Building Instances of TTM Immune to the Goubin-Courtois Attack and the Ding-Schmidt Attack

T. Moh*
Math Department, Purdue University, West Lafayette, Indiana 47907-1395

J.M.Chen
Math Department, National Taiwan U, Taipei, Taiwan

Boyin Yang
Math Department, Tamkang University, Taipei, Taiwan

July 21, 2004

Abstract

We think that there are two main attacks on TTM cryptosystem; the Goubin-Courtois attack ([6]) and the Ding-Schmidt attack ([5]). The paper of Goubin-Courtois is not clearly written. Their arguments (with many gaps) depend on an parameter r which is never defined. It is nature to take their parameter r to be the index s used in our "lock polynomials" (see section 1). Later on Courtois implies otherwise in his website. In their paper ([6]) or in his website, Courtois simply declares that TTM is of rank 2 (i.e., $r = 2$) without any justification. In this paper, we will illustrate another example (cf Example below) satisfies both requirements, i.e., the index s used in our "lock polynomials" (see section 1) is 7, and the number of variables in all quartic forms is 4 which shows that Goubin-Courtois' unsubstantial claim: "TTM is rank 2" invalid. Thus we settle this question of Goubin-Courtois attack once for all. To guard against "high rank attack", in this Example every variable appears 9 times in 9 different polynomials. On the other hand J. Ding and D. Schmidt show ([5]) how to construct an interesting attack on some implementations of TTM ([10,11]) based on Patarin's idea ([14]) of bilinear relations created by the structure in the kernel equations in an implementation of TTM. The success of this attack is accidental. In our Example, the attack fails. we will describe a *mixed implementation* which will make any attack, which is sensitive to the size of the ground field, ineffective. In this paper, the Example is strong (i.e., $\geq 2^{148}$) against both Goubin-Courtois attack and Ding-Schmidt attack as well as other previously proposed incomplete attacks like XL($> 2^{97}$), FXL($> 2^{112}$).

1 Introduction

Many recent advances in cryptography have been based on finite-field arithmetic. TTM is a family of multivariate public-key encryption schemes based on the concept of Tame Transformations. It is one of the fastest encryption schemes known and is fast enough that it does not need to be used in conjunction with a symmetric block cipher (like 3DES or AES) for a medium size document (one million bytes or less per second).

For reader's convenience, we will fix the notations and re-introduce the TTM (*Tame Transformation Method*) in the following subsections.

* e-mail ttm@math.purdue.edu

1.1 Tame Transformations and TTM: Lock Polynomials

A *Tame Transformation* $\phi : \mathbf{x}(\in K^n) \mapsto \mathbf{y}(\in K^{n+k})$ is usually given as a set of relations (where each q_i is a polynomial, and the subscripts can be permuted):

$$
\begin{array}{rcl}
y_1 &=& x_1; \\
y_2 &=& x_2 + q_2(x_1); \\
\vdots & \vdots & \vdots \\
y_i &=& x_i + q_i(x_1, x_2, \dots, x_{i-1}); \\
\vdots & \vdots & \vdots \\
y_n &=& x_n + q_n(x_1, x_2, \dots, x_{n-1}); \\
y_{n+1} &=& q_{n+1}(x_1, x_2, \dots, x_n); \\
\vdots & \vdots & \vdots \\
y_{n+k} &=& q_{n+k}(x_1, x_2, \dots, x_n).
\end{array}
$$

Tame Transformations are injective; when $k = 0$ they are also bijective and are called *Tame Automorphisms*. Tame Transformations had been first researched extensively in the field of algebraic geometry, but its was first proposed by T. Moh for use in public-key cryptography infrastructure ([10]). They possess the desirable property that

1. A preimage $\mathbf{x} = \phi^{-1}(\mathbf{y})$ can be computed quickly by solving for each component serially, but:
2. an explicit polynomial form for ϕ^{-1} is hard to write out in full and of very high degree:

$$
\begin{array}{rcl}
x_1 &=& y_1; \\
x_2 &=& y_2 - q_2(x_1); \\
x_3 &=& y_3 - q_3(x_1,\, x_2) = y_3 - q_3(y_1,\, y_2 - q_2(y_1)); \\
\vdots & \vdots & \vdots \\
x_n &=& y_n - q_n(x_1,\, x_2, \dots,\, x_{n-1}) = y_n - q_n(y_1, y_2 - q_2(y_1) \dots y_{n-1} - q_{n-1}(\cdots)).
\end{array}
$$

In all proposed TTM cryptosystems, the *base field* is unspecified. As usual, we amy consider $K = \mathrm{GF}(2^8)$ or $K = GF(2^{16})$ or even larger, When at first suggested, the TTM cryptosystem has a bijective public map that looks like $\mathbf{w} \mapsto \mathbf{z} = L_2 \circ T \circ L_1(\mathbf{w})$, where L_1 and L_2 are affine (linear) and T is tame and inhomogeneous quadratics for each q_i, and an expansion rate of 1 during encryption. This turned out to be insecure, due to the attacks by P. Montgomery and A. Sathaye ([13]) because the first coordinate in the tame portion is fixed, as is practically the second coordinate, since q_2 is essentially constricted to $x \mapsto x^2$.

The Montgomery-Sathaye attack are **rank attacks**, due to differing rank in the kernel equations. Coppersmith, Stern, and Vaudenay broke Shamir's Birational Permutation Schemes ([4]) in a similar manner by looking for common kernel spaces. Theobald issue a warning ([16]) that "varying ranks of quadratic forms may cause security concerns." Briefly, the attack looks for linear combinations of the z_i so that there are no quadratic terms in the $\mathbf{w}$; then it looks for linear combinations with no square-free quadratic terms (crossterms $w_i w_j$). That gives y_1 and y_2. The whole scheme then unravels.

To get around this flaw, since the initial published implementations TTM ([10]) has always used an LTTL form $\mathbf{y} = \phi_4 \circ \phi_3 \circ \phi_2 \circ \phi_1(\mathbf{x})$. The maps ϕ_1 and ϕ_4 are affine. ϕ_2 **is from K^n to K^m with**

$n < m$**, and** ϕ_3 **is bijective in** K^m **with this form (where** $1 < s < m$):

$$\begin{aligned}
y_1 &= x_1 + p_1(x_2, x_3, \dots, x_m);\\
\vdots &= \vdots\\
y_s &= x_s + p_s(x_{s+1}, \dots, x_m);\\
y_{s+1} &= x_{s+1};\\
\vdots &= \vdots\\
y_m &= x_m;
\end{aligned}$$

such that the degrees of the polynomials p_j's are suitably large but the composition $\phi_3 \circ \phi_2 : K^n \to K^m$ are quadratic in each component of the image. Thus, $\phi = \phi_4 \circ \phi_3 \circ \phi_2 \circ \phi_1$ looks like a generic quadratic with m degree-2 equations of n variables each, and is given in the composite form as the public key. The functions $p_1, \dots, p_s$ are called **lock polynomials** for that particular instance of TTM. They protect TTM from rank-related attacks such as Montgomery-Sathaye.

1.2 About Goubin-Courtois Minrank Attack

What is the rank r of a cryptosystem? Goubin-Courtois claim unsubstancially that TTM is of rank 2 (i.e., r=2) in their paper. There are possibly two independent interpretations of the number r, since the term is not defined; either the number r is the index s in our **lock polynomials** or the minimal number of variables in all quadratic forms q_i's and their linear combinations. In section 4 we will give an example (see the Example below) to show that according to either interpretations, the number r is ≥ 4. It establishs that their important claim is wrong. Moreover their attack will be ineffective even according to their claimed formula of $q^{(rank)\lceil \frac{m}{n} \rceil} \times m^3$.

To say the most, their attack is about the key sizes of TTM and not about the security. The reason is as follows; let the matrix M we are searching for be with rank r. Then the probability of finding a random vector in its kernel is

$$q^{-r}$$

where q is the size of the ground field. If we change q from 2^8 to 2^{16}, then the probability decreases a lot. If we enlarge the ground field to GF(2^{16}), there is no doubt that it will speed up the encryption by the new larger capacity and it will slow down because the complication of multiplication and the size of the keys. All these troubles can be easily handled by a "mixed implementation"; which means the "diagram working" (or "table walking") is used for multiplication and "subfield reduction" which is simply put coefficients in $GF(2^{16})$ at some limited number of positions while keeping the remaining coeffients in some smaller subfields. In the following we will show by the discussions of Example (below) that this "mixed implementation" can sometimes reduce the total length of the program and sometimes even speed up encryption. Since TTM encryptsystem reaches a speed of tens of millions bits per second, an acceleration might be significant.

1.3 The Ding-Schmidt Attack

The attack by Ding and Schmidt can be called Trivialization of the **Lock**. It is based on the same idea as Patarin's. For illustrative purposes, suppose (noting that we are dealing with a characteristic 2 field) $y_9 = x_9 + x_3x_8$, $y_{37} = x_9 + x_7x_8$, and $y_{57} = x_9 + x_8x_{11}$, then we have $(y_{37} + y_9)(x_{11} + x_3) = (y_{57} + y_9)(x_7 + x_3)$. Suppose with $\mathbf{y} = \phi_3 \circ \phi_2(\mathbf{x})$, there are relations

$$a_i + \mathbf{x}^T\mathbf{b}_i + \mathbf{c}_i^T\mathbf{y} + \mathbf{x}^T D_i\mathbf{y} = 0 \tag{1}$$

between the variables x_i and y_j; and if invertible affine maps ϕ_1 and ϕ_4 relate $\mathbf{w}$ to $\mathbf{x}$ and $\mathbf{y}$ to $\mathbf{z}$ respectively, we have similar bilinear relations between $\mathbf{w}$ and $\mathbf{z}$, heretofore termed **Patarin Relations**:

$$a'_i + \mathbf{w}^T \mathbf{b}'_i + (\mathbf{c}')_i^T \mathbf{z} + \mathbf{w}^T D'_i \, \mathbf{z} = 0. \tag{2}$$

As a result, given a public map $\mathbf{w} \mapsto \mathbf{z}$, we can try to construct all the Patarin relations in the forms of Eq. (2), by generating random $\mathbf{w}$'s and computing corresponding $\mathbf{z}$'s. Each $(\mathbf{w}, \mathbf{z})$ pair is an equation of the unknowns $(a', \mathbf{b}', \mathbf{c}', D')$. Using enough samples and Gaussian elimination, we can eventually find a maximal independent set of relations in the form of Eq. (2). According to [13], it almost never takes many more tries than the number of variables.

Having found this maximal independent set, for any given ciphertext $\mathbf{z}$, we can attempt to decrypt it by substituting the $\mathbf{z}$'s into all Patarin relations we found and solving for $\mathbf{w}$. In general, this is insufficient to determine all the w_j. However, it often leads to **the trivialization of the lock**.

What does this mean? Unbeknownst to the attacker, underneath each Patarin relation is a relation of the form Eq.(2) between the $\mathbf{x}$ and the $\mathbf{y}$. These are in turn linear combinations of relations shown above, that result from the basic structure of that particular TTM instance. Of course, he is unable to unscramble them from his collection of Patarin relations. However, the intrinsic effect of solving the set of Patarin relations for $\mathbf{w}$ is to find linear relations among the $\mathbf{x}$. Occationally, all the independent variables x_i in the **lock polynomials** p_1 and p_2 may become fixed.

It is not fatal that Patarin relations exist, because after using them to eliminate the unnecessary variables in $\mathbf{w}$, there are still a lot of independent w_i left, and the TTM structure remains largely intact. An attacker still need to solve a big set of equations. However, it is fatal if p_1 and p_2 become constants. Indeed, it would be fatal if they become perfect squares. Because then the Montgomery-Sathaye attack again works, and will lead to the set of plaintext $\mathbf{w}$ that generates the ciphertext $\mathbf{z}$. Worse, it is not a sure thing but it might well provide a successful decomposition of the public map.

The above attack, using Patarin's bilinear relations, is called a Ding-Schmidt attack of the first level. There is a second level of attack. We can extend the Patarin relation to something like this:

$$a'_i + \mathbf{w}^T \mathbf{b}'_i + (\mathbf{c}')_i^T \mathbf{z} + \mathbf{w}^T D'_i \mathbf{z} + \mathbf{z}^T E'_i \mathbf{z} = 0.$$

Where E'_i are upper-triangular matrices. We can expect to find additional relations and proceed further in the elimination of variables. Indeed, we may push the concept further and look for this:

$$a + \sum_i b_i w_i + \sum_j c_i z_i + \sum_{i,j} d_{ij} w_i z_j + \sum_{i \le j} e_{ij} z_i z_j + \sum_{i,j \le k} f_{ijk} w_i z_j z_k + \sum_{i \le j \le k} g_{ijk} z_i z_j z_k = 0.$$

This is what we call an attack of the fourth level for obvious reasons. While increasing the level of the attack leads to more linear relations and more attacking power, there is a point of diminishing returns. Also, we estimate that at the fifth level, where we need to have terms $h_{ijkl} w_i z_j z_k z_l$, the number of terms become too many for a machine to hold in main memory.

We may select an TTM cryptosystem immuned to the above attack. The basic problem is that there are not enough equations of the form (2) above. Usually Ding-Schmidt attack can not trivialize the lock as in the Example (see below).

1.4 Some Comments on the Attack as Presented in [5].

Clearly, there are something missing in their description. Their claims: "we can show that it takes about 2^{32} computations on a finite field of size 2^8 to defeat the scheme and we performed a computation example on a PC (450Mhz) and defeated it in a few hours". It is easy to see that a PC of 450Mhz can perform 2^{32} computations in less than 9 seconds. What does they mean by "a few hours", any number of hours between 1 and 24, maybe 12 hours. Than it is off by a factor of 5,000. There might be something unusual.

They were talking about the challenge problems of USDSi.com. The challenge problems of USDSi.com are about the private keys as stated clearly by that webpage. The present attack of Ding and Schmidt is about a different matter, i.e., to crack the plaintext giving a cyphertext in a few hours. So it is unrelated to the webpage of USDSi.com. and unlikely to solve those problems. It is a mild surprise to us that Goubin and Courtois misunderstood also the nature of the challege problems of USDSi.com. It is known that the private keys can be used for some other purposes as personal identification in a signature scheme. Usually, an instantaneous solution is demanded. A few hours time to response will automatically identify a forger.

The attack of Ding and Schmidt relies on the presence of the **whole** public key. As point out in [10], the **signature** scheme of TTM depends on the hidden subspaces, which is a totally different matter. The **signature** schemes of TTM as presented (which is known as TTS) by the two junior authors of this paper in [1] & [2] hide a portion of $\{y_i\}$ and the part of the **lock** of TTM, hence immune from the attack of Ding and Schmidt.

2 Diagram-walking, and mixed implementation

2.1 Diagram-walking and Multiplication

Let us fix the fields. Let $F_0 = GF(2), F_1 = GF(2^2), F_2 = GF(2^4)$, $F_3 = GF(2^8), F_4 = GF(2^{16})$ and $F_1 = F_0(a), F_2 = F_1(b), F_3 = F_2(c), F_4 = F_3(d)$ where a, b, c, d satisfying the following equations,

$$\begin{array}{ll} a^2+a+1=0, & b^2+ab+1=0 \\ c^2+bc+1=0, & d^2+cd+1=0 \end{array}$$

For instance, an element $\delta_0 + \delta_1 a + (\delta_2 + \delta_3 a)b \in F_2$ where $\delta_i = 0, 1$ will be represented as $[\delta_0, \delta_1, \delta_2, \delta_3]$ in the computer.

Note that

$$\begin{array}{l} [\delta_0, \delta_1] \times [\epsilon_0, \epsilon_1] \\ = [\delta_0, 0] \times [\epsilon_0, \epsilon_1] + [0, \delta_1] \times [\epsilon_0, \epsilon_1] \\ = \delta_0 \times [\epsilon_0, \epsilon_1] + \delta_1 \times [\epsilon_1, \epsilon_0 + \epsilon_1] \end{array}$$

The last line of computation can be easily gotten by considering $[\delta_0, \delta_1]$ as "instruction". For the second elements of the above multiplication in a larger field, one may simply treat it as many 2-blocks and do acordingly. This method can be extended to multiplications invloving one element from $F_2 = GF(2^4)$ as follows, first we treat the second element n the multiplication in the larger field as many 4-blocks, second for each block we do the following,

$$\begin{array}{l} [\delta_0, \delta_1, \delta_2, \delta_3] \times [\epsilon_0, \epsilon_1, \epsilon_2, \epsilon_3] \\ = [\delta_0, \delta_1, 0, 0] \times [\epsilon_0, \epsilon_1, \epsilon_2, \epsilon_3] + [0, 0, \delta_2, \delta_3] \times [\epsilon_0, \epsilon_1, \epsilon_2, \epsilon_3] \\ = [\delta_0, \delta_1] \times [\epsilon_0, \epsilon_1, \epsilon_2, \epsilon_3] + [\delta_2, \delta_3] \times [\epsilon_2, \epsilon_3, (\epsilon_0 + \epsilon_3), (\epsilon_1 + \epsilon_2 + \epsilon_3)] \end{array}$$

Note that the last line consists of multiplication involving elements in $GF(2^2)$. For multiplication involving an element in $F_3 = GF(2^8)$, one usually uses "table lookup". We either have a multiplication table $F_3 \times F_3$ (64 k byte) or a table of log-exp (512 byte). It can be summarised up as

Case 1: At least one of x, y is in F_3, the other is in F_3 or F_4.

(A): For multiplicative table. Let $x, y \in F_3$. Then one (1) lookup for $x \times y$. Let $x \in F_3, y \in F_4$. Then two (2) lookups for $x \times y$.

(B): For a table of log-exp. Let $x, y \in F_3$. Then

$$x \times y = exp(log\, x + log\, y)$$

Clearly, it takes three (3) lookups. Let $x \in F_3, y \in F_4$. Then it takes six (6) lookups.

Case 2: Both x, y are in F_4.

Let us consider multiplications involved two elements in F_4. We have

$$\begin{aligned}&(g_0 + g_1 d)(h_0 + h_1 d)\\&= (g_0 h_0 + g_1 h_1) + (g_1 h_0 + g_0 h_1 + c g_1 h_1)d\end{aligned}$$

(A): For multiplicative table. Then six (6) lookups for $x \times y$.

(B): For a table of log-exp. Since c is common, log c will be remembered. Therefore we need four lookups for log and five lookups for exp. Totally we need nine (9) lookups.

The above method is well-known as "**Diagram-walking or table-walking**". We may easily extend the set of fields to include $F_5 = GF(2^{32})$ etc..

2.2 Mixed implimentation

The cryptosystem TTM consists of map $\phi_4 \circ \phi_3 \circ \phi_2 \circ \phi_1$, where ϕ_1, ϕ_4 are *affine linear*,i.e., $L_1 + v_1, L_4 + v_4$ with L_1, L_4 linear over the vector spaces of 1-forms. The vector v_4 is determined by property that the total map sends the the origin of the affine space to the origin of the affine space, i.e.,. the resulting polynomials are without constant terms. If we ignore the constant terms, then we may ignore v_4. The mixed implementation is a strategy which put different coefficients in differnt fields to improve the security, to sometimes reduce the key length and to sometimes increase the speed. We will introduce a simple version of it and abuse the language to call it **mixed implementation**. It is the following;

(A): We select ϕ_3, ϕ_2, ϕ_1 as over a small field, say F_1 or F_2.

(B): We concentrate on the linear transformation L_4. It is an $m \times m$ invetible matrix. Let it be written as $L \times P$, where P is a general permutaion matrix. Let us only take L to have the first s_1 rows from F_1, the next s_2 rows from F_2, the next s_3 rows from F_3 and the next s_4 rows from F_4, (clearly we may extend this sequence to F_5 etc.), here $s_1 + s_2 + s_3 + s_4 = m$.

Note that every polynomial has $n(n+3)/2$ terms. The collection of all coefficients of all polynomials is the "public-key-length" is of the length,

$$(n(n+3)/2)(s_1/4 + s_2/2 + s_3 + 2s_4)$$

The plaintexts are from the field F_4. The speed can be roughly measured by the number of lookups. Let us use an unmixed implementation over F_3 as a yardstick with 2 round of encryption. The substitution of the values w into a polynomial can be written as

$$wMw^T + wv^T = w(Mw^T + v^T)$$

where M is an upper triangular matrix. Note that an umixed implementation, the number of multiplications of 2 rounds of encryption is

$$2(1/2)n(n+3)m = n(n+3)m$$

As for mixed implementation, note that for the first $s_1 + s_2$ polynomials for the mixed implementation, the part Mw^T can be carried out by "shifts", since all coefficients in M are from the smaller fields F_1 or F_2, therefore there are only $(s_1 + s_2)n$ multiplications in F_4. For the next s_3 polynomials, there are $2(1/2)n(n+1)s_3$ multiplications over F_3 (since the plaintext is over the field F_4, we have the factor 2 above), and ns_3 multiplications over the field F_4. For the next s_4 polynomials, there are $(1/2)n(n+1)s_4$ multiplications between elements in F_4, ns_4 multiplications over F_4 . Therefore the total lookups for encryption are in the following two cases;

(A): For a multiplication table. In this case, the number of lookups for encryption is

$$\begin{aligned}&6n(s_1+s_2)+n(n+1)s_3+6ns_3+(12/2)n(n+1)s_4+6ns_4\\&=(1/2)n(n+1)(2s_3+6s_4)+6nm\end{aligned}$$

While two rounds of unmixed implementation over F_3 requires

$$(1/2)n(n+3)2m$$

lookups for encryption for a comparison.

(B): For a table of log-exp: In this case, the number of lookups for encryption is

$$\begin{aligned}&9n(s_1+s_2)+3n(n+1)s_3+9ns_3+(9/2)n(n+1)s_4+9ns_4\\&=(1/2)n(n+1)(6s_3+9s_4)+9nm\end{aligned}$$

While two rounds of unmixed implementation over F_3 requires

$$(1/2)n(n+3)6m$$

lookups for encryption for a comparison.

In general, we have a sequence of quadratic field extension $F_0 \subset F_1 \subset \cdots \subset F_k$, and non-negative integers $s_0, s_1, \cdots, s_k$ with $\sum_0^k s_i = m$ and $s_k \geq 1$. We will define

Definition 1 *: If we have an $m \times m$ matrix L_4 with the first s_0 rows with entries from F_0, next s_1 rows with entries from F_1, $\cdots$, last s_k rows with entries from F_k. Then we have a mixed implementation, $MI(s_0, s_1, \cdots, s_k)$.*

■

The mixed implementation is effective against any attack (say, Goubin-Courtois' minrank attack) which is sensitive to the size of the ground field, since we may push the size of the field to $GF(2^{128})$, or the field F_7. The programmer may wish to keep the field as small as possible. We will give a detailed discussion about the security after the section of concrete implementations.

3 Example

We will give an example of TTM(55,110) with a mixed implementation $MI = (s_2, s_3, s_4) = (0, 96, 14)$ *or* $(28, 68, 14)$. The map $\pi = \phi_4 \circ \phi_3 \circ \phi_2 \circ \phi_1$ with ϕ_3, ϕ_2, ϕ_1 over $F_1 = GF(2^2)$. The map ϕ_2 is defined by the following system of quadratic polynomials over $F_1 = GF(2^2)$ in the variables $\{x_0, \cdots, x_{i-1}\}$, The quadratic polynomials $f_i = f_i(x_0, \cdots, x_{i-1})$ are random.

$$\begin{array}{ll}y_0 := x_0; & y_1 := f_1 + x_1;\\ y_2 := f_2 + x_2; & y_3 := f_3 + x_3;\\ y_4 := f_4 + x_4; & y_5 := f_5 + x_5;\\ y_6 := f_6 + x_6; & y_7 := f_7 + x_7;\\ y_8 := f_8 + x_8; & y_9 := f_9 + x_9;\\ y_{10} := f_{10} + x_{10}; & y_{11} := f_{11} + x_{11};\end{array}$$

$y_{12} := f_{12} + x_{12};$
$y_{13} := f_{13} + x_{13};$
$y_{14} := f_{14} + x_{14};$
$y_{15} := f_{15} + x_{15};$
$y_{16} := f_{16} + x_{16};$
$y_{17} := f_{17} + x_{17};$
$y_{18} := f_{18} + x_{18};$
$y_{19} := f_{19} + x_{19};$
$y_{20} := f_{20} + x_{20};$
$y_{21} := f_{21} + x_{21};$
$y_{22} := x_0x_5 + x_1x_4 + x_8 + x_{22};$
$y_{23} := x_0x_6 + x_2x_4 + x_{23};$
$y_{24} := x_1x_6 + x_2x_5 + x_{24};$
$y_{25} := x_3x_5 + x_1x_7 + x_{25};$
$y_{26} := x_3x_4 + x_0x_7 + x_{26};$
$y_{27} := x_2x_9 + x_{22}x_3 + x_{27};$
$y_{28} := x_6x_9 + x_{22}x_7 + x_{28};$
$y_{29} := x_{10}x_7 + x_8x_6 + x_{29};$
$y_{30} := x_{10}x_3 + x_2x_8 + x_{30};$
$y_{31} := x_{11}x_{16} + x_{12}x_{15} + x_{19} + x_{31};$
$y_{32} := x_{11}x_{17} + x_{13}x_{15} + x_{32};$
$y_{33} := x_{12}x_{17} + x_{13}x_{16} + x_{33};$
$y_{34} := x_{14}x_{16} + x_{12}x_{18} + x_{34};$
$y_{35} := x_{14}x_{15} + x_{11}x_{18} + x_{35};$
$y_{36} := x_{13}x_{20} + x_{31}x_{14} + x_{36};$
$y_{37} := x_{17}x_{20} + x_{31}x_{18} + x_{37};$
$y_{38} := x_{21}x_{18} + x_{19}x_{17} + x_{38};$
$y_{39} := x_{21}x_{14} + x_{13}x_{19} + x_{39};$
$y_{40} := x_{11}x_{12} + x_1x_0 + x_{39} + x_{40};$
$y_{41} := x_{11}x_2 + x_{13}x_0 + x_{41};$
$y_{42} := x_1x_2 + x_{13}x_{12} + x_{42};$
$y_{43} := x_3x_{12} + x_1x_{14} + x_{43};$
$y_{44} := x_3x_0 + x_{11}x_{14} + x_{44};$
$y_{45} := x_{32}x_{18} + x_{14}x_{33} + x_{13}x_{34} + x_{17}x_{35} + x_{45};$
$y_{46} := x_{37}x_{13} + x_{39}x_{18} + x_{36}x_{17} + x_{38}x_{14} + x_{46};$
$y_{47} := x_{23}x_7 + x_3x_{24} + x_2x_{25} + x_6x_{26} + x_{47};$
$y_{48} := x_{28}x_2 + x_{30}x_7 + x_{27}x_6 + x_{29}x_3 + x_{48};$
$y_{49} := x_{14}x_{41} + x_3x_{42} + x_{13}x_{43} + x_2x_{44} + x_{49};$
$y_{50} := x_{46}x_{48} + x_{47}x_{49} + x_{50};$
$y_{51} := x_{45}x_{47} + x_{48}x_{49} + x_{51};$
$y_{52} := x_{45}x_{48} + x_{46}x_{49} + x_{52};$
$y_{53} := x_{45}x_{49} + x_{46}x_{47} + x_{53};$
$y_{54} := x_{45}x_{46} + x_{47}x_{48} + x_{54};$
$y_{55} := x_{11}x_{42} + x_1x_{41} + x_{13}x_{39} + x_3x_{30};$
$y_{56} := x_0x_{43} + x_{12}x_{44} + x_2x_{29} + x_{14}x_{40};$
$y_{57} := x_0x_{42} + x_{12}x_{41} + x_2x_{39} + x_{14}x_{30};$
$y_{58} := x_{11}x_{43} + x_1x_{44} + x_{13}x_{29} + x_3x_{40};$
$y_{59} := x_{42}x_{44} + x_{41}x_{43};$
$y_{60} := x_{42}x_{29} + x_{39}x_{43} + x_{14};$
$y_{61} := x_{41}x_{29} + x_{39}x_{44} + x_3;$
$y_{62} := x_{30}x_{44} + x_{41}x_{40} + x_{13};$
$y_{63} := x_{30}x_{29} + x_{39}x_{40} + x_1 + x_0;$
$y_{64} := x_3x_2 + x_{13}x_{14};$
$y_{65} := x_{30}x_{43} + x_{42}x_{40} + x_2;$
$y_{66} := x_{11}x_{33} + x_{12}x_{32} + x_{13}x_{19} + x_{14}x_{21};$
$y_{67} := x_{15}x_{34} + x_{16}x_{35} + x_{17}x_{20} + x_{18}x_{31};$
$y_{68} := x_{15}x_{33} + x_{16}x_{32} + x_{17}x_{19} + x_{18}x_{21};$
$y_{69} := x_{11}x_{34} + x_{12}x_{35} + x_{13}x_{20} + x_{14}x_{31};$
$y_{70} := x_{33}x_{35} + x_{32}x_{34};$
$y_{71} := x_{33}x_{20} + x_{19}x_{34} + x_{18};$
$y_{72} := x_{32}x_{20} + x_{19}x_{35} + x_{14};$
$y_{73} := x_{21}x_{35} + x_{32}x_{31} + x_{13};$
$y_{74} := x_{21}x_{20} + x_{19}x_{31} + x_{12} + x_{15};$
$y_{75} := x_{14}x_{17} + x_{13}x_{18};$
$y_{76} := x_{21}x_{34} + x_{33}x_{31} + x_{17};$
$y_{77} := x_{11}x_{13} + x_{15}x_{17} + x_{38}x_{31} + x_{37}x_{21};$
$y_{78} := x_{12}x_{14} + x_{16}x_{18} + x_{39}x_{20} + x_{36}x_{19};$
$y_{79} := x_{12}x_{13} + x_{16}x_{17} + x_{39}x_{31} + x_{36}x_{21};$
$y_{80} := x_{11}x_{14} + x_{15}x_{18} + x_{38}x_{20} + x_{37}x_{19};$
$y_{81} := x_{11}x_{39} + x_{38}x_{12} + x_{17};$
$y_{82} := x_{15}x_{39} + x_{38}x_{16} + x_{13};$
$y_{83} := x_{37}x_{16} + x_{15}x_{36} + x_{14};$
$y_{84} := x_{37}x_{39} + x_{38}x_{36};$
$y_{85} := x_{37}x_{12} + x_{11}x_{36} + x_{18};$
$y_{86} := x_0x_{24} + x_1x_{23} + x_2x_8 + x_3x_{10};$
$y_{87} := x_4x_{25} + x_5x_{26} + x_6x_9 + x_7x_{22};$
$y_{88} := x_4x_{24} + x_5x_{23} + x_6x_8 + x_7x_{10};$
$y_{89} := x_0x_{25} + x_1x_{26} + x_2x_9 + x_3x_{22};$
$y_{90} := x_{24}x_{26} + x_{23}x_{25};$
$y_{91} := x_{24}x_9 + x_8x_{25} + x_7;$
$y_{92} := x_{23}x_9 + x_8x_{26} + x_3;$
$y_{93} := x_{10}x_{26} + x_{23}x_{22} + x_2;$
$y_{94} := x_{10}x_9 + x_8x_{22} + x_1 + x_4;$
$y_{95} := x_3x_6 + x_2x_7;$
$y_{96} := x_{10}x_{25} + x_{24}x_{22} + x_6;$
$y_{97} := x_0x_2 + x_4x_6 + x_{29}x_{22} + x_{28}x_{10};$
$y_{98} := x_1x_3 + x_5x_7 + x_{30}x_9 + x_{27}x_8;$
$y_{99} := x_1x_2 + x_5x_6 + x_{30}x_{22} + x_{27}x_{10};$
$y_{100} := x_0x_3 + x_4x_7 + x_{29}x_9 + x_{28}x_8;$
$y_{101} := x_0x_{30} + x_{29}x_1 + x_6;$
$y_{102} := x_4x_{30} + x_{29}x_5 + x_2;$
$y_{103} := x_{28}x_5 + x_4x_{27} + x_3;$
$y_{104} := x_{28}x_{30} + x_{29}x_{27};$
$y_{105} := x_{28}x_1 + x_0x_{27} + x_7;$
$y_{106} := f_{106};$
$y_{107} := f_{107};$
$y_{108} := f_{108};$
$y_{109} := f_{109};$

There are six polynomials $\{G_i\}$ defined as follows

$$
\begin{aligned}
&R_1 := y_{66}y_{67} + y_{68}y_{69} + y_{70}y_{31} + y_{71}y_{32} + y_{72}y_{33} + y_{73}y_{34} + y_{74}y_{75} + y_{76}y_{35} + y_{45};\\
&R_2 := y_{77}y_{78} + y_{79}y_{80} + y_{75}y_{31} + y_{36}y_{81} + y_{37}y_{82} + y_{38}y_{83} + y_{74}y_{84} + y_{39}y_{85} + y_{46};\\
&R_3 := y_{86}y_{87} + y_{88}y_{89} + y_{90}y_{22} + y_{91}y_{23} + y_{92}y_{24} + y_{93}y_{25} + y_{94}y_{95} + y_{96}y_{26} + y_{47};\\
&R_4 := y_{97}y_{98} + y_{99}y_{100} + y_{95}y_{22} + y_{27}y_{101} + y_{28}y_{102} + y_{29}y_{103} + y_{94}y_{104} + y_{30}y_{105} + y_{48};\\
&R_5 := y_{55}y_{56} + y_{57}y_{58} + y_{59}y_{40} + y_{60}y_{41} + y_{61}y_{42} + y_{62}y_{43} + y_{63}y_{64} + y_{65}y_{44} + y_{49};\\
&S_1 := R_2R_4 + R_3R_5 + y_{50} = x_{50};\\
&S_2 := R_1R_3 + R_4R_5 + y_{51} = x_{51};\\
&S_3 := R_1R_4 + R_2R_5 + y_{52} = x_{52};\\
&S_4 := R_1R_5 + R_2R_3 + y_{53} = x_{53};\\
&S_5 := R_1R_2 + R_3R_4 + y_{54} = x_{54};\\
&G_0 := S_2S_4 + S_3S_5 = x_{51}x_{53} + x_{52}x_{54};\\
&G_1 := S_1S_3 + S_4S_5 = x_{50}x_{52} + x_{53}x_{54};\\
&G_2 := S_1S_4 + S_2S_5 = x_{50}x_{53} + x_{51}x_{54};\\
&G_3 := S_1S_5 + S_2S_3 = x_{50}x_{54} + x_{51}x_{52};\\
&G_4 := S_1S_2 + S_3S_4 = x_{50}x_{51} + x_{52}x_{53};\\
&G_5 := R_1S_1 + R_2S_2 + R_3S_3 + R_4S_4 + R_5S_5 = x_{50}x_{45} + x_{51}x_{46} + x_{52}x_{47} + x_{53}x_{48} + x_{54}x_{49};\\
&G_6 := R_1S_2 + R_2S_3 + R_3S_4 + R_4S_5 + R_5S_1 = X_{50}x_{46} + x_{51}x_{47} + x_{52}x_{48} + x_{53}x_{49} + x_{54}x_{45}
\end{aligned}
$$

The map ϕ_3 is defined by the following system of equations,

$$
\begin{aligned}
&z_j = y_j + G_j,\ for\ j = 0, 1, 2, 3, 4, 5, 6;\\
&z_i = y_i,\ for\ i = 7, \cdots, 109.
\end{aligned}
$$

As in the previous section, we write ϕ_4 as $LP + v_4$, where L is an 110×110 invertible matrix. We will use mixed implementations $MI(s_2, s_3, s_4) = (0, 96, 14)\ or\ (28, 68, 14)$, i.e., with first 0 (or 28) rows in F_2, the next 96 (or 68) rows in F_3, and the last 14 rows in F_4. We shall discuss the securities of the above mixed implementations.

3.1 Security

Mainly, we shall consider the four attacks, Goubin-Courtois, XL, FXL and Ding-Schmidt.

(I): Against Goubin-Courtois' minrank attack. According to their unproved formula, the security is

$$q^{r \times \lceil \frac{m}{n} \rceil} m^3 = 2^{128} \times 110^3 \approx 2^{148}$$

(2): Against XL. The method of XL is incomplete in the sense that the critical index D can not be found easily. It is the risk the attacker has to take to assume that D is minimal possible. In this case, we have $D \geq 9$ and the security is higher than

$$\binom{64}{9}^{2.8} \approx 2^{97}$$

(3): Against FXL. In this case, let f(=the number of free variables) be $1, 2$ respectively, then we $D \geq 9, 9$ respectively, and the security is higher than

$$
\begin{aligned}
2^{16} \times \binom{63}{9}^{2.8} \approx 2^{112}\\
2^{32} \times \binom{62}{9}^{2.8} \approx 2^{126}
\end{aligned}
$$

respectively.

(4): Against Ding-Schmidt attack. The Ding-Schmidt attack can be separated into 2 steps: (1). Test run to see if the lock can be trivialized by part of the central polynomials. (2). If the answer is yes, then simulate a whole size attack. As for step (1), this attack can not trivialize the lock even upto level 2. For level 3, the data overflows for our computer. Note that the orginal attack of Ding-Schmidt is of level 1 ([5]) in our terminology. In [5], they claims that level 2 attack (in our terminology) is " hard but not impossible". It is obvious that if necessary, we may double or quadruple the degrees of the lock polynomials (which is easy) to make the higher level attacks ineffective.

3.2 Key-lengths and speeds

We have two cases: (1) $MI(s_2, s_3, s_4) = (0, 96, 14)$. (2) $MI(s_2, s_3, s_4) = (28, 68, 14)$. Let us compare the total program length (including public-key, private-key and the program) and the numbers of lookups for encryption of the mixed implementation and an unmixed (i.e., normal) implementation over $GF(2^8)$.

We note that an umixed implementation over $F_3 = GF(2^8)$ will have public-key length=$(1/2)(55\times 58) \times 110 = 175,450$ bytes, a private-key length =15, 125 bytes, a program length=500 bytes and v_4 length of $6,050$ bytes for decrypting purpose. Total space needed is $197,125$ bytes. As for speed, we have $350,900$ lookups for two rounds of encryption if we have a table of multiplication, $1,052,700$ lookups if we only have a log-exp table.

Case 1: For the mixed implementation $(0, 96, 14)$, the public-key length is $197,780$ bytes, the private-key length=$16,132$ bytes, a program length=500 bytes and v_4 length of $6,820$ bytes. The total space needed is $221,232$ bytes. As for the speed, we have,

(A): For a multiplicative table. The number of lookups for encrption is

$$(1/2)n(n+1)(2s_3 + 6s_4) + 6nm = 465,960$$

So that the decreasing factor of the length of the program and the increasing factor the the speed are

$$\begin{array}{l} decreasing\ factor\ of\ programs = 221,232/197,125 = 1.12 \\ increasing\ factor\ of\ speeds = 350,900/465,960 = 0.75 \end{array}$$

Note that for an unmixed implementation over $F_4 = GF(2^{16})$, the public key length will increase by 8-fold, and speed will be reduced to the 2/9. The above data indicates a good result.

(B): For only log-exp table. The number is

$$(1/2)n(n+1)(6s_3 + 9s_4) + 9nm = 1,142,460$$

$$\begin{array}{l} decreasing\ factor\ of\ programs = 223,131/197,125 = 1.13 \\ increasing\ factor\ of\ speeds = 1,052,700/1,142,460 = 0.92 \end{array}$$

Case 2: For the mixed implementation $(28, 68, 14)$, the public-key length is $175,450$ bytes, the private-key length=$12,856$ bytes, a program length=500 bytes and v_4 length of $6,050$ bytes. The total space needed is $194,856$ bytes. As for the speed, we have,

(A): For a multiplicative table. The number of lookups for encrption is

$$(1/2)n(n+1)(2s_3 + 6s_4) + 6nm = 375,100$$

So that the decreasing factor of the length of the program and the increasing factor the the speed are

$$\begin{array}{l} decreasing\ factor\ of\ programs = 194,856/197,125 = 0.99 \\ increasing\ factor\ of\ speeds = 350,900/375,100 = 0.93 \end{array}$$

Note that for an unmixed implementation over $F_4 = GF(2^{16})$, the public key length will increase by 8-fold, and speed will be reduced to the 2/9. The above data indicates a good result.

(B): For only log-exp table. The number is

$$(1/2)n(n+1)(6s_3 + 9s_4) + 9nm = 876,810$$

$$decreasing\ factor\ of\ programs = 194,856/197,125 = 0.99$$
$$increasing\ factor\ of\ speeds = 1,052,700/876,810 = 1.20$$

If we use only log-exp table, the mixed implementation will increase the security, shorten the length of the program and accelerate the speed of encryption.

References

[1] J.-M. Chen and B.-Y. Yang, *Tame Transformation Signatures with Topsy-Turvy Hashes*, proc. IWAP '02, Taipei, a summary available at the IACR e-Print acrhive at `http://eprint.iacr.org/2003/160`.

[2] J.-M. Chen and B.-Y. Yang, *A More Secure and Efficacious TTS Scheme*, preprint, available at the IACR e-Print archive.

[3] C.-Y. Chou, D.-J. Guan, and J.-M. Chen, *A Systematic Construction of a Q_{2^k}-module in TTM*, Communications in Algebra, 30 (2002), pp. 551–562.

[4] D. Coppersmith, J. Stern, and S. Vaudenay, *The Security of the Birational Permutation Signature Schemes*, Journal of Cryptology, 10(3), 1997, pp. 207–221.

[5] J. Ding, D. Schmidt, *A Defect Of The Implementation Schemes Of The TTM Cryptosystem*, available at `http://eprint.iacr.org/2003/086`

[6] L. Goubin and N. Courtois, *Cryptanalysis of the TTM Cryptosystem*, ASIACRYPT 2000, LNCS v. 1976, pp. 44–57.

[7] H. Hironaka, *Resolution of singularities of an algebraic variety over a field of characteristic zero, Parts I and II,* Annals of Mathematics, 79 (1964), pp. 109-203, 205-326.

[8] H. Imai and T. Matsumoto, *Algebraic Methods for Constructing Asymmetric Cryptosystems*, AAECC-3, LNCS v. 229, pp. 108–119.

[9] B. Lucier, *Cryptography, Finite Fields, and AltiVec*, `http://www.simdtech.org/apps/group_public/download.php/22/Cryptography.pdf`

[10] T. Moh, *A Public Key System with Signature and Master Key Functions*, Communications in Algebra, 27 (1999), pp. 2207–2222.

[11] T. Moh and J.-M. Chen, *On the Goubin-Courtois Attack on TTM*, available at `http://eprint.iacr.org/2001/072`

[12] H. Hironaka, *Resolution of singularities of an algebraic variety over a field of characteristic zero, Parts I and II,* Annals of Mathematics, 79 (1964),

[13] P. Montgomery and A. Sathaye, private communication.

[14] J. Patarin, *Cryptanalysis of the Matsumoto and Imai Public Key Scheme of Eurocrypt'88*, CRYPTO'95, LNCS V. 963, pp. 248–261.

[15] J. Patarin, *Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): Two New Families of Asymmetric Algorithms*, EUROCRYPT'96, LNCS V. 1070, pp. 33–48.

[16] T. Theobald, *How to Break Shamir's Asymmetric Basis*, CRYPTO'95, LNCS V. 963, pp. 136–147.