Tractable Rational Map Signature

Lih-Chung Wang[1,⋆], Yuh-Hua Hu[2], Feipei Lai[3],
Chun-Yen Chou[4,⋆⋆], and Bo-Yin Yang[5,⋆⋆⋆]

[1] Department of Applied Mathematics,
National Donghwa University, Hualien 974, Taiwan
lcwang@mail.ndhu.edu.tw
[2] Department of Computer Science and Information Engineering,
National Taiwan University, Taipei 106, Taiwan
d92015@csie.ntu.edu.tw
[3] Departments of Electrical Engineering &
of Computer Science and Information Engineering,
National Taiwan University, Taipei 106, Taiwan
flai@ntu.edu.tw
[4] Department of Mathematical Education,
National Hualien Teachers College, Hualien 970, Taiwan
choucy@mail.nhltc.edu.tw
[5] Dept. of Mathematics, Tamkang University, Tamsui 251, Taiwan
by@moscito.org

Abstract. Digital signature schemes are crucial for applications in electronic commerce. The effectiveness and security of a digital signature scheme rely on its underlying public key cryptosystem. Trapdoor functions are central to public key cryptosystems. However, the modular exponentiation for RSA or the discrete logarithms for ElGamal/DSA/ECC, as the choice of the trapdoor functions, are relatively slow in performance. Some multivariate schemes has potentially much higher performance than other public key cryptosystems. We present a new multivariate digital signature scheme (TRMS) based on tractable rational maps. We also give some security analysis and some actual implementation data in comparison to some other signature schemes.

Keywords: multivariate, public key, digital signature, finite field, tractable rational maps

1 Introduction

Digital signature schemes are crucial for applications in electronic commerce. For example, to improve the efficiency and maintain the order of stock exchange, each on-line transaction needs to be verified to be validated. The effectiveness and security of a digital signature scheme rely on its underlying public key cryptosystem. Trapdoor functions are central to public key cryptosystems. Only a

⋆ Partially supported by National Science Council Grant NSC-93-2115-M-259-003.
⋆⋆ Partially supported by National Science Council Grant NSC-93-2115-M-026-001.
⋆⋆⋆ Partially supported by National Science Council Grant NSC-93-2115-M-032-008.

S. Vaudenay (Ed.): PKC 2005, LNCS 3386, pp. 244–257, 2005.

handful of the many schemes attempted reached practical deployment. However, the modular exponentiation for RSA or the discrete logarithms for ElGamal/DSA/ECC, as the choice of the trapdoor functions, are relatively slow in performance. One main reason is the size of the single operand which (at the required security levels) tends to be huge, and this slows the performance.

Some multivariate schemes distinguish themselves from other public key cryptosystems by showing potential for higher performance. For example, Courtois, Goubin and Patarin proposed SFLASH, which has been selected by Nessie Consortium and recommended for low-cost smart cards. The newest version of this signature scheme, SFLASH^{v3} may be found in [12]. Also, Chen and Yang gave a class of signature (TTS) scheme based on tame transformations in [4, 5, 38]. The newest version of TTS, called Enhanced TTS, outperforms ([40]) all previously known digital signature schemes of comparable security levels, including SFLASH^{v3}. A summary of this newest instance may be found in [38].

Here we will present a new class of multivariate digital signature scheme (TRMS) based on tractable rational maps. TRMS has similar security and performance as Enhanced-TTS. However there is a small yet non-negligible chance (around 7%) that signing takes perceptibly longer in the newer versions of TTS. In contrast, the signing time for TRMS is constant, which can do no harm and may be an improvement.

Fix a finite field $\mathbb{K}$ and a natural number n. Tractable rational maps on $\mathbb{K}^n$ are invertible affine transformations or, after a rearrangement of indices if necessary, functions of the following form $\varphi : \mathbb{K}^n \to \mathbb{K}^n$,

$$\begin{cases} y_1 = r_1(x_1) \\ y_2 = r_2(x_2)\dfrac{p_2(x_1)}{q_2(x_1)} + \dfrac{f_2(x_1)}{g_2(x_1)} \\ \quad \vdots \\ y_k = r_k(x_k)\dfrac{p_k(x_1, x_2, \ldots, x_{k-1})}{q_k(x_1, x_2, \ldots, x_{k-1})} + \dfrac{f_k(x_1, x_2, \ldots, x_{k-1})}{g_k(x_1, x_2, \ldots, x_{k-1})} \\ \quad \vdots \\ y_n = r_n(x_n)\dfrac{p_n(x_1, x_2, \ldots, x_{n-1})}{q_n(x_1, x_2, \ldots, x_{n-1})} + \dfrac{f_n(x_1, x_2, \ldots, x_{n-1})}{g_n(x_1, x_2, \ldots, x_{n-1})} \end{cases}$$

where for $i = 2, 3, \ldots, n$, p_i, q_i, f_i, g_i are polynomials, and for $i = 1, 2, \ldots, n$, r_i is a permutation polynomial on $\mathbb{K}$. That is, r_i is a polynomial function which is also a bijection from $\mathbb{K}$ onto itself.

Let $S = \{(x_1, x_2, \ldots, x_n) \mid \prod_{j=2}^{n} p_j q_j g_j \neq 0\}$. For any point in the image set of S, it is very easy to find point-wise inverse for tractable rational maps: Given a point $(y_1, y_2, \ldots, y_n) \in \varphi(S)$, we can easily compute $(x_1, x_2, \ldots, x_n) \in \mathbb{K}^n$ such that $\varphi(x_1, x_2, \ldots, x_n) = (y_1, y_2, \ldots, y_n)$. When φ is an invertible affine transformation, we can easily write the inverse transformation φ^{-1} in an explicit and simultaneous way. That is, we have an explicit formula from which we can compute $x_1, x_2, \ldots, x_n$ simultaneously. When φ is not an invertible affine transformation, although it is computationally infeasible to write the inverse in an explicit and simultaneous way, given any point $(y_1, y_2, \ldots, y_n) \in \varphi(S)$, it

is very easy to compute $x_1, x_2, \ldots, x_n$ in a sequential way. We simply apply a sequence of substitutions as follows. We refer to this as substitution property.

$$\begin{cases} x_1 = r_1^{-1}(y_1) \\ x_2 = r_2^{-1}\left((y_2 - \dfrac{f_2(x_1)}{g_2(x_1)})\dfrac{q_2(x_1)}{p_2(x_1)}\right) \\ \quad \vdots \\ x_k = r_k^{-1}\left((y_k - \dfrac{f_k(x_1, x_2, \ldots, x_{k-1})}{g_k(x_1, x_2, \ldots, x_{k-1})})\dfrac{q_k(x_1, x_2, \ldots, x_{k-1})}{p_k(x_1, x_2, \ldots, x_{k-1})}\right) \\ \quad \vdots \\ x_n = r_n^{-1}\left((y_n - \dfrac{f_n(x_1, x_2, \ldots, x_{n-1})}{g_n(x_1, x_2, \ldots, x_{n-1})})\dfrac{q_n(x_1, x_2, \ldots, x_{n-1})}{p_n(x_1, x_2, \ldots, x_{n-1})}\right) \end{cases}$$

Note that, by Lagrange interpolation, any map over a finite field is a polynomial map. There are both computational and categorical reasons that we put our maps in rational form. For computational reasons, it is faster to compute the division between two function values by low degree polynomial maps than to compute a single function value by a much higher degree polynomial map. For example, it is much easier to compute $\dfrac{1}{x}$ than to compute x^{254} over $GF(256)$. And categorically, even given a tractable rational map without denominator, by the direct computation above, the inverse of that map is most naturally described as a rational map. Therefore we choose to put the map in the rational form. For details, see [36].

TRMS is the result of exploring the combination of substitution property of tractable rational maps and other mathematical ideas into application of digital signatures.

In [26], T. Moh invented a public key cryptosystem (TTM) based on tame automorphisms which also have the substitution property. It is easily seen that tame transformations are special cases of tractable rational maps with the term $r_k(x_k)\dfrac{p_k(x_1, x_2, \ldots, x_{k-1})}{q_k(x_1, x_2, \ldots, x_{k-1})}$ replaced by x_k. Therefore it is not surprising at all that TRMS based on tractable rational maps can achieve similar security and performance as TTS based on tame automorphisms. However, there are also substantial differences between TRMS and TTS with respect to other mathematical ideas and designs.

In section 2, we give the details of TRMS. In section 3, we give some actual implementation data. In section 4, we give some analysis and compare TRMS to other signature schemes, in particular, including TTS.

2 Details of TRMS

We show an implement scheme of TRMS. It can be seen that there are a variety of schemes of TRMS which are all based on tractable rational maps.

Let $\mathbb{K} = GF(2^8)$. We will construct 3 maps $\varphi_1 : \mathbb{K}^{28} \to \mathbb{K}^{28}$, $\varphi_2 : \mathbb{K}^{28} \to \mathbb{K}^{20}$, $\varphi_3 : \mathbb{K}^{20} \to \mathbb{K}^{20}$ where φ_1, φ_3 are invertible affine transformations, $\varphi_2 = \pi \circ \widetilde{\varphi_2} \circ i$

with π a projection, i an imbedding, and $\widetilde{\varphi_2}$ identified as a tractable rational map over some extension field over $\mathbb{K}$. All the details are given below.

The public key or the verification map V is the result of the composition map $\varphi_3 \circ \varphi_2 \circ \varphi_1$. Therefore the public key will only be seen as 20 quadratic polynomials in 28 variables whose size is about 8.7KB as shown below.

The private key or the key part in the signing map S is the triple $(\varphi_1, \varphi_2, \varphi_3)$ in some specified structured form whose size is about 0.4KB as shown below. As mentioned in the introduction, each φ_i gives direct instruction to find the point-wise inverse for any concrete instance. Therefore the private key holder or the signer can directly apply φ_i^{-1} point-wisely.

To sign a message M, first find its hash $\mathbf{z} = H(M) \in \mathbb{K}^{20}$ by a publicly agreed hash function. Then do $\mathbf{y} = \varphi_3^{-1}(\mathbf{z})$, where the indices of $\mathbf{y}$ is starting at 9. Then choose 8 nonzero random numbers $r_1, r_2, \ldots, r_8$. Then get $\mathbf{x}$ by identifying it with $(\widetilde{\varphi_2}^{-1} \circ i)(r_1, r_2, \ldots, r_8, \mathbf{y})$ which is computed by a sequence of substitutions. Then get the signature $\mathbf{w} = \varphi_1^{-1}(\mathbf{x})$.

To verify a signature $\mathbf{w}$, simply check if $V(\mathbf{w}) = (\varphi_3 \circ \varphi_2 \circ \varphi_1)(\mathbf{w}) = (\varphi_3 \circ \pi \circ \widetilde{\varphi_2} \circ i)(\mathbf{x}) = (\varphi_3 \circ \pi)(r_1, r_2, \ldots, r_8, \mathbf{y}) = \varphi_3(\mathbf{y}) = \mathbf{z} = H(M)$.

2.1 Details of φ_1 and φ_3

Since $GF(2^{32})$ is finite extension fields of $\mathbb{K}$ of degree 4, therefore we can identify an element in $\mathbb{K}^4$ as an element in $GF(2^{32})$. Furthermore, we can decompose $(x_1, x_2, \ldots, x_{28}) \in \mathbb{K}^{28}$ into seven groups: for $i = 1, 2, \ldots, 7$, $X_i = (x_{4i-3}, x_{4i-2}, x_{4i-1}, x_{4i})$ and identify $X_i \in GF(2^{32})$, $i = 1, 2, \ldots, 7$. Hence we can identify $\mathbb{K}^{28}$ with $GF(2^{32})^7$. Similarly, we can identify $\mathbb{K}^{20}$ with $GF(2^{32})^5$.

Let φ_1, φ_3 be invertible affine maps on $\mathbb{K}^{28}$ and $\mathbb{K}^{20}$ respectively such that $\varphi_1 = S_1 \circ T_1 \circ L_1 \circ D_1 \circ U_1$ and $\varphi_3 = T_3 \circ L_3 \circ D_3 \circ U_3 \circ S_3$ where

1. S_1 is a circular shift on $\mathbb{K}^{28}$ and S_3 is a circular shift on $\mathbb{K}^{20}$.
2. T_1 is a translation on $\mathbb{K}^{28}$ and T_3 is a translation on $\mathbb{K}^{20}$. T_3 is used to cancel the constant terms in the public key. Therefore T_3 is not chosen but determined.
3. L_1 is a 7×7 lower triangular matrix over $GF(2^{32})$ and L_3 is a 5×5 lower triangular matrix over $GF(2^{32})$ such that both with diagonal entries equal to $1 \in GF(2^{32})$.
4. D_1 is a 28×28 invertible upper triangular matrix over $\mathbb{K}$ and D_3 is a 20×20 invertible upper triangular matrix over $\mathbb{K}$ in the following form:

$$D_1 = \begin{pmatrix} d_1 & d_1^2 & d_1^3 & \ldots & d_1^{28} \\ 0 & d_2 & d_2^2 & \ldots & d_2^{27} \\ 0 & 0 & d_3 & \ldots & d_3^{26} \\ \vdots & \vdots & & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & d_{28} \end{pmatrix}$$

5. U_1 is a 7×7 upper triangular matrix over $GF(2^{32})$ and U_3 is a 5×5 upper triangular matrix over $GF(2^{32})$ such that both with diagonal entries equal to $1 \in GF(2^{32})$.

Note that circular shifts on $\mathbb{K}^n$ are indeed linear transformations on $\mathbb{K}^n$ and each T_i above represents the translation part in the corresponding affine transformation. The LDU decomposition above covers quite a part of general invertible linear transformations. Moreover, our construction enjoys some benefits in key size. With L_1, U_1 linear on $GF(2^{32})^7$ and L_3, U_3 linear on $GF(2^{32})^5$, key size of the private key is reduced. Also, the calculation speed of additions is optimized on current 32-bit computer hardware structure. The diagonal entries in L's and U's are 1 implies that when we solve $L\mathbf{u} = \mathbf{v}$ or $U\mathbf{u} = \mathbf{v}$ we only have to do additions and multiplications and don't have to bother to do any division. Furthermore, with D_1 and D_3 both linear over $\mathbb{K}$ but not on $GF(2^{32})$, and also the circular shifts over $\mathbb{K}^n$, we can choose φ_1, φ_3 linear over $\mathbb{K}$, but not linear over $GF(2^{32})$. The purpose is to maintain security at the level over $\mathbb{K}$.

2.2 Details of φ_2

Let $\mathbb{L}, \mathbb{L}', \mathbb{L}''$ be the finite extension fields of $\mathbb{K}$ such that $\mathbb{K} \subset \mathbb{L}'' \subset \mathbb{L}' \subset \mathbb{L}$ and $[\mathbb{L}'' : \mathbb{K}] = 2$, $[\mathbb{L}' : \mathbb{L}''] = 3$, $[\mathbb{L} : \mathbb{L}'] = 3$. Therefore we can identify an element in $\mathbb{K}^2$ as an element in $\mathbb{L}' = GF(2^{16}) \subset \mathbb{L}' \subset \mathbb{L}$, an element in $\mathbb{K}^6$ as an element in $\mathbb{L}' = GF(2^{48}) \subset \mathbb{L}$, and an element in $\mathbb{K}^{18}$ as an element in $\mathbb{L} = GF(2^{144})$.

Decompose $(x_1, x_2, \ldots, x_{28}) \in \mathbb{K}^{28}$ into five groups: $X_1 = (x_1, x_2, \ldots, x_8)$, $X_2 = (x_9, x_{10}, x_{11}, x_{12}, x_{13}, x_{14})$, $X_3 = (x_{15}, x_{16})$, $X_4 = (x_{17}, x_{18}, x_{19})$ and $X_5 = (x_{20}, x_{21}, \ldots, x_{28})$. Identify X_1 with $(0, \ldots, 0, x_1, x_2, \ldots, x_8) \in \mathbb{L}$. Identify $X_2 \in \mathbb{K}^6$ as an element in $\mathbb{L}' \subset \mathbb{L}$. Identify $X_3 \in \mathbb{K}^2$ as an element in $\mathbb{L}'' \subset \mathbb{L}' \subset \mathbb{L}$ and $X_4 \in \mathbb{K}^3$ with $(0, x_{17}, 0, x_{18}, 0, x_{19}) \in \mathbb{L}'' \subset \mathbb{L}$. Identify $X_5 \in \mathbb{K}^9$ with $(0, x_{20}, 0, x_{21}, \ldots, 0, x_{28})$ as an element in $\mathbb{L}$. Hence we have a natural imbedding $i : \mathbb{K}^{28} \hookrightarrow \mathbb{L}^5$ by $i(x_1, x_2, \ldots, x_{28}) = (X_1, X_2, X_3, X_4, X_5)$. Similarly, decompose $(y_9, y_{10}, \ldots, y_{32}) \in \mathbb{K}^{20}$ into four groups: $Y_2 = (y_9, y_{10}, y_{11}, y_{12}, y_{13}, y_{14})$, $Y_3 = (y_{15}, y_{16})$, $Y_4 = (y_{17}, y_{18}, y_{19})$ and $Y_5 = (y_{20}, y_{21}, \ldots, y_{28})$ and identify them as elements in $\mathbb{L}$. For any $r_i \in \mathbb{K}$, $i = 1, 2, \ldots, 8$, identify $R_1 = (r_1, r_2, \ldots, r_8) \in \mathbb{K}^8$ with $(0, \ldots, 0, r_1, r_2, \ldots, r_8) \in \mathbb{L}$. Then we also have

$$i(r_1, r_2, \ldots, r_8, y_9, y_{10}, \ldots, y_{28}) = (R_1, Y_2, Y_3, Y_4, Y_5) \in \mathbb{L}^5.$$

Furthermore, since $\mathbb{K}^{20}$ is a subspace of $\mathbb{L}^5 = \mathbb{K}^{90}$, we have the projection $\pi : \mathbb{L}^5 \to \mathbb{K}^{20}$ such that $(\pi \circ i)(r_1, r_2, \ldots, r_8, y_9, y_{10}, \ldots, y_{28}) = (y_9, y_{10}, \ldots, y_{28})$

Let $\widetilde{\varphi_2} : \mathbb{L}^5 \to \mathbb{L}^5$ be a tractable rational map of the following form.

$$\begin{cases} R_1 = X_1 \\ Y_2 = X_2\ p_2(X_1)\ +\ f_2(X_1) \\ Y_3 = r_3(X_3)\ +\ f_3(X_1, X_2) \\ Y_4 = X_4\ p_4(X_1, X_2, X_3)\ +\ f_4(X_1, X_2, X_3) \\ Y_5 = X_5\ p_5(X_1, X_2, X_3, X_4)\ +\ f_5(X_1, X_2, X_3, X_4) \end{cases}$$

such that $\varphi_2 = \pi \circ \widetilde{\varphi_2} \circ i$, and we have the following in φ_2:

1. $R_1 = X_1$ induces $(r_1, r_2, \ldots, r_8) = (x_1, x_2, \ldots, x_8)$.
2. $Y_2 = X_2\ p_2(X_1)\ +\ f_2(X_1)$ induces

$$\begin{pmatrix} y_9 \\ y_{10} \\ \vdots \\ y_{14} \end{pmatrix} = \begin{pmatrix} x_9 \\ x_{10} \\ \vdots \\ x_{14} \end{pmatrix} *_6 \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_6 \end{pmatrix} + \begin{pmatrix} c_1x_1x_2 \\ c_2x_2x_3 \\ \vdots \\ c_6x_6x_7 \end{pmatrix} + \begin{pmatrix} c_7x_3 \\ c_8x_4 \\ \vdots \\ c_{12}x_8 \end{pmatrix}$$

where c_i's are constant parameters of user's choice and $\mathbf{u} *_n \mathbf{v}$ denotes first identifying $\mathbf{u}, \mathbf{v} \in \mathbb{K}^n$ in the extension field with degree n then carrying out the multiplication there. For details see Appendix.

3. $Y_3 = r_3(X_3) \ + \ f_3(X_1, X_2)$ induces

$$\begin{pmatrix} y_{15} \\ y_{16} \end{pmatrix} = \begin{pmatrix} x_{15} \\ x_{16} \end{pmatrix}^2 + \begin{pmatrix} c_{13}x_1x_2 + c_{14}x_3x_4 + \cdots + c_{19}x_{13}x_{14} \\ c_{20}x_{14}x_1 + c_{21}x_2x_3 + \cdots + c_{26}x_{12}x_{13} \end{pmatrix} + \begin{pmatrix} c_{27}x_1 \\ c_{28}x_2 \end{pmatrix}$$

where $\begin{pmatrix} x_{15} \\ x_{16} \end{pmatrix}^2 = \begin{pmatrix} x_{15} \\ x_{16} \end{pmatrix} *_2 \begin{pmatrix} x_{15} \\ x_{16} \end{pmatrix}$ and c_i's are constant parameters of user's choice.

4. $Y_4 = X_4 \ p_4(X_1, X_2, X_3) \ + \ f_4(X_1, X_2, X_3)$ induces

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

where c_i's are constant parameters of user's choice.

5. $Y_5 = X_5 \ p_5(X_1, X_2, X_3, X_4) \ + \ f_5(X_1, X_2, X_3, X_4)$ induces

$$\begin{pmatrix} y_{20} \\ y_{21} \\ \vdots \\ y_{28} \end{pmatrix} = \begin{pmatrix} x_{20} \\ x_{21} \\ \vdots \\ x_{28} \end{pmatrix} *_9 \begin{pmatrix} x_1 \\ x_2 + x_6 + x_{11} \\ x_3 + x_7 + x_{12} \\ x_4 + x_8 + x_{13} \\ x_5 + x_9 + x_{14} \\ x_{10} + x_{14} + x_{16} \\ x_{11} + x_{15} + x_{17} \\ x_{12} + x_{16} + x_{18} \\ x_{13} + x_{17} + x_{19} \end{pmatrix} + \begin{pmatrix} c_{35}x_{18}x_{19} \\ c_{36}x_{17}x_{13} \\ c_{37}x_{16}x_{14} \\ c_{38}x_{12}x_{13} \\ c_{39}x_{15}x_{14} \\ c_{40}x_{19}x_{12} \\ c_{41}x_{18}x_{10} \\ c_{42}x_{12}x_6 \\ c_{43}x_{13}x_5 \end{pmatrix} + \begin{pmatrix} c_{44}x_1 \\ c_{45}x_2 \\ \vdots \\ c_{52}x_9 \end{pmatrix}$$

where c_i's are constant parameters of user's choice.

The reason why the formulas in the above assignments represents a permutation polynomial r_3 and polynomials $p_2, f_2, f_3, p_4, f_4, p_5, f_5$ is as follows.

1. We identify $X_3 = (x_{15}, x_{16})$ as an element in $\mathbb{L}'' = GF(2^{16})$ which is of characteristic 2. For any finite field of characteristic 2, $X \mapsto X^2$ is an automorphism. Hence let $r_3(X) = X^2$, then r_3 is an automorphism on $\mathbb{L}''$, hence a permutation polynomial. And $\begin{pmatrix} x_{15} \\ x_{16} \end{pmatrix} \mapsto \begin{pmatrix} x_{15} \\ x_{16} \end{pmatrix}^2$ surely represents r_3.
2. For polynomials p_2, f_2, f_3, p_4, f_4, p_5, f_5, simply notice that on a finite field, any map is a polynomial map. See [36] for details. For example, we show the case of p_2 for illustration. Consider a map $\mathcal{P}$ on $\mathbb{L}$ as follows

$$\mathcal{P}(X_1) = \begin{cases} \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 0 \\ 0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{pmatrix} & \text{if } X_1 = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{pmatrix}, \\ \overrightarrow{0} & \text{otherwise.} \end{cases}$$

Simply let p_2 to be the polynomial representation for $\mathcal{P}$.

It is worth to mention the following.

1. For theoretical purpose we showed above that φ_2 is viewed as $\pi \circ \widetilde{\varphi_2} \circ i$ where $\widetilde{\varphi_2} : \mathbb{L}^5 \to \mathbb{L}^5$ is a tractable rational map with polynomials p_2, f_2, f_3, p_4, f_4, p_5, f_5 possibly very complicated. Computationally, we actually follow the other way around. That is, φ_2 is a computationally efficient representation for $\widetilde{\varphi_2}$ when restricted to the subspace $i(\mathbb{K}^{28})$. We get benefits on calculation speed due the following. The second assignment in φ_2 can be carried out in the subfield $GF(2^{48})$ instead of in $\mathbb{L} = GF(2^{144})$. For details see appendix. Similarly, the third assignment in φ_2 can be carried out in $\mathbb{L}'' = GF(2^{16})$ instead of in $\mathbb{L} = GF(2^{144})$. Both these contribute on calculation speed.
2. It is easily seen that our φ_2 representation is quadratic in x_i's. Since φ_1, φ_3 are affine maps, the public key is 20 general quadratic polynomials in 28 variables without constant terms.

2.3 Information on Keys

As shown above, $\varphi_1 = S_1 \circ T_1 \circ L_1 \circ D_1 \circ U_1$, $\varphi_3 = T_3 \circ L_3 \circ D_3 \circ U_3 \circ S_3$, and there are 52 parameters $c_1, c_2, \ldots, c_{52}$ for the private key user to choose in φ_2. Therefore the size for private key is $[0 + 28 + 4(1 + 2 + 3 + 4 + 5 + 6) + 28 + 4(6 + 5 + 4 + 3 + 2 + 1)] + [20 + 4(1 + 2 + 3 + 4) + 20 + 4(4 + 3 + 2 + 1) + 0] + 52 = 396$ Bytes. However, T_3 in φ_3 is not chosen but determined. Hence it is to choose 376 nonzero elements in $\mathbb{K}$ to generate the private key.

Also, since the public key is 20 general quadratic polynomials in 28 variables without constant terms, its size is $20 \cdot (\frac{28 \cdot 29}{2} + 28) = 8680$ bytes. In general, there are two ways to generate the public keys. One way is the method of undetermined coefficients, the other one is to make the composition by direct computation. Both have many optimized variants. Our major concern is on the structure of TRMS, therefore we did not put much effort in the optimization of the key generation.

3 Performance

Test Platform: CPU: P4 2.4GHz; RAM: 1024MB; OS: Linux + gcc 3.3;
ARG: gcc -O3 -march=pentium4 -fomit-frame-pointer

Scheme Name	Signature size (byte)	Public Key Size (byte)	Private Key Size (byte)	Sign (μs)	Verify (μs)	Key Generation (ms)
TTS(20,28)	28	8680	1399	7	20	2.2
TRMS(20,28)	28	8680	396	4.8	20	1.2

Table: NESSIE signature report, TTS and TRMS tested as above
Unit: { Signature/key size:Bytes, Sign/Verify/Key Generation: cycles/invocation

Scheme Name	Signature size	Public Key Size	Private Key Size	Sign	Verify	Key Generation
ECDSA	48	48	24	1971K	5415K	1758K
ESgin	144	145	96	4434K	936K	269M
RSA-PSS	128	128	320	82M	1587K	3206M
$SFLASH_{v2}$	37	$\approx$ 15K	$\approx$ 28K	5106K	765K	2929M
SQARTZ	16	$\approx$ 71K	$\approx$ 4K	6261M	144K	3167M
ACESign	425	620	748	26M	20M	9645M
TTS(20,28)	28	$\approx$ 8.7K	$\approx$ 1.4K	16.8K	48K	5.28M
TRMS(20,28)	28	$\approx$ 8.7K	396	11.4K	48K	2.67M

4 Analysis and Comparison

4.1 Security Analysis

For brevity, we fix the following notations for our TRMS example:

- $m = 20$ denotes the dimension of the hash space.
- $n = 28$ denotes the dimension of the signature space.
- $q = 2^8$ denotes the size of the base field $GF(256)$.

There are several known attacks for multivariate cryptosystems.

Rank Attack: Goubin and Courtois shows that the MinRank attack for Triangular-Plus-Minus systems. Yang and Chen generalized the idea to Rank attack for multivariate systems in [38]. The complexity of the Rank attack is about $q^r \cdot \frac{(m^2(\frac{n}{2} - \frac{m}{6}) + mn^2)}{k}$ multiplications, where k is the number of linear combinations of the components of φ_2 which reach the minimal rank r. The minimal rank for our example is at least 12, and k is 6. Therefore the complexity is about 2^{107} multiplications or 2^{101} 3DES units (1 unit of 3DES $\approx 2^6$ multiplications).

Dual Rank Attack: Coppersmith et al first ([6]) used the Dual Rank attack against multivariate scheme of Shamir; Yang and Chen to generalize this attack to all tame-like multivariate systems in [38]. The complexity of the Dual Rank attack is about $q^u(un^2 + \frac{n^3}{6})$ multiplications where u is the minimal number of appearances in φ_2 for any variable x_i. When $u = 9$ for our sample scheme, the complexity is about 2^{86} multiplications or 2^{80} 3DES units.

Unbalanced Oil and Vinegar Attack: As in [38], Let an "oil-set" be any set of independent variables x_i, such that any of their cross-products never appears in any equation in φ_2. Suppose the maximum size of an oil set is k, then then we may determine in time k^4q^{n-2k-1} the "vinegar" and the "oil" subspaces. After that, several possible techniques may be used to find a solution. If case $k = 9$, so the time taken to identify the vinegar and oil subspaces is about 2^{86} multiplications, or 2^{80} 3DES units.

Patarin Relations Attack for C^* Family: In φ_2 of our TRMS example, there is no Patarin relation, which means the attack for C^* family is not feasible for our system.

Affine Parts Distillation: Geiselmann et al. in [19, 20] pointed out the possibility that if the middle portion of any multivariate system is homogeneous of degree two, then it is possible to find the constant parts of both affine mappings easily. The φ_2 in our TRMS example is not homogeneous.

XL Family and Gröbner Bases: Courtois et al proposed the XL method for solving overdetermined quadratic system (which can be viewed as a refinement of the relinearization method by Kipnis-Shamir, [24]) and its variant FXL in [11]. Faugère ([15, 16]) have been improving algorithms for computing Gröbner Bases, and the current state-of-the art variant is $\mathbf{F_5}$, which was used as the critical equation solver in breaking the HFE challenge 1 ([17]). The consensus of current research ([1–3, 13, 39, 41]) is that Gröbner/XL-like equation solvers on generic equations are exponential in the number of variables. The best variant will be F$\mathbf{F_5}$ if $O(n^{2+\varepsilon})$ timing can be achieved, and FXL otherwise. The time complexity for the two methods on a system with $m = 20$ equations will be respectively 2^{74} and 2^{76} 3DES units, still better than RSA-1024 (see [29]). If $m = 24$, then we would get 2^{80} and 2^{81} respectively.

Remark: The speed estimates on nongeneric equations are still being debated, but the converse to Moh's lemma was proved in [39], which shows that it is likely that all Gröbner/XL-like equation solvers will run into trouble if the dimension of the projective solution set at infinity (denoted $\dim H_\infty$) is non-zero. It is not very easy to benefit from this, however, because the UOV attack means that the last stage of our sample TRMS scheme or something similar cannot be too large, and the dual rank attack dictates that it cannot be too small! Thus for $m = 20$, we cannot benefit $\dim H_\infty > 0$, because the last stage is forced to be 9 variables. For larger TRMS schemes, say $m = 28$ upwards, we can start to do better with optimal selection of parameters.

Finding Minus and Vinegar Variables: These are very specialized methods designed against what is generally called "Big-Field" multivariate schemes such as C^{*--}. They do not work against tame-like multivariates with non-constant central parts.

Patarin's IP Approach: Patarin et al proposed an attack method for fixed middle map schemes in [31, 32]. Since there are variable parameters in the middle map, the IP attack is not applicable.

Search Methods: Courtois et al proposed some search methods at PKC 2002 in [7]. However, they are mainly designed for small finite fields, and we may follow the computations of [4] to find a complexity of 2^{120} 3DES units.

4.2 Comparison to Enhanced-TTS

The structure of the latest version of TTS, Enhanced-TTS is as follows. Fix a finite field $\mathbb{K}$. Choose three natural numbers m, n, k such that $m < n$ and $k < n - m$. Let φ_1, φ_3 are invertible affine maps on $\mathbb{K}^n$ and $\mathbb{K}^m$ respectively. Let $\varphi_2 : \mathbb{K}^n \to \mathbb{K}^m$ be of the following form. (Below f_i's are all quadratic and $\mathbf{y} = (y_{n-m+1}, \ldots, y_n)$.)

$$
\begin{cases}
r_1 = x_1 \\
r_2 = x_2 \\
\vdots \quad \vdots \\
r_{n-m} = x_{n-m} \\
\begin{pmatrix} y_{n-m+1} \\ y_{n-m+2} \\ \vdots \\ y_{n-k-j} \end{pmatrix} = \begin{pmatrix} \text{invertible} \\ \text{matrix of} \\ \text{linear} \\ \text{expressions of} \\ x_1, \ldots, x_{n-m} \end{pmatrix} \begin{pmatrix} x_{n-m+1} \\ x_{n-m+2} \\ \vdots \\ x_{n-k-j} \end{pmatrix} + \begin{pmatrix} \text{column} \\ \text{vector of} \\ \text{quadratic} \\ \text{expressions of} \\ x_1, \ldots, x_{n-m} \end{pmatrix} \\
y_{n-k-j+1} = x_{n-k-j+1} + f_{n-k-j+1}(x_1, x_2, \ldots, x_{n-k-j}) \\
\vdots \quad \vdots \\
y_{n-k} = x_{n-k} + f_{n-k}(x_1, x_2, \ldots, x_{n-k-1}) \\
\begin{pmatrix} y_{n-k+1} \\ y_{n-k+2} \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} \text{invertible} \\ \text{matrix of} \\ \text{linear} \\ \text{expressions of} \\ x_1, \ldots, x_{n-k} \end{pmatrix} \begin{pmatrix} x_{n-k+1} \\ x_{n-k+2} \\ \vdots \\ x_n \end{pmatrix} + \begin{pmatrix} \text{column} \\ \text{vector of} \\ \text{quadratic} \\ \text{expressions of} \\ x_1, \ldots, x_{n-k} \end{pmatrix}
\end{cases}
$$

The verification map V can be decomposed as $\mathbf{w} \in \mathbb{K}^n \overset{\varphi_1}{\mapsto} \mathbf{x} \overset{\varphi_2}{\mapsto} \mathbf{y} \overset{\varphi_3}{\mapsto} \mathbf{z} \in \mathbb{K}^m$. That is, $V = \varphi_3 \circ \varphi_2 \circ \varphi_1$, where $\mathbf{x} = \varphi_1(\mathbf{w}) = \mathsf{M}_1\mathbf{w} + \mathbf{c}_1$, $\mathbf{z} = \varphi_3(\mathbf{y}) = \mathsf{M}_3\mathbf{y} + \mathbf{c}_3$ and $(r_1, r_2, \ldots, r_{n-m}, \mathbf{y}) = \varphi_2(\mathbf{x})$.

To sign a message, Enhanced-TTS needs to solve two systems of equations for finding one inverse image point of the middle map. There is about 1/25 chance of redoing the signing procedure for the implement in [38]. However, our TRMS example has constant signing time, since the non-zero element in a field is always invertible.

Regarding to signing time, TRMS is better than TTS. One reason is that TRMS utilizes special field extension structure to reduce the computation time for φ_2^{-1}, the details is in the Appendix, while TTS only uses the common method of Gaussian elimination. Another reason is that during computation of the affine transformations φ_1, φ_3, part of it is also carried out in a larger field, which will benefit the computation, too. We like to point out that there are a lot of ways to construct φ_1, φ_3. One reason for us to use the LU-decomposition is that it has advantages when implemented on smart cards.

The main external differences between TRMS(20,28) and Enhanced-TTS(20,28) can be tabulated as follows.

1. The private key size for TTS is 1.4KB, while for TRMS it is 396 bytes.
2. Regarding to signing time, TRMS is better than TTS.
3. TTS has at most 7% chance of redoing the signing procedure while the signing time for TRMS is constant.

5 Appendix: Implement of Field Extension

Firstly, $GF(2) = \{(0)_2, (1)_2\}$, where $(\cdot)_2$ means the binary representation. Then $t^2 + t + (1)_2$ is irreducible over $GF(2)$. Let $GF(4) = GF(2)[t]/(t^2 + t + (1)_2)$ and $(ab)_2$ denote the equivalent class of $at + b$. Then we have the following multiplication table.

	$(00)_2$	$(01)_2$	$(10)_2$	$(11)_2$
$(00)_2$	$(00)_2$	$(00)_2$	$(00)_2$	$(00)_2$
$(01)_2$	$(00)_2$	$(01)_2$	$(10)_2$	$(11)_2$
$(10)_2$	$(00)_2$	$(10)_2$	$(11)_2$	$(01)_2$
$(11)_2$	$(00)_2$	$(11)_2$	$(01)_2$	$(10)_2$

Similarly, we have $t^2 + t + (10)_2$ is irreducible over $GF(4)$. Let $GF(16) = GF(4)[t]/(t^2+t+(10)_2)$ and $(abcd)_2$ denote the equivalent class of $(ab)_2t+(cd)_2$. Then we can construct a multiplication table of size 16×16.

Similarly, we have $t^2+t+(1000)_2$ is irreducible over $GF(16)$. Let $GF(256) = GF(16)[t]/(t^2 + t + (1000)_2)$ and $(abcdefgh)_2$ denote the equivalent class of $(abcd)_2t+(efgh)_2$. Then we can construct a multiplication table of size 256×256.

Similarly, we have $t^2 + t + (1000,0000)_2$ is irreducible over $GF(256)$. Let $\alpha_1 = (1000,0000)_2$. Let $GF(2^{16}) = GF(256)[t_1]/(t_1^2 + t_1 + \alpha_1)$. However, we do not construct the multiplication table of $GF(2^{16})$. For $a, b, c, d \in GF(256)$, $(at_1 + b)(ct_1 + d) = act_1^2 + (ad + bc)t_1 + bd = ac(t_1 + \alpha_1) + (ad + bc)t_1 + bd = [(a + b)(c + d) + bd]t_1 + [ac\alpha_1 + bd]$.

Similarly, we have $t^2+t+(1000,0000,0000,0000)_2$ is irreducible over $GF(2^{16})$. Let $\alpha_2 = (1000,0000,0000,0000)_2$. Let $GF(2^{32}) = GF(2^{16})[t_2]/(t_2^2 + t_2 + \alpha_2)$. For $A, B, C, D \in GF(2^{16})$, $(At_2 + B)(Ct_2 + D) = [(A + B)(C + D) + BD]t_2 + [AC\alpha_2 + BD]$.

Note that we now have a recursive definition for $GF((2^8)^{(2^i)})$. With a proper choice of α_i, we let $GF((2^8)^{(2^i)}) = GF((2^8)^{(2^{i-1})})[t_i]/(t_i^2+t_i+\alpha_i)$. For $a, b, c, d \in GF((2^8)^{(2^{i-1})})$,

$$(at_i + b)(ct_i + d) = [(a + b)(c + d) + bd]t_i + [ac\alpha_i + bd]$$

where the addition is the bitwise XOR and the multiplication of expressions of a, b, c, d and α_i are done in $GF((2^8)^{(2^{i-1})})$.

To find the inverse of $at_i + b$, first we let $(at_i + b)(At_i + B) = 1$, that is, $(aA + aB + Ab)t_i + aA\alpha_i + bB = 1$ or, in vector form, by considering $\{t_i, 1\}$ as a basis, $\begin{pmatrix} a+b & a \\ a\alpha_i & b \end{pmatrix}\begin{pmatrix} A \\ B \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$. Hence $\begin{pmatrix} A \\ B \end{pmatrix} = \begin{pmatrix} a+b & a \\ a\alpha_i & b \end{pmatrix}^{-1}\begin{pmatrix} 0 \\ 1 \end{pmatrix} = (ab + b^2 + a^2\alpha_i)^{-1}\begin{pmatrix} b & a \\ a\alpha_i & a+b \end{pmatrix}\begin{pmatrix} 0 \\ 1 \end{pmatrix} = (ab + b^2 + a^2\alpha_i)^{-1}\begin{pmatrix} a \\ a+b \end{pmatrix}$. Therefore $(at_i + b)^{-1} = (at_i + a + b)(ab + b^2 + a^2\alpha_i)^{-1}$.

Here we give an example of field extension of degree 12 to illustrate how we can accelerate the computation of large field. We let $\mathbb{K} = GF(2^8)$ and $\mathbb{L}, \mathbb{L}', \mathbb{L}''$ be the finite extension fields of $\mathbb{K}$ such that $\mathbb{K} \subset \mathbb{L}'' \subset \mathbb{L}' \subset \mathbb{L}$ and $[\mathbb{L}'' : \mathbb{K}] = 2$, $[\mathbb{L}' : \mathbb{L}''] = 2$, $[\mathbb{L} : \mathbb{L}'] = 3$. Therefore $\mathbb{L}' = GF(2^{16})$, $\mathbb{L}' = GF(2^{32}) \subset \mathbb{L}$, and $\mathbb{L} = GF(2^{96})$ and we need to discuss the field extension of degree 3 below.

Since $t^3 + t + 1$ is irreducible over $GF(2^{32})[t]$, we can identify $GF(2^{96})$ with $GF(2^{32})[t]/(t^3 + t + 1)$. If we use $\begin{pmatrix} a \\ b \\ c \end{pmatrix}$ to represent $at^2 + bt + c$, then

$$\begin{pmatrix} a \\ b \\ c \end{pmatrix} *_{12} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} (a+c) & b & a \\ (a+b) & (a+c) & b \\ b & a & c \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

where $*_{12}$ denotes the multiplication in $\mathbb{L}$ and the right hand side is just the usual matrix multiplication. In signing a message, we need to solve $\mathbf{ax} = \mathbf{y}$ for $\mathbf{x}$ in $\mathbb{L}$. That is, to solve

$$\begin{pmatrix} (a+c) & b & a \\ (a+b) & (a+c) & b \\ b & a & c \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}$$

for x_1, x_2, x_3. Therefore, we have

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \frac{1}{\triangle}\left[\mathbf{adj}\begin{pmatrix} (a+c) & b & a \\ (a+b) & (a+c) & b \\ b & a & c \end{pmatrix}\right]\begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}.$$

Write out $\mathbf{adj}\begin{pmatrix} (a+c) & b & a \\ (a+b) & (a+c) & b \\ b & a & c \end{pmatrix}$ as $\begin{pmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{pmatrix}$, then

$$\begin{aligned} A_{31} &= A_{12} = A_{23} = a^2 + bc \\ A_{11} &= A_{22} = a(b + c) + c^2 \\ A_{32} &= A_{13} = ac + (a + b)^2 \\ A_{21} &= A_{31} + A_{13} \\ A_{33} &= A_{22} + A_{32} \\ \triangle &= aA_{31} + bA_{32} + cA_{33} \end{aligned}$$

According to the calculation above, to solve $\mathbf{ax} = \mathbf{y}$, we need 21 multiplications and one inverse operation in $GF(2^{32})$, which is roughly 342 multiplications

in $\mathbb{K}$. Comparing to TTS, doing the Gaussian elimination for two 9×9 matrices, it takes at least about $2 \times 9^3/3 \approx 500$ multiplications in $\mathbb{K}$.

Note: There will be a extended version at IACR eprint archive.

References

1. G. Ars and J.-C. Faugère, *Comparison of XL and Gröbner Bases Algorithms over Finite Fields*, preprint. Will appear as one half of an article at Asiacrypt 2004 and LNCS.
2. M. Bardet, J.-C. Faugère, and B. Salvy, *Complexity of Gröbner Basis Computations for Regular Overdetermined Systems*, INRIA Rapport de Recherche No. 5049; a slightly modified preprint is accepted by the International Conference on Polynomial System Solving.
3. M. Bardet, J.-C. Faugère, B. Salvy, and B.-Y. Yang, *Asymptotic Complexity of Gröbner Basis Algorithms for Semi-regular Overdetermined Systems over Large Fields*, manuscript in preparation.
4. J.-M. Chen and B.-Y. Yang, *Tame Transformations Signatures With Topsy-Turvy Hashes*, proc. IWAP 2002, Taipei.
5. J.-M. Chen and B.-Y. Yang, *A More Secure and Efficacious TTS Scheme*, ICISC 2003, LNCS v. 2971, pp. 320-338; full version at `eprint.iacr.org/2003/160`.
6. D. Coppersmith, J. Stern, and S. Vaudenay, *Attacks on the Birational Permutation Signature Schemes*, Crypto 1993, LNCS v. 773, pp. 435–443.
7. N. Courtois, L. Goubin, W. Meier, and J. Tacier, *Solving Underdefined Systems of Multivariate Quadratic Equations*, PKC 2002, LNCS v. 2274, pp. 211-227
8. N. Courtois, *Generic Attacks and the Security of Quartz*, PKC 2003, LNCS v. 2567, pp. 351-364.
9. N. Courtois, *Algebraic Attacks over $GF(2^k)$, Cryptanalysis of HFE Challenge 2 and SFLASH^{v2}*, accepted for PKC 2004.
10. N. Courtois, A. Klimov, J. Patarin, and A. Shamir, *Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations*, EUROCRYPT 2000, LNCS v. 1807, pp. 392-407.
11. N. Courtois and J. Patarin, *About the XL Algorithms over $GF(2)$*, CT-RSA 2003, LNCS v. 2612, pp. 141-157.
12. N. Courtois, L. Goubin, and J. Patarin, *SFLASH^{v3}, a Fast Asymmetric Signature Scheme*, preprint
13. C. Diem, *The XL-algorithm and a Conjecture from Commutative Algebra*, preprint (to appear Asiacrypt 2004 and LNCS) and private communication.
14. W. Diffie and M. Hellman, *New Directions in Cryptography*, IEEE Trans. Info. Theory, vol. IT-22, no. 6, pp. 644-654.
15. J.-C. Faugére, *A New Efficient Algorithm for Computing Gröbner Bases (F4)*, Journal of Pure and Applied Algebra, 139 (1999), pp. 61–88.
16. J.-C. Faugère, *A New Efficient Algorithm for Computing Gröbner Bases without Reduction to Zero (F5)*, Proc. ISSAC 2002, pp. 75-83, ACM Press 2002.
17. J.-C. Faugère and A. Joux, *Algebraic Cryptanalysis of Hidden Field Equations (HFE) Cryptosystems Using Gröbner Bases*, Crypto 2003, LNCS v. 2729, pp. 44-60.
18. M. Garey and D. Johnson, *Computers and Intractability, A Guide to the Theory of NP-completeness*, 1979, p. 251.
19. W. Geiselmann, R. Steinwandt, and T. Beth, *Attacking the Affine Parts of SFLASH*, 8th International IMA Conference on Cryptography and Coding, LNCS v. 2260, pp. 355-359.

20. W. Geiselmann, R. Steinwandt, and T. Beth, *Revealing the 441 Key Bits of SFLASH^{v2}*, Third NESSIE Workshop, 2002.
21. L. Goubin and N. Courtois, *Cryptanalysis of the TTM cryptosystem*, Asiacrypt 2000, LNCS v. 1976, pp. 44-57.
22. A. Kipnis and A. Shamir, *Cryptanalysis of the Oil and Vinegar Signature Scheme*, Crypto'98, LNCS v. 1462, pp. 257-266
23. A. Kipnis, J. Patarin, and L. Goubin, *Unbalanced Oil and Vinegar Sigature Schemes*, Crypto'99, LNCS v. 1592, pp. 206-222
24. A. Kipnis and A. Shamir, *Cryptanalysis of the HFE Public Key Cryptosystem by Relinearization*, Crypto'99, LNCS v. 1666, pp. 19-30
25. T. Matsumoto and H. Imai, *Public Quadratic Polynomial-Tuples for Efficient Signature-Verification and Message-Encryption*, EUROCRYPT'88, LNCS v. 330, pp. 419-453.
26. T. Moh, *A Public Key System with Signature and Master Key Functions, Communications in Algebra*, 27 (1999), pp. 2207-2222.
27. T. Moh and J. -M. Chen, *On the Goubin-Courtois Attack on TTM*, published electronically by Cryptology ePrint Archive (2001/072).
28. *New European Schemes for Signatures, Integrity, and Encryption*, project homepage at http://www.cryptonessie.org.
29. *Performance of Optimized Implementations of the NESSIE primitives, version 2.0* http://www.cryptonessie.org.
30. J. Patarin, *Cryptanalysis of the Matsumoto and Imai Public Key Scheme of Eurocrypt'88*, Crypto'95, LNCS v. 963, pp. 248-261.
31. 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.
32. J. Patarin, L. Goubin, N. Courtois, *Improved Algorithm for Isomorphisms of Polynomials*, EUROCRYPT'98, LNCS v. 1403, pp. 184-200.
33. J. Patarin, N. Courtois, and L. Goubin, *QUARTZ, 128-Bit Long Digital Signatures*, CT-RSA 2001, LNCS v. 2020, pp. 282-297. Updated version available at http://www.cryptonessie.org.
34. J. Patarin, N. Courtois, and L. Goubin, FLASH, *a Fast Multivariate Signature Algorithm*, CT-RSA 2001, LNCS v. 2020, pp. 298-307. Updated version available at http://www.cryptonessie.org.
35. A. Shamir and E. Tromer, *Factoring Large Numbers with the TWIRL Device*, Crypto 2003, LNCS v. 2729, pp. 1-26.
36. Lih-Chung Wang and Fei-Hwang Chang, *Tractable Rational Map Cryptosystem*, available at http://eprint.iacr.org/2004/046.
37. C. Wolf, *Efficient Public Key Generation for Multivariate Cryptosystems, preprint*, available at http://eprint.iacr.org/2003/089.
38. B.-Y. Yang and J.-M. Chen, *Rank Attacks and Defence in Tame-Like Multivariate PKC's*, see http://eprint.iacr.org/2004/061.
39. B.-Y. Yang and J.-M. Chen, *All in the XL Family: Theory and Practice*, to appear at ICISC 2004 and LNCS.
40. B.-Y. Yang, Y.-H. Chen, and J.-M. Chen, *TTS: High-Speed Signatures on a Low-End Smart Card*, Proc. CHES '04, LNCS v. 3156, pp. 371-385.
41. B.-Y. Yang, J.-M. Chen, and N. Courtois, *On Asymptotic Security Estimates in XL and Gröbner Bases-Related Algebraic Cryptanalysis*, ICICS 2004, LNCS v. 3269, pp. 401-413.