper degree because the Hilbert series is a infinte series. In that case, In this case, the following algorithm is useful.

Algorithm 2 Calculating the number of Gröbner bases for each degree d

```
Require: A regular homogeneous sequence, $F = \{f_1, \ldots, f_m\}$
Ensure: The number of a Gröbner bases for each degree $d$, $\{N_d\}_{0 \le d \le d_u}$
 1: $H_{R/\langle F \rangle}(z) \leftarrow \frac{\prod_{i=1}^{m}(1-z^{d_i})}{(1-z)^n}, (d_i = \deg(f_i))$
 2: $LT_G \leftarrow \varnothing$
 3: $d \leftarrow 0$
 4: while $H_{R/I}(z) \neq H_{R/\langle \mathrm{LT}(F) \rangle}$ do
 5:     $B_d \leftarrow \{m \in M_d \mid m \notin \langle LT_G \rangle\}$
 6:     $N_d \leftarrow \#B_d - h_{R/\langle F \rangle}(d)$
 7:     $LT_G \leftarrow LT_G \cup \{$top $N_d$ elements from $B_d$ in descending order$\}$
 8:     $d \leftarrow d + 1$
 9: end while
10: return $\{N_d\}_{0 \le d \le d_u}$
```