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

Yasufumi HASHIMOTO (ISIT, Japan)

Partially supported by JST Strategic Japanese-Indian Cooperative Programme on multidisciplinary Research Field, which combines Information and Communications Technology with Other Fields, entitled "Analysis of cryptographic algorithms and evaluation on enhancing network security Based on Mathematical Science"

Introduction

MQ (multivariate quadratic) problem.

Let $n, m \geq 1$ be integers and $f_1, \cdots, f_m$ quadratic forms of $x = (x_1, \cdots, x_n)$.

MQ problem is the problem to find $x = (x_1, \cdots, x_n)$ such that $f_1(x) = 0, \cdots, f_m(x) = 0$ for given $f_1, \cdots, f_m$.

This problem over finite fields is used to cryptography.
(ex. Matsumoto-Imai, HFE, UOV, STS, TTM etc).

Known: MQ prob. is NP-hard.
(Secure against quantum attacks?)

However, not all quadratic equations are difficult to be solved (some MQ cryptosystems were already broken).

Q. Which equations are weak?
How to characterize its weakness?

Kipnis-Patarin-Goubin[Eurocrypt'99] found that if $n \geq m^2 + m$ then the equations can be solved
(i) in polynomial time for **even** char. field,
(ii) with $2^m \times$ (polyn.) complexity for **odd** char. field.

Q. Is the condition between n and m improved?

Main results.

1. When $n \geq$ (about)$m^2 - 2m^{3/2} + 2m$,
the equations can be solved in polynomial time
(**both** for even and odd char. fields).

2. When $n \geq \frac{1}{2}m(m+1) + 1$,
the equation can be solved
with $O(2^m)$ complexity for **even** char. cases,
and with $O(3^m)$ complexity for **odd** char. cases.

Notations.

q: a prime power,

k: a finite field of order q,

$m, n \geq 1$: integers,

$x = (x_1, \cdots, x_n)^t \in k^n$: variables,

$f_1(x), \cdots, f_m(x) \in k$ **:** quadratic forms of x**,**

F_i: $n \times n$ matrix such that $f_i(x) = x^t F_i x +$ (linear).

(e.g. If $f_i(x) = x_1^2 + 4x_1x_2 + 3x_2^2 + x_1 + 2x_2 + 1$

then $F_i = \begin{pmatrix} 1 & 2 \\ 2 & 3 \end{pmatrix}$)

Kipnis-Patarin-Goubin's method

1. Take linear transforms $m-1$ times such that, for $1 \leq i \leq m$,

$$F_i \mapsto \left(\begin{array}{cc|c} * & 0 & \\ 0 & * & * \\ \hline & * & * \end{array}\right) \mapsto \left(\begin{array}{ccc|c} * & 0 & 0 & \\ 0 & * & 0 & * \\ 0 & 0 & * & \\ \hline & * & & * \end{array}\right) \mapsto \cdots$$

$$\cdots \mapsto \left(\begin{array}{ccc|c} * & 0 & \cdots & 0 & \\ & \ddots & \ddots & \vdots & \\ 0 & \ddots & \ddots & & * \\ \vdots & \ddots & \ddots & 0 & \\ 0 & \cdots & 0 & * & \\ \hline & * & & & * \end{array}\right).$$

Finding such transforms requires to solve at most $m(m-1)$ linear equations with n variables.

Then we have

$$f_i(x) \mapsto \sum_{j=1}^{m} \alpha_{ij} x_j^2 + \sum_{j=1}^{m} x_j((x_{m+1}, \cdots, x_n)\text{-linear}) + ((x_{m+1}, \cdots, x_n)\text{-quadratic}).$$

2. Substitute values into $x_{m+1}, \cdots, x_n$ such that $f_i(x) \mapsto \sum_{j=1}^{m} \alpha_{ij} x_j^2 + \beta$.

This requires to solve m^2 linear equations with $n - m$ variables (thus $n \geq m(m+1)$).

3. The problem to solve

$$f_1(x) = 0, \cdots, f_m(x) = 0$$

is reduced to the problem to solve

$$x_1^2 = \gamma_1, \cdots, x_m^2 = \gamma_m.$$

This can be solved

(i) in polynomial time when q is even,

(ii) with $2^m \times$ polyn.-complexity when q is odd.

When $n \geq$(about)$m^2 - 2m^{3/2} + 2m$

Additional notations.

For $x = (x_1, \cdots, x_n)^t$,
$\tilde{x} := (x_0, x_1, \cdots, x_n)^t$,

For $f_1(x), \cdots, f_m(x)$,
$\tilde{f}_1(\tilde{x}), \cdots, \tilde{f}_m(\tilde{x})$: homogeneous quadratic forms such that
$\tilde{f}_i(1, x_1, \cdots, x_n) = f_i(x_1, \cdots, x_n)$,

(e.g. If $f(x_1) = 3 + 2x_1 + x_1^2$,
then $\tilde{f}(x_0, x_1) = 3{x_0}^2 + 2x_0x_1 + x_1^2$),

$\tilde{F}_i$: $(n+1) \times (n+1)$ matrix such that $\tilde{f}_i(\tilde{x}) = \tilde{x}^t \tilde{F}_i \tilde{x}$.

Elementary fact.

Let $U = (u_{ij})_{0 \leq i,j \leq n}$ be an invertible $(n+1) \times (n+1)$ matrix. If U satisfies that $u_{00} \neq 0$ and

$$U^t \tilde{F}_i U = \left(\begin{array}{c|c} 0_{1\times 1} & * \\ \hline * & * \end{array} \right)$$

for $1 \leq i \leq m$, then $x = (u_{00}^{-1} u_{01}, \cdots, u_{00}^{-1} u_{0n})$ is a solution of $f_1(x) = 0, \cdots, f_m(x) = 0$.

We will find such U instead of solving the equations.

Algorithm A.

Let G be a square matrix. Then one can find an invertible matrix U such that

$$U^t G U = \begin{pmatrix} 0 & \cdots & 0 & * \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \ddots & & \vdots \\ * & \cdots & \cdots & * \end{pmatrix}$$

in polynomial time.

How to do? A variant of triangulation.

Algorithm B.

Suppose that $n > m^2$ and $F_1, \cdots, F_m$ are $n \times n$ square matrices. Then one can find an invertible U such that

$$U^t F_i U = \begin{pmatrix} 0_{m\times m} & * \\ * & * \end{pmatrix}$$

for $1 \leq i \leq m$ in polynomial time.

Step 1. Find U_1 such that

$$U_1^t F_1 U_1 = \begin{pmatrix} 0_{m\times m} & * \\ * & * \end{pmatrix}.$$

Step 2. We "want to" find U_2 such that

$$U_2^t F_1 U_2, U_2^t F_2 U_2 = \begin{pmatrix} 0_{m\times m} & * \\ * & * \end{pmatrix}.$$

Step 3. We "want to" find U_3 such that

$$U_3^t F_1 U_3, U_3^t F_2 U_3, U_3^t F_3 U_3 = \begin{pmatrix} 0_{m\times m} & * \\ * & * \end{pmatrix}.$$

$$\vdots$$

Assume that, until Step $N-1$, we can find U_{N-1} such that

$$U_{N-1}^t F_i U_{N-1} = \begin{pmatrix} 0_{m\times m} & * \\ * & * \end{pmatrix}$$

for $1 \leq i \leq N-1$.

Step N.

N-0. Find $m \times m$ matrix V_N such that

$$\begin{pmatrix} V_N^t & \\ & I \end{pmatrix} F_N \begin{pmatrix} V_N & \\ & I \end{pmatrix} = \left(\begin{array}{cccc|c} 0 & \cdots & 0 & * & \\ \vdots & \ddots & \iddots & \vdots & \\ & & & \vdots & * \\ 0 & \ddots & & \vdots & \\ * & \cdots & \cdots & * & \\ \hline & & * & & * \end{array}\right)$$

by Alg. A. Note that

$$F_1, \cdots, F_{N-1} \mapsto \begin{pmatrix} 0_{m\times m} & * \\ * & * \end{pmatrix}.$$

$N-1$. Choose a linear transform $x_m \mapsto a_1 x_1 + \cdots + a_n x_n$ such that

$$F_1, \ldots, F_{N-1} \mapsto \begin{pmatrix} 0_{m\times m} & * \\ * & * \end{pmatrix},$$

$$F_N \mapsto \left(\begin{array}{ccccc|c} 0 & \cdots & \cdots & 0 & 0 & \\ \vdots & & \iddots & * & * & \\ \vdots & \iddots & \iddots & \iddots & \vdots & * \\ 0 & * & \iddots & & \vdots & \\ 0 & * & \cdots & \cdots & * & \\ \hline & & * & & & * \end{array}\right).$$

This requires to solve

(1) $(N-1) \times (m-1)$ linear equations of $(x_{m+1}, \ldots, x_n)$,

(2) a linear equation of $(x_m, \ldots, x_n)$,

(3) $N-1$ quadratic equations of $(x_1, \ldots, x_n)$, given by

$$\sum_{i=1}^{m} x_i((x_{m+1}, \ldots, x_n)\text{-linear}) + ((x_{m+1}, \ldots, x_n)\text{-quadratic}) = 0$$

Thus this can be solved by the linear operations.

$$\vdots$$

Repeating such computations, we can find an invertible linear transform U such that

$$U^t F_i U = \begin{pmatrix} 0_{m\times m} & * \\ * & * \end{pmatrix}$$

for $1 \leq i \leq \min(n/m, m)$ in polynomial time.

Thus this works when $n > m^2$.

Solving quadratic equations.

$N_0 = n + 1$.

Step 1. Choose $M_1 < \sqrt{N_0}$ and put $N_1 := \lfloor N_0/M_1 \rfloor$.
Find $N_0 \times N_0$ matrix U_1 such that

$$U_1^t \tilde{F}_i U_1 = \begin{pmatrix} 0_{N_1 \times N_1} & * \\ * & * \end{pmatrix}$$

for $1 \leq i \leq M_1$.

Step 2. Choose $M_2 < \sqrt{N_1}$ and put $N_2 = [N_1/M_2]$.
Find $N_1 \times N_1$ matrix U_2 such that

$$\begin{pmatrix} U_2^t & \\ & I \end{pmatrix} \tilde{F}_i \begin{pmatrix} U_2 & \\ & I \end{pmatrix} = \begin{pmatrix} 0_{N_2 \times N_2} & * \\ * & * \end{pmatrix}$$

for $M_1 + 1 \leq i \leq M_1 + M_2$.

$$\vdots$$

Repeat such operations and stop when $M_t = 1$.
Then we can find $U = (u_{ij})_{0 \leq i,j \leq n}$ such that

$$U^t \tilde{F}_i U = \begin{pmatrix} 0_{1 \times 1} & * \\ * & * \end{pmatrix}$$

for $1 \leq i \leq M_1 + M_2 + \cdots + M_t$.

Thus we can solve equations in polynomial time when

$$m \leq M_1 + M_2 + \cdots \sim n^{1/2} + n^{1/4} + \cdots ,$$

namely

$$n \geq \text{ (about)} m^2 - 2m^{3/2} + 2m.$$

Kipnis-Patarin-Goubin: $n \geq m^2 + m$.

m	$\cdots$	14	15	16	17	18	19	$\cdots$
n for Alg. 1	$\cdots$	100	121	144	156	169	196	$\cdots$
n for [KPG]	$\cdots$	210	240	272	306	342	380	$\cdots$

When $n \geq \frac{1}{2}m(m+1)+1$.

Step 1. Find a linear transform $x_0 \mapsto a_0x_0 + a_1x_1 + \cdots + a_nx_n$ such that

$$\tilde{F}_1 \mapsto \left(\begin{array}{c|c} 0_{1\times 1} & * \\ \hline * & * \end{array}\right).$$

This requires to solve one quadratic equation.

It should be $n \geq 2$.

Step 2. Find a linear transform

$x_1 \mapsto a_0x_0 + a_1x_1 + \cdots + a_nx_n$ such that

$$\tilde{F}_1 \mapsto \left(\begin{array}{cc|c} 0 & 0 & * \\ 0 & 0 & \\ \hline * & & * \end{array}\right), \qquad \tilde{F}_2 \mapsto \left(\begin{array}{cc|c} * & 0 & * \\ 0 & * & \\ \hline * & & * \end{array}\right).$$

This requires to solve 2 linear eq. and one quadratic eq. Then it must be $n \geq 4$.

Next, find a transform $x_0 \mapsto a_1x_0 + a_2x_1$ such that

$$\tilde{F}_2 \mapsto \left(\begin{array}{cc|c} 0 & * & * \\ * & * & \\ \hline * & & * \end{array}\right).$$

Step 1 and 2 solve 2 quadratic equations with more than 4 variables.

Step 3. Find $x_1 \mapsto a_0x_0 + a_1x_1 + \cdots + a_nx_n$ such that

$$\tilde{F}_1, \tilde{F}_2 \mapsto \left(\begin{array}{cc|c} 0 & 0 & * \\ 0 & 0 & \\ \hline * & & * \end{array}\right), \qquad \tilde{F}_3 \mapsto \left(\begin{array}{cc|c} * & 0 & * \\ 0 & * & \\ \hline * & & * \end{array}\right).$$

This requires to solve 3 linear eq. and 2 quadratic eq.
Then it must be $n \geq 7$.

Next, find $x_0 \mapsto a_1x_0 + a_2x_1$ such that

$$\tilde{F}_3 \mapsto \left(\begin{array}{cc|c} 0 & * & * \\ * & * & \\ \hline * & & * \end{array}\right).$$

Step 1 - 3 solve 3 quadratic eq with more than 7 variables.

$$\vdots$$

Repeat such operations.
We see that the quadratic equations can be solved when $n \geq \frac{1}{2}m(m+1)+1$.

However, the computational task is roughly estimated by $O(2^m)$ when q is even,
and $O(3^m)$ when q is odd.

Application: Analysis of UOV.

UOV [Patarin(1998), Kipnis-Patarin-Goubin(1999)]:

One of MQ signature schemes.

The signature is generated by linear operations.

If q^{n-2m} is small, UOV is broken.

Suggested parameter by KPG (2003):

$$q = 2^4, \quad m = 16, \quad n = 48 \text{ or } 64.$$

Our attack. Apply the first algorithm directly.

This solves 9 equations for 48 variables,
and 11 equations for 64 variables.

Inserte exhausitive searches for the remaining 7 or 5 equations.

$$n = 48, \qquad q^4 \left(\omega(49) + q^2 \left(\omega(7) + q\omega(3)\right)\right) \sim 2^{36.4},$$
$$n = 64, \qquad q \left(\omega(65) + q^3 \left(\omega(8) + q\omega(3)\right)\right) \sim 2^{26.4},$$

where $\omega(n) \sim n(n-m)^3/3$ is the complexity for Alg. B.

Comparison to past attacks.

(q, m, n)	$(2^4, 16, 48)$	$(2^4, 16, 64)$
exhaustive	2^{64}	2^{64}
Courtois et al (PKC'02)	2^{46}	2^{42}
Faugére-Perret(SCC'08)	$2^{40.5}$	$2^{40.5}$
Our attack	$2^{36.4}$	$2^{26.4}$

Conclusion

We found algorithms to solve multivariate quadratic equations

1. in polynomial time when $n \geq$ (about) $m^2 - 2m^{3/2} + 2m$,

and

2. with $O(2^m)$ or $O(3^m)$ complexities when $n \geq \frac{1}{2}m(m+1) + 1$.

Open problem.

How can one reduce the lower bound of n solving m equations effectively?