Algorithms to solve massively under-defined systems of multivariate quadratic equations

Yasufumi Hashimoto [*]

Abstract

It is well known that the problem to solve a set of randomly chosen multivariate quadratic equations over a finite field is NP-hard. However, when the number of variables is much larger than the number of equations, it is not necessarily difficult to solve equations. In fact, when $n \geq m(m+1)$ (n, m are the numbers of variables and equations respectively) and the field is of even characteristic, there is an algorithm to solve equations in polynomial time (see [Kipnis et al, Eurocrypt'99] and also [Courtois et al, PKC'02]). In the present paper, we give two algorithms to solve quadratic equations; one is for the case of $n \geq$ (about) $m^2 - 2m^{3/2} + 2m$ and the other is for the case of $n \geq m(m+1)/2 + 1$. The first algorithm solves equations over any finite field in polynomial time. The second algorithm requires exponential time operations. However, the number of required variables is much smaller than that in the first one, and the complexity is much less than the exhaustive search. As an application, we propose an analysis of the unbalanced oil and vinegar signature scheme (UOV) with a parameter suggested in [Kipnis et al, Eurocrypt'99]. This analysis is more effective than known attacks in [Courtois et al, PKC'02] and [Faugére and Perret, SCC'08], and shows that the UOV with such a parameter is not secure at all.

1 Introduction

It is well known that the problem to solve a set of randomly chosen multivariate quadratic equations over a finite field is NP-hard. Then the cryptosystems based on multivariate quadratic equations (Matsumoto-Imai, HEF, UOV, STS, TTM and so on, see e.g. [5], [9] and their references) have been expected to be secure against the quantum attacks. However, not all quadratic equations are difficult to be solved while the problem itself is NP-hard. In fact, some of such cryptosystems were already broken and some others of them are weaker than expected when they were proposed. Thus it is important to study

*Partially supported by JSPS Grant-in-Aid for Young Scientists (B) no. 20740027.

Keywords. under-defined multivariate quadratic equations, the oil and vinegar signature scheme

which quadratic equations are solved easily and how to characterize its difficulty for the practical use of quadratic equations in cryptology.

For this topic, there has been several works in the view of the relation between the numbers of variables and equations. In fact, Courtois et al. ([2] and [3]) have studied how to solve the equations when m is much larger than n (where m, n are the numbers of equations and variables respectively). On the other hand, Kipnis et al. [6] studied the case when n is much larger than m. In fact, they found an algorithm to solve quadratic equations when $n \geq m(m+1)$ and the characteristic of the field is even. Note that the characteristic is odd, their algorithm requires the complexity $O(2^m \times (\text{polynomial}))$. Although Courtois et al. [1] modified it more effectively for odd characteristic cases, its modification requires much more variables.

In the present paper, we give two algorithms to solve quadratic equation when n is sufficiently larger than m. The first algorithm solves equations over any finite field in polynomial time when $n \geq$ (about) $m^2 - 2m^{3/2} + 2m$. The number of variables required in this algorithm is less than that in [6], and this works in polynomial time both for even and odd characteristic fields. The second algorithm solves equations for $n \geq m(m+1)/2+1$. The complexity of the second algorithm is roughly estimated by $O(2^m)$ or $O(3^m)$. While it is in exponential time, it is much better than the exhaustive search, especially for large order fields, and furthermore the number of required variables is much less than that in the first algorithm.

As an application, we use our first algorithm to attack the unbalanced oil and vinegar signature scheme (UOV) with the parameters suggested in [6]. Our attack is more effectively than several known attack by Courtois et al. [1] and Faugére-Perret [4], and the UOV with such a parameter is not secure at all.

2 Notations

Throughout this paper, we use the following notations.

q: a power of prime.
k: a finite field of order q.
$n, m \geq 1$: integers.
$x = (x_1, \cdots, x_n) \in k^n$.
$\tilde{x} = (x_0, x_1, \cdots, x_n) \in k^{n+1}$.
$f_l(x) \in k$ $(1 \leq l \leq m)$: a quadratic form of x.
$\tilde{f}_l(\tilde{x}) \in k$ $(1 \leq l \leq m)$: the homogeneous quadratic form of $\tilde{x}$ such that $\tilde{f}_l(1, x_1, \cdots, x_n) = f_l(x_1, \cdots, x_n)$.
$U_i : k^n \to k^n$: an invertible linear map such that $x_i \mapsto a_1x_1 + \cdots + a_nx_n$ for $a_1, \cdots, a_n \in k$ $(a_i \neq 0)$ and $x_j \mapsto x_j$ for $j \neq i$.
$\tilde{U}_i : k^{n+1} \to k^{n+1}$: an invertible linear map such that $x_i \mapsto a_0x_0 + a_1x_1 + \cdots + a_nx_n$ for $a_0, a_1, \cdots, a_n \in k$ $(a_i \neq 0)$ and $x_j \mapsto x_j$ for $j \neq i$.

3 Kipnis-Patarin-Goubin's algorithm for $n \geq m(m+1)$

In this section, we give an algorithm proposed by Kipnis-Patarin-Goubin [6] to solve m quadratic equations with n variables for $n > m(m+1)$.

First, find U_2 such that the coefficients of x_1x_2 in $f_1(U_2x), \cdots, f_m(U_2x)$ are zero. This requires to solve m linear equations with n variables. Put $f_l^{(2)}(x) := f_l(U_2x)$. Next, find U_3 such that the coefficients of x_1x_3, x_2x_3 in $f_1^{(1)}(U_3x), \cdots, f_m^{(1)}(U_3x)$ are zero. This requires to solve $2m$ linear equations with n variables. Put $f_l^{(3)}(x) := f_l^{(2)}(U_3x)$. Recursively, find $U_4, \cdots, U_m$ such that the coefficients of x_ix_j $(1 \leq i < j \leq m)$ in all polynomials are zero. Such computations require to solve at most $m(m-1)$ linear equations of n variables.

Put $U := U_m \cdots U_3U_2$ and $g_l(x) := f_l(Ux)$. Then the coefficients of x_ix_j $(1 \leq i < j \leq m)$ in $g_l(x)$ are zero. Take $x_{m+1}, \cdots, x_n \in k$ such that $g_1(x), \cdots, g_m(x)$ are written by linear combinations of $x_1^2, \cdots, x_m^2$ and constants. Finding such $x_{m+1}, \cdots, x_n$ requires to solve m^2 linear equations of $n-m$ variables. Since $g_l(x)$'s are linear combinations of $x_1^2, \cdots, x_m^2$ and constants, the problem to solve $f_1(x) = 0, \cdots, f_m(x) = 0$ is reduced that to solve $x_1^2 = (\text{const}), \cdots, x_m^2 = (\text{const})$.

When q is even, all elements of k have square roots. Then the algorithm works in polynomial time. When q is odd, almost half elements of k have square roots. Then the algorithm requires roughly $2^m \times$ (polynomial) operations.

Note that Courtois et al. modified this algorithm for odd characteristic k under the assumptions that 2^{40}-times operations is feasible and that $n \geq 2^{m/7}(m+1)$. See [1] for the detail.

4 Solving quadratic equations for $n \geq$ (about) $m^2 - 2m^{3/2} + 2m$

In this section, we propose an algorithm to solve equations for $n \geq$ (about) $m^2 - 2m^{3/2} + 2m$. For the algorithm, we first prepare the following elementary fact.

Fact 1. Let $U = (u_{ij})_{0 \leq i,j \leq n}$ be an invertible matrix over k. If U satisfies that $u_{00} \neq 0$ and the coefficient of x_0^2 of $\tilde{f}_i(Ux)$ is zero for $1 \leq l \leq n$, then $(u_{00}^{-1}u_{10}, \cdots, u_{00}^{-1}u_{n0})$ is a solution of $f_1(x) = 0, \cdots, f_m(x) = 0$.

Then the target of this algorithm to find such a transform U. We also prepare the following two algorithms.

Algorithm A.
Aim. Let $g(x)$ be a quadratic form of $x \in k^n$. Find an invertible linear transform U such that the coefficients of x_ix_j for $i + j \leq n$ in $g(Ux)$ are zero.
Step 1. Find U_1 such that the coefficients of x_1x_i $(1 \leq i \leq n-1)$ in $g(U_1x)$ are zero. This requires to solve $n-2$ homogeneous linear equations and a homogeneous quadratic equation with n variables. Put $g^{(1)}(x) := g(U_1x)$.
Step 2. Find U_2 such that the coefficients of x_2x_i $(1 \leq i \leq n-2)$ in $g(U_2x)$ are zero. This requires to solve $n-3$ homogeneous linear equations and a homogeneous quadratic

equation with $n-1$ variables. Put $g^{(2)}(x) := g^{(1)}(U_2x)$.

Recursively, find U_3 such that the coefficients of x_3x_i $(1 \le i \le n-3)$ are zero, U_4 such that the coefficients of x_4x_i $(1 \le i \le n-4)$ are zero and so on. Then one can find an invertible transform $U : k^n \to k^n$ such that the coefficients of x_ix_j $(1 \le i, j \le n,\ i+j \le n)$ in $g(Ux)$ are zero.

Algorithm B.
Aim. Let $1 \le M, L \le n/2$ be integers and $g_1(x), \cdots, g_M(x)$ quadratic forms of $x \in k^n$. Find an invertible linear $U : k^n \to k^n$ such that the coefficients of x_ix_j for $1 \le i, j \le L$ in $g_1(x), \cdots, g_M(x)$ are zero when

$$M \le \begin{cases} \left\lfloor \dfrac{n-L}{L-1} \right\rfloor, & n \le L^2 - L, \\ L-1, & L^2 - L + 1 \le n \le L^2, \\ L, & n \ge L^2 + 1. \end{cases}$$

Step 1. Find an invertible linear map $V_1 : k^n \to k^n$ such that the coefficients of x_ix_j $(1 \le i, j \le L)$ in $g_1(V_1x)$ are zero. This can be done by Algorithm A. Put $g_l^{(1)}(x) := g_l(V_1x)$.

In Step2, we want to find V_2 such that the coefficients of x_ix_j $(1 \le i, j \le L)$ in $g_1^{(1)}(V_2x)$ and $g_2^{(1)}(V_2x)$ are zero, and in Step 3, we want to find V_3 such that the coefficients of x_ix_j $(1 \le i, j \le L)$ in $g_1^{(2)}(V_3x), g_2^{(2)}(V_3x), g_3^{(2)}(V_3x)$ are zero, and so on. To consider recursively, we assume that, until Step $N-1$, we can find V' such that the coefficients of x_ix_j $(1 \le i, j \le L)$ in $g_1(V'x), \cdots, g_{N-1}(V'x)$ are zero. Put $g_l^{(N-1)}(x) := g_l(V'x)$. We will describe how to find an invertible V_N such that the coefficients of x_ix_j $(1 \le i, j \le L)$ in $g_1^{(N-1)}(V_Nx), \cdots, g_N^{(N-1)}(V_Nx)$ are zero in Step N.

Step N.
$N-$**0.** Using Algorithm A, find an invertible linear map $W_1 : k^L \to k^L$ such that the coefficients of x_ix_j $(1 \le i, j \le L,\ i+j \le L)$ in $g_N^{(N-1)}(\tilde{W}_1x)$ are zero (where $\tilde{W}_1 := \begin{pmatrix} W_1 & \\ & I \end{pmatrix}$).
Note that the coefficients of x_ix_j $(1 \le i, j \le L)$ in $g_1^{(N-1)}(\tilde{W}_1x), \cdots, g_{N-1}^{(N-1)}(\tilde{W}_1x)$ are zero. Put $g_l^{(N-1,1)}(x) := g_l^{(N-1)}(\tilde{W}_1x)$.
$N-$ **1.** Find U_L such that the coefficients of x_ix_j $(1 \le i, j \le L)$ in $g_l^{(N-1,1)}(U_Lx)$ for $1 \le l \le N-1$ and of x_1x_L in $g_N^{(N-1,1)}(U_Lx)$ are zero. This requires to solve $(L-1)(N-1)$ homogeneous linear equations of $(x_{L+1}, \cdots, x_n)$, a homogeneous linear equation of $(x_L, \cdots, x_n)$ and $N-1$ homogeneous quadratic equations written by

$$\sum_{i=1}^{L} x_i \times \big(\text{linear form of } (x_{L+1}, \cdots, x_n)\big) + \big(\text{quadratic form of } (x_{L+1}, \cdots, x_n)\big) = 0.$$

It is easy to see that, when $N \le L$ and $(L-1)(N-1)+1 < n-L$, such equations can be solved by the linear operations. After finding such U_L, find an invertible linear map $W_2 : k^{L-1} \to k^{L-1}$ such that the coefficients of x_ix_j $(1 \le i, j \le L,\ i+j \le L+1)$

in $g_N^{(N-1,1)}(\tilde{W}_2 U_L x)$ are zero (where $\tilde{W}_2 = \begin{pmatrix} 1 & & \\ & W_2 & \\ & & I \end{pmatrix}$) by using Algorithm A. Put $g_l^{(N-1,2)}(x) := g_l^{(N-1,1)}(\tilde{W}_2 U_L x)$.

$N - $ **2.** Find U_L such that the coefficients of $x_i x_j$ $(1 \leq i, j \leq L)$ in $g_l^{(N-1,2)}(U_L x)$ for $1 \leq l \leq N-1$ and of $x_1 x_L, x_2 x_L$ in $g_N^{(N-1,2)}(U_L x)$ are zero. This requires to solve $(L-1)(N-1)+1$ homogeneous linear equations of $(x_{L+1}, \cdots, x_n)$, a homogeneous linear equation of $(x_L, \cdots, x_n)$ and $N-1$ homogeneous quadratic equations written by

$$\sum_{i=1}^{L} x_i \times \big(\text{linear form of } (x_{L+1}, \cdots, x_n)\big) + \big(\text{quadratic form of } (x_{L+1}, \cdots, x_n)\big) = 0.$$

It is easy to see that, when $N \leq L$ and $(L-1)(N-1)+2 < n-L$, this can be solved by the linear operations. After finding such U_L, find an invertible linear map $W_3 : k^{L-2} \to k^{L-2}$ such that the coefficients of $x_i x_j$ $(1 \leq i, j \leq L,\ i+j \leq L+2)$ in $g_N^{(N-1,2)}(\tilde{W}_3 U_L x)$ are zero (where $\tilde{W}_3 = \begin{pmatrix} I_2 & & \\ & W_3 & \\ & & I \end{pmatrix}$). Put $g_l^{(N-1,3)}(x) := g_l^{(N-1,2)}(\tilde{W}_3 U_L x)$.

After that, find U_L and W_4 such that the coefficients of $x_i x_j$ for $1 \leq i, j \leq L$, $i+j \leq L+3$ in $g_N^{(N-1,3)}(\tilde{W}_4 U_L x)$ and $x_i x_j$ for $1 \leq i, j \leq L$ in $g_l^{(N-1,3)}(\tilde{W}_4 U_L x)$ $(1 \leq l \leq N-1)$ are zero. And find U_L and W_5 similarly. Continue such operations until we will find an invertible linear $V'_N : k^n \to k^n$ such that the coefficients of $x_i x_j$ $(1 \leq i, j \leq L)$ in $g_1^{(N-1)}(V'_N x), \cdots, g_{N-1}^{(N-1)}(V'_N x)$ and the coefficients of $x_i x_j$ $(1 \leq i, j \leq L, (i,j) \neq (m,m))$ in $g_N^{(N-1)}(V'_N x)$ are zero. After putting $g_l^{(N-1,L-1)}(x) := g_l^{(N-1)}(V'_N x)$, find U_L such that $x_i x_j$ $(1 \leq i, j \leq L)$ in $g_1^{(N-1,L-1)}(V'_N x), \cdots, g_N^{(N-1,L-1)}(V'_N x)$. This requires to solve $(L-1)N$ homogeneous linear equation of (x_{L+1}, x_n), $N-1$ homogeneous quadratic equations written by

$$\sum_{i=1}^{L} x_i \times \big(\text{linear form of } (x_{L+1}, \cdots, x_n)\big) + \big(\text{quadratic form of } (x_{L+1}, \cdots, x_n)\big) = 0$$

and a homogeneous quadratic equation written by

$$x_L^2 + \sum_{i=1}^{L} x_i \times \big(\text{linear form of } (x_{L+1}, \cdots, x_n)\big) + \big(\text{quadratic form of } (x_{L+1}, \cdots, x_n)\big) = 0.$$

To solve them, we need $(L-1)N \leq n-L+1, N \leq L$ or $(L-1)N \leq n-L, \quad N \leq L-1$, namely

$$N \leq \begin{cases} \left\lfloor \dfrac{n-L}{L-1} \right\rfloor, & n \leq L^2 - L, \\ L-1, & L^2 - L + 1 \leq n \leq L^2, \\ L, & n \geq L^2 + 1. \end{cases}$$

Thus we can get an invertible U such that the coefficients of the coefficients of x_ix_j $(1 \leq i, j \leq L)$ in $g_l(Ux)$ for

$$1 \leq l \leq \begin{cases} \left\lfloor \dfrac{n-L}{L-1} \right\rfloor, & n \leq L^2 - L, \\ L-1, & L^2 - L + 1 \leq n \leq L^2, \\ L, & n \geq L^2 + 1. \end{cases}$$

It is easy to see that the complexity of this algorithm is $O\big(n^{w+1}\big)$ where $2 \leq w < 3$ is the exponent of the Gaussian reduction to solve linear equations.

Based on two algorithms above, we give an algorithm to solve quadratic equations for $n \geq$ (about) $m^2 - 2m^{3/2} + 2m$

Algorithm 1.
Aim. Find a solution $x \in k^n$ of the equations $f_1(x) = 0, \cdots, f_m(x) = 0$.
Step 1. Choose $N_1 < \sqrt{n+1}$ and put

$$M_1 := \min\left(\left\lfloor \frac{n+1}{2} \right\rfloor, \left\lfloor \frac{n - N_1 + 1}{N_1 - 1} \right\rfloor \right)$$

Using Algorithm B, find and invertible linear map $V_1 : k^{n+1} \to k^{n+1}$ such that the coefficients of x_ix_j $(0 \leq i, j \leq M_1 - 1)$ in $\tilde{f}_l(V_1\tilde{x})$ for $1 \leq l \leq N_1$ are zero. Put $\tilde{f}_l^{(1)}(\tilde{x}) := \tilde{f}_l(V_1\tilde{x})$.
Step 2. Choose $N_2 < \sqrt{M_1}$ and put

$$M_2 := \min\left(\left\lfloor \frac{M_2}{2} \right\rfloor, \left\lfloor \frac{M_1 - N_2 + 1}{N_2 - 1} \right\rfloor \right).$$

Using Algorithm B, find an invertible linear map $V_2 : k^{M_1} \to k^{M_1}$ such that the coefficients of x_ix_j $(0 \leq i, j \leq M_2 - 1)$ in $\tilde{f}_l(\tilde{V}_2\tilde{x})$ for $N_1 + 1 \leq l \leq N_1 + N_2$ are zero (where $\tilde{V}_2 := \begin{pmatrix} V_2 & \\ & I \end{pmatrix}$). Put $\tilde{f}_l^{(2)}(\tilde{x}) := \tilde{f}_l^{(1)}(\tilde{V}_2\tilde{x})$.

Continuing such operations until $M_t = 1$, we can get an invertible linear map $U = (u_{ij})_{0 \leq i,j \leq n}$ such that the coefficients of x_0^2 in $\tilde{f}_l(U\tilde{x})$ for $1 \leq l \leq N_1 + N_2 + \cdots + N_t$ are zero. From Fact 1, we see that this algorithm solves the equations when $m \leq N_1 + \cdots + N_t \sim n^{1/2} + n^{1/4} + \cdots$, namely $n \geq$ about $m^2 - 2m^{3/2} + 2m$. Note that the complexity of this algorithm is $O\big(N_0^{w+1}\big)$.

Remark that, for a fixed number of variables, the number of equations solved by this algorithm depends on the choice of $N_1, N_2, \cdots$. Consider the case of $n = 49$ $(n+1 = 50)$ for example. When we put $(N_1, M_1) = (7, 7)$, $(N_2, M_2) = (2, 3)$ and $(N_3, M_3) = (1, 1)$, we can solve 10 equations. However, when we put $(N_1, M_1) = (3, 23)$, $(N_2, M_2) = (3, 10)$, $(N_3.M_3) = (2, 5)$, $(N_4, M_4) = (2, 2)$ and $(N_5, M_5) = (1, 1)$, we can solve 11 equations. The following is the table of the number of variables required to solve m equations.

m	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	$\cdots$
n	1	3	4	7	9	12	18	22	28	40	48	60	77	88	100	121	140	$\cdots$

5 Solving quadratic equations for $n \geq m(m+1)/2+1$

In this section, we propose an algorithm to solve equations for $n \geq m(m+1)/2+1$. The algorithm is as follows.

Algorithm 2.
Aim. Find a solution $x \in k^n$ of the equations $f_1(x) = 0, \cdots, f_m(x) = 0$.
Step 1. Find $\tilde{U}_0$ such that the coefficient of x_0^2 in $\tilde{f}(\tilde{U}_0\tilde{x})$ is zero. This requires to solve a homogeneous quadratic equation of $n+1$ variables. Then this is valid when when $n \geq 2$. Put $\tilde{f}_l^{(1)}(x) := \tilde{f}_l(\tilde{U}_0\tilde{x})$.
Step 2. Find $\tilde{U}_1$ such that the coefficients of x_0x_1 and x_1^2 in $\tilde{f}_1^{(1)}(\tilde{U}_1\tilde{x})$ and the coefficient of x_0x_1 in $\tilde{f}_2^{(1)}(\tilde{U}_1\tilde{x})$ are zero. This requires to solve two homogeneous linear equations and a homogeneous quadratic equation of $n+1$ variables. Then we need $n \geq 4$. Put $\tilde{f}_l^{(1,1)}(\tilde{x}) := \tilde{f}_l^{(1)}(\tilde{U}_1\tilde{x})$. If there is a solution $z_2 \in k$ of $\tilde{f}_2^{(1,1)}(1, z_2, 0, \cdots, 0) = 0$, denote by $V_2 := \begin{pmatrix} 1 & 0 & \\ z_2 & 1 & \\ & & I \end{pmatrix}$ and put $\tilde{f}_l^{(2)}(\tilde{x}) := \tilde{f}_l^{(1,1)}(V_2\tilde{x})$. If there are no such z_2, take another $\tilde{U}_1$ and repeat until such $z_2 \in k$ appears. It is easy to see that the coefficients of x_0^2, x_0x_1, x_1^2 in $\tilde{f}_1^{(2)}(\tilde{x})$ and x_0^2 in $\tilde{f}_2^{(2)}(\tilde{x})$ are zero. Note that Step 1 and Step 2 solves 2 equations of at least 4 variables.
Step 3. Find $\tilde{U}_1$ such that the coefficients of x_0x_1 and x_1^2 in $\tilde{f}_1^{(2)}(\tilde{U}_1\tilde{x}), \tilde{f}_2^{(2)}(\tilde{U}_1\tilde{x})$ and the coefficient of x_0x_1 in $\tilde{f}_3^{(2)}(\tilde{U}_1\tilde{x})$ are zero. This requires to solve 3 homogeneous linear equations and 2 homogeneous quadratic equations of $n+1$ variables. When $n \geq 7$, this can be done by Step 1 and Step 2. Put $\tilde{f}_l^{(2,1)}(\tilde{x}) := \tilde{f}_l^{(2)}(\tilde{U}_1\tilde{x})$. If there is a solution $z_3 \in k$ of $\tilde{f}_3^{(2,1)}(1, z_3, 0, \cdots, 0) = 0$, denote by $V_3 := \begin{pmatrix} 1 & 0 & \\ z_3 & 1 & \\ & & I \end{pmatrix}$ and put $\tilde{f}_l^{(3)}(\tilde{x}) := \tilde{f}_l^{(2,1)}(V_3\tilde{x})$. If there are no such z_3, take another U_1 and repeat until such $z_3 \in k$ appears. It is easy to see that the coefficients of x_0^2, x_0x_1, x_1^2 in $\tilde{f}_1^{(3)}(\tilde{x})$ and $\tilde{f}_2^{(3)}(\tilde{x})$ and x_0^2 in $\tilde{f}_3^{(3)}(\tilde{x})$ are zero. Note that Step 1 to Step 3 solves 3 equations of at least 7 variables.

We now suppose that, until Step $N-1$, we can find an invertible linear map $U' : k^{n+1} \to k^{n+1}$ such that the coefficients of x_0^2, x_0x_1, x_1^2 in $\tilde{f}_1(U'\tilde{x}), \cdots, \tilde{f}_{N-2}(U'\tilde{x})$ and of x_0^2 in $\tilde{f}_{N-1}(U'\tilde{x})$ are zero when $n \geq N(N-1)/2+1$. This also means that Step 1 to Step $N-1$ solves $N-1$ quadratic equations of at least $n \geq N(N-1)/2+1$ variables. Put $\tilde{f}_l^{(N-1)}(\tilde{x}) := \tilde{f}_l(U'\tilde{x})$.
Step N. Find $\tilde{U}_1$ such that the coefficients of x_0x_1 and x_1^2 in $\tilde{f}_l^{(N-1)}(\tilde{U}_1\tilde{x})$ for $1 \leq l \leq N-1$ and the coefficient of x_0x_1 in $\tilde{f}_N^{(N-1)}(\tilde{U}_1\tilde{x})$ are zero. This requires to solve N homogeneous linear equations and $N-1$ homogeneous quadratic equations of $n+1$ variables. If $n \geq N(N+1)/2+1$, this can be done by Step 1 to Step $N-1$. Put $\tilde{f}_l^{(N-1,1)}(\tilde{x}) := \tilde{f}_l^{(N-1)}(\tilde{U}_1\tilde{x})$. If there is a solution $z_N \in k$ of $\tilde{f}_N^{(N-1,1)}(1, z_N, 0, \cdots, 0) = 0$, denote by $V_N := \begin{pmatrix} 1 & 0 & \\ z_N & 1 & \\ & & I \end{pmatrix}$ and put $\tilde{f}_l^{(N)}(\tilde{x}) := \tilde{f}_l^{(N-1,1)}(V_N\tilde{x})$. If there are no such z_N, take another U_1 and repeat

until such $z_N \in k$ appears. It is easy to see that the coefficients of x_0^2, x_0x_1, x_1^2 in $\tilde{f}_l^{(N)}(\tilde{x})$ for $1 \le l \le N-1$ and x_0^2 in $\tilde{f}_N^{(N)}(\tilde{x})$ are zero. Note Step 1 to Step N solves N equations of at least $N(N+1)/2+1$ variables.

Thus we see that, if $n \ge m(m+1)/2+1$, we can solve quadratic equations.

We now estimate the complexity of this algorithm. Let c_N be the complexity in the N-th step. For simplicity, assume that one computes U_1 once if q is even and twice if q is odd in all steps, because the probability that univariate quadratic equation has a solution is almost 1 if q is even and $1/2$ if q is odd. Since N-th step requires to solve $N-1$ linear equations and $N-1$ quadratic equations, we see that

$$c_N = \begin{cases} c_1 + c_2 + \cdots + c_{N-1} + (\text{polyn}), & (2 \mid q), \\ 2(c_1 + c_2 + \cdots + c_{N-1}) + (\text{polyn}), & (2 \nmid q). \end{cases}$$

Thus we see that $c_N = O(2^N)$ when q is even and $c_N = O(3^N)$ when q is odd. Since the complexity of this algorithm is $c_1 + \cdots + c_m$, we can roughly estimate the complexity by $O(2^m)$ when q is even and $O(3^m)$ when q is odd.

Remark that, when we use Algorithm 1 instead of Step 1 to Step 12 in Algorithm 2, we can solve equations for $m \ge 12$ and $n \ge m(m+1)/2 - 18$. While its complexity is about $O(2^m)$ or $O(3^m)$, its practical computational task will be much less than that of the original Algorithm 2.

6 Analysis of UOV

The unbalanced oil and vinegar signature scheme (UOV, see [8] and [6]) is one of signature schemes based on multivariate quadratic equations. According to [7] and [6], UOV can be broken if $n \le 2m$ or q^{n-2m} is small. Thus, for UOV, n should be sufficiently larger than $2m$. The authors of [6] suggested UOV with $q = 2^4, m = 16$ and $n = 48$ or 64.

To break UOV with such parameters, we use the algorithms proposed in Section 4. Assume that the complexity of Algorithm B is $n^{3.5}$. The attack is as follows.

The case of $n = 64$.
Step 1. Use Algorithm B to find $V_1 : k^{65} \to k^{65}$ such that the coefficients of x_ix_j $(0 \le i, j \le 30)$ in $\tilde{f}_1(V_1\tilde{x}), \tilde{f}_2(V_1\tilde{x}), \tilde{f}_3(V_1\tilde{x})$ are zero. The complexity in this step is $64^{3.5}$. Put $x^{(1)} := (x_0, \cdots, x_{30})$ and $f_l^{(1)}(x^{(1)}) := \tilde{f}_l\big(V_1(x_0, \cdots, x_{30}, 0, \cdots, 0)\big)$. Since $f_1^{(1)}(x^{(1)}) = 0$, $f_2^{(1)}(x^{(1)}) = 0$ and $f_3^{(1)}(x^{(1)}) = 0$ for any $x^{(1)}$, we see that the problem to solve equations with $(n, m) = (64, 16)$ is reduced to the problem to find $x^{(1)} = (1, x_1, \cdots, x_{30})$ satisfying $f_4^{(1)}(x^{(1)}) = 0, \cdots, f_{16}^{(1)}(x^{(1)}) = 0$.
Step 2. Use Algorithm B to find $V_2 : k^{31} \to k^{31}$ such that the coefficients of x_ix_j $(0 \le i, j \le 13)$ in $f_4^{(1)}(V_2x^{(1)}), f_5^{(1)}(V_2x^{(1)}), f_6^{(1)}(V_2x^{(1)})$ are zero. The complexity in this step is $31^{3.5}$. Put $x^{(2)} := (x_0, \cdots, x_{13})$ and $f_l^{(2)}(x^{(2)}) := f_l^{(1)}\big(V_2(x_0, \cdots, x_{13}, 0, \cdots, 0)\big)$. Since $f_4^{(2)}(x^{(2)}) = 0$, $f_5^{(2)}(x^{(2)}) = 0$ and $f_6^{(2)}(x^{(2)}) = 0$ for any $x^{(2)}$, we see that the problem to solve equations with $(n, m) = (30, 13)$ is reduced to the problem to find $x^{(2)} = (1, x_1, \cdots, x_{14})$ satisfying $f_7^{(2)}(x^{(2)}) = 0, \cdots, f_{16}^{(2)}(x^{(2)}) = 0$.

Step 3. Use Algorithm B to find $V_3 : k^{14} \to k^{14}$ such that the coefficients of $x_i x_j$ $(0 \leq i, j \leq 4)$ in $f_7^{(2)}(V_3 x^{(2)}), f_8^{(2)}(V_3 x^{(2)}), f_9^{(2)}(V_3 x^{(2)})$ are zero. The complexity in this step is $14^{3.5}$. Put $x^{(3)} := (x_0, \cdots, x_4)$ and $f_l^{(3)}(x^{(3)}) := f_l^{(2)}\big(V_3(x_0, \cdots, x_4, 0, \cdots, 0)\big)$. Since $f_7^{(3)}(x^{(3)}) = 0$, $f_8^{(3)}(x^{(3)}) = 0$ and $f_9^{(3)}(x^{(3)}) = 0$ for any $x^{(3)}$, we see that the problem to solve equations with $(n, m) = (13, 10)$ is reduced to the problem to find $x^{(3)} = (1, x_1, \cdots, x_4)$ satisfying $f_{10}^{(3)}(x^{(3)}) = 0, \cdots, f_{16}^{(3)}(x^{(3)}) = 0$ if there exists a solution.
Step 4. Use Algorithm 1 to find $x^{(3)} = (1, x_1, \cdots, x_4)$ such that $f_{10}^{(3)}(x^{(3)}) = 0$, $f_{11}^{(3)}(x^{(3)}) = 0$ and $f_{12}^{(3)}(x^{(3)}) = 0$. The complexity is about $5^{3.5} = 2^{10.5}$. If $f_{13}^{(3)}(x^{(3)}) = 0$ is not satisfied, try Algorithm 1 again and repeat it until the solution $x^{(3)}$ of $f_{10}^{(3)}(x^{(3)}) = 0$, $f_{11}^{(3)}(x^{(3)}) = 0$ and $f_{12}^{(3)}(x^{(3)}) = 0$ will be also a solution of $f_{13}^{(3)}(x^{(3)}) = 0$. Since the probability that y satisfies such a condition is roughly q^{-1}, the complexity in this step is about $2^4 \times 5^{3.5}$.

Note that Step 1 to 4 find a solution $x \in k^n$ of $f_1(x) = 0, \cdots, f_{13}(x) = 0$. If $f_{14}(x) = 0$, $f_{15}(x) = 0$, $f_{16}(x) = 0$ are not satisfied, try Step 3 and Step 4 again and repeat it until the solution of $f_1(x) = 0, \cdots, f_{13}(x) = 0$ will be a solution of $f_{14}(x) = 0$, $f_{15}(x) = 0$ and $f_{16}(x) = 0$. Since the probability that x satisfies such a condition is roughly q^{-3}, we see that the complexity of this attack is about

$$64^{3.5} + 31^{3.5} + 2^{12} \times (14^{3.5} + 2^4 \times 5^{3.5}) \sim 2^{25.9}.$$

The case of $n = 48$.
Step 1. Use Algorithm B to find $V_1 : k^{49} \to k^{49}$ such that the coefficients of $x_i x_j$ $(0 \leq i, j \leq 22)$ in $\tilde{f}_1(V_1 \tilde{x}), \tilde{f}_2(V_1 \tilde{x}), \tilde{f}_3(V_1 \tilde{x})$ are zero. The complexity in this step is $49^{3.5}$. Put $x^{(1)} := (x_0, \cdots, x_{22})$ and $f_l^{(1)}(x^{(1)}) := \tilde{f}_l\big(V_1(x_0, \cdots, x_{22}, 0, \cdots, 0)\big)$. Since $f_1^{(1)}(x^{(1)}) = 0$, $f_2^{(1)}(x^{(1)}) = 0$ and $f_3^{(1)}(x^{(1)}) = 0$ for any $x^{(1)}$, we see that the problem to solve equations with $(n, m) = (48, 16)$ is reduced to the problem to find $x^{(1)} = (1, x_1, \cdots, x_{22})$ satisfying $f_4^{(1)}(x^{(1)}) = 0, \cdots, f_{16}^{(1)}(x^{(1)}) = 0$.
Step 2. Use Algorithm B to find $V_2 : k^{23} \to k^{23}$ such that the coefficients of $x_i x_j$ $(0 \leq i, j \leq 9)$ in $f_4^{(1)}(V_2 x^{(1)}), f_5^{(1)}(V_2 x^{(1)}), f_6^{(1)}(V_2 x^{(1)})$ are zero. The complexity in this step is $23^{3.5}$. Put $x^{(2)} := (x_0, \cdots, x_9)$ and $f_l^{(2)}(x^{(2)}) := f_l^{(1)}\big(V_2(x_0, \cdots, x_9, 0, \cdots, 0)\big)$. Since $f_4^{(2)}(x^{(2)}) = 0$, $f_5^{(2)}(x^{(2)}) = 0$ and $f_6^{(2)}(x^{(2)}) = 0$ for any $x^{(2)}$, we see that the problem to solve equations with $(n, m) = (22, 13)$ is reduced to the problem to find $x^{(2)} = (1, x_1, \cdots, x_9)$ satisfying $f_7^{(2)}(x^{(2)}) = 0, \cdots, f_{16}^{(2)}(x^{(2)}) = 0$ if there exists a solution.
Step 3. Use Algorithm B to find $V_3 : k^{10} \to k^{10}$ such that the coefficients of $x_i x_j$ $(0 \leq i, j \leq 4)$ in $f_7^{(2)}(V_3 x^{(2)}), f_8^{(2)}(V_3 x^{(2)})$ are zero. The complexity in this step is $10^{3.5}$. Put $x^{(3)} := (x_0, \cdots, x_4)$ and $f_l^{(3)}(x^{(3)}) := f_l^{(2)}\big(V_3(x_0, \cdots, x_4, 0, \cdots, 0)\big)$. Since $f_7^{(3)}(x^{(3)}) = 0$ and $f_8^{(3)}(x^{(3)}) = 0$ for any $x^{(3)}$, we see that the problem to solve equations with $(n, m) = (9, 10)$ is reduced to the problem to find $x^{(3)} = (1, x_1, \cdots, x_4)$ satisfying $f_9^{(3)}(x^{(3)}) = 0, \cdots, f_{16}^{(3)}(x^{(3)}) = 0$ if there exists a solution.
Step 4. Use Algorithm 1 to find $x^{(3)} = (1, x_1, \cdots, x_4)$ such that $f_9^{(3)}(x^{(3)}) = 0$, $f_{10}^{(3)}(x^{(3)}) = 0$ and $f_{11}^{(3)}(x^{(3)}) = 0$. The complexity is about $5^{3.5} = 2^{10.5}$. If $f_{12}^{(3)}(x^{(3)}) = 0$ is not satisfied, try Algorithm 1 again and repeat it until the solution $x^{(3)}$ of $f_9^{(3)}(x^{(3)}) = 0$, $f_{10}^{(3)}(x^{(3)}) = 0$

and $f_{11}^{(3)}(x^{(3)}) = 0$ will be also a solution of $f_{12}^{(3)}(x^{(3)}) = 0$. Since the probability that y satisfies such a condition is roughly q^{-1}, the complexity in this step is about $2^4 \times 5^{3.5}$.

Note that Step 1 to 4 find a solution $x \in k^n$ of $f_1(x) = 0, \cdots, f_{12}(x) = 0$. If $f_{13}(x) = 0$, $f_{14}(x) = 0$, $f_{15}(x) = 0$ are not satisfied, try Step 3 and Step 4 again and repeat it until the solution of $f_1(x) = 0, \cdots, f_{12}(x) = 0$ will be a solution of $f_{13}(x) = 0$, $f_{14}(x) = 0$ and $f_{15}(x) = 0$. Note that the probability that x satisfies such a condition is roughly q^{-3}.

If such x does not satisfies $f_{16}(x) = 0$, try choose another parameter in Step 2, try Step 3 and 4 again and repeat it until the solution of $f_1(x) = 0, \cdots, f_{15}(x) = 0$ will be a solution of $f_{16}(x) = 0$. Since the probability that x satisfies such a condition is roughly q^{-1}, we see that the complexity of this attack is about

$$49^{3.5} + 2^4 \times (23^{3.5} + 2^{12} \times (10^{3.5} + 2^4 \times 5^{3.5}) \sim 2^{28.9}.$$

We now summarize the complexities of our attack proposed in this section and other known attacks in the table below. From this table, we see that our attack is much better than the others and the UOV with such parameters are not secure at all.

(q, m, n)	$(2^4, 16, 48)$	$(2^4, 16, 64)$
exhaustive	2^{64}	2^{64}
Courtois et al. [1]	2^{46}	2^{42}
Faugére-Perret [4]	$2^{40.5}$	$2^{40.5}$
Our attack	$2^{28.9}$	$2^{25.9}$

7 Conclusion

In this paper, we propose two algorithms to solve massively under-defined systems of quadratic equations and give an analysis of UOV with the parameter suggested in [6] as an application. It is obvious that the cryptosystems based on quadratic equations with $n \gg m$ is not good for the efficiency. Also the work in [6] and our work show that such cryptosystems are not good also for the security. Of course, in most multivariate public key cryptosystems, n is not much larger than m (see [5], [9] and their references). However, our algorithm might be able to be improved in the future, or might help some experimental attacks (e.g. the Gröbner basis attacks and the XL algorithms) to be more effectively. Thus we claim that it is important to establish better algorithms and to study the lower bound of n such that a set of m equations is solved effectively.

References

[1] N. Courtois, L. Goubin, W. Meier and J. Tacier, *Solving underdefined systems of multivariate quadratic equations*, PKC'02, LNCS **2274**, pp.211–227.

[2] N. Courtois, A. Klimov, J. Patarin and A. Shamir, *Efficient algorithms for solving overdefined systems of multivariate polynomial equations*, Eurocrypt'00, LNCS **1807**, pp.392–407.

[3] N. Courtois and J. Pieprzyk, *Cryptanalysis of Block Ciphers with Overdefined Systems of Equations*, Asiacrypt'02, LNCS **2501**, pp. 267–287.

[4] J. Faugére and L. Perret, *On the security of UOV*, Proceedings of SCC'08, pp.103–109.

[5] J. Ding, J. Gower and D. Schmidt, *Multivariate public key cryptosystems*, Advances in Information Security, Springer, 2006.

[6] A. Kipnis, J. Patarin and L. Goubin, *Unbalanced Oil and Vinegar Signature Schemes*, Eurocrypt'99, LNCS **1592** (1999), pp. 206–222, extended in *citeseer/231623.html, 2003-06-11.*

[7] A. Kipnis and A. Shamir, *Cryptanalysis of the Oil and Vinegar signature scheme*, Crypto'98, LNCS **1462** (1998), pp. 257–267.

[8] J. Patarin, *The Oil and Vinegar algorithm for signatures*, Dagstuhl Workshop on Cryptography, 1997.

[9] S. Tsujii, T. Kaneko, K. Tadaki and M. Gotaishi, *Design Policy of MPKC based on Piece in Hand Concept (in Japanese)*, IEICE Technical Report **108** (2008), pp.15–22.

HASHIMOTO, Yasufumi
Institute of Systems, Information Technologies and Nanotechnologies,
7F 2-1-22, Momochihama, Fukuoka 814-0001, JAPAN
e-mail:hasimoto@isit.or.jp