New Differential-Algebraic Attacks and Reparametrization of Rainbow

Jintai Ding[1], Bo-Yin Yang[2], Chia-Hsin Owen Chen[2], Ming-Shing Chen[2], and
Chen-Mou Cheng[3]

[1] Dept. of Mathematical Sciences, University of Cincinnati, USA, `ding@math.uc.edu`
[2] IIS, Academia Sinica, Taiwan, `[byyang,owenhsin,mschen]@iis.sinica.edu.tw`
[3] Dept. of Elec. Eng., Nat'l Taiwan University, Taiwan, `ccheng@cc.ee.ntu.edu.tw`

Abstract. A recently proposed class of multivariate Public-Key Cryptosystems, the Rainbow-Like Digital Signature Schemes, in which successive sets of central variables are obtained from previous ones by solving linear equations, seem to lead to efficient schemes (TTS, TRMS, and Rainbow) that perform well on systems of low computational resources. Recently SFLASH (C^{*-}) was broken by Dubois, Fouque, Shamir, and Stern via a differential attack. In this paper, we exhibit similar algebraic and diffential attacks, that will reduce published Rainbow-like schemes below their security levels. We will also discuss how parameters for Rainbow and TTS schemes should be chosen for practical applications.

Keywords: rank, differential attack, algebraic attack, oil-and-vinegar

Note: This is an update to the paper to appear at ACNS 2008, New York

1 Outline

Multivariate Public-Key Cryptosystems (MPKCs, or trapdoor $\mathcal{MQ}$ schemes) are cryptosystems for which the public key is a set of polynomials $\mathcal{P} = (p_1, \ldots, p_m)$ in variables $\mathbf{x} = (x_1, \ldots, x_n)$ where all variables and coefficients are in $\mathbb{K} = GF(q)$. In practice this is always accomplished via

$$\mathcal{P} : \mathbf{w} = (w_1, \ldots, w_n) \in \mathbb{K}^n \overset{S}{\mapsto} \mathbf{x} = M_S\mathbf{w} + \mathbf{c}_S \overset{\mathcal{Q}}{\mapsto} \mathbf{y} \overset{T}{\mapsto} \mathbf{z} = M_T\mathbf{y} + \mathbf{c}_T = (z_1, \ldots, z_m) \in \mathbb{K}^m$$

In any given scheme, the *central map* $\mathcal{Q}$ belongs to a certain class of quadratic maps whose inverse can be computed relatively easily. The maps S, T are affine. The polynomials giving y_i in $\mathbf{x}$ are called the central polynomials, and the x_j are called the central variables.

In 1999, the Unbalanced Oil-and-Vinegar multivariate structure is proposed by Patarin *et al* [16]. Lately the Rainbow class of signatures [7, 20, 25], based on repeated applications of the Unbalanced Oil-and-Vinegar principle, shows some promise on systems of low computational resources.

Given that the well-known C^{*-} class of signature schemes including SFLASH was broken by differential attacks [8], we examine similar attacks on Rainbow, with the following conclusions:

- Differentials improve on the High-Rank attacks on Rainbow-like systems.

- Differentials also helps with randomized brute-force searches for S and T.
- We can assess how Rainbow-like schemes needs to be amended in view of recent developments.
- The results are in line with experiments run on small scale systems.

In Sec. 2 we recap Rainbow-like multivariates and what is known about the security of MPKC before the appearance of Rainbow in Sec. 3. In Sec. 4, we describe the new differential attack, which is related to the high-rank attack, and in Sec. 5 we present new paramters for Rainbow construction, we tabulate what we know about the security of Rainbow-like schemes, in particular, the security against the two new recent attacks specially targeted against the Rainbow schemes, and we design schemes with new parameters for practical applications. Finally, in Sec. 6, we present the conclusion.

2   Rainbow-like Multivariate Signatures

We characterize a Rainbow type PKC with u stages:

- The segment structure is given by a sequence $0 < v_1 < v_2 < \cdots < v_{u+1} = n$. For $l = 1, \ldots, u+1$, set $S_l := \{1, 2, \ldots, v_l\}$ so that $|S_l| = v_l$ and $S_0 \subset S_1 \subset \cdots \subset S_{u+1} = S$.
- Denote by $o_l := v_{l+1} - v_l$ and $O_l := S_{l+1} \setminus S_l$ (i.e., $v_l < k \le v_{l+1}$ if $k \in O_l$) for $l = 1 \cdots u$. The central map $\mathcal{Q} : \mathbf{x} = (x_1, \ldots, x_n) \mapsto \mathbf{y} = (y_{v_1+1}, \ldots, y_n)$, where each $y_i := q_i(\mathbf{x})$ is a quadratic polynomial in $\mathbf{x}$ of the following form

$$q_k = \sum_{i<j \le v_l} \alpha_{ij}^{(k)} x_i x_j + \sum_{i \le v_l < j < v_{l+1}} \alpha_{ij}^{(k)} x_i x_j + \sum_{i < v_{l+1}} \beta_i^{(k)} x_i, \quad \text{if } v_l < k \le v_{l+1}.$$

  In every q_k, $k \in O_l$, there is no cross-term $x_i x_j$ where both i and j are in O_l at all. So given all the y_i with $v_l < i \le v_{l+1}$, and all the x_j with $j \le v_l$, we can compute $x_{v_l+1}, \ldots, x_{v_{l+1}}$.
- To expedite computations, some coefficients α_{ijk}'s may be fixed (e.g., set to zero), chosen at random (and included in the private key), or be interrelated in a predetermined manner.
- To invert $\mathcal{Q}$, determine (usu. at random) $x_1, \ldots x_{v_1}$, i.e., all x_k, $k \in S_1$. From the components of $\mathbf{y}$ that corresponds to the polynomials $q_{v_1+1}, \ldots q_{v_2}$, we obtain a set of o_1 equations in the variables x_k, $(k \in O_1)$. We may repeat the process to find all remaining variables.

In this form, we can see that Rainbow can only be a signature scheme. We can see a good example of what can go wrong in [15] if we try to construct an encryption scheme, where the initial vinegar variables is determined through an initial block of equations.

Example 1. enTTS(20,28) of [25] has structure $(8, 9, 1, 1, 9)$ and this central map:

$$
\begin{aligned}
y_i &= x_i + \sum_{j=1}^{7} p_{ij} x_j x_{8+(i+j \bmod 9)}, \ i = 8 \cdots 16; \\
y_{17} &= x_{17} + p_{17,1} x_1 x_6 + p_{17,2} x_2 x_5 + p_{17,3} x_3 x_4 \\
&\quad + p_{17,4} x_9 x_{16} + p_{17,5} x_{10} x_{15} + p_{17,6} x_{11} x_{14} + p_{17,7} x_{12} x_{13}; \\
y_{18} &= x_{18} + p_{18,1} x_2 x_7 + p_{18,2} x_3 x_6 + p_{18,3} x_4 x_5 \\
&\quad + p_{18,4} x_{10} x_{17} + p_{18,5} x_{11} x_{16} + p_{18,6} x_{12} x_{15} + p_{18,7} x_{13} x_{14}; \\
y_i &= x_i + p_{i,0} x_{i-11} x_{i-9} + \sum_{j=19}^{i-1} p_{i,j-18} \ x_{2(i-j)-(i \bmod 2)} \ x_j + p_{i,i-18} x_0 x_i \\
&\quad + \sum_{j=i+1}^{27} p_{i,j-18} \ x_{i-j+19} \ x_j, \ i = 19 \cdots 27.
\end{aligned}
\tag{1}
$$

If $x_0, \ldots, x_7$ is decided, one can solve first for $x_8, \ldots, x_{16}$, then x_{17}, x_{18}, then $x_{19}, \ldots, x_{27}$. Note: x_0 *does not appear until the last block, which will be significant later.*

Example 2. The proposed Rainbow scheme in [7] is an essentially generic stage-wise UOV construction with layers $(6, 6, 5, 5, 11)$. The first six central equations is a generic UOV construction with six vinegar $(x_1, \ldots x_6)$ and six oil $(x_7, \ldots, x_{12})$ variables; the next five has 12 vinegars and 5 oils $(x_{13}, \ldots, x_{17})$; the next five has 17 vinegars and 5 oils $(x_{18}, \ldots, x_{22})$, and the last 11 has 22 vinegars and 11 oils $(x_{23}, \ldots, x_{33})$.

Rainbow schemes where most of the crossterm coefficients $\alpha_{ij}^{(k)}$ are zero are said to be TTS instances. TTS schemes have a relatively small private key and even better efficiency, but may be exposed to additional risks. Regardless, the same techniques that we shall describe below are security concerns for all schemes of the rainbow type including TTS, TRMS, and Rainbow [7, 20, 25].

3 The Security of Multivariates and Prior Attacks

The name of the class came from the "Multivariate Quadratics" problem:

Problem $\mathcal{MQ}$: Solve the system $p_1 = p_2 = \cdots = p_m = 0$, where each p_i is a quadratic polynomial in $\mathbf{x} = (x_1, \ldots, x_n)$ and coefficients and variables are in $\mathbb{K} = \mathrm{GF}(q)$.

Generic $\mathcal{MQ}$ is NP-hard [12], and consensus pegs it as a difficult problem to solve even probabilistically. However, to use $\mathcal{MQ}$ as the underlying hard problem in a PKC, one need a trapdoor built into the public map $\mathcal{P}$. So the security of the cryptosystem also depends on the following:

Problem EIP: (Extended Isomorphism of Polynomials) Given a class of central maps $\mathfrak{C}$ and a map $\mathcal{P}$ expressible as $\mathcal{P} = T \circ \mathcal{Q} \circ S$, where $\mathcal{Q} \in \mathfrak{C}$, and S, T are affine, make such a decomposition.

There are two interesting twists here:

– If $\mathcal{Q}$ is constant, this is known as the IP problem. J.-C. Faugère showed that in some cases simple IP is not NP-hard at Eurocrypt 2006 [11].

– The EIP problem where $\mathfrak{C}$ is the set of homogeneous quadratic maps is easy [13]. Equivalently, if $\mathcal{Q}$ is homogeneous (e.g., as in SFLASH=C^{*-}) we can set $\mathbf{c}_S = \mathbf{c}_T = 0$.

If $\mathcal{Q}$ fundamentally involves a map in a field $\mathbb{L} = \mathbb{K}^k$ that is of a size significantly bigger than $\mathbb{K}$, we call the scheme "big field" or "dual field". This order includes derivatives of Matsumoto-Imai (C^*) and Hidden Field Equations. Otherwise we call the scheme a "true multivariate" (sometimes "single field"). This includes the Unbalanced Oil-and-Vinegar and stagewise triangular structures.

One of the biggest concerns of multivariate cryptography is the lack of provable security results. Today security in MPKC is still very much *ad hoc*. Proposed schemes are evaluated against known attacks security estimates obtained for various parameters. The designers then tries to juggle the system parameters so as to have some requisite security level under every known attack.

With that, we list the standard attacks known for MPKCs today:

1. Rank (or Low Rank, MinRank) attack, which finds a central equation with least rank [25].

$$C_{\text{low rank}} \approx \left\lceil q^{r\lceil m/n \rceil} m(n^2/2 - m^2/6)/\mu \right\rceil \mathfrak{m}.$$

Here as below, the unit $\mathfrak{m}$ is a multiplications in $\mathbb{K}$, and r is that lowest rank ("MinRank", [14]). μ is the number of linear combinations of central equations [25] at that minimal rank.

2. Dual Rank (or High Rank) attack [5, 14], which finds a variable appearing the fewest number of times in a central equation cross-term. If this least number is s, [25] gives

$$C_{\text{high rank}} \approx \left\lceil q^s n^3/6 \right\rceil \mathfrak{m}.$$

3. Oil-and-Vinegar Separation [16, 17, 22], which finds an Oil subspace that is sufficiently large (estimates as corrected in [25]).

$$C_{\text{UOV}} \approx \left\lceil q^{n-2o-1} o^4 + (\text{some residual term bounded by } o^3 q^{m-o}/3) \right\rceil \mathfrak{m}.$$

o is the max. *oil set* size, i.e., there is a set of o central variables which are never multiplied together in the central equations, and no more.

4. Trying for a direct solution (i.e., going for the $\mathcal{MQ}$ as opposed to the EIP or "structural" problem). Best known methods are the Lazard-Faugère family of solvers (the Gröbner Bases methods $\mathbf{F_4}$-$\mathbf{F_5}$ or XL) whose complexities [6, 9, 10, 24] are very hard to evaluate; some recent asymptotic formulas can be found in [1, 2, 24].

4   New Differential attacks

One key point of our new attack is to use the differentials (first used, as far as we know, with MPKC in [18] and recently to break SFLASH [8]).

Given the public key of a MPKC, which we denote as $\mathcal{P}(\mathbf{x})$, a set of quadratic polynomials, its differential $D\mathcal{P}(\mathbf{x})$ is defined as

$$D\mathcal{P}(\mathbf{x}) = \mathcal{P}(\mathbf{x} + \mathbf{c}) - \mathcal{P}(\mathbf{x}) - \mathcal{P}(\mathbf{c}),$$

a set of linear functions in $\mathbf{x}$.

The key is to use the hidden structures in the differential to attack the cryptosystem. The observation is that the differential can be used to improve the old high-rank attack when there are too many variables that don't appear in the final block of equations (for y_i, where $i \in O_u$). First, we will reformulate an existing attack in terms of the differentials.

Let H_i be the symmetric matrix corresponding to the quadratic part of $z_i(\mathbf{w})$. Without loss of generality, we may let the fewest number of appearances of all variables in the cross-terms of the central equations be the last variable x_n appearing s times.

Algorithm 0 (High or Dual Rank Attack) *as described by Goubin-Courtois and Yang-Chen [14, 25]:*

1. *Compute the differential $\mathcal{P}(\mathbf{x}+\mathbf{c}) - \mathcal{P}(\mathbf{x}) - \mathcal{P}(\mathbf{c})$ and take its j-th component (which is bilinear in $\mathbf{x}$ and $\mathbf{c}$) as $\mathbf{c}^T H_j \mathbf{x}$. H_k is representing the quadratic crossterms in the k-th polynomial of the public key. Note that the H_i are always symmetric and if $\mathrm{char}\,\mathbb{K} = 2$, and $\mathbf{x}^T H_i \mathbf{x} = 0$.*
2. *Form an arbitrary linear combination $H = \sum_i \alpha_i H_i$. Find $V = \ker H$.*
3. *When $\dim V = 1$, set $(\sum_j \lambda_j H_j)V = \{\mathbf{0}\}$ and check if the solution set $\hat{V}$ of the (λ_i) form a subspace dimension $m - s$. Note: Since a matrix in $K^{n \times n}$ can have at most n different eigenvalues, less than n/q of the time we would need to do this.*
4. *With probability q^{-s} we have $V = U = \{\mathbf{x} : x_1 = \cdots = x_{v_u} = 0\}$.*

As each trial run consists of running an elimination and some testing, we can realistically do this with $\sim \left(sn^2 + \frac{n^3}{6}\right) q^s$ field multiplications, by taking linear combinations from only $(s+1)$ of the matrices H_i and hope not to get too unlucky. An upper bound is $\left[mn^2 + \frac{n^3}{6} + \frac{n}{q}(m^3/3 + mn^2)\right] q^s$.

The above formulation of the high rank attack is designed to defeat "plus"-modified Triangular systems. We first present some notations before describing how we can improve this attack further:

Let P_l be the linear space of quadratic polynomials spanned by polynomials of the form

$$\sum_{i \in O_l, j \in S_l} \alpha_{i,j} x_i x_j + \sum_{i,j \in S_l} \alpha_{i,j} x_i x_j + \sum_{i \in S_{l+1}} \beta_i x_i + \eta$$

We can see that these are Oil and Vinegar type of polynomials such that x_i, $i \in O_l$ are the Oil variables and x_i, $i \in S_l$ are the Vinegar variables. We call x_i, $i \in O_l$ an l-th layer Oil variable and x_i, $i \in S_l$ an l-th layer Vinegar variable.

We call any polynomial in P_l an l-th layer Oil and Vinegar polynomial. Clearly we have $P_i \subset P_j$ for $i < j$. Let W_i be the space of linear functions of variables $x_1, ..., x_{v_i}$. Then we have

$$W_1 \subset P_1 \subset W_2 \subset P_2 \cdots \subset W_u \subset P_u \subset W_{u+1}.$$

Now we present the new attack:

Algorithm 1 *The Improved High-Rank Attack using differentials:*

1. *Pick random $\mathbf{c}, \mathbf{c}' \in \mathbb{K}^n$, compute $\mathcal{P}(\mathbf{w}+\mathbf{c}) - \mathcal{P}(\mathbf{w}) - \mathcal{P}(\mathbf{c})$, and we will denote its components as $(t_1, t_2, \ldots, t_m)$. Similarly we compute $(t'_1, t'_2, \ldots, t'_m) = \mathcal{P}(\mathbf{w} + \mathbf{c}') - \mathcal{P}(\mathbf{w}) - \mathcal{P}(\mathbf{c}')$, then*

$$U = \mathrm{span}(t_1, t_2, \ldots, t_m) \cap \mathrm{span}(t'_1, t'_2, \ldots, t'_m).$$

2. *Guess at a linear form $f \in U$; find coefficients a_i and a'_i such that $f = \sum a_i t_i = \sum a'_i t'_i$.*
3. *Use a_i and a'_i as the guessed α_i in the High Rank Attack (Algorithm 0) above.*

Proposition 1. *The expected complexity of Algorithm 1 is $\sim q^d \cdot (\text{cubic-time elimination})$ where (the last block of equations is the ones whose solutions gives O_u)*

$$d \leq s - [\# \text{ vars appearing in crossterms only in the last block}]. \qquad (2)$$

Proof. Let
$$F = (F_1, \ldots, F_m) = \mathcal{Q} \circ S$$

be the mapping from $\mathbf{x} \mapsto \mathbf{z}$. Let

$$F(\mathbf{x} + \mathbf{b}) - F(\mathbf{x}) - F(\mathbf{b}) := G = (G_1, G_2, \ldots, G_n),$$

where $\mathbf{b} = (b_1, b_2, \ldots b_i, \ldots, b_n)$ is randomly chosen. Pick another $\mathbf{b}'$ and form

$$H = (H_1, \ldots, H_n) = F(\mathbf{x} + \mathbf{b}') - F(\mathbf{x}) - F(\mathbf{b}'),$$

then

1. if $i \in O_j$, then $G_i, H_i \in W_{j+1}$;
2. $\overline{W}_{j+1} := \mathrm{span}\{G_i\}_{i \in O_j} \subset W_{j+1}$, and similarly $\widehat{W}_{j+1} := \mathrm{span}\{H_i\}_{i \in O_j} \subset W_{j+1}$;
3. $\overline{W}_2 \subset ... \subset \overline{W}_{u+1}$ and $\widehat{W}_2 \subset ... \subset \widehat{W}_{u+1}$.

Clearly $(\widehat{W}_u \bigcap \overline{W}_u) \subset (\widehat{W}_{u+1} \bigcap \overline{W}_{u+1})$, and we observe that: if the dimensions of the two subspaces differ by d, then we can break the system with $\propto q^d \cdot (\text{one guess})$ computations.

How so? Because the relationship between $\mathcal{P}$ and F, is the same as that between the $\mathbf{w}$-space and $\mathbf{x}$-space, i.e., the linear transformation S. So there is a 1-in-q^d chance that both $\sum a_i z_i$ and $\sum a'_i z_i$ correspond to a linear form in

W_u. The odds are now decided by q^{-d} instead of q^{-s}. In a Rainbow-like system, $s = o_u = n - v_u$. For Alg. 1 to be worthwhile, we must show that $d \leq s$.

In fact, it is not so hard to describe how to determine d. $\overline{W}_{u+1}$ and $\widehat{W}_{u+1}$ are two m-dimensional subspaces in the n-dimensional vector space W_{u+1}. Most of the time they intersect in a $2m - n$ dimensional subspace, hence

$$\dim \overline{W}_u = \dim \widehat{W}_u = m - o_u$$

which equals **the number of variables appearing in cross-terms in equations *not* of the final block**, which is equivalent to Eq. 2.

Example 3. Consider enTTS(20,28) as in Eq. 1. Here $\dim(\overline{W}_{u+1} \bigcup \widehat{W}_{u+1}) = 20 + 20 - 28 = 12$, while $\dim(\overline{W}_u \bigcup \widehat{W}_u) = 11 + 11 - 17 = 5$. Therefore we need only $\sim 2^{56}$ instead of 2^{72} guesses, which is a speed increase of $2^{16} \times$ over Algorithm 0. Since each guess takes about 2^8 time units (standard is to use time of a 3DES block encryption, between 2^6 to 2^8 multiplications), this gives complexity 2^{64} instead of 2^{80}, too weak to be "strong" crypto.

What went wrong? Generically $\dim W_u = n - o_u$ and the intersection is of dimension $2(m - o_u) - (n - o_u) = 2m - n - o_u$, making $d = (2m - n) - (2m - n - o_u) = o_u = s$. *The lesson: watch out for variable not in the final oil set that does not occur prior to the last block of equations.* In enTTS(20,28), x_0 and x_{18} did not appear in any earlier equations than the final block.

4.1 Experimentation with mini-versions

We experimented in smaller fields with three different schemes: Rainbow (6,6,5,5,11), the enTTS(20,28) scheme above, and its miniaturized sister version enTTS(16,22) [structure (6,7,1,1,7)].

Scheme	Structure	q	Alg. 0	Alg. 1	ratio
EnTTS(20,28)	(8,9,1,1,9)	8	93	4.4	0.047
EnTTS(20,28)	(8,9,1,1,9)	16	42435	496	0.012
EnTTS(16,22)	(6,7,1,1,7)	16	102	3.5	0.034
Rainbow [7]	(6,6,5,5,11)	8	8454	17123	2.028

Table 1. Timing (sec) on 16 of 3GHz P4 machines guessing in parallel

The results are fairly constant over many tests [except the enTTS(20,28) test which we only ran a few times]. Clearly, not having all vinegar variables of the last segment appearing previously in cross-terms is a big minus. Rainbow (6,6,5,5,11) does not have the same problem and Algorithm 1 is no improvement of the High Rank Attack against it.

5 New Rainbow Parameters for Practical Applications

For practical applications, we will propose the following Rainbow Structures.

1. $(20, 10, 4, 10)$, where the public key has 44 variables and 24 polynomials.
2. $(18, 12, 12)$, where the public key has 42 variables and 24 polynomials.
3. $(20, 14, 14)$, where the public key has 48 variables and 28 polynomials.

We will first formalize a twist on the regular Rainbow construction, which is somewhat more general. In the previous constructions, in each new layer, previously appeared variables will only be Vinegar variables, the new variables appearing only as Oil variables. We can also consider adding new Vinegar variables as we add Oil variables. This also implies that in the signing process, we guess at the new vinegar variables as they appear, while in the previous Rainbow construction, we only guess the Vinegar variables in the first layer once. In this case, we can also write for each layer two parameters, (v'_i, o_i), where the v'_i counts the new vinegar variables we introduce. In this layer, we will have $v_i + v'_i$ Vinegar variables (where v_i counts the number of all previous appearing variables) and o_i the number of Oil variables.

If all the v'_i are zero, this is precisely the original Rainbow construction. We might call this new construction the extended Oil-Vinegar construction. From the viewpoint of the attacker we can see this as a specialization of the Rainbow construction, since the new vinegar variables might as well have been part of the initial block of vinegar variables, but simply never have been used before. However, it is different in an operative sense, in that if we use the new vinegar variables properly, we could always find a signature, as implicitly used in TTS constructions earlier.

So, in this language, we would propose scheme: $((15, 10), (4, 4), (1, 10))$, $((17, 12), (1, 12))$, and $((19, 14), (1, 14))$.

For these new schemes, we could also choose to use the generic sparse polynomials or special sparse polynomials as in the case of TTS [25]. For generic sparse polynomials, we think it is a good idea to choose $3L_i$ terms for each layer, where L_i is the sum of number of Oil and vinegar variables in each layer.

For these new schemes, we need to take into two new recent special attacks against Rainbow.

5.1 The Reconciliation Attack

In the following attack we attempt to find a sequence of change of basis that let us invert the public map. In this sense it can be considered an improved brute force attack.

Suppose we have an oil-and-vinegar structure, then the quadratic part of each component q_i in the central map from $\mathbf{x}$ to $\mathbf{y}$, when expressed as a symmetric

matrix, looks like

$$
M_i := \left[\begin{array}{ccc|ccc}
\alpha_{11}^{(i)} & \cdots & \alpha_{1v}^{(i)} & \alpha_{1,v+1,}^{(i)} & \cdots & \alpha_{1n}^{(i)} \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
\alpha_{v1}^{(i)} & \cdots & \alpha_{vv}^{(i)} & \alpha_{v,v+1,}^{(i)} & \cdots & \alpha_{vn}^{(i)} \\ \hline
\alpha_{v+1,1,}^{(i)} & \cdots & \alpha_{v+1,v,}^{(i)} & 0 & \cdots & 0 \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
\alpha_{n1}^{(i)} & \cdots & \alpha_{nv}^{(i)} & 0 & \cdots & 0
\end{array} \right]
\tag{3}
$$

First, no matter what M_T is, it won't change the basic shape, so we let T be the identity map for the moment. What can S be like? Suppose we pick M_S as totally random, most often (see below) it decompose to

$$
M_S := \begin{bmatrix} *_{v \times v} & *_{v \times o} \\ *_{o \times v} & *_{o \times o} \end{bmatrix} = \begin{bmatrix} 1_{v \times v} & *_{v \times o} \\ 0_{o \times v} & 1_{o \times o} \end{bmatrix} \begin{bmatrix} *_{v \times v} & 0_{v \times o} \\ *_{o \times v} & *_{o \times o} \end{bmatrix}
\tag{4}
$$

where 1 means identity matrix, 0 means just zeros and $*$ means random or anything. In fact, this decomposition always hold unless the lower-right $o \times o$ submatrix is singular. It should be clear that the $\begin{bmatrix} *_{v \times v} & 0_{v \times o} \\ *_{o \times v} & *_{o \times o} \end{bmatrix}$ portion of M_S, as a coordinate change leaves the M_i's with the same shape. That is, if we can find the correct $\begin{bmatrix} 1_{v \times v} & *_{v \times o} \\ 0_{o \times v} & 1_{o \times o} \end{bmatrix}$ portion and perform the basis change in reverse, we will again make the resulting public map into the same form (all zeroes on the lower right) and be easily inverted. Hence, no more security at all. More about this phenomenon ("equivalent keys") in MPKCs can be seen in, say, [23].

Let this essential portion of M_S that we wish to recreate be P, that is, the linear transformation $\mathbf{w} \mapsto \mathbf{x} = P\mathbf{w}$ will create all zeroes on the lower right. We can decompose this P into a product of $P := P_{v+1}P_{v+2} \cdots P_n$, where each of the matrices look like

$$
P_n = 1_n + \left[\begin{array}{ccc|c}
0 & \cdots & 0 & a_1 \\
0 & \cdots & 0 & a_2 \\
\vdots & \ddots & \vdots & \vdots \\
0 & \cdots & 0 & a_v \\ \hline
0 & \cdots & 0 & 0 \\
\vdots & \ddots & \vdots & \vdots \\
0 & \cdots & 0 & 0
\end{array} \right] \; ; \quad
P_{n-1} = 1_n + \left[\begin{array}{ccc|c|c}
0 & \cdots & 0 & a_1' & 0 \\
0 & \cdots & 0 & a_2' & 0 \\
\vdots & \ddots & \vdots & \vdots & \vdots \\
0 & \cdots & 0 & a_v' & 0 \\ \hline
0 & \cdots & 0 & 0 & 0 \\
\vdots & \ddots & \vdots & \vdots & \vdots \\
0 & \cdots & 0 & 0 & 0
\end{array} \right] \; ; \cdots
$$

Indeed, the multiplication is actually commutative among the various P_i's. *Suppose, then, that we start with the differential matrices H_i and simultaneously transform them to make their lower-right corner a square of 0's using exactly such P_i's.*

Algorithm 2 (UOV Reconciliation) *The following gives the <u>Reconciliation Attack</u> against a UOV scheme with o oil and $v = n - o$ vinegar variables (which has the smaller indices):*

1. *Perform basis change $w_i := w'_i - \lambda_i w'_n$ for $i = 1 \cdots v$, $w_i = w'_i$ for $i = v + 1 \cdots n$. Evaluate $\mathbf{z}$ in $\mathbf{w}'$.*
2. *Let all coefficients of $(w'_n)^2$ be zero and solve for the λ_i. We may use any method such as $\mathbf{F_4}/\mathbf{F_5}$ or FXL. There will be m equations in v unknowns.*
3. *Repeat the process to find P_{n-1}. Now we set $w'_i := w''_i - \lambda_i w''_{n-1}$ for $i = 1 \cdots v$, and set every $(w''_{n-1})^2$ and $w''_n w''_{n-1}$ term to zero (i.e., more equations in the system) after making the substitution. This time there are $2m$ equations in v unknowns.*
4. *Continue similarly to find $P_{n-2}, \ldots, P_{v+1}$ with more and more equations.*

Given what we know about system-solving today, we can expect the complexity to be determined in solving the initial system. Hence, if $v < m$, solving m equations in v variables will be easier than m equations in n equations, and we achieve a simplification.

Proposition 2. *The Reconciliation Attack works with probability $\approx \left(1 - \frac{1}{q-1}\right)$.*

Proof (Sketch). Provided that lower-right $o \times o$ submatrix of M_S is non-singular, we can see that the construction of P_n will eliminate the quadratic term in the last variable. P_{n-1} will eliminate all quadratic terms in the last two variables, and so on, and each sequential construction will not disturb the structure built by the prior transformations. The number of nonsingular $k \times k$ matrices in over $\mathrm{GF}(q)$ is $(q^k - 1)(q^k - q)(q^k - q^2) \cdots (q^k - q^{k-1})$, because the first row has 1 possibility to be zero, the second row q possibilities to be a multiple of the first, the third row q^2 possibilities to be dependent on the first two, etc., so the chance that the above attack works is roughly

$$\left(1 - \frac{1}{q}\right)\left(1 - \frac{1}{q^2}\right) \cdots \left(1 - \frac{1}{q^k}\right) > 1 - \left(\frac{1}{q} + \frac{1}{q^2} + \cdots + \frac{1}{q^k}\right) > 1 - \frac{1}{q-1}.$$

Here we will use formulas from [26] for all our estimates as shown below.

Example 4. We attack enTTS(20,28) as in Eq. 1. Originally we must solve a 20-equation, 20-variable (we can guess 8 out of the original 28) MQ system. With $v_u = 19$, the rate-determining step of the Reconciliation Attack is a 20-equation, 19-variable system. This is easier by a factor of exactly 2^8 if we are using FXL or $\mathrm{F}\mathbf{F_4}$ [1, 24], since we will guess exactly one fewer variable.

Since we expect a direct attack on enTTS(20,28) to have $\sim 2^{72}$ complexity, Alg. 2 should take $\sim 2^{64}$. The construction process and odds as given above have been tested and verified on miniature versions (cf. [25]) of TTS schemes such as enTTS(16,22) as well as other Rainbow-like instances.

Example 5. TRMS [20] can be reduced to 2^{64} via the same attack (a faster attack given below) because it has rainbow layer parameters of $(8, 6, 2, 3, 9)$, with a last block of the same size as TTS.

Example 6. We implemented enTTS(16,22) over GF(128), the initial system has 16 equations and $22 - 7 = 15$ variables. We ran FXL with Wiedemann solver (as in [26]) with one fixed variable on an assembly of machines with 128 total P4 cores at 3.0GHz, each guessing 1 value out of 128. Here $D = 8$ [24], and the number of monomials is $T = 319770$, with a total of 73799040 terms which took only 288MB of storage at every core. Solving a system known to have a solution should take around $3(T^2 n(n + 3)/2) \approx 2^{45}$ multiplications, which at about 16 cycles a multiplication about 2.0×10^4 seconds, but we discovered that there is guesswork in generating a system, so we dare not run more than one value on a given CPU.

In practice we were not so unlucky and were able to solve 15 variables in 16 equations in GF(128) in what was in fact closer about 3 days, probably due to non-optimal programming. After that, solving the remaining systems is a piece of cake [real CPU time estimated at less than two hours], and we can then decompose an enTTS(16,22) instance.

Example 7. We now attack the proposed Rainbow instance in [7]. Since $v_u = 22 < m = 26$, solving this one is significantly easier: using F$\mathbf{F_5}$ [24], the expected time use is 2^{56} (3DES blocks) instead of 2^{81}. $\mathbf{F_5}$ is not generally available but we should be able to achieve $\sim 2^{64}$ cycles using FXL on a large SMP system.

We can easily see that we must be very careful choosing our parameters for security against one attack may expose it to another. *Our selected parameters are all tuned against this particular attack and this attack is no better or worse than direct attack, which have complexity of solving 24, and 28 equations in as many variables over* GF(256), *or roughly* 2^{83} *and* 2^{98} *respectively.*

But this is just a *unbalanced oil and vinegar attack*. The more efficiently implemented systems are Rainbow and have multiple layers. If we look at the Rainbow construction, it looks more like

$$
M_i := \begin{bmatrix} \alpha_{11}^{(i)} & \cdots & \alpha_{1v}^{(i)} & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \alpha_{v1}^{(i)} & \cdots & \alpha_{vv}^{(i)} & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & 0 \end{bmatrix} \text{, if } i \leq m-o; \quad \begin{bmatrix} \alpha_{11}^{(i)} & \cdots & \alpha_{1v}^{(i)} & \alpha_{1,v+1,}^{(i)} & \cdots & \alpha_{1n}^{(i)} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \alpha_{v1}^{(i)} & \cdots & \alpha_{vv}^{(i)} & \alpha_{v,v+1,}^{(i)} & \cdots & \alpha_{vn}^{(i)} \\ \alpha_{v+1,1,}^{(i)} & \cdots & \alpha_{v+1,v,}^{(i)} & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \alpha_{n1}^{(i)} & \cdots & \alpha_{nv}^{(i)} & 0 & \cdots & 0 \end{bmatrix} \text{, otherwise.}
$$

$$(5)$$

That is, only the last o equations looks like Eq. 3, the initial $m - o$ equations actually have non-zero entries in the upperleft submatrix — which actually looks like a UOV matrix itself, i.e., has a block of zeros on the lower right. We don't bother with that detail. Can we exploit this property? Yes we can.

At this point, we should no longer consider T as the identity. Let us think about what the matrix M_T does in Rainbow. At the moment that we distill the P_n portion out, $m - o$ of the new M_i's should show a zero last column. *However we don't;* M_T *mixes the* M_i's *together so that they in fact don't – we will see*

most of the time only the lower right entry as zero. But if we take any $o + 1$ of those last columns, there will be a non-trivial linear dependency. We can verify that by setting one of those columns as the linear combination as the other o, the resulting equations are still quadratic!

Algorithm 3 (Rainbow Band Separation) *Reconciliation may be extended for a Rainbow scheme where the final stage has o oil and $v = n - o$ vinegar variables (which has the smaller indices):*

1. *Perform basis change $w_i := w'_i - \lambda_i w'_n$ for $i = 1 \cdots v$, $w_i = w'_i$ for $i = v + 1 \cdots n$. Evaluate $\mathbf{z}$ in $\mathbf{w}'$.*
2. *Find m equations by setting all coefficients of $(w'_n)^2$ to be zero; there are v variables in the λ_i's.*
3. *Set all cross-terms involving w'_n in $\mathbf{z}_1 - \sigma_1^{(1)} \mathbf{z}_{v+1} - \sigma_2^{(1)} \mathbf{z}_{v+2} - \cdots - \sigma_o^{(1)} \mathbf{z}_m$ to be zero and find $n - 1$ more equations. Note that $(w'_n)^2$ terms are assumed gone already, so we can no longer get a useful equation.*
4. *Solve $m + n - 1$ quadratic equations in $o + v = n$ unknowns. We may use any method (e.g., $\mathbf{F_4}$ or XL).*
5. *Repeat the process to find P_{n-1}. Now set $w'_i := w''_i - \lambda_i w''_{n-1}$ for $i = 1 \cdots v$, and set every $(w''_{n-1})^2$ and $w''_n w''_{n-1}$ term to zero after making the substitution. Also set $\mathbf{z}_2 - \sigma_1^{(2)} \mathbf{z}_{v+1} - \sigma_2^{(2)} \mathbf{z}_{v+2} - \cdots - \sigma_o^{(2)} \mathbf{z}_m$ to have a zero second-to-last column. This time there are $2m + n - 2$ equations in n unknowns.*
6. *Continue in the same vein to find $P_{n-2}, \ldots, P_{v+1}$.*

The idea was mentioned by Mr. Yu-Hua Hu to one of the authors in a conversation, for which we are indebted. And this attack explains why the current parameter set suggested looks like that in Sec. 5.

Example 8. We run the attack on an instance of enTTS$(16, 22)$ [25] which has the shape $(6, 7, 1, 1, 7)$. The algebraic portion of the attack results in a system with 22 variables and 37 equations. This with XL at degree $D_{XL} = 6$ can be solved using 400MB (actually 415,919,856 bytes) of memory and 123,257 seconds on a 16-core, 2.2GHz Opteron machine with a total of 1,877,572 seconds of K8-CPU time. The number of multiplications is about 2^{47}, or ~ 16 cycles a multiplication.

On a single core, a K8 machine running XL-Wiedemann can average one multiplication in GF(2^8) in about 9 cycles. The slowdown comes from the communications requirement between cores.

Example 9. The attack on an instance of enTTS$(20, 28)$ [25] should result in a system with 22 variables and 37 equations. This with XL at degree $D_{XL} = 7$ should be solvable in 15GB of main memory and about 2^{56} multiplications. This is under the design complexity of 2^{72}.

We are also testing the prowess of other system-solving methods like Magma's $\mathbf{F_4}$.

5.2   Interlinked/Accumulating Kernels and MinRank

As noted in [25] and recapped in Sec. 3, if μ combinations of central equations stays at the minrank, a Rank attack often speed up μ-fold, and which is termed *interlinking* or *accumulation* of kernels.

Recently Billet and Gilbert [4] cryptanalyzed the Rainbow instance of [7] in $\sim 2^{64}$ 3DES unit times (they stated 2^{71}, but GF(256)-multiplications is a very small unit; NESSIE for example counted 3DES units) using the same principle. While we exhibit a faster attack on that rainbow instance above, the same extended accumulating-kernel minrank attack is more widely applicable:

Proposition 3 (Billet-Gilbert). *Kernels of the initial block of equations in a rainbow-like multivariate always accumulate such that any vector in **x**-space with the initial vinegar components all vanishing has at least a $1/q$ probability of being found by the MinRank attack.*

Example 10. We can cryptanalyze enTTS(20,28) [25] in 2^{64} via the accumulating kernels attack.

In fact, this pitfall is sometimes easy to overlook:

Proposition 4. *We can cryptanalyze TRMS from [20] in $\sim 2^{62}$ via the accumulating kernels attack.*

Proof. The central map has this piece with $*_3$ meaning multiplication in GF(2^{24}):

$$\begin{pmatrix} y_{17} \\ y_{18} \\ y_{19} \end{pmatrix} = \begin{pmatrix} x_{17} \\ x_{18} \\ x_{19} \end{pmatrix} *_3 \begin{pmatrix} x_8 \\ x_9 + x_{11} + x_{12} \\ x_{13} + x_{15} + x_{16} \end{pmatrix} + \begin{pmatrix} c_{29}x_4x_{16} \\ c_{30}x_5x_{10} \\ c_{31}x_{15}x_{16} \end{pmatrix} + \begin{pmatrix} c_{32}x_9 \\ c_{33}x_{10} \\ c_{34}x_{11} \end{pmatrix}.$$

Each of these equations are only of rank 8 (the minrank) in GF(256), and the y_{17} and y_{19} form a pair of equations that has $q = 256$ interlinked kernels. Evaluating as in Sec. 3 gives $\approx 2^{62}$.

In our schemes, the attack has complexity roughly q to the number of equations in the first block times change, which comes out to about $2^{85}, 2^{100}, 2^{118}$.

5.3   The challenge

From all the above, we can see that we need to be very careful in our design of the parameter for Rainbow like schemes.

Proposition 5. *To build a scheme with design security C over the base field GF(q), we let ℓ be the smallest integer such that $q^{\ell+1} \gtrsim C$, then:*

— *The initial segment must contain $\ell - 1$ or more vinegar variables. The final segment must contain $\ell - 1$ or more equations and exactly as many as there are total vinegar variables.*

- *There should be enough equations to avoid direct solution via a Lazard-Faugère solver.*
  Current estimate [24] is that 20 underdetermined equations in $\mathrm{GF}(2^8)$ *achieves* 2^{72}*; 24 equations achieves* 2^{82}*; each extra equation roughly gives a factor* $\gtrsim 2^{2.5}$ *to the complexity [24].*

We conclude that all three Rainbow like schemes we propose below have security levels above 2^{80} elementary operations. The best attack is with Algorithm 3, and the expected complexity in $\mathrm{GF}(2^8)$ multiplications is 2^{84}, 2^{87}, 2^{80} respectively.

1. Rainbow (20,10,4,10), in the extended form $((15, 10), (4, 4), (1, 10))$
2. Rainbow (18,12,12), in the extended form $((17, 12), (1, 12))$
3. Rainbow (20,14,14), in the extended form $((19, 14), (1, 14))$.

Of course, without using the extended form, the security level would not be any lower, the extended form merely guarantees the existence of a signature always.

We hasten to add that the form given above is not much slower in signing than the previous TTS. In preliminary runs, a single signature for (20,10,4,10) version averages to about $157\mu s$, still way faster than any competitor.

6   Conclusion

In this paper, we present a new differential attack and a new Rainbow constructions. We design new schemes for practical applications.

With these constructions, we note that the design security of the system would still go up exponentially as the length of the hash in both generic (rainbow) and sparse (TTS) variants. *Perhaps, we might even say that the kinks of this approach is being ironed out, and multivariate cryptographers are finally beginning to understand Rainbow-like Multivariate Signatures*

Another development that affects Rainbow-like schemes is the fact that SHA-1 is being phased out in the wake of recent results [21]. This means that hashes and hence signatures might become longer in a hurry. ECC is affected in much the same way, because 163- or 191-bit ECC may be obsoleted when everyone switches to SHA-2 (no one really wants to use a truncated hash if it can be helped). Even such state-of-the-art work as [3] would force the slightly uncomfortable SHA-224. With multivariate signature schemes, an additional problem is the large (and sometimes redundant, cf. [23]) keys. One might look toward other base fields such as GF(16) to help with the key size problem, but this would also pose new challenges in optimization. Another way is to look for a safe TTS (built on the similar layer structures as specified above), now that hash sizes has gotten longer. Though the new attacks are found on Rainbow schemes, these attacks can be easily prevented by adjusting the parameter. All in all, we think that multivariates including Rainbow-like schemes still deserve a good look as the age of quantum computers approaches.

Acknowledgements

Comments and correspondence: Please address to BY at `by@moscito.org`.

References

1. M. Bardet, J.-C. Faugère, and B. Salvy. On the complexity of Gröbner basis computation of semi-regular overdetermined algebraic equations. In *Proceedings of the International Conference on Polynomial System Solving*, pages 71–74, 2004. Previously INRIA report RR-5049.

2. M. Bardet, J.-C. Faugère, B. Salvy, and B.-Y. Yang. Asymptotic expansion of the degree of regularity for semi-regular systems of equations. In P. Gianni, editor, *MEGA 2005 Sardinia (Italy)*, 2005.

3. Daniel J. Bernstein. Curve25519: New diffie-hellman speed records. In *Public Key Cryptography*, volume 3958 of *Lecture Notes in Computer Science*, pages 207–228. Moti Yung et al, editors, Springer, 2006. ISBN 3-540-33851-9.

4. Olivier Billet and Henri Gilbert. Cryptanalysis of rainbow. In *Security and Cryptography for Networks*, volume 4116 of *LNCS*, pages 336–347. Springer, September 2006.

5. Don Coppersmith, Jacques Stern, and Serge Vaudenay. The security of the birational permutation signature schemes. *Journal of Cryptology*, 10:207–221, 1997.

6. Nicolas T. Courtois, Alexander Klimov, Jacques Patarin, and Adi Shamir. Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In *Advances in Cryptology — EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 392–407. Bart Preneel, editor, Springer, 2000. Extended Version: `http://www.minrank.org/xlfull.pdf`.

7. Jintai Ding and Dieter Schmidt. Rainbow, a new multivariable polynomial signature scheme. In *Conference on Applied Cryptography and Network Security — ACNS 2005*, volume 3531 of *Lecture Notes in Computer Science*, pages 164–175. Springer, 2005.

8. Vivien Dubois, Pierre-Alain Fouque, Adi Shamir, and Jacques Stern. Practical cryptanalysis of sflash. In *Crypto*, volume 4622 of *LNCS*, pages 1–12. Springer, 2007.

9. Jean-Charles Faugère. A new efficient algorithm for computing Gröbner bases (F_4). *Journal of Pure and Applied Algebra*, 139:61–88, June 1999.

10. Jean-Charles Faugère. A new efficient algorithm for computing Gröbner bases without reduction to zero (F_5). In *International Symposium on Symbolic and Algebraic Computation — ISSAC 2002*, pages 75–83. ACM Press, July 2002.

11. Jean-Charles Faugère and Ludovic Perret. Polynomial equivalence problems: Algorithmic and theoretical aspects. In Serge Vaudenay, editor, *EUROCRYPT*, volume 4004 of *Lecture Notes in Computer Science*, pages 30–47. Springer, 2006.

12. Michael R. Garey and David S. Johnson. *Computers and Intractability — A Guide to the Theory of NP-Completeness.* W.H. Freeman and Company, 1979. ISBN 0-7167-1044-7 or 0-7167-1045-5.
13. W. Geiselmann, R. Steinwandt, and Th. Beth. Attacking the affine parts of SFlash. In *Cryptography and Coding - 8th IMA International Conference*, volume 2260 of *Lecture Notes in Computer Science*, pages 355–359. B. Honary, editor, Springer, 2001. Extended version: http://eprint.iacr.org/2003/220/.
14. Louis Goubin and Nicolas T. Courtois. Cryptanalysis of the TTM cryptosystem. In *Advances in Cryptology — ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 44–57. Tatsuaki Okamoto, editor, Springer, 2000.
15. Antoine Joux, Sébastien Kunz-Jacques, Frédéric Muller, and Pierre-Michel Ricordel. Cryptanalysis of the tractable rational map cryptosystem. In PKC [19], pages 258–274.
16. Aviad Kipnis, Jacques Patarin, and Louis Goubin. Unbalanced Oil and Vinegar signature schemes. In *Advances in Cryptology — EUROCRYPT 1999*, volume 1592 of *Lecture Notes in Computer Science*, pages 206–222. Jacques Stern, editor, Springer, 1999.
17. Aviad Kipnis and Adi Shamir. Cryptanalysis of the oil and vinegar signature scheme. In *Advances in Cryptology — CRYPTO 1998*, volume 1462 of *Lecture Notes in Computer Science*, pages 257–266. Hugo Krawczyk, editor, Springer, 1998.
18. Jacques Patarin and Louis Goubin. Trapdoor one-way permutations and multivariate polynomials. In *International Conference on Information Security and Cryptology 1997*, volume 1334 of *Lecture Notes in Computer Science*, pages 356–368. International Communications and Information Security Association, Springer, 1997. Extended Version: http://citeseer.nj.nec.com/patarin97trapdoor.html.
19. Serge Vaudenay, editor. *Public Key Cryptography — PKC 2005*, volume 3386 of *Lecture Notes in Computer Science*. Springer, 2005. ISBN 3-540-24454-9.
20. Lih-Chung Wang, Yuh-Hua Hu, Feipei Lai, Chun yen Chou, and Bo-Yin Yang. Tractable rational map signature. In PKC [19], pages 244–257. ISBN 3-540-24454-9.
21. Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding collisions in the full sha-1. In *CRYPTO*, volume 3621 of *Lecture Notes in Computer Science*, pages 17–36. Victor Shoup, editor, Springer, 2005. ISBN 3-540-28114-2.
22. Christopher Wolf, An Braeken, and Bart Preneel. Efficient cryptanalysis of RSE(2)PKC and RSSE(2)PKC. In *Conference on Security in Communication Networks — SCN 2004*, volume 3352 of *Lecture Notes in Computer Science*, pages 294–309. Springer, September 8–10 2004. Extended version: http://eprint.iacr.org/2004/237.
23. Christopher Wolf and Bart Preneel. Superfluous keys in $\mathcal{M}$ultivariate $\mathcal{Q}$uadratic asymmetric systems. In PKC [19], pages 275–287. Extended version http://eprint.iacr.org/2004/361/.
24. Bo-Yin Yang and Jiun-Ming Chen. All in the XL family: Theory and practice. In *ICISC 2004*, volume 3506 of *Lecture Notes in Computer Science*, pages 67–86. Springer, 2004.
25. Bo-Yin Yang and Jiun-Ming Chen. Building secure tame-like multivariate public-key cryptosystems: The new TTS. In *ACISP 2005*, volume 3574 of *Lecture Notes in Computer Science*, pages 518–531. Springer, July 2005.
26. Bo-Yin Yang, Owen Chia-Hsin Chen, Daniel J. Bernstein, and Jiun-Ming Chen. Analysis of QUAD. In *Fast Software Encryption — FSE 2007*, volume 4593 of *Lecture Notes in Computer Science*, pages 290–307. Alex Biryukov, editor, Springer, 2007. workshop record available.