A New Efficient Algorithm for Solving Systems of Multivariate Polynomial Equations

Xijin Tang and Yong Feng

Laboratory for Automated Reasoning and Programming
Chengdu Institute of Computer Applications
Chinese Academy of Sciences
610041 Chengdu, P. R. China
tangxij@mails.gucas.ac.cn

Abstract. The security of many recently proposed cryptosystems is based on the difficulty of solving large systems of quadratic multivariate polynomial equations. The classical algorithm for solving such a system is Buchberger's algorithm for constructing Gröbner bases. Another algorithm for solving such a system is XL algorithm. For sparse system, Buchberger's algorithm benefits from sparsity of the system, but its complexity is impractical and hard to determine. XL could not make a good use of sparse structure of the system, since XL has no good strategy of choosing the multiply monomials.
In this paper, based on Extended Dixon Resultants, a new algorithm DR is proposed to solve systems of multivariate polynomial equations. The basic idea of DR is to apply Extended Dixon Resultants method to system of multivariate polynomial equations, by taking $x_1 \ldots x_{n-1}$ as variables and x_n as parameter. The time complexity of DR technique is evaluated, it seems to be polynomial when the system is sparse and $m = n$ and mixed volume is polynomial. As far as we know, it is the first algorithm which has better behavior than exhaustive search for some sparse systems over large field. Moreover, DR technique is compared with Buchberger's algorithm and XL technique in this paper. It is shown that DR is far more efficient than Buchberger's algorithm and XL when $m = n$. DR is a quite efficient algorithm, it makes a good use of the sparsity of the sparse system. Besides its efficiency, another advantage of DR is that its complexity is easy to determine.
Key words: multivariate cryptography, cryptography, polynomial equations over finite field, algebraic attack, Dixon Resultants, DR.

1 Introduction

Solving systems of multivariate quadratic polynomial equations is a hot topic in cryptology now, since AES encryption can be described by an extremely sparse overdefined multivariate quadratic system over GF(2^8) [8], and a large number of multivariate schemes had been proposed over the last few years, for example the HFE family [7].

The classical algorithm for solving such a system is Buchberger's algorithm [3]for constructing Gröbner bases, and its many variants. The algorithm first fixes a monomial order (typically in lexicographic order), and by computing the S-polynomial of two equations eliminates the top monomial. This process is repeated until find the Gröbner bases, and then solves the remaining univariate polynomial equation (e.g., by using Berlekamp's algorithm over the original or an extension field). Unfortunately, the degrees of the remaining monomials increase rapidly during the elimination process, thus the time complexity of the algorithm makes it often impractical even for a modest number of variables. In the worst case Buchberger's algorithm is known to run in double exponential time, and on average its running time seems to be single exponential. An efficient variant of this algorithm which we are aware of is F4 [10, 9].

Another algorithm for solving such a system is XL [5]algorithm, and its variants. It was proposed as a technique which can be viewed as a combination of bounded degree Gröbner bases and linearization. The basic idea of this technique is to generate from each polynomial equation a large number of higher degree variants by multiplying it with all the possible monomials of some bounded degree, and then to linearize the expanded system. In [5], the time complexity of the XL technique was analyzed and they had provided a strong theoretical and practical evidence that the expected running time of XL technique is:

– Polynomial when the number m of (random) equations is at least εn^2, and this for all $\varepsilon > 0$
– Subexponential if m exceeds n even by a small number.

In this paper we are interested in the problem of solving systems of multivariate polynomial equations in which the number of equation m is equal to the number of variables n, especially the system that is sparse. We make sure that the system has at least one solution, since systems that often occur in multivariate cryptographic schemes are of this type.

To solve sparse system, Gröbner base techniques benefits from the sparsity, but its complexity is still impractical and hard to determine. XL could not make a good use of sparsity of the system, since XL has no good strategy of choosing the multiply monomials. The complexity of XL is also impractical.

Dixon Resultant [1, 2] method is widely used in algebraic geometry and automated reasoning, it has good efficiency and easy determined complexity. For sparse systems, Dixon Resultant naturally make a good use of the sparsity of the system, its efficiency is extremely well to solve such systems.

In this paper, we propose a new efficient algorithm DR to solve systems of multivariate polynomial equations. The basic idea of DR is to apply Extended Dixon Resultants method to system of multivariate polynomial equations.

In the next section we give the preliminaries - the Dixon Resultant method, KSY method, mixed volume, and MQ problem. In section 3, we give the detailed description of DR.

After given a toy example of DR in section 4, we divide the MQ problems into three types in section 5, and give the corresponding experiment resultants. In section 6, we evaluate the time complexity of the DR technique and provide strong theoretical and practical evidence that the expected running time of this technique is:

– Polynomial when $m = n$ and the system is sparse and system's mixed volume is polynomial ;
– $2^{\omega \times n}$ when $m = n$ and the system is sparse and system's mixed volume is exponential ;
– $C^{\omega \times n}$ when $m = n$ and the system is general. When n increases, $C \to 4$.

Where $\omega = 3$ in the usual gauss reduction algorithms, and $\omega = 2.3766$ in improved algorithms.

Moreover, We compare DR technique with Buchberger's algorithm and XL in section 7. We will show that DR is far more efficient than Buchberger's algorithm and XL when $m = n$, no mater the system is sparse or not. We conclude this paper in the last section.

2 Preliminary

2.1 Dixon method and KSY method

Let $F = \{p_1(x_1, x_2, \cdots, x_n), \cdots, p_{n+1}(x_1, x_2, \cdots, x_n)\}$ be the set of $n+1$ generic ndegree polynomials in n variables. One determinant is formed as follows:

$$\varDelta(x_1, \cdots, x_n, \alpha_1, \cdots, \alpha_n) = \begin{vmatrix} p_1(x_1, x_2, \cdots, x_n) & \cdots & p_n(x_1, x_2, \cdots, x_n) \\ p_1(\alpha_1, x_2, \cdots, x_n) & \cdots & p_n(\alpha_1, x_2, \cdots, x_n) \\ p_1(\alpha_1, \alpha_2, \cdots, x_n) & \cdots & p_n(\alpha_1, \alpha_2, \cdots, x_n) \\ \cdots & \cdots & \cdots \\ p_1(\alpha_1, \alpha_2, \cdots, \alpha_n) & \cdots & p_n(\alpha_1, \alpha_2, \cdots, \alpha_n) \end{vmatrix}$$

Accordingly, one can get following polynomial:

$$\delta(x_1, \cdots, x_n, \alpha_1, \cdots, \alpha_n) = \frac{\varDelta(x_1, \cdots, x_n, \alpha_1, \cdots, \alpha_n)}{(x_1 - \alpha_1) \cdots (x_n - \alpha_n)}$$

The above polynomial δ is defined as the **Dixon polynomial**. The Dixon polynomial vanishes at any common zero of F, no matter what the values of $\alpha_1, \cdots, \alpha_n$ are. Hence, all the coefficients of the various power products of $\alpha_1, \cdots, \alpha_n$ in the Dixon polynomial vanish. We have polynomials in $x_1, x_2, \cdots, x_n$ which are coefficients of the power products of $\alpha_1, \cdots, \alpha_n$ in δ, denoted by ε'. If one views each power product of $x_1, \cdots, x_n$ as a new variable v_i $(i = 1, \cdots, s)$, one can get a system of s homogeneous linear equations in s variables:

$$\varepsilon \equiv D(v_1, v_2, \cdots, v_s)^T = (0, 0, \cdots, 0)^T,$$

where s is the number of power products of $x_1, \cdots, x_n$, Matrix D is called **Dixon matrix** and its determinant **Dixon Resultant**. Vanishing of the Dixon Resultant is a necessary condition for polynomials to have an affine zero. However, The Dixon's matrix is often singular, yielding no information for the polynomials. So, Deepak Kapur, Tushar Saxena and Lu Yang provided an approach (KSY method) to deal with Dixon Resultant being identically zero.

Given a set of any arbitrary polynomials, construct $\varDelta$, δ and the set of linear equations ε from Dixon polynomial as before, except that ε may have less than or equal to s equations in less than or equal to s variables. We still call it Dixon matrix D which may be an $s_1 \times s_2$ matrix, where $s_1 \leq s$ and $s_2 \leq s$. A set of constraints C on the variables $x_1, \cdots, x_n$ of the form $x_1 \neq 0 \wedge \cdots \wedge x_n \neq 0$ are given. Let the i-th column of the Dixon matrix be denoted by m_i and $monom(m_i)$ the monomial corresponding to m_i. Also, for given a set of constraints C, let $nvcol(C)$ denote the set of all columns m_i such that $C \Rightarrow monom(m_i) \neq 0$.

Let F be a set of $n+1$ polynomials with parameter coefficients and D the Dixon matrix of F. Let N_1 be the set of all $s_1 \times (s_2 - 1)$ matrices obtained from D by deleting a column which is an element of $nvcol(C)$. Let $\phi : a_1, \cdots, a_m \to Q$ be a mapping which gives values to the parameters from the algebraic closure of field of rational number Q. $\phi(F)$,$\phi(D)$, and $\phi(N_1)$ are the results of substituting those values for the parameters in F,D and N_1 respectively. Finally, let R=$\{Y|Y$ is an $r \times r$ nonsingular submatrix of $D\}$ it holds that[2]

Theorem 1. *If $\exists X \in N_1$ s.t. $rank(X) < rank(D)$ then for all $Y \in R$, $\phi(det(Y))$ vanishes if $\phi(F)$ has a common affine zero which satisfies C.*

According to theorem 1, one can obtain an algorithm as follows: Check if $\exists X \in N_1$ s.t. $rank(X) < rank(D)$, which is called **RSC criteria**. If this is true, any element of R is called **KSY Dixon Matrix**, then the determinant of **KSY Dixon Matrix** called **Extended Dixon Resultant** gives the required polynomial. However, this algorithm is not efficient. Kapur et al. [2] provided another algorithm to perform the check, obtain an element of R and compute its determinant.

Algorithm 1 *Compute Extended Dixon Resultant as follows:*

```
Step 1: Set up the $s_1 \times s_2$ Dixon matrix $D$ of $F$.
Step 2: Solve the matrix equation $D\bar{w} = \bar{0}$, where $\bar{w} = (w_1, \cdots, w_{s_2})^T$.
Step 3: Find out if there exists a $w_i$ in $\bar{w}$ such that $w_i = 0$ and also $C \Rightarrow monom(m_i) \neq 0$.
  If such a $w_i$ exists then compute $D_{row}$ and return the product of all the pivots of $D_{row}$.
Step 4: Else,return failure.
```

2.2 Mixed Volume

The convex hull of the support of a polynomial f is called its Newton polytope, and will be denoted as $\mathcal{N}(f)$. One can relate the Newton polytopes of a polynomial system to the number of its roots.

Definition 1. *([12, 13]).The mixed volume function $\mu(\mathcal{Q}_1, \ldots, \mathcal{Q}_d)$, where $\mathcal{Q}_i$ is convex hull, is a unique function which is multilinear with respect to the Minkowski sum and scaling operations, and is defined to have the multilinear property*

$$\mu(\mathcal{Q}_1, \ldots, a\mathcal{Q}'_k + b\mathcal{Q}''_k, \ldots, \mathcal{Q}_d) = a\ \mu(\mathcal{Q}_1, \ldots, \mathcal{Q}'_k, \ldots, Q_d) + b\ \mu(\mathcal{Q}_1, \ldots, \mathcal{Q}''_k, \ldots, \mathcal{Q}_d)$$

to ensure uniqueness, $\mu(\mathcal{Q}, \ldots, \mathcal{Q}) = d!Vol(\mathcal{Q})$, where $Vol()$ is the Euclidean volume of the polytope .

2.3 MQ problem

In this paper we consider the problem of solving a system of m multivariate quadratic equations with n variables over a finite field $GF(q)$. The input variables are denoted by x_i and belong to $GF(q)$. The equations are denoted by l_i and are quadratic, which may include linear and constant terms. The system to solve will be:

$$A = \begin{cases} l_1(x_1, x_2, \ldots, x_n) = 0 \\ \quad\quad . \\ \quad\quad . \\ \quad\quad . \\ l_m(x_1, x_2, \ldots, x_n) = 0 \end{cases}$$

Given m, n, q, we call MQ the problem of finding one (not necessarily all) solution to such a system chosen at random.

3 DR Algorithm

The algorithm DR (which stands for Dixon Resultants) applies Extended Dixon Resultants method to system of multivariate polynomial equations by taking $x_1 \dots x_{n-1}$ as variables and the x_n as parameter. For Dixon Resultants method requires that the number of equations should be equal to the number of variables plus one, so DR algorithm will be efficient to solve MQ problem with $m = n$, but DR could also be applied to overdefined system of multivariate polynomial equations. As shown later, DR could get all the common solutions in most cases.

Let A denotes a system of multivariate quadratic equations, with $m = n$, all $x_{i,i=1..n}$ is over $GF(q)$. $(a_1, a_2, \dots a_n) \in GF(q)^n$ denotes a common solution of A. Let $V(A)$ denotes all common solutions of A. Let $\pi_{k,k=1..n}$ denote a projection operator, $\pi_k((a_1, a_2, \dots a_n)) = a_k$.

Algorithm 2 ***DR***

```
    Input: A system A of m multivariate quadratic equations with n variables over a finite field GF(q), and m = n.
    Output: At least one common solution of the input system.
    step 1. Taking $x_1 \dots x_{n-1}$ as variables and $x_n$ as parameter, Computer the Dixon matrix of A;
    step 2. Run subprogram RSC to check RSC Criteria and select rows and columns that needed for constructing KSY Dixon Matrix;
    step 3. Construct the KSY Dixon Matrix;
    step 4. Compute the determinant of the KSY Dixon Matrix;
    step 5. Solve the equation gotten in step 4 over $GF(q)$(e.g., with Berlekamp's algorithm). There may be several roots, the set of these roots is called s;
    step 6. For each root of $x_n$, substitute it to the KSY Dixon Matrix gotten in step 3, then solve the linear equation to find the values of all the other monomials, in particular for all the other variables $x_i$.
    step 7. If fail to find a common solution of A in step 6, let $s = \{0, p(used\ in\ subprogram\ RSC)\}$, run step 6.
```

Algorithm 3 ***RSC***

```
    Input: A Dixon Matrix M of dimension $s_1 \times s_2$.
    Output: The rows and columns that needed for constructing the KSY Dixon Matrix.
    step 1. Substitute a random value p in $GF(q)$ for $x_n$ in the Dixon matrix;
    step 2. Perform gauss elimination on the matrix gotten in step 1, assume the result is $M'$ and the rank of $M'$ is r;
    step 3. If $M'$ is a square and full rank matrix then return all the rows and columns in M;
    step 4. For each column m of matrix $M'$ do
        construct a submatrix $M_s$ of $M'$ of dimension $s_1 \times (s_2 - 1)$ by deleting m;
        if rank of $M_s < r$ then break this loop;
    step 5. If step 3 finds a submatrix $M_s$, whose rank is less than r
    then
        choose the columns needed for constructing a $r \times r$ submatrix of $M'$ and whose rank is r;
        transpose $M'$ and perform gauss elimination, then choose the rows needed for constructing a $r \times r$ submatrix of $M'$ and whose rank is r;
        return the rows and columns;
```

```
else
    goto step 1.
```

Remark 1: In most cases of MQ problem, the RSC criteria always holds. If it does not hold, our algorithm will fail, but in our large number of stimulations we have not met this situation.

Remark 2: We adopt the original KSY Dixon Resultants computation algorithm in RSC, our algorithm makes a good use of the efficiency of numeric computation, new algorithm is more efficient than the original algorithm.

Remark 3: In step 4 of algorithm RSC, we construct the submatrix by the matrix after gauss elimination, so we need a little computation to reduce the matrix to row echelon form. The complexity of this Guass elimination is quite less than $O(s_1 \times (s_2 - 1) \times (s_2 - 1))$.

Remark 4: If we only want to find one common solution of system A, when we fail in algorithm RSC step 4, we could return to algorithm DR and omit step 3, step 4, step 5, straightly let $s = \{p\}$ and run step 6. Because the RSC criteria always holds, when we failed in algorithm RSC step 4, it means $p \in \pi_n(V(A))$.

Remark 5: In practice, it is found that by running the following step instead of step 4,step 5 and step 6 in algorithm DR, computation time could be reduced.

```
step 4&5&6: for each value $p$ in $GF(q)$ do
    substitute $p$ for $x_n$ in KSY Dixon Matrix and we get $M'$;
    compute thedeterminant of $M'$, if determinant = 0 then recover other variables;
```

The reason is that the numeric computation is far more efficient than symbolic computation. We guess it will hold over $GF(p^k)$. If this also holds in $GF(p^k)$, we could replace step 4,step 5 and step 6 in algorithm DR with step 4&5&6.

The Correctness of DR Algorithm:

In our stimulation, the RSC criteria always holds, and the column that corresponds to the monomial $1 = x_1^0 x_2^0 ...$ of the Dixon matrix is usually not a linear combination of the remaining ones, so usually there is no constrain on the common solutions that DR gets. If monomial 1 is not satisfies, there are quite many columns satisfies RSC criteria. By this fact, we could minimize our constrain, in most cases, we could minimize the constrains with no constrain on x_n. If we can't make this, we could simply add 0 into s.

In algorithm RSC step 4, if $p \in \pi_n(V(A))$, the RSC check may fail or not. If we fail in the check, we assign another value to x_n, and go on our algorithm, we could get all of $\pi_n(V(A))$. If we don't fail in the check, we may omit the p. To attack this problem, we could simply add p into s.

Making a summary of all the discussions above, we could come to a conclusion that DR algorithm could get all common solution of A in most cases.

4 A Toy Example of DR

Consider the following problem of solving:

$x_{i,i=1..5} \in GF(127)$

$$A = \begin{cases} l_1 : 9x_1 + 37x_3 + 17x_1x_2 + 120x_2x_3 + 18x_3x_5 + 58x_4^2 + 87 = 0 \\ l_2 : 46x_1 + 43x_3 + 117x_5 + 43x_1x_2 + 93x_1x_3 + 61x_3x_4 + 48 = 0 \\ l_3 : 32x_1x_2 + 54x_1x_4 + 56x_2x_3 + 93x_3x_5 + 60x_5^2 + 45 = 0 \\ l_4 : 124x_1 + 93x_1x_3 + 78x_1x_4 + 45x_1x_5 + 39x_2x_3 + 38x_2x_4 + 46 = 0 \\ l_5 : 27x_2 + 95x_2x_5 + 85x_3^2 + 74x_3x_4 + 46x_3x_5 + 77x_4x_5 + 66 = 0 \end{cases}$$

Computing $A's$ Dixon Matrix we get 23×23 matrix, its determinant is identically zero.

In subprogram RSC we assign 19 to x_n, we find the RSC criteria holds with only constructing one submatirx by deleting the last column. Then we select the following columns: 1, 2, 3, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23 and the following rows: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 23.

In step 3, we get a 20×20 matrix. in step 4, we compute the determinant of the matrix gotten in step 3 then we have

$88 + 75x_5 + 22x_5^2 + 15x_5^3 + 47x_5^4 + 98x_5^5 + 64x_5^27 + 99x_5^6 + 67x_5^7 + 108x_5^8 + 126x_5^9 + 69x_5^{10} + 29x_5^{11} + 53x_5^{12} + 51x_5^{13} + 57x_5^{14} + 102x_5^{15} + 69x_5^{16} + 118x_5^{17} + 124x_5^{18} + 107x_5^{19} + 11x_5^{20} + 68x_5^{21} + 44x_5^{22} + 100x_5^{23} + 63x_5^{24} + 34x_5^{25} + 72x_5^{26} + 47x_5^{28} + 110x_5^{29} + 17x_5^{30}$

After solving the above equation over $GF(127)$, we get 4 solutions 5,10,20,108. Then by recovering other variables we get 2 common solutions $[x_1 = 1, x_2 = 2, x_3 = 3, x_4 = 4, x_5 = 5], [x_1 = 93, x_2 = 100, x_3 = 23, x_4 = 54, x_5 = 20]$.

5 Experimental Results on DR

5.1 Experimental results with general MQ problem

In this part of experiment, we solve MQ problem that is randomly generated by *randpoly* command in Maple, and we make sure it has at least one common solution.

Table 1. DR over $GF(127)$ for general MQ problem

n	3	4	5	6	7	8
m	3	4	5	6	7	8
terms	9	14	19	24	28	30
mixed volume	8	14	26	54	124	254
M	(5,5)	(14,14)	(41,38)	(113,109)	(236,222)	(406,430)
M'	(5,5)	(12,12)	(29,29)	(71,71)	(150,150)	(278,278)

Legend:
n: number of variables
m: number of equations
terms:number of monomials
mixed volume: mixed volume of the MQ problem to be solved
M: the matrix size of Dixon Matrix
M': the matrix size of KSY Dixon matrix

5.2 Experimental results with sparse MQ problem

As we know, mixed volume is a key factor in the size of resultant matrix. According to the way the sparse MQ problems' mixed volume increase, we divide them into two types.

– type A: the mixed volume is polynomial;
– type B: the mixed volume is exponential.

Type A In this part of experiment, we give a series of example of type A, each l_i has the form $l_i = x_i + x_{(i \ mod \ n)+1} \times x_{((i+1) \ mod \ n)+1} + b_i$, and we make sure it has at least one common solution. For example, when $n = 3$, the system to be solved may equal to $\{x_1 + x_2 \times x_3 + 4, x_2 + x_3 \times x_1 + 114, x_3 + x_1 \times x_2 + 106\}$.

Table 2. DR over $GF(127)$ for sparse MQ problem of type A (notations as for Table 1)

n	3	4	5	6	7	8	9	10	11	12
m	3	4	5	6	7	8	9	10	11	12
terms	7	9	11	13	15	17	19	21	23	25
mixed volume	5	6	12	17	30	46	77	122	200	321
M	(2,2)	(4,4)	(7,7)	(12,12)	(20,20)	(33,33)	(54,54)	(88,88)	(143,143)	(232,232)
M'	(2,2)	(4,4)	(7,7)	(12,12)	(20,20)	(33,33)	(54,54)	(88,88)	(143,143)	(232,232)

Type B In this part of experiment, we give a series of example of type B, each l_i has the form $l_i = x_i + x_{(i \ mod \ n)+1} + {x_{((i+1) \ mod \ n)+1}}^2 + b_i$, and we make sure it has at least one common solution. For example, when $n = 3$, the system to be solved may equal to $\{x1 + x2 + x3^2 + 25, x2 + x3 + x1^2 + 119, x3 + x1 + x2^2 + 116\}$.

Table 3. DR over $GF(127)$ for sparse MQ problem of type B(notations as for Table 1)

n	3	4	5	6	7	8	9	10	11
m	3	4	5	6	7	8	9	10	11
terms	7	9	11	13	15	17	19	21	23
mixed volume	8	16	32	64	128	256	512	1024	2048
M	(4,4)	(8,8)	(16,16)	(32,32)	(64,64)	(128,128)	(256,256)	(512,512)	(1024,1024)
M'	(4,4)	(8,8)	(16,16)	(32,32)	(64,64)	(128,128)	(256,256)	(512,512)	(1024,1024)

6 Complexity Evaluation of DR

The complexity of DR depends on the Dixon matrix size. We assign a random value p to x_n in step 1 of RSC, if $p \notin \pi_n(V(A))$, we may perform Gauss elimination on Dixon matrix twice and on KSY Dixon matrix serval times(the time is dependent on the root number we got in step 5). if $p \in \pi_n(V(A))$ we usually need perform three times Gauss elimination on Dixon matrix and on

KSY Dixon matrix serval times , for the possibility of $first\ p \in \pi_n(V(A)) \bigwedge second\ p \in \pi_n(V(A))$ is really low, less than $\frac{5}{127} \times \frac{4}{127}$ (we usually could find less than 5 solutions of equation gotten in step 4). It is known that, the complexity of usual Gauss reduction algorithm is n^3 on matrix of dimension $n \times n$, and $n^{2.3766}$ in improved algorithms. The Dixon matrix generated in DR is a approximate square matrix. So the complexity of DR is at least $min(s_1, s_2)^{\omega}$, where $\omega = 3$ in the usual gauss reduction algorithms, and $\omega = 2.3766$ in improved algorithms.

When the MQ problem is full – all possible monomials occurred in every original equation, from the the construction of the Dixon Matrix, we could find the Dixon matrix size is equal to the number of terms in polynomial $\prod_{i=1}^{n-1}(\sum_{j=1}^{i}(1 + x_j))$. The the number of terms in $\prod_{i=1}^{n-1}(\sum_{j=1}^{i}(1 + x_j))$ is $\frac{(2\times n)!}{n!\times(n+1)!}$, so the Dixon Matrix size is $\frac{(2\times n)!}{n!\times(n+1)!}$. Assume $f(x) = \frac{(2\times x)!}{x!\times(x+1)!}$, then $\frac{f(x+1)}{f(x)} = 2\,\frac{2\,x+1}{x+2}$, and $\lim_{x\to\infty}(2\frac{2\,x+1}{x+2}) = 4$, so the Dixon matrix size is always less than 4^n. What's more, we would not reach the worst situation. Since if the system to be solved is full, we could perform a Gauss-Jordan Elimination on it, by that we could eliminate $n - 1$ terms in every original equation l_i.

For general MQ problem, Dixon matrix's size is less than or equal to $\frac{(2\times n)!}{n!\times(n+1)!}$. Transforming the equation into C^n form, we get that the Dixon matrix size is $C^n, C \to 4$.

$$C \approx \begin{cases} 2 & n = 1..8, \\ 3 & n = 9..22, \\ 3.5 & n = 22..80, \\ 4 & n > 80. \end{cases}$$

When the system is sparse the Dixon matrix size depends on the structure of the problem. Our stimulations show that when the mixed volume is polynomial, the Dixon matrix size is probably polynomial too. When the mixed volume is exponential, the Dixon matrix size is probably exponential too, and the Dixon matrix size is equal to 2^n.

Making a summarize of the discussion above, we could except running time of DR is:

- Polynomial when $m = n$ and the system is sparse and system's mixed volume is polynomial;
- $2^{\omega\times n}$ when $m = n$ and the system is sparse and system's mixed volume is exponential;
- $C^{\omega\times n}$ when $m = n$ and the system is general, $C \to 4$;

Where $\omega = 3$ in the usual gauss reduction algorithms, and $\omega = 2.3766$ in improved algorithms.

7 Comparison Between DR and Other Algorithm

7.1 comparison between DR and Buchberger's algorithm

For the complexity of Buchberger's algorithm is hard to determine, and no value could identify the scale of it, we just compare the time DR and Buchberger's algorithm taken to solve same problem randomly generated by Maple. We used two algorithms for constructing Gröbner bases, one is *GroebnerBasis* command in Maple 9.5, which uses classical Buchberger's algorithm, the another is *slimgb* command in Singular 3.0 [11], which uses a variant of F4. DR is implemented in Maple 9.5. Computations are on Intel P4 1.4G , 256Mb RAM.

Table 4. comparison between DR and Buchberger's algorithm (1)

	general					
n	3	4	5	6	7	8
m	3	4	5	6	7	8
$terms$	9	14	16	22	28	30
DR	1.32	1.61	6.5	43.39	118.6	557.75
$G1$	0.29	269.52	> 3600	?	?	?
$G2$	< 0.01	0.13	971.02	?	?	?

Table 5. comparison between DR and Buchberger's algorithm (2)

	sparse of type A										sparse of type B								
n	3	4	5	6	7	8	9	10	11	12	3	4	5	6	7	8	9	10	11
m	3	4	5	6	7	8	9	10	11	12	3	4	5	6	7	8	9	10	11
$terms$	7	9	11	13	15	17	19	21	23	25	7	9	11	13	15	17	19	21	23
DR	0.20	0.28	0.30	0.35	0.86	1.80	4.22	9.25	21.51	51.05	0.19	0.31	0.45	1.42	4.65	15.86	94.45	406.20	1593.25
$G1$	0.03	0.03	0.25	1.36	1.39	114.42	> 7200	?	?	?	0.02	0.19	1.00	11.58	?	?	?	?	?
$G2$	< 0.01	< 0.01	< 0.01	< 0.01	0.02	0.07	3.80	1489.23	> 18000	?	< 0.01	< 0.01	0.01	0.05	0.75	320.56	4421.42	?	?

Legend:

DR: the seconds DR taken

G1: the seconds that Buchberger's Gröbner bases algorithm taken in Maple 9.5

G2: the seconds that F4 algorithm taken in Singular 3.0

?: the time is too long or run out of memory, we don't know the exact time

7.2 Comparison between DR and XL

We compare the matrix size between DR and FXL(a variant of XL), for both DR's and XL's complexity depends on the matrix size. We terminate FXL once the rank of matrix is equal to the number of columns, since when that condition is reached we could get a equation in x_n, then solve it and recover other variables just like in DR.

Table 6. comparison between DR and XL (1)

	general					
n	3	4	5	6	7	8
m	3	4	5	6	7	8
$terms$	9	14	16	22	28	30
M	(5,5)	(13,14)	(32,32)	(65,72)	(224,179)	(406,330)
M'	(5,5)	(12,12)	(25,25)	(52,52)	(130,130)	(278,278)
FXL	(18,15)	(80,56)	(127,125)	(1512,792)	(?)	(?)

Legend:

FXL: the matrix size of FXL

?: the matrix size is too large, we don't know the exact size

Table 7. comparison between DR and XL (2)

	sparse of type A									
n	3	4	5	6	7	8	9	10	11	12
m	3	4	5	6	7	8	9	10	11	12
$terms$	7	9	11	13	15	17	19	21	23	25
M	(2,2)	(4,4)	(7,7)	(12,12)	(20,20)	(33,33)	(54,54)	(88,88)	(143,143)	(232,232)
M'	(2,2)	(4,4)	(7,7)	(12,12)	(20,20)	(33,33)	(54,54)	(88,88)	(143,143)	(232,232)
FXL	(9,9)	(16,15)	(75,57)	(126,100)	(588,392)	(960,661)	(?)	(?)	(?)	(?)

Table 8. comparison between DR and XL (3)

	sparse of type B								
n	3	4	5	6	7	8	9	10	11
m	3	4	5	6	7	8	9	10	11
$terms$	7	9	11	13	15	17	19	21	23
$mixed\ volume$	8	16	32	64	128	256	512	1024	2048
M	(4,4)	(8,8)	(16,16)	(32,32)	(64,64)	(128,128)	(256,256)	(512,512)	(1024,1024)
M'	(4,4)	(8,8)	(16,16)	(32,32)	(64,64)	(128,128)	(256,256)	(512,512)	(1024,1024)
FXL	(18,15)	(80,56)	(350,210)	(1512,792)	(?)	(?)	(?)	(?)	(?)

8 Conclusion

In this paper, we propose a new algorithm DR for solving systems of multivariate polynomial equations based on Extended Dixon Resultant. DR is a quite efficient algorithm, especially for the spare MQ problem, because it make a good of the sparsity of the sparse system. The running time seems to be polynomial when the system is sparse and $m = n$ and mixed volume is polynomial. As far as we konw, DR is the first algorithm which we known has better behavior than exhaustive search for sparse system of type A over large field. However in many practical cases, the best known algorithms are still close to exhaustive search over large field.

Finally besides DR's efficiency, another advantage of DR is that its complexity is easy to determine. Unlike XL and Buchberger's algorithm, before starting computation, we could anticipate the time it would take.

References

1. Dixon, A. L., The eliminant of three quantics in two independent variables. Proc. London Mathematical Society, 6:468-478, 1908.
2. Deepak Kapur, Tushar Saxena, and Lu Yang. Algebraic and geometric reasoning using dixon resultants. In ACM ISSAC 94, pages 99–107, Oxford, England, 1994.
3. B. Buchberger, Grobner bases: An Algorithmic method in Polynomial Ideal theory, in Multidimensional Systems Theory, N.K. Bose, cd., D. Reidel Publishing Co., 1985.
4. Nicolas Courtois. Algebraic attacks over GF(2^k), application to hfe challenge 2 and sflash-v2. PKC 2004, springer, pages 201–217, 2004.
5. Nicolas Courtois, Alexander Klimov, Jacques Patarin, and Adi Shamir. Efficient algorithms for solving overdefined systems of multivariate polynomial equations. Lecture Notes in Computer Science, EUROCRYPT, pages 392–407, 2000.
6. Deepak Kapur. Automated geometric reasoning: Dixon resultants, gröbner bases, and characteristic sets. Automated Deduction in Geometry, International Workshop on Automated Deduction in Geometry, Toulouse, France, September 27-29, 1996, Selected Papers, Springer, pages 1–36, 1996.
7. Jacques Patarin. Hidden fields equations (HFE) and isomorphisms of polynomials (IP): Two new families of asymmetric algorithms. EUROCRYPT, pages 33–48, 1996.
8. Sean Murphy, Matthew J.B. Robshaw, Essential Algebraic Structure within the AES, Lecture Notes in Computer Science, Volume 2442, pages 1 - 16, Jan 2002

9. J.-C. Faugeere, A New Efficient Algorithm for Computing Gröbner Bases without Reduction to Zero (F5), Proceedings of ISSAC 2002, pages 75-83, ACM Press 2002.
10. J.-C. Faugere, A New Efficient Algorithm for Computing Gröbner Bases (F4), Journal of Pure and Applied Algebra, 139 (1999), pages 61-88.
11. G.-M. Greuel, G. Pfister, and H. Schönemann. SINGULAR 3.0. A Computer Algebra System for Polynomial Computations. Centre for Computer Algebra, University of Kaiserslautern (2001). `http://www.singular.uni-kl.de`.
12. Gelfand, I.M., Kapranov, M.M., Zelevinsky, A.V., Discriminants, Resultants and Multidimensional Determinants. first edition. Birkhauser, Boston 1994.
13. Cox, D., Little, J., OShea, D., Using Algebraic Geometry. first edition. Springer-Verlag, New York 1998.