Fast Enumeration Algorithm for Multivariate Polynomials over General Finite Fields

Hiroki Furue and Tsuyoshi Takagi

Department of Mathematical Informatics, The University of Tokyo, Tokyo, Japan
{furue-hiroki261,takagi}@g.ecc.u-tokyo.ac.jp

Abstract. The enumeration of all outputs of a given multivariate polynomial is a fundamental mathematical problem and is incorporated in some algebraic attacks on multivariate public key cryptosystems. For a degree-d polynomial in n variables over the finite field with q elements, solving the enumeration problem classically requires $O\left(\binom{n+d}{d} \cdot q^n\right)$ operations. At CHES 2010, Bouillaguet et al. proposed a fast enumeration algorithm over the binary field $\mathbb{F}_2$. Their proposed algorithm covers all the inputs of a given polynomial following the order of Gray codes and is completed by $O(d \cdot 2^n)$ bit operations. However, to the best of our knowledge, a result achieving the equivalent efficiency in general finite fields is yet to be proposed.
In this study, we propose a novel algorithm that enumerates all the outputs of a degree-d polynomial in n variables over $\mathbb{F}_q$ with a prime number q by $O(d \cdot q^n)$ operations. The proposed algorithm is constructed by using a lexicographic order instead of Gray codes to cover all the inputs. This result can be seen as an extension of the result of Bouillaguet et al. to general finite fields and is almost optimal in terms of time complexity. We can naturally apply the proposed algorithm to the case where q is a prime power. Notably, our enumeration algorithm differs from the algorithm by Bouillaguet et al. even in the case of $q = 2$.

Keywords: multivariate polynomial, finite fields, enumeration algorithm, exhaustive search, MQ problem, MPKC

1 Introduction

Currently used public key cryptosystems such as RSA and ECC can be broken in polynomial time by Shor's algorithm [15] using a quantum computer. Thus, the amount of research conducted on post-quantum cryptography (PQC), which is secure against quantum computing attacks, has been accelerating. Indeed, the U.S. National Institute for Standards and Technology (NIST) has initiated a PQC standardization project [14]. Among various PQC candidates, multivariate public key cryptosystem (MPKC) is one of the main categories. MPKCs are cryptosystems constructed based on the difficulty of solving a system of multivariate quadratic polynomial equations over a finite field (the multivariate quadratic (MQ) problem). The MQ problem is proven to be NP-complete [11] and is thus likely to be secure against quantum computers.

The security of MPKCs is strongly dependent on the difficulty of solving some algebraic problems in addition to the MQ problem. Especially, there have been proposed many key recovery attacks on MPKCs solving the MinRank problem [1, 3, 12, 16], which finds one low-rank matrix from linear combinations of given matrices. An *enumeration problem* is also one of the algebraic problems relevant to the security of MPKCs. The enumeration problem is defined over the finite field $\mathbb{F}_q$ with a prime power q as follows: Given a single-degree-d polynomial f in $\mathbb{F}_q[x_1, \ldots, x_n]$, evaluate f over all the points in $\mathbb{F}_q^n$ (to find all the zeros of f). Indeed, there exist many algebraic attacks partly using enumeration algorithms, such as the claw finding attack, Crossbred algorithm [13], and polynomial XL [10]. Therefore, improving the complexity of solving the enumeration problem directly improves the complexity of some algebraic attacks on MPKCs and strongly affects the security of MPKCs. In the rest of this paper, we focus on the theoretical asymptotic complexity of algorithms for solving the enumeration problems.

Fast exhaustive search over $\mathbb{F}_2$. At CHES 2010, Bouillaguet et al. [7] proposed a fast enumeration algorithm in $\mathbb{F}_2$ and a way of solving non-linear systems using this enumeration algorithm. Given a single-degree-d polynomial in n variables over $\mathbb{F}_2$, their enumeration algorithm requires $O(d \cdot 2^n)$ bit operations. This complexity is smaller than that of the classical exhaustive search $O\left(\binom{n}{d} \cdot 2^n\right)$. In their algorithms, all the inputs are covered in the order of Gray codes. Here, Gray codes are orderings of the elements of $\mathbb{F}_2^n$ such that two consecutive elements differ in only one bit. For consecutive two elements $\mathbf{x}_0$ and $\mathbf{x}_1$ in the order of Gray codes, which only differs in the k-th bit, their enumeration algorithm computes the output $f(\mathbf{x}_1)$ as follows using the derivative $\partial_k f$ with respect to the k-th variable

$$f(\mathbf{x}_1) = f(\mathbf{x}_0) + \partial_k f(\mathbf{x}_0),$$

where $+$ denotes addition in the binary field. By using this method recursively, outputs can be updated by $O(d)$ bit operations. Efficient implementations of their proposed algorithms are given in [6, 8].

Our study aims to extend the results of Bouillaguet et al. to general finite fields. In [7], their enumeration algorithm uses one property of polynomials over $\mathbb{F}_2$ such that derivatives $\partial_k f$ do not include the variable x_k because polynomials over $\mathbb{F}_2$ are represented as a sum of monomials where the exponent of each variable is at most one (due to $x_i^2 = x_i$). Such a property does not hold over general finite fields, and this renders naturally extending the results of Bouillaguet et al. to general finite fields difficult.

Our Contributions. We propose a novel efficient enumeration algorithm over general finite fields $\mathbb{F}_q$ with a prime number q. Given a single-degree-d polynomial in n variables over $\mathbb{F}_q$, the proposed algorithm enumerates all the outputs of the given polynomial by $O(d \cdot q^n)$ operations with an initialization phase. This result achieves efficiency comparable to the enumeration algorithm proposed by

Bouillaguet et al. Furthermore, we also show a method of applying the proposed algorithm to the case of $\mathbb{F}_{p^r}$ with a prime number p and a positive integer r in Remark 3.

From a theoretical point of view, the main difference between the proposed enumeration algorithm and one of Bouillaguet et al. is the order to cover all the inputs $\mathbb{F}_q^n$. In the proposed algorithm, an order like a lexicographic one is used instead of Gray codes. For example, in the case of $q = 3$ and $n = 2$, all the inputs are ordered as follows: $(0,0) \to (0,1) \to (0,2) \to (1,0) \to (1,1) \to (1,2) \to (2,0) \to (2,1) \to (2,2)$. However, unlike the algorithm proposed by Bouillaguet et al., we compute not all the outputs by adding some derivatives into the output to the last input. In the proposed algorithm, if an input has the form $(x_1, \ldots, x_k, 0, \ldots, 0)$ with $x_k \neq 0$, then the output to $(x_1, \ldots, x_k, 0, \ldots, 0)$ is computed as follows:

$$\begin{aligned} &f(x_1, \ldots, x_k, 0, \ldots, 0) = \\ &f(x_1, \ldots, x_k - 1, 0, \ldots, 0) + \partial_k f(x_1, \ldots, x_k - 1, 0, \ldots, 0). \end{aligned}$$

The proposed algorithm following this rule covers all the inputs by a branching structure, unlike the algorithm proposed by Bouillaguet et al. (See Figure 1.) We then can achieve the time complexity limited to a small value as that proposed by Bouillaguet et al. Note that our enumeration algorithm differs from the algorithm proposed by Bouillaguet et al. even in the case of $q = 2$.

From a practical point of view, as we mentioned above, our enumeration algorithm can revise the complexity of attacks including the enumeration part, such as the claw finding attack, Crossbred, and polynomial XL, in finite fields with odd characteristics. This has an impact on MPKCs since some multivariate-based signature schemes in finite fields with odd characteristics have been proposed recently [2, 9]. Furthermore, by using the method proposed in [7], our enumeration algorithm can be applied to solve systems of polynomial equations, and its complexity is estimated as $O\left(d^2 \cdot \log n \cdot q^n\right)$. (See Remark 4.)

We finally discuss the optimality of the proposed enumeration algorithm in terms of time complexity. The lower bound of the complexity of the exhaustive search is conjectured to be $O\left(q^n\right)$ because outputs have to be computed q^n times for all the inputs in $\mathbb{F}_q^n$. Therefore, in the case of $d \ll n$, the proposed enumeration algorithm with $O\left(d \cdot q^n\right)$ operations can be considered to be almost optimal in terms of time complexity.

Organizations. Section 2 reviews a classical enumeration algorithm and its complexity. After describing the enumeration algorithm proposed by Bouillaguet et al. in Section 3, the proposed algorithm is detailed in Section 4. Finally, Section 5 presents the conclusion, which summarizes the key points and suggests possible future directions.

2 Classical Approach

Let q be a prime power, and n and d be positive integers. This section deals with the enumeration problem on a polynomial $f \in \mathbb{F}_q[x_1, \ldots, x_n]$ with degree d. The conventional method to solve the enumeration problem is to evaluate the given polynomial f at all the points of $\mathbb{F}_q^n$. This section explains the conventional approach and its complexity.

Given a polynomial f in n variables with degree d, f can be decomposed as follows:

$$f(x_1, \ldots, x_n) = x_1 \cdot f_1(x_1, \ldots, x_n) + f_2(x_2, \ldots, x_n), \tag{2.1}$$

where the degrees of $f_1(x_1, \ldots, x_n)$ and $f_2(x_2, \ldots, x_n)$ are at most $d-1$ and d, respectively. Then, the output $f(x_1, \ldots, x_n)$ can be obtained by evaluating $f_1(x_1, \ldots, x_n)$ and $f_2(x_2, \ldots, x_n)$. Therefore, the enumeration problem in n variables with degree d can be reduced to the problem in n variables with degree $d-1$ and the problem in $n-1$ variables with degree d. From the discussion, the following theorem can be explained for evaluating complexity.

Lemma 1. *Let f be a polynomial in n variables with degree d over $\mathbb{F}_q$. Denoting by $T(n,d)$ the upper bound of the number of additions and multiplications over $\mathbb{F}_q$ for the evaluation of f for any input with the aforementioned approach, we have*

$$T(n,d) = 2 \cdot \left(\binom{n+d}{d} - 1 \right). \tag{2.2}$$

Proof. The aforementioned statement can be proved by induction. We have $T(n,1) = 2n$ and $T(1,d) = 2d$, which satisfy the aforementioned statement. We assume that, for $n, d \geq 2$, $T(n,d-1)$ and $T(n-1,d)$ satisfy equation (2.2). Since we have $T(n,d) = T(n,d-1) + T(n-1,d) + 2$ from equation (2.1), the following equation can be obtained:

$$\begin{aligned} T(n,d) &= 2 \cdot \left(\binom{n+d-1}{d-1} - 1 \right) + 2 \cdot \left(\binom{n-1+d}{d} - 1 \right) + 2 \\ &= 2 \cdot \left(\binom{n+d-1}{d-1} + \binom{n+d-1}{d} - 1 \right) \\ &= 2 \cdot \left(\binom{n+d}{d} - 1 \right). \end{aligned}$$

Thus, we confirmed $T(n,d)$ satisfies equation (2.2), and the aforementioned statement holds for any n and d. □

From this lemma, the time complexity of the evaluation of f at each point of $\mathbb{F}_q^n$ is given as $O\left(\binom{n+d}{d} \right)$. Therefore, the complexity of the classical approach to the enumeration problem is given as follows:

$$O\left(\binom{n+d}{d} \cdot q^n \right).$$

3 Enumeration Algorithm of Bouillaguet et al.

This section describes an enumeration algorithm over the binary field proposed by Bouillaguet et al. [7]. After presenting some notations, we describe their enumeration algorithm in Subsection 3.2. We change some notations from [7] for readability and consistency with the description of the proposed algorithm in Section 4.

3.1 Notations

We here give some notations about the vector space over the binary field, *Gray codes*, and *Derivatives*. These Gray codes and derivatives play crucial roles in their enumeration algorithm given in Subsection 3.2,

Binary vector. For the n-dimensional vector space $\mathbb{F}_2^n$ over the binary field, the indices are allocated from 1 to n from the most left bit to the most right bit such as $(x_1, \ldots, x_n)$. For a vector $\mathbf{a} \in \mathbb{F}_2^n$ and two integers $0 \leq i \leq 2^n - 1$ and $1 \leq k \leq n$, we use the following notations:

- $i_{(2)}$: an n-dimensional vector over $\mathbb{F}_2$ representing i in base-2
- $\mathbf{e}_k$: the k-th canonical basis in $\mathbb{F}_2^n$
- $\mathbf{a} \ll k$ (resp. $\mathbf{a} \gg k$): the binary left (resp. right) shift of a vector $\mathbf{a}$ by k bits.
- $\rho(\mathbf{a})$ (resp. $\sigma(\mathbf{a})$): the index of the most left (resp. right) nonzero bit of $\mathbf{a}$ (If $\mathbf{a} = \mathbf{0}$, then $\rho(\mathbf{a}) = \sigma(\mathbf{a}) = 0$.)

Gray codes. The Gray code is an ordering of the binary vector space such that two successive values differ in only one bit. For the vector space $\mathbb{F}_2^n$, several orderings satisfy the aforementioned condition. However, in this study, we defined the Gray code $\mathrm{GC}(i)$ for $0 \leq i \leq 2^n - 1$ uniquely as follows:

$$\mathrm{GC}(i) = i_{(2)} + (i_{(2)} \gg 1). \tag{3.1}$$

Then, it can be easily confirmed that $\mathrm{GC}(i)$ and $\mathrm{GC}(i+1)$ differ in only one bit.

Derivatives. Finally, for an integer $1 \leq k \leq n$, $\mathbb{F}_2$ derivative $\partial_k f$ is defined as follows:

$$\partial_k f(\mathbf{x}) = f(\mathbf{x} + \mathbf{e}_k) + f(\mathbf{x}).$$

We can easily confirm that if the degree of f is d, then that of $\partial_k f$ is at most $d - 1$.

3.2 Enumeration Algorithm

This subsection recalls an enumeration algorithm proposed in [7]. The input f is a polynomial in $\mathbb{F}_2[x_1, \ldots, x_n]$ with degree d.

First, a method of storing some information through the enumeration is given. For any $\mathbf{a} = (a_1, \ldots, a_n) \in \mathbb{Z}_{\geq 0}^n$ with $|\mathbf{a}| = \sum_{i=1}^n a_i \leq d$, $D[\mathbf{a}]$ contains the information of the derivative $\partial^{\mathbf{a}} f = \prod_{i=1}^n \partial_i^{a_i} f$. Here, $\mathbf{a}$ is restricted by $|\mathbf{a}| \leq d$ because $\partial^{\mathbf{a}} f = 0$ for any $\mathbf{a}$ with $|\mathbf{a}| > d$. In their enumeration algorithm, three values can be read from $D[\mathbf{a}]$: $D[\mathbf{a}].\mathbf{x} \in \mathbb{F}_2^n$, $D[\mathbf{a}].y \in \mathbb{F}_2$, and $D[\mathbf{a}].i \in \mathbb{Z}_{\geq 0}$. These invariants satisfy relationships that $D[\mathbf{a}].y = \partial^{\mathbf{a}} f(D[\mathbf{a}].\mathbf{x})$ and $D[\mathbf{a}].i$ is used to update $D[\mathbf{a}].\mathbf{x}$.

The enumeration algorithm is composed of three functions: INIT, MAIN, and NEXT. The main function MAIN first performs INIT and sets the initial values of each derivative. After the initialization phase, MAIN derives the outputs of f from D and uses NEXT to update D at each point. See Algorithm 1, 2, and 3 for details.

The main idea of this algorithm is to update the input $\mathbf{x}$ of f following Gray codes and obtain $f(\mathbf{x} + \mathbf{e}_i)$ by adding a derivative $\partial_i f(\mathbf{x})$ into $f(\mathbf{x})$. Such a construction is feasible because two successive values differ in only one bit in the Gray code order. In the following, we show a way of evaluating $\partial_i f$ for some inputs to update f. From the definition of the Gray code in equation (3.1), we have the following equation:

$$\mathrm{GC}(i+1) = \mathrm{GC}(i) + \mathbf{e}_{\sigma((i+1)_{(2)})}.$$

From this equation, $\mathrm{GC}(i+1)$ is equal to $\mathrm{GC}(i) + \mathbf{e}_k$ in the case where $i = j \cdot 2^{n-k+1} + 2^{n-k} - 1$ with $0 \leq j \leq 2^{k-1} - 1$. Namely, the enumeration algorithm requires the output of $\partial_k f$ at each point $\left(j \cdot 2^{n-k+1} + 2^{n-k} - 1\right)_{(2)}$ for $0 \leq j \leq 2^{k-1} - 1$. The following lemma can be easily derived from Lemma 3 in [7]

Lemma 2. *For* $0 \leq j \leq 2^{k-1} - 1$,

$$\mathrm{GC}\left(j \cdot 2^{n-k+1} + 2^{n-k} - 1\right) = \begin{cases} \mathrm{GC}(2^{n-k} - 1) + (\mathrm{GC}(j) \ll (n-k+1)) & (j \text{ is even}) \\ \mathrm{GC}(2^{n-k} - 1) + (\mathrm{GC}(j) \ll (n-k+1)) + \mathbf{e}_k & (j \text{ is odd}) \end{cases}.$$

From this lemma, the first $(k-1)$ bits of $\mathrm{GC}\left(j \cdot 2^{n-k+1} + 2^{n-k} - 1\right)$ for $0 \leq j \leq 2^{k-1} - 1$ behaves like Gray codes in $(k-1)$ variables, and the last $(n-k)$ bits of them are constants. The k-th bit of $\mathrm{GC}\left(j \cdot 2^{n-k+1} + 2^{n-k} - 1\right)$ is determined by the parity of j. However, the k-th bit does not affect the value of $\partial_k f\left(\mathrm{GC}\left(j \cdot 2^{n-k+1} + 2^{n-k} - 1\right)\right)$ because we have $\partial_k f(\mathbf{x} + \mathbf{e}_k) = \partial_k f(\mathbf{x})$ for any $\mathbf{x}$. Therefore, $\partial_k f\left(\mathrm{GC}\left(j \cdot 2^{n-k+1} + 2^{n-k} - 1\right)\right)$ for $0 \leq j \leq 2^{k-1} - 1$ can be enumerated in the order of Gray codes in the first $(k-1)$ variables. By applying this reduction recursively, the enumeration in the Gray code order can be realized for any $\partial^{\mathbf{a}} f$ with $|\mathbf{a}| \leq d$.

Following Theorem 1 in [7], the time and space complexities of the enumeration algorithm of Bouillaguet et al. are estimated as follows:

Theorem 1. *All the zeros of a single polynomial f in n variables with degree d over the binary field can be found in essentially $O\left(d \cdot 2^n\right)$ bit operations, using $O\left(n^d\right)$ bits memory, after an initialization phase of negligible complexity $O\left(n^{2d}\right)$.*

Algorithm 1 MAIN(f)

$D \leftarrow \mathrm{INIT}(D, f, \mathbf{0}, \mathbf{0})$
for $i = 0, \ldots, 2^n - 1$ **do**
  "$f\,(D[\mathbf{0}].\mathbf{x}) = D[\mathbf{0}].y$"
  $D \leftarrow \mathrm{NEXT}(D, \mathbf{0})$
end for

Algorithm 2 INIT($D, f, \mathbf{a}, \mathbf{x}_0$)

$D[\mathbf{a}].i \leftarrow 0$
$D[\mathbf{a}].\mathbf{x} \leftarrow \mathbf{x}_0$
$D[\mathbf{a}].y \leftarrow f(\mathbf{x}_0)$
if $|\mathbf{a}| < d$ **then**
  $k_0 \leftarrow \rho(\mathbf{a})$ ($k_0 \leftarrow n + 1$ if $\mathbf{a} = \mathbf{0}$.)
  for $k = 1, \ldots, k_0 - 1$ **do**
    $D \leftarrow \mathrm{INIT}(D, \partial_k f, \mathbf{a} + \mathbf{e}_k, \mathbf{x}_0 + \mathbf{e}_{k+1})$ ($\mathbf{e}_{n+1} = \mathbf{0}$)
  end for
end if
return D

Algorithm 3 NEXT($D, \mathbf{a}$)

$D[\mathbf{a}].i \leftarrow D[\mathbf{a}].i + 1$
$k \leftarrow \rho(\mathbf{a}) + \sigma((D[\mathbf{a}].i)_{(2)}) - n - 1$
$D[\mathbf{a}].\mathbf{x} \leftarrow D[\mathbf{a}].\mathbf{x} + \mathbf{e}_k$
$D[\mathbf{a}].y \leftarrow D[\mathbf{a}].y + D[\mathbf{a} + \mathbf{e}_k].y$
if $|\mathbf{a}| < d - 1$ **then**
  $D \leftarrow \mathrm{NEXT}(D, \mathbf{a} + \mathbf{e}_k)$
end if
return D

Remark 1 (Case of $\mathbb{F}_q$ with a prime number $q \neq 2$). It is critical to consider the extension of the enumeration algorithm in Subsection 3.2 to general finite fields $\mathbb{F}_q$ with a prime number $q \neq 2$. Indeed, there exist Gray codes over $\mathbb{F}_q$ called q-array Gray codes, and derivatives are feasible over $\mathbb{F}_q$. However, the enumeration algorithm of Bouillaguet et al. cannot be simply extended to $\mathbb{F}_q$.

Subsection 3.2 reveals that some outputs of $\partial_k f$ required to enumerate f can be enumerated in the order of Gray codes. A similar result as Lemma 2 holds in $\mathbb{F}_q$ with $q \neq 2$. However, it does not hold that $\partial_k f(\mathbf{x} + \mathbf{e}_k) = \partial_k f(\mathbf{x})$ in $\mathbb{F}_q$, and thus, some required outputs of $\partial_k f$ cannot be enumerated using Gray codes in the first $(k-1)$ variables. Therefore, applying the proposed enumeration recursively for a polynomial in $\mathbb{F}_q$ is difficult.

4 Our Proposed Algorithms

This section proposes a novel efficient enumeration algorithm on a degree-d polynomial $f \in \mathbb{F}_q[x_1, \ldots, x_n]$ with a prime number q. This section is organized as follows: Subsection 4.1 prepares some notations. Subsection 4.2 explains the order used in the proposed algorithm instead of Gray codes. Subsection 4.3 introduces the data structure used in the proposed algorithm. Subsection 4.4 prepares a function to classify the inputs used in the main algorithm. Subsection 4.5 describes the details of the proposed algorithm. Subsection 4.6 discusses the time and space complexities.

4.1 Notations

This subsection gives some notations for the proposed enumeration algorithm over $\mathbb{F}_q$ as in Subsection 3.1 for the enumeration algorithm of Bouillaguet et al.

For the n-dimensional vector space over the finite field $\mathbb{F}_q$, the indices are allocated from 1 to n from the most left bit to the most right bit as described in Section 3. For a vector $\mathbf{a} \in \mathbb{F}_q^n$ and two integers $0 \leq i \leq q^n - 1$ and $1 \leq k \leq n$, we use the following notations

- $\mathbf{e}_k$: the k-th canonical basis in $\mathbb{F}_q^n$
- $\rho(\mathbf{a})$: the index of the most left nonzero bit of $\mathbf{a}$ (If $\mathbf{a} = \mathbf{0}$, then $\rho(\mathbf{a}) = n$.)

Furthermore, derivatives over $\mathbb{F}_q$ are defined as follows: For an integer $1 \leq k \leq n$, the $\mathbb{F}_q$ derivative $\partial_k f(\mathbf{x})$ for $\mathbf{x} \in \mathbb{F}_q^n$ is

$$\partial_k f(\mathbf{x}) = f(\mathbf{x} + \mathbf{e}_k) - f(\mathbf{x}).$$

For example, given $f(x_1, x_2) = 2x_1^2 + x_1x_2 + x_1 + 2 \in \mathbb{F}_3[x_1, x_2]$, it holds $\partial_1 f(x_1, x_2) = x_1 + x_2$. As in the case of $\mathbb{F}_2$, if the degree of f is d, then that of $\partial_k f$ is at most $d - 1$.

4.2 Enumeration Order

We here introduce the order in all inputs $\mathbb{F}_q^n$ used to enumerate a given polynomial in the proposed algorithm. The enumeration of Bouillaguet et al. computes all the outputs of a polynomial consecutively in the order of Gray codes. On the other hand, the proposed algorithm follows a different order from Gray codes, like a lexicographic order.

The proposed algorithm enumerates all the inputs in $\mathbb{F}_q^n$ following the order like a lexicographic order starting from $(0,\ldots,0)$ to $(q-1,\ldots,q-1)$ as follows:

$$\begin{aligned}
&(0,\ldots,0) \to (0,\ldots,0,1) \to \cdots \to (0,\ldots,0,q-1) \to \\
&(0,\ldots,0,1,0) \to \cdots \to (0,\ldots,0,1,q-1) \to \\
&\quad\vdots \\
&(q-1,\ldots,q-1,0) \to \cdots \to (q-1,\ldots,q-1).
\end{aligned}$$

We then compute all the outputs in this order as follows: We first prepare $f(\mathbf{0})$ as an initial point. For any input with the form of $(x_1,\ldots,x_k,0,\ldots,0)$ with $x_k \neq 0$, the output is computed by

$$\begin{aligned}
&f(x_1,\ldots,x_k,0,\ldots,0) = \\
&f(x_1,\ldots,x_k-1,0,\ldots,0) + \partial_k f(x_1,\ldots,x_k-1,0,\ldots,0),
\end{aligned} \tag{4.1}$$

using derivatives $\partial_k f$. If we have the value of $\partial_k f(x_1,\ldots,x_k-1,0,\ldots,0)$, then this computation is clearly feasible since the value of $f(x_1,\ldots,x_k-1,0,\ldots,0)$ is computed before the computation of $f(x_1,\ldots,x_k,0,\ldots,0)$ in the lexicographic order. In the rest of this paper, we call by a lexicographic order the order given above including the way of computation with derivatives like equation (4.1).

We also use the lexicographic order to enumerate each derivative $\partial^{\mathbf{a}} f$ with $\mathbf{a} \in \mathbb{Z}_{\geq 0}^n$. To realize the enumeration of f, it is not necessary to evaluate $\partial^{\mathbf{a}} f$ at all the inputs in $\mathbb{F}_q^n$. Indeed, in the example on $f \in \mathbb{F}_3[x_1,x_2]$, $\partial_1 f$ and $\partial_2 f$ are only evaluated at $(0,0),(1,0)$ and $(0,0),(0,1),(1,0),(1,1),(2,0),(2,1)$, respectively. (See Figure 1.) For any $\mathbf{a} = (a_1,\ldots,a_n) \in \mathbb{Z}_{\geq 0}^n$ with $|\mathbf{a}| \leq d$ satisfying $a_1,\ldots,a_{\alpha-1} = 0$ and $a_\alpha \neq 0$, we then can define the subset $A_{\mathbf{a}} \subseteq \mathbb{F}_q^n$ composed by elements at which the proposed algorithm evaluates $\partial^{\mathbf{a}} f$ as follows:

$$A_{\mathbf{a}} = \{(x_1,\ldots,x_\alpha,0,\ldots,0) \mid x_1,\ldots,x_{\alpha-1} \in \mathbb{F}_q, x_\alpha \in \{0,\ldots,q-a_\alpha-1\}\}. \tag{4.2}$$

One can confirm the correctness of this definition of $A_{\mathbf{a}}$ from the following three points:

- For any $\mathbf{a} \in \mathbb{Z}_{\geq 0}^n$, $A_{\mathbf{a}}$ can be covered by the lexicographic order.
- $A_{\mathbf{0}} = \mathbb{F}_q^n$.
- For any $\mathbf{a} \in \mathbb{Z}_{\geq 0}^n$, $A_{\mathbf{a}+\mathbf{e}_k}$ is derived from $A_{\mathbf{a}}$.

We finally prepare some subsets of $A_{\mathbf{a}}$ to realize our enumeration in the lexicographic order. For any $\mathbf{a} = (a_1,\ldots,a_n) \in \mathbb{Z}_{\geq 0}^n$ with $|\mathbf{a}| \leq d$ satisfying

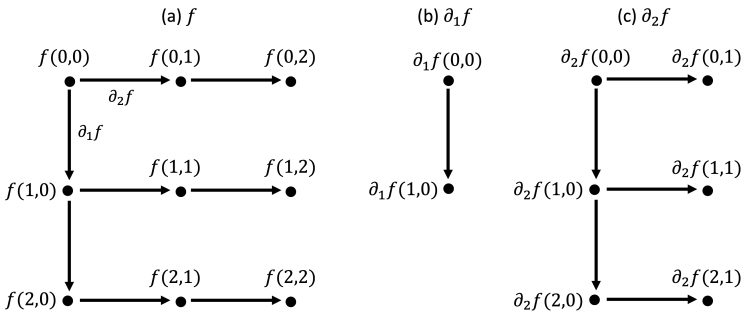

Fig. 1. The order of our proposed enumeration on a polynomial $f \in \mathbb{F}_3[x_1, x_2]$ with degree 2.

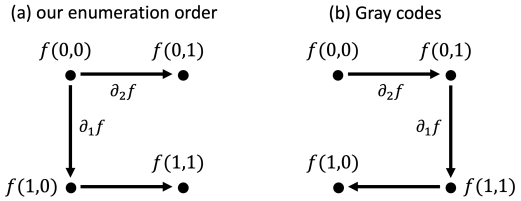

Fig. 2. Comparison of the order to cover all the outputs of $f \in \mathbb{F}_2[x_1, x_2]$ in our enumeration order and Gray codes

$a_1, \ldots, a_{\alpha-1} = 0$ and $a_\alpha \neq 0$, we define subsets $B_1^{\mathbf{a}}, \ldots, B_\alpha^{\mathbf{a}}$ of $A_{\mathbf{a}}$ as follows:

$$\begin{aligned} B_1^{\mathbf{a}} &= \{(x_1, 0, \ldots, 0) \mid x_1 \in \mathbb{F}_q\}, \\ B_i^{\mathbf{a}} &= \{(x_1, \ldots, x_i, 0, \ldots, 0) \mid x_1, \ldots, x_{i-1} \in \mathbb{F}_q, x_i \neq 0\}. \quad (2 \leq i \leq \alpha - 1), \\ B_\alpha^{\mathbf{a}} &= \{(x_1, \ldots, x_\alpha, 0, \ldots, 0) \mid x_1, \ldots, x_{\alpha-1} \in \mathbb{F}_q, x_\alpha \in \{1, \ldots, q - a_\alpha - 1\}\}. \end{aligned} \tag{4.3}$$

We then have $B_i^{\mathbf{a}} \cap B_j^{\mathbf{a}} = \phi$ if $i \neq j$ and $B_1^{\mathbf{a}} \cup \cdots \cup B_\alpha^{\mathbf{a}} = A_{\mathbf{a}}$. In the proposed algorithm given in Subsection 4.5 below, outputs for inputs in each $B_i^{\mathbf{a}}$ are stored separately to realize the branching structure due to the lexicographic order. (See Figure 1.)

Remark 2. For a polynomial over the binary field $\mathbb{F}_2$, this remark shows the difference of the behaviors of the proposed enumeration order in Subsection 4.2 and Gray codes used in the enumeration algorithm of Bouillaguet et al. Figure 2 shows the difference of enumerating all the outputs $f \in \mathbb{F}_2$ in two variables. As displayed in Figure 2, our enumeration order computes $f(0,1)$ and $f(1,0)$ by adding derivatives into $f(0,0)$ and computes $f(1,1)$ from $f(1,0)$. By contrast, the enumeration of Bouillaguet et al. computes all the outputs in the order $f(0,1)$, $f(0,1)$, $f(0,1)$, $f(0,1)$ successively. These examples indicate that the enumeration order and Gray codes are definitely different even in the case of the binary field.

4.3 Data Structure

Our proposed algorithm uses a different way of holding some data from the enumeration algorithm of Bouillaguet et al. For any $\mathbf{a} = (a_1, \ldots, a_n) \in \mathbb{Z}^n_{\geq 0}$ with $|\mathbf{a}| \leq d$, $D[\mathbf{a}]$ corresponding to the derivative $\partial^{\mathbf{a}} f$ stores following values

- α $(= \rho(\mathbf{a}))$: the index of the most left nonzero elements of $\mathbf{a}$ (If $\mathbf{a} = \mathbf{0}$, then $\alpha = n$.),
- $y_1, \ldots, y_\alpha \in \mathbb{F}_q$: an output of $\partial^{\mathbf{a}} f$ for an input as an element of each $B^{\mathbf{a}}_1, \ldots, B^{\mathbf{a}}_\alpha$,
- $i \in \{1, \ldots, \alpha\}$: the index that indicates that the input is an element of $B^{\mathbf{a}}_i$,
- $\mathbf{t} = (t_1, \ldots, t_n) \in \{0, \ldots, n-1\}^n$, $\mathbf{u} = (u_1, \ldots, u_n) \in \{0, \ldots, q-1\}^n$: two vectors used in the classification of elements of $A_{\mathbf{a}}$ into the subsets $B^{\mathbf{a}}_1, \ldots, B^{\mathbf{a}}_\alpha$.

Compared with the data structure used in the enumeration by Bouillaguet et al., we add some data to realize the enumeration in the lexicographic order given in Subsection 4.2. Note that these additional data only provide a small cost of memory complexity as discussed in Subsection 4.6 below.

4.4 Successive Classification of Inputs

We here prepare a subroutine CLASS used in the main algorithm described in Subsection 4.5 below. This CLASS classifies given elements of $A_{\mathbf{a}}$ into the subsets $B^{\mathbf{a}}_1, \ldots, B^{\mathbf{a}}_\alpha$ defined in equation (4.3) successively in the lexicographic order in constant time. This computation is mainly equivalent to the successive computation of the index of the most right nonzero bit for vectors in $\mathbb{F}_q^{n-1}$ in the lexicographic order due to the definition of $B^{\mathbf{a}}_1, \ldots, B^{\mathbf{a}}_\alpha$. The following algorithm is constructed by revising an algorithm for Gray codes proposed in [4].

We first describe a way of successively computing the index of the lowest nonzero bit for vectors in $\mathbb{F}_q^{n-1}$ using a stack as the data structure. A stack is an abstract data type with two main operations as follows: *push* which adds an element to the collection and *pop* which removes the most recently added element. For a stack and $u_1, \ldots, u_{n-1} \in \{0, \ldots, q-1\}$, the computation is realized as follows: The stack initially contains $1, \ldots, n-1$ with the $n-1$ on top and we set $u_1 = \cdots = u_{n-1} = 0$. Next, the top element i is popped off and added into the sequence of the index to be found. If $a_i < q-2$, then the elements $i, \ldots, n-1$ are pushed onto the stack and we increase u_i by one. If $a_i = q-2$, then the elements $i+1, \ldots, n-1$ are pushed onto the stack and we set $u_i = 0$. Comparing this algorithm with the algorithm proposed in [4], we introduce $u_1, \ldots, u_{n-1}$ to represent the structure of the base-q positional system.

In the above algorithm, the stack can be replaced by an array $t_1, \ldots, t_n \in \{0, \ldots, n-1\}$. For this array, t_j denotes the element below j on the stack if j is on the stack, t_j is set as $j-1$ if j is not on the stack, and t_n points to the top element of the stack. By following these rules, the way of updating the stack is changed as follows: If $u_i < q-2$, then we set $t_n = n-1$. On the other hand, in the case of $u_i = q-2$, we set $t_n = n-1$, $t_{i+1} = t_i$, and $t_i = i-1$ since i is removed from the top and $i+1, \ldots, n-1$ are pushed onto the top.

Algorithm 4 CLASS $(\mathbf{a} = (a_1, \ldots, a_n), \alpha, (t_1, \ldots, t_n), (u_1, \ldots, u_n))$

if $u_n < q - a_\alpha - 1$ **then**
  $i \leftarrow n$
  $u_n \leftarrow u_n + 1$
else
  $i \leftarrow t_n$
  if $\alpha + i - n = 0$ **then**
    return $0, (t_1, \ldots, t_n), (u_1, \ldots, u_n)$
  end if
  $u_n \leftarrow 0$
  $t_n \leftarrow n - 1$
  if $u_i < q - 2$ **then**
    $u_i \leftarrow u_i + 1$
  else
    $t_{i+1} \leftarrow t_i$
    $t_i \leftarrow i - 1$
    $u_i \leftarrow 0$
  end if
end if
return $\alpha + i - n, (t_1, \ldots, t_n), (u_1, \ldots, u_n)$

Then, our classification algorithm is constructed by combining the above algorithm with $t_1, \ldots, t_n$ and $u_1, \ldots, u_{n-1}$ and one counter $u_n \in \{0, \ldots, q-1\}$ to represent the structure of $A_{\mathbf{a}}$. In $A_{\mathbf{a}}$, only the $D[\mathbf{a}].\alpha$-th bit is carried up by $q - a_{D[\mathbf{a}].\alpha}$ from the definition, and u_n represents the value of this $D[\mathbf{a}].\alpha$-th bit. In the case where $u_n = q - a_{D[\mathbf{a}].\alpha} - 1$, we carry up the $D[\mathbf{a}].\alpha$-th bit, that is, we update $t_1, \ldots, t_n$ and $u_1, \ldots, u_{n-1}$ and set $u_n = 0$. By contrast, in the case where $u_n < q - a_{D[\mathbf{a}].\alpha} - 1$, we increase u_n by one and output $D[\mathbf{a}].\alpha$ as the index of $B_1^{\mathbf{a}}, \ldots, B_{D[\mathbf{a}].\alpha}^{\mathbf{a}}$.

Finally, we describe our classification algorithm CLASS. After two arrays $t_1, \ldots, t_n$ and $u_1, \ldots, u_n$ are initially set as $(0, 1, \ldots, n-1)$ and $(0, \ldots, 0)$, respectively, following the above discussion, these arrays are updated each time through CLASS. Other than these arrays, $\mathbf{a} = (a_1, \ldots, a_n)$ and α following the notation in Subsection 4.3 are included in the inputs. The outputs are given as the classification index of $B_i^{\mathbf{a}}$ and the updated two arrays. (See Algorithm 4 for more details.)

4.5 Our Enumeration Algorithm

This subsection describes the proposed efficient enumeration algorithm over finite fields $\mathbb{F}_q$. As in the enumeration of Bouillaguet et al. in Subsection 3.2, the proposed enumeration algorithm is composed of three functions, namely MAIN,

INIT, and NEXT. In the following, we will describe these three algorithms. See Algorithm 5, 6, and 7 for more details.

MAIN. From Algorithm 1, our main algorithm differs in the following two points: First, we cover all the inputs in $\mathbb{F}_q^n$ in the lexicographic order. Second, from the definition, $D[\mathbf{0}]$ holds n outputs $y_1, \ldots, y_n$ and we select one output from them as an output of f. The value $D[\mathbf{0}].i$ indicates that $y_{D[\mathbf{0}].i}$ is updated in the preceding NEXT as an output for an input $\mathbf{x}$. Thus, we take $y_{D[\mathbf{0}].i}$ as an output in line 3.

INIT. As in Algorithm 2, the function INIT sets initial values for $D[\mathbf{a}]$. Line 1 sets α following the aforementioned definition. Line 2-3 set initial values for two arrays $\mathbf{t}$ and $\mathbf{u}$ according to the description in Subsection 4.4. Because $\mathbf{0} \in B_1^{\mathbf{a}}$, we set $i = 1$ and store $f(\mathbf{0})$ in y_1. If $|\mathbf{a}| = d$, then we store $f(\mathbf{0})$ in $y_1, \ldots, y_\alpha$ for convenience. In the case where $|\mathbf{a}| < d$, the function proceeds to $\partial_1 f, \ldots, \partial_\alpha f$ in line 8, because, for any element of $A_{\mathbf{a}}$, the last $n - \alpha$ values are always zeros from the definition in (4.2).

NEXT. First, we show the basic strategy of the function NEXT. For any $\mathbf{a}$, we shift over an input $\mathbf{x}$ in $A_{\mathbf{a}}$ following the lexicographic order and store an output for the input in one of $y_1, \ldots, y_\alpha$ following the definition of $B_1^{\mathbf{a}}, \ldots, B_\alpha^{\mathbf{a}}$. The update of outputs of $\partial^{\mathbf{a}} f$ is performed according to the following equation

$$\begin{aligned} &\partial^{\mathbf{a}} f(x_1, \ldots, x_k, 0, \ldots, 0) = \\ &\partial^{\mathbf{a}} f(x_1, \ldots, x_k - 1, 0, \ldots, 0) + \partial^{\mathbf{a}+\mathbf{e}_k} f(x_1, \ldots, x_k - 1, 0, \ldots, 0), \end{aligned} \tag{4.4}$$

like equation (4.1) for any input $(x_1, \ldots, x_k, 0, \ldots, 0) \in B_k^{\mathbf{a}} \subset A_{\mathbf{a}}$ with $x_k \neq 0$. In NEXT, every time updating outputs of $\partial^{\mathbf{a}} f$ using $\partial^{\mathbf{a}+\mathbf{e}_k} f$, we call NEXT$(D, \mathbf{a}+\mathbf{e}_k)$ and update outputs of $\partial^{\mathbf{a}+\mathbf{e}_k} f$. By doing so, we always have the following: When we try to add $\partial^{\mathbf{a}+\mathbf{e}_k} f(\mathbf{x})$ into $\partial^{\mathbf{a}} f(\mathbf{x})$ for the update of $\partial^{\mathbf{a}} f$, the output $\partial^{\mathbf{a}+\mathbf{e}_k} f(\mathbf{x})$ for $\mathbf{x}$ is the value updated in the last NEXT$(D, \mathbf{a}+\mathbf{e}_k)$. This is because every $A_{\mathbf{a}}$ is covered following the same order.

We then show a practical way of realizing the update like the above discussion. The update requires two indices i and i' that denote

$$(x_1, \ldots, x_i, 0, \ldots, 0) \in B_i^{\mathbf{a}},\ (x_1, \ldots, x_i - 1, 0, \ldots, 0) \in B_{i'}^{\mathbf{a}}.$$

If we have these two values, then the update like (4.4) is realized as follows:

$$D[\mathbf{a}].y_i \leftarrow D[\mathbf{a}].y_{i'} + D[\mathbf{a} + \mathbf{e}_k].y_{i'}.$$

From the definition, this i is represented as $D[\mathbf{a}].i$. Furthermore, from the above discussion, this i' is equal to the index i of $D[\mathbf{a} + \mathbf{e}_k]$ in the last time updating.

We finally explain the construction of Algorithm 7. We first determine the index $D[\mathbf{a}].i$ by using the classification algorithm CLASS described in Subsection 4.4. In line 2-5, we update $D[\mathbf{a}].y_{(D[\mathbf{a}].i)}$,and the correctness of this part directly follows the aforementioned discussion. If $|\mathbf{a}| = d$, we do not update $D[\mathbf{a}].y_{(D[\mathbf{a}].i)}$ in line 2-5, because $\partial^{\mathbf{a}} f$ is constant. However, even in this case, we compute $D[\mathbf{a}].i$ for the correctness of NEXT$(D, \mathbf{a}')$ with $|\mathbf{a}'| = d - 1$.

Algorithm 5 MAIN(f)

$D \leftarrow \text{INIT}(D, f, \mathbf{0})$
for $\mathbf{x} \in \mathbb{F}_q^n$ (in the lexicographic order) **do**
  "$f(\mathbf{x}) = D[\mathbf{0}].y_{D[\mathbf{0}].i}$"
  $D \leftarrow \text{NEXT}(D, \mathbf{0})$
end for

Algorithm 6 INIT($D, f, \mathbf{a}$)

$D[\mathbf{a}].\alpha \leftarrow \rho(\mathbf{a})$
$D[\mathbf{a}].\mathbf{t} \leftarrow (0, 1, \ldots, n-1)$
$D[\mathbf{a}].\mathbf{u} \leftarrow (0, \ldots, 0)$
$D[\mathbf{a}].i \leftarrow 1$
if $|\mathbf{a}| < d$ **then**
  $D[\mathbf{a}].y_1 \leftarrow f(\mathbf{0})$
  for $k = 1, \ldots, \alpha$ **do**
    $D \leftarrow \text{INIT}(D, \partial_k f, \mathbf{a} + \mathbf{e}_k)$
  end for
else
  $D[\mathbf{a}].y_1, \ldots, y_\alpha \leftarrow f(\mathbf{0})$
end if
return D

Algorithm 7 NEXT($D, \mathbf{a} = (a_1, \ldots, a_n)$)

$D[\mathbf{a}].i, D[\mathbf{a}].\mathbf{t}, D[\mathbf{a}].\mathbf{u} \leftarrow \text{CLASS}(\mathbf{a}, D[\mathbf{a}].\alpha, D[\mathbf{a}].\mathbf{t}, D[\mathbf{a}].\mathbf{u})$
if $|\mathbf{a}| < d$ and $D[\mathbf{a}].i \neq 0$ **then**
  $i' \leftarrow D[\mathbf{a} + \mathbf{e}_{(D[\mathbf{a}].i)}].i$
  $D[\mathbf{a}].y_{(D[\mathbf{a}].i)} \leftarrow D[\mathbf{a}].y_{i'} + D[\mathbf{a} + \mathbf{e}_{(D[\mathbf{a}].i)}].y_{i'}$
  $D \leftarrow \text{NEXT}(D, \mathbf{a} + \mathbf{e}_{(D[\mathbf{a}].i)})$
end if
return D

4.6 Complexity

This subsection considers the time and space complexities of the initial part and the enumeration part of the proposed algorithm in Subsection 4.5.

Theorem 2. *For a single polynomial f in n variables of degree d over $\mathbb{F}_q$ with a prime number q, the enumeration algorithm proposed in Subsection 4.5 can be performed in $O\left(d \cdot q^n\right)$ operations after an initialization phase of negligible complexity $O\left(\binom{n+d}{d}^2\right)$ using $O\left(\log\left(q \cdot n\right) \cdot n \cdot \binom{n+d}{d}\right)$ bits memory.*

Proof. We first consider the time complexity of INIT. The computation of derivatives $\partial_k f$ in line 8 is clearly dominant in terms of the time complexity. This means that the time complexity of INIT is estimated as that of computing $\partial^{\mathbf{a}} f$ for any $\mathbf{a} \in \mathbb{Z}_{\geq 0}^n$ with $|\mathbf{a}| \leq d$. We here estimate that the number of $\mathbf{a}$ satisfying the condition of $|\mathbf{a}| \leq d$ is $\binom{n+d}{d}$ and the number of operations required to compute derivatives is at most $O\left(\binom{n+d}{d}\right)$. Therefore, the time complexity of the initial phase is given as $O\left(\binom{n+d}{d}^2\right)$ over $\mathbb{F}_q$.

We then estimate the time complexity of the enumeration part. The function NEXT can be performed in constant time excluding the recursive part in line 5, since the function CLASS is clearly completed by constant operations from Algorithm 4. Therefore, the time complexity of $\mathrm{NEXT}(D, \mathbf{0})$ is given as $O(d)$, and that of the enumeration part is estimated by $O(d \cdot q^n)$.

Finally, the space complexity of the proposed algorithm is discussed. Through the whole algorithm, for each $\mathbf{a}$, $D[\mathbf{a}]$ consumes $O\left(\log\left(q \cdot n\right) \cdot n\right)$ bit memory from the description in Subsection 4.3. Therefore, the space complexity consumed by D is given as $O\left(\log\left(q \cdot n\right) \cdot n \cdot \binom{n+d}{d}\right)$. Other than memory consumption by D, we consider the space of derivatives $\partial^{\mathbf{a}} f$ consumed in INIT. From the description of Algorithm 6, we prepare the memory that can store polynomials of degree d to degree 0 simultaneously. The size of this memory is estimated as follows:

$$\log_2 q \cdot \left(\binom{n+d}{d} + \cdots + \binom{n+1}{1} + 1\right) \approx O\left(\log q \cdot \binom{n+d}{d}\right).$$

In conclusion, the space complexity of the proposed enumeration algorithm is given as $O\left(\log\left(q \cdot n\right) \cdot n \cdot \binom{n+d}{d}\right)$. □

The comparison between these complexities and those of the enumeration algorithm of Bouillaguet et al. reveals that the proposed enumeration algorithm is as efficient as that of Bouillaguet et al. at the expense of a small amount of memory consumption.

Remark 3 (Case of $q = p^r$). In this paper, we only discuss the case in which the number q of elements of the finite field is a prime number. This remark explains a method to apply the enumeration algorithm to the case of $q = p^r$ with a prime number p and a positive integer r.

One polynomial in n variables over $\mathbb{F}_{p^r}$ can be clearly regarded as r polynomials in $n \cdot r$ variables over $\mathbb{F}_p$ (i.e., if $\theta_1, \ldots, \theta_r$ are basis of $\mathbb{F}_{p^r}$ over $\mathbb{F}_p$, for each variable x_i over $\mathbb{F}_{p^r}$, we set r variables $x_1^{(i)}, \ldots, x_r^{(i)}$ over $\mathbb{F}_p$ satisfying $x_i = \sum_{j=1}^{r} x_j^{(i)} \theta_j$). After performing this transformation, our enumeration algorithm can be applied to each one of the resulting r polynomials over $\mathbb{F}_p$. Then, the time complexity is given as $O\left(r \cdot d \cdot p^{r \cdot n}\right) = O\left(r \cdot d \cdot q^n\right)$, whereas the space complexity is given as $O\left(r \cdot \log\left(p \cdot n \cdot r\right) \cdot n \cdot r \cdot \binom{n \cdot r + d}{d}\right)$.

Remark 4 (Application to solving polynomial equations). By using the method proposed in [7], our enumeration algorithm can be applied to solve systems of polynomial equations, and its complexity is estimated as $O\left(d^2 \cdot \log n \cdot q^n\right)$. (See Appendix A.) Unfortunately, this is not the theoretically best algorithm for solving multivariate non-linear systems, since Lokshtanov et al. proposed an algorithm over general finite fields with the time complexity $O\left(q^{n \cdot (1-\epsilon)}\right)$, where $\epsilon > 0$. However, our method of solving non-linear systems would be more practical than the algorithm proposed by Lokshtanov et al. due to our simple structure as in the FES algorithm proposed by Bouillaguet et al. We leave optimizing our implementation as our future work.

5 Conclusion

This paper proposes a novel enumeration algorithm over finite fields $\mathbb{F}_q$ with a prime number q. Given a single-degree-d polynomial in n variables over $\mathbb{F}_q$, the proposed algorithm evaluates the given function at all the inputs with the time complexity $O\left(d \cdot q^n\right)$. The proposed enumeration algorithm is constructed by using a lexicographic order instead of Gray codes used in the enumeration algorithm by Bouillaguet et al. over $\mathbb{F}_2$. Compared with the enumeration algorithm by Bouillaguet et al., the proposed method achieves the equivalent efficiency with a small cost of memory consumption. Note that this small cost of the memory complexity is caused by our brunching structure of enumeration due to our lexicographic order. Furthermore, the proposed algorithm can be easily applied to the case where q is a prime power.

This paper discusses only the theoretical side, and thus one of our future works is to realize an efficient implementation of the proposed algorithm.

Acknowledgements

This work was supported by JST CREST Grant Number JPMJCR2113, Japan, and JSPS KAKENHI Grant Number JP22KJ0554, Japan.

References

1. W. Beullens. Improved cryptanalysis of UOV and Rainbow. In *EUROCRYPT 2021*, pages 348–373. Springer, 2021.

2. W. Beullens. MAYO: Practical post-quantum signatures from oil-and-vinegar maps. In *SAC 2021*, pages 355–376. Springer, 2021.
3. O. Billet and H. Gilbert. Cryptanalysis of Rainbow. In *SCN 2006*, pages 336–347. Springer, 2006.
4. J. R. Bitner, G. Ehrlich, and E. M. Reingold. Efficient generation of the binary reflected Gray code and its applications. *Commun. ACM*, 19(9):517–521, 1976.
5. W. Bosma, J. Cannon, and C. Playoust. The Magma algebra system. I. The user language. *J. Symb. Comput.*, 24(3-4):235–265, 1997.
6. C. Bouillaguet. Boolean polynomial evaluation for the masses. Cryptology ePrint Archive, Paper 2022/1412, 2022. `https://eprint.iacr.org/2022/1412`.
7. C. Bouillaguet, H.-C. Chen, C.-M. Cheng, T. Chou, R. Niederhagen, A. Shamir, and Yang B.-Y. Fast exhaustive search for polynomial systems in $\mathbb{F}_2$. In *CHES 2010*, pages 203–218. Springer, 2010.
8. C. Bouillaguet, C.-M. Cheng, T. Chou, R. Niederhagen, and Yang B.-Y. Fast exhaustive search for quadtaric systems in $\mathbb{F}_2$ on FPGAs. In *SAC 2013*, pages 205–222. Springer, 2014.
9. H. Furue, Y. Ikematsu, Y. Kiyomura, and T Takagi. A new variant of unbalanced oil and vinegar using quotient ring: QR-UOV. In *ASIACRYPT 2021*, pages 187–217. Springer, 2021.
10. H. Furue and M. Kudo. Polynomial XL: A variant of the XL algorithm using Macaulay matrices over polynomial rings. Cryptology ePrint Archive, Paper 2021/1609, 2021. `https://eprint.iacr.org/2021/1609`.
11. M.-R. Garey and D.-S. Johnson. *Computers and intractability: a guide to the theory of NP-completeness*. W. H. Freeman, 1979.
12. L. Goubin and N. Courtois. Cryptanalysis of the TTM cryptosystem. In *ASIACRYPT 2000*, pages 44–57. Springer, 2000.
13. A. Joux and V. Vitse. A Crossbred algorithm for solving boolean polynomial systems. In *NuTMiC 2017*, pages 3–21. Springer, 2017.
14. NIST. Post-quantum cryptography CSRC. `https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization`.
15. P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997.
16. C. Tao, A. Petzoldt, and J. Ding. Efficient key recovery for all HFE signature variants. In *CRYPTO 2021*, pages 70–93. Springer, 2021.

A Application to Solving Polynomial Equations

In this section, we consider finding common zeros of m polynomials $f_1, \ldots, f_m$ in n variables with degree d. This section first recalls a way of applying the enumeration algorithm of Bouillaguet et al. described in Subsection 3.2, and shows that our enumeration algorithm in Subsection 4.5 is also applicable. We here only consider the case of $n = m$ case, because if $n > m$, then $n-m$ variables can be specified, and if $n < m$, then we can focus on n equations as they should have a constant number of solutions.

We here roughly describe the way of applying an enumeration algorithm to find common zeros of $f_1, \ldots, f_n$ proposed in [7]. Let Z_i be the set of common zeros of $f_1, \ldots, f_i$, and then it is clear that Z_n is the set of the solutions of the system. For an integer $1 \leq k \leq n$, the proposed algorithm is described as follows:

(1) Find Z_k using the enumeration algorithm on each $f_1, \ldots, f_k$.
(2) Compute $Z_{k+1}, \ldots, Z_n$ one by one by substituting each value of Z_i for f_{i+1}.

The enumeration algorithm in Section 4 can be clearly applied to find common zeros of $f_1, \ldots, f_n$ over $\mathbb{F}_q$ in a similar manner. In the following, we estimate the time complexity of solving a system. As estimated in [7], the time complexity of the first step is estimated by $O(k \cdot d \cdot q^n)$ from the statement of Theorem 2. Furthermore, because the expected cardinality of Z_i is q^{n-i}, the time complexity of the second step is given as $\sum_{i=k}^{n-1} \binom{n+d}{d} \cdot q^{n-i} \approx O\left(\binom{n+d}{d} \cdot q^{n-k}\right)$ due to the complexity of the classical evaluation described in Section 2. Therefore, the optimal k minimizing the complexity is given by solving $k \cdot d \cdot q^n = \binom{n+d}{d} \cdot q^{n-k}$, and k is estimated by $d \log n$ when $n \to \infty$. By substituting $k = d \log n$ in $k \cdot d \cdot q^n$, the whole complexity of solving a polynomial system is given as $O\left(d^2 \cdot \log n \cdot q^n\right)$ over $\mathbb{F}_q$.

B Toy Example of Our Enumeration

We give an example of the behavior of the proposed enumeration algorithm on a concrete function. We take $f = x_1^2 + x_1 x_2 + x_2 + 2 \in \mathbb{F}_3[x_1, x_2]$ with degree $d = 2$ as an input. Then, derivatives of f are computed as follows:

$$\begin{aligned}
\partial_1 f &= 2x_1 + x_2 + 1, \\
\partial_2 f &= x_1 + 1, \\
\partial_1^2 f &= 2, \\
\partial_1 \partial_2 f &= 1, \\
\partial_2^2 f &= 0.
\end{aligned}$$

From these derivatives, for any $\mathbf{a} \in \mathbb{Z}_{\geq 0}$ with $|\mathbf{a}| \leq d = 2$, D is obtained as follows after $\mathrm{INIT}(D, f, \mathbf{0}, \mathbf{0})$ in line 1 of Algorithm 5:

$$\begin{aligned}
&D[(0,0)].(\alpha, \mathbf{t}, \mathbf{u}, (y_1, y_2), i) = (2, (0,1), (0,0), (2, \cdot), 1), \\
&D[(1,0)].(\alpha, \mathbf{t}, \mathbf{u}, y_1, i) = (1, (0,1), (0,0), 1, 1), \\
&D[(0,1)].(\alpha, \mathbf{t}, \mathbf{u}, (y_1, y_2), i) = (2, (0,1), (0,0), (1, \cdot), 1), \\
&D[(2,0)].(\alpha, \mathbf{t}, \mathbf{u}, y_1, i) = (1, (0,1), (0,0), 2, 1), \\
&D[(1,1)].(\alpha, \mathbf{t}, \mathbf{u}, y_1, i) = (1, (0,1), (0,0), 1, 1), \\
&D[(0,2)].(\alpha, \mathbf{t}, \mathbf{u}, (y_1, y_2), i) = (2, (0,1), (0,0), (0,0), 1).
\end{aligned}$$

Note that, for the case of $\mathbf{a} = (0,0)$ and $(0,1)$, $D[\mathbf{a}].y_2$ is not determined in $\mathrm{INIT}(D, f, \mathbf{0}, \mathbf{0})$. In the following, we show how D is updated for each $\mathbf{x}$ in line 2-5 of Algorithm 5 after the aforementioned initialization phase. We here omit $D[\mathbf{a}].\alpha$ because this value is not changed through the algorithm. We also omit values stored in $D[(2,0)]$, $D[(1,1)]$, and $D[(0,2)]$ because $D[\mathbf{a}].i$ with $|\mathbf{a}| = 2$ does not change from 1 in this case due to the relationship of q and d. Here, D

is updated as follows:

$$
\begin{aligned}
&\mathbf{x} = (0,1),\\
&\qquad D[(0,0)].(\mathbf{t},\mathbf{u},(y_1,y_2),i) = ((0,1),(0,1),(2,0),2),\\
&\qquad D[(1,0)].(\mathbf{t},\mathbf{u},y_1,i) = ((0,1),(0,0),1,1),\\
&\qquad D[(0,1)].(\mathbf{t},\mathbf{u},(y_1,y_2),i) = ((0,1),(0,1),(1,1),2),\\
&\mathbf{x} = (0,2),\\
&\qquad D[(0,0)].(\mathbf{t},\mathbf{u},(y_1,y_2),i) = ((0,1),(0,2),(2,1),2),\\
&\qquad D[(1,0)].(\mathbf{t},\mathbf{u},y_1,i) = ((0,1),(0,0),1,1),\\
&\qquad D[(0,1)].(\mathbf{t},\mathbf{u},(y_1,y_2),i) = ((0,1),(1,0),(2,1),1),\\
&\mathbf{x} = (1,0),\\
&\qquad D[(0,0)].(\mathbf{t},\mathbf{u},(y_1,y_2),i) = ((0,1),(1,0),(0,1),1),\\
&\qquad D[(1,0)].(\mathbf{t},\mathbf{u},y_1,i) = ((0,1),(0,1),0,1),\\
&\qquad D[(0,1)].(\mathbf{t},\mathbf{u},(y_1,y_2),i) = ((0,1),(1,0),(2,1),1),\\
&\mathbf{x} = (1,1),\\
&\qquad D[(0,0)].(\mathbf{t},\mathbf{u},(y_1,y_2),i) = ((0,1),(1,1),(0,2),2),\\
&\qquad D[(1,0)].(\mathbf{t},\mathbf{u},y_1,i) = ((0,1),(0,1),0,1),\\
&\qquad D[(0,1)].(\mathbf{t},\mathbf{u},(y_1,y_2),i) = ((0,1),(1,1),(2,2),2),\\
&\mathbf{x} = (1,2),\\
&\qquad D[(0,0)].(\mathbf{t},\mathbf{u},(y_1,y_2),i) = ((0,1),(1,2),(0,1),2),\\
&\qquad D[(1,0)].(\mathbf{t},\mathbf{u},y_1,i) = ((0,1),(0,1),0,1),\\
&\qquad D[(0,1)].(\mathbf{t},\mathbf{u},(y_1,y_2),i) = ((0,0),(0,0),(0,2),1),\\
&\mathbf{x} = (2,0),\\
&\qquad D[(0,0)].(\mathbf{t},\mathbf{u},(y_1,y_2),i) = ((0,0),(0,0),(0,1),1),\\
&\qquad D[(1,0)].(\mathbf{t},\mathbf{u},y_1,i) = ((0,1),(0,1),0,0),\\
&\qquad D[(0,1)].(\mathbf{t},\mathbf{u},(y_1,y_2),i) = ((0,0),(0,0),(0,2),1),\\
&\mathbf{x} = (2,1),\\
&\qquad D[(0,0)].(\mathbf{t},\mathbf{u},(y_1,y_2),i) = ((0,0),(0,1),(0,0),2),\\
&\qquad D[(1,0)].(\mathbf{t},\mathbf{u},y_1,i) = ((0,1),(0,1),0,0),\\
&\qquad D[(0,1)].(\mathbf{t},\mathbf{u},(y_1,y_2),i) = ((0,0),(0,1),(0,0),2),\\
&\mathbf{x} = (2,2),\\
&\qquad D[(0,0)].(\mathbf{t},\mathbf{u},(y_1,y_2),i) = ((0,0),(0,2),(0,0),2),\\
&\qquad D[(1,0)].(\mathbf{t},\mathbf{u},y_1,i) = ((0,1),(0,1),0,0),\\
&\qquad D[(0,1)].(\mathbf{t},\mathbf{u},(y_1,y_2),i) = ((0,0),(0,1),(0,0),0).
\end{aligned}
$$

Then, by seeing $D[(0,0)].\left(y_{D[(0,0)].i}\right)$ for each $\mathbf{x} \in \mathbb{F}_3^2$, one can confirm that the proposed algorithm enumerates the outputs of f correctly as follows:

$\mathbf{x}$	$(0,0)$	$(0,1)$	$(0,2)$	$(1,0)$	$(1,1)$	$(1,2)$	$(2,0)$	$(2,1)$	$(2,2)$
$f(\mathbf{x})$	2	0	1	0	2	1	0	0	0

C Magma Code

We here provides a code of the proposed enumeration algorithm in Magma [5]. Note that this implementation is not an optimized one.

Listing 1.1. Magma code for the proposed algorithm

```
//q: the number of elements of the finite field
//n: the number of variables
//f: a given polynomial
//d: the degree of f

// Given a polynomial f and an index k,
// compute the k-th derivative of f.
function deriv(f,k)
    P<[x]>:=Parent(f);
    n:=#x;
    Hom := hom<P->P|[x[i]: i in [1..k-1]] cat [x[k]+1] cat [x[i
        ]: i in [k+1..n]]>;
    ff:=Hom(f)-f;
    return ff;
end function;

// Given a vector a representing the derivative \partial^a f and
    the degree d,
// compute the index of data structures corresponding to the
    derivatives of a.
function accD(a,d)
    n:=#a;
    b:=0;
    c:=0;
    for i in [1..n] do
        b:=b+ &+([Binomial((n-i)+(d-c-j),(d-c-j)):j in [0..a[i
            ]-1]] cat [0]);
        c:=c+a[i];
    end for;
    if c gt d then
        return 0;
    end if;
    return b+1;
end function;

// Given a vector a,
// output the index of the most left nonzero bit of a.
function bit_a(a)
    n:=#a;
    for i in [1..n] do
        if a[i] ne 0 then
            return i;
        end if;
```

```
    end for;
    return n;
end function;

//a: a vector representing the derivative \partial^a f
//D: outputs y_1,...y_\alpha for any a
//aa: \alpha=\rho(a) for any a
//ii: indices D[a].i for any a
//tt: vectors D[a].t for any a
//uu: vectors D[a].u for any a

function CLASS(q,n,d,a,aa,tt,uu)
    k:=accD(a,d);
    if uu[k][n] lt q-a[aa[k]]-1 then
        i:=n;
        uu[k][n]:=uu[k][n]+1;
    else
        i:=tt[k][n];
        if aa[k]+i-n eq 0 then
            return 0, tt, uu;
        end if;
        uu[k][n]:=0;
        tt[k][n]:=n-1;
        if uu[k][i] lt q-2 then
            uu[k][i]:=uu[k][i]+1;
        else
            tt[k][i+1]:=tt[k][i];
            tt[k][i]:=i-1;
            uu[k][i]:=0;
        end if;
    end if;
    return aa[k]+i-n, tt, uu;
end function;

function INIT(f,a,D,d,aa)
    P<[x]>:=Parent(f);
    n:=#x;
    aa[accD(a,d)]:=bit_a(a);
    if &+[a[i]:i in [1..n]] lt d then
        D[accD(a,d)][1]:=Evaluate(f,[0: i in [1..n]]);
        for k in [1..aa[accD(a,d)]] do
            ab:=a;
            ab[k]:=ab[k]+1;
            D,aa:=INIT(deriv(f,k),ab,D,d,aa);
        end for;
    else
        DD:=Evaluate(f,[0: i in [1..n]]);
        for i in [1..aa[accD(a,d)]] do
            D[accD(a,d)][i]:=DD;
        end for;
    end if;
    return D,aa;
end function;

function NEXT(q,n,d,D,a,aa,ii,tt,uu)
```

```
95      ii[accD(a,d)],tt,uu:=CLASS(q,n,d,a,aa,tt,uu);
96      if &+[a[i]:i in [1..n]] lt d and ii[accD(a,d)] ne 0 then
97          a1:=a;
98          a1[ii[accD(a,d)]]:=a1[ii[accD(a,d)]]+1;
99          i1:=ii[accD(a1,d)];
100         D[accD(a,d)][ii[accD(a,d)]]:=D[accD(a,d)][i1]+D[accD(a1,
                d)][i1];
101         D,ii,tt,uu:=NEXT(q,n,d,D,a1,aa,ii,tt,uu);
102     end if;
103     return D,ii,tt,uu;
104 end function;
105
106 function MAIN(q,n,d,f)
107     P:=Parent(f);
108     K:=BaseRing(P);
109     D:=ZeroMatrix(K,Binomial(n+d,d),n);
110     aa:=[0:i in [1..Binomial(n+d,d)]];
111     ii:=[1:i in [1..Binomial(n+d,d)]];
112     tt:=[[0..n-1]:i in [1..Binomial(n+d,d)]];
113     uu:=[[0: i in [1..n]]:i in [1..Binomial(n+d,d)]];
114     D,aa:=INIT(f,[0:i in [1..n]],D,d,aa);
115     solution:=[];
116     for i in [0..q^n-1] do
117         Append(~solution,D[accD([0:i in [1..n]],d)][ii[accD([0:i
                in [1..n]],d)]]);
118         D,ii,tt,uu:=NEXT(q,n,d,D,[0:i in [1..n]],aa,ii,tt,uu);
119     end for;
120     return solution;
121 end function;
```