

A More Secure and Efficacious TTS Signature Scheme

Jiun-Ming Chen¹ and Bo-Yin Yang²

¹ Chinese Data Security, Inc. *and* Mathematics Department, National Taiwan University, Taipei, Taiwan. jmchen@math.ntu.edu.tw

² Mathematics Department, Tamkang University, Tamsui, Taiwan. by@moscito.org*

Abstract. In 2002 the authors introduced the new genre of digital signature scheme TTS (Tame Transformation Signatures) along with a sample scheme TTS/2. TTS is from the family of multivariate cryptographic schemes to which the NESSIE primitive SFLASH also belongs. It is a realization of T. Moh's theory ([31]) for digital signatures, based on Tame Transformations or Tame Maps. Properties of multivariate cryptosystems are determined mainly by their central maps. TTS uses Tame Maps as their central portion for even greater speed than C^* -derived SFLASH family of schemes, which uses monomials in a large field for the central portion, previously usually acknowledged as fastest.

We show a small flaw in TTS/2 and present an improved TTS implementation which we call TTS/4. We will examine in some detail how well TTS/4 performs, how it stands up to previously known attacks, and why it represents an advance over TTS/2. Based on this topical assessment, we consider TTS in general and TTS/4 in particular to be competitive or superior in several aspects to other schemes, partly because the theoretical roots of TTS induce many good traits. One specific area in which TTS/4 should excel is in low-cost smartcards. It seems that the genre has great potential for practical deployment and deserves further attention by the cryptological community.

Key Words: multivariate, public-key, digital signature, finite field, tame transformation.

1 Introduction

Trapdoor mappings are central to Public-Key Cryptography. As such, cryptographers have studied trapdoor permutations and maps since the dawn of public key cryptography ([13]). A handful of the many schemes attempted reached practical deployment. However, *the critical trapdoor maps are often very slow, and that is frequently due to the sheer size of the algebraic structure.* Typical are the modular exponentiation in RSA or the discrete logarithms in ElGamal/DSA/ECC.

Multivariate public-key cryptosystems were born partly to circumvent this limitation. Great names like Shamir ([36]) and Diffie ([17]) abound among the pioneers, but the first scheme to show promise and grab everyone's attention was C^* by Imai and Matsumoto ([24, 30]). Unfortunately a few years later C^* was broken by Patarin ([37]) who in turn introduced many of his own innovations, some (HFE/ C^* - families) of which are still extant. The variations on this theme seem endless.

* Corresponding author

TTS (Tame Transformation Signatures) digital signature schemes belong to this extended family. We propose TTS/4, an improvement variant of TTS and discuss some design, performance, and security issues associated with the TTS genre. Multivariates mainly differ in their central maps, or *kernels*, that determine the trapdoors and hence their security. We aim to show that the Tame Transformation, a *biregular map*³ first introduced by T. Moh to cryptography, is a viable kernel for trapdoor permutations with appreciable gains versus other schemes, and one that warrants further investigation.

Sec. 2 is a succinct summary of TTS with theory and example. Like other multivariate schemes, TTS can be flexible in terms of hash length and is easily adaptable to 256-bit or longer hashes if needed, but TTS/4 is designed to work with current 160-bit hashes like SHA-1. We will quantify how well TTS/4 does by various metrics of speed and key size in Sec. 3. It compares well with some better-known alternatives. We see that TTS/4 is especially fast in signing and should be very suitable for use on a smartcard as seen in a point-by-point comparison with the SFLASH^{v2} scheme recommended by the NESSIE (New European Schemes for Signatures, Integrity, and Encryption) project for the same purpose ([1, 35]).

Avoiding the pitfalls that ensnared other schemes should be central to design decisions taken in present and future schemes, hence the multiplicity of techniques of Sec. 4 serves as an illustrative backdrop to TTS/4. We show that the TTS family can be made safe against current attacks.

2 Tame Transformation and TTS

While clock speeds went up according to Moore’s law, unfortunately so did the complexity skyrocket and key lengths exponentiate. In a quarter-century, no alternative to the venerable RSA ever showed enough of a speed gain to become the heir apparent. Indeed, multivariate public-key cryptography arose out of this need for faster algorithms.

Cryptographers also applied their considerable talents to seeking faster alternatives in *birational* permutations ([17, 43]) over two decades. Birational implies *being polynomial or rational, with a polynomial or rational inverse*. Regrettably, an explicit low-degree inverse that brings swiftness in execution sometimes has the undesirable side effect of engendering vulnerabilities ([7, 8, 17]).

T. Moh first took *Tame Transformations* into the landscape of Cryptography from their native habitat of Algebraic Geometry ([31]). A Tame Transformation over a field K , which shall be $\text{GF}(2^8)$ unless otherwise specified, is either invertible and affine ($\mathbf{x} \mapsto \mathbf{y} = \mathbf{M}\mathbf{x} + \mathbf{c}$) or given by a set of polynomial relations of this form $\phi : \mathbf{x}(\in K^n) \mapsto \mathbf{y}(\in K^n)$:

$$\begin{aligned} y_1 &= x_1; \\ y_2 &= x_2 + q_2(x_1); \\ y_3 &= x_3 + q_3(x_1, x_2); \\ &\vdots \quad \quad \quad \vdots \\ y_n &= x_n + q_n(x_1, x_2, \dots, x_{n-1}); \end{aligned}$$

³ a bijective map that is polynomial both ways.

the indices of x_i can be permuted. Basic properties of tame transformations are:

- We can compute the preimage $\mathbf{x} = \phi^{-1}(\mathbf{y})$ as easily as $\mathbf{y} = \phi(\mathbf{x})$; but
- it is difficult to write \mathbf{x} explicitly as a function of \mathbf{y} . As we solve for each x_i serially, the degree of the polynomials expressing x_i in y_j escalate quickly — exponentially, even if most q_k 's are merely quadratic:

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

In this paper we loosely call a map *tame-like* or just *tame* when it is either a tame transformation, or if it retains, at least where it matters, the property of having *at least one* preimage with easy serial computation through substitution or the solution of only linear equations but not an explicit inverse of low degree. So a tame-like map can be neither injective nor surjective.

In general the verification map of any TTS scheme is $V : \text{GF}(2^8)^n \rightarrow \text{GF}(2^8)^m$ where $n > m$ and use only a single tame map. In contrast TTM, the family of public-key encryption scheme that is companion to TTS, uses two tame maps. We illustrate with an example first.

2.1 A Toy Example

We use a verification map $V = \phi_3 \circ \phi_2 \circ \phi_1 : \text{GF}(2)^5 \rightarrow \text{GF}(2)^3$ composed thusly:

$$\begin{array}{ccc} \phi_3 & \phi_2 & \phi_1 \\ \begin{bmatrix} z_0 \\ z_1 \\ z_2 \end{bmatrix} = \mathbf{M}_3 \begin{bmatrix} y_2 \\ y_3 \\ y_4 \end{bmatrix} + \mathbf{c}_3 & \begin{aligned} y_2 &= x_2 + a_2 x_0 x_1 \\ y_3 &= x_3 + a_3 x_1 x_2 \\ y_4 &= x_4 + a_4 x_2 x_3 \end{aligned} & \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \mathbf{M}_1 \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix} + \mathbf{c}_1 \end{array}$$

Normally, in $\text{GF}(2^8)$, we choose arbitrarily the nonzero parameters a_i . Here each a_i has to be 1. We can pick any \mathbf{c}_1 , and the invertible matrices \mathbf{M}_1 and \mathbf{M}_3 , but we will compute a \mathbf{c}_3 so that all the constant terms vanish. Suppose

$$\mathbf{c}_1 = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}; \quad \mathbf{M}_1 = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix}; \quad \mathbf{M}_3 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}.$$

We compose them to get $\mathbf{c}_3 = (0, 1, 0)$ and this $\mathbf{z} = V(\mathbf{w})$ (note: $w_i^2 = w_i$ in $\text{GF}(2)$).

$$z_0 = w_0 + w_1 + w_2 + w_3 + w_0w_1 + w_0w_2 + w_1w_3 + w_1w_4 + w_2w_4 + w_3w_4;$$

$$z_1 = w_2 + w_4 + w_0w_3 + w_1w_2 + w_1w_3 + w_1w_4 + w_2w_3 + w_2w_4 + w_3w_4;$$

$$z_2 = w_0 + w_2 + w_0w_2 + w_0w_3 + w_0w_4 + w_1w_2 + w_1w_3 + w_1w_4 + w_2w_3 + w_3w_4.$$

These quadratic polynomials form our verification function or public key. The private key would be the a_i 's, \mathbf{c}_1 , \mathbf{c}_3 , and the inverses M_1^{-1} and M_3^{-1} . Suppose a mini-hash value $\mathbf{z} = (1, 1, 0)$ is given, we will compute a set of $\mathbf{w} = (w_0, w_1, w_2, w_3, w_4)$ that satisfies these equations. We can compute the signature $S(\mathbf{z}) = \phi_1^{-1}(\phi_2^{-1}(\phi_3^{-1}(\mathbf{z})))$ by solving three sequential sets of solutions:

1. $\mathbf{y} = M_3^{-1}(\mathbf{z} - \mathbf{c}_3) = (1, 1, 1)$ is straightforward⁴.
2. The solution for \mathbf{x} is clearly not unique, but assigning values to x_0 and x_1 will force the rest and there are four possible values for \mathbf{x} : $(0, 0, 1, 1, 0)$, $(0, 1, 1, 0, 1)$, $(1, 0, 1, 1, 0)$, $(1, 1, 0, 1, 1)$.
3. As $\mathbf{w} = M_1^{-1}(\mathbf{x} - \mathbf{c}_1)$, we can compute all four \mathbf{w} values, which turns out to be $(1, 1, 0, 1, 1)$, $(1, 0, 0, 1, 1)$, $(1, 0, 0, 0, 1)$, $(1, 1, 1, 0, 1)$. Each could be the signature attached to the message. The recipient can verify that the signature and the hash value indeed satisfy $\mathbf{z} = V(\mathbf{w})$.

In this toy example a brute-force search takes no time. However, assuming that V does not easily decompose into its original components, and that solving for \mathbf{w} is hard with real-life sized parameters then we have a secure signature that cannot easily be forged. We can see also that our trapdoor provides for a very fast signing procedure, taking not much more time than two matrix multiplications.

2.2 A Generic Form for TTS

Let K be a field. The verification map $V = \phi \circ \phi_2 \circ \phi_1 : K^n \rightarrow K^m$ is the public key, where $\phi_1 : \mathbf{w} \mapsto \mathbf{x} = M_1\mathbf{w} + \mathbf{c}_1$ and $\phi_3 : \mathbf{y} \mapsto \mathbf{z} = M_3\mathbf{y} + \mathbf{c}_3$ are invertible affine maps in K^n and K^m respectively. $\phi_2 : K^n \rightarrow K^m$ is a tame map, called the central map or the kernel, and contains a lot of parameters. $S = \phi_1^{-1} \circ \phi_2^{-1} \circ \phi_3^{-1}$, where ϕ_2^{-1} takes *any* preimage of ϕ_2 and does not have to be deterministic, is the signing map. The information content of $(\phi_1^{-1}, \phi_2^{-1}, \phi_3^{-1})$ is the private key.

As pointed out in [31], using a bijective affine-tame-affine public map is inadvisable because the initial equations $y_1 = x_1$ and $y_2 = x_2 + ax_1^2$ represent intrinsic vulnerabilities. TTM must resort to using another tame map because a public encryption map must retain all information content. In fact, T. Moh's prototype signature scheme ([31]) also used two tame maps, but TTS was designed with a more lightweight single tame-map approach, concentrating on being a good signature scheme.

In TTS, the signing map S may add extra information that we need not preserve through the verification process V . Hence, some initial dimensions can and will be collapsed by the central map ϕ_2 :

⁴ all minus signs in this section could just have been pluses because of characteristic 2.

$$\phi_2 = [\text{projection collapsing initial } (n - m) \text{ coordinates}] \circ [\text{tame-like map}].$$

Otherwise stated, we discard the first $n - m$ coordinates after a tame or tame-like transformation. Geometrically, given a message digest, we hide an $(n - m)$ -dimensional *algebraic variety* in K^n consisting of all possible digital signatures. The probability of guessing a point on the variety correctly is $\approx |K|^{-m}$, e.g., $\approx 2^{-160}$ as $K = \text{GF}(2^8)$ and $m = 20$ in our proposal. Suppose our central map takes this form:

$$\begin{aligned} \phi_2 : \mathbf{x} = (x_0, x_1, \dots, x_{n-1}) &\mapsto \mathbf{y} = (y_{n-m}, \dots, y_{n-1}), \\ y_k &= x_k + f_k(x_0, \dots, x_{k-1}), \text{ for } k = n - m, \dots, n - 1, \end{aligned}$$

where f_k 's are quadratic polynomials, then ϕ_2^{-1} can be computed thus when signing:

$$\begin{aligned} x_k &= \text{random variable } r_k \text{ in } K, \text{ for } k = 0, \dots, n - m - 1, \\ x_k &= y_k - f_k(x_0, \dots, x_{k-1}), \text{ for } k = n - m, \dots, n - 1. \end{aligned}$$

For security reasons, the tame-like maps used in current TTS deviate slightly from the form above, while retaining the basic properties of being *flexible in hash sizes* (because n and m can be adjusted easily) and being *serially and quickly solvable in each x_k* . Note that *the polynomials f_k can contain many arbitrary parameters* which will be randomly chosen and incorporated into the private keys.

2.3 The Old: TTS/2, a Previously Proposed Variant

Hereafter we fix $K = \text{GF}(2^8)$. The signing map is $S = \phi_1^{-1} \circ \phi_2^{-1} \circ \phi_3^{-1} : K^{20} \rightarrow K^{28}$, where $\phi_1 : \mathbf{w} \mapsto \mathbf{x} = \mathbf{M}_1 \mathbf{w} + \mathbf{c}_1$ and $\phi_3 : \mathbf{y} \mapsto \mathbf{z} = \mathbf{M}_3 \mathbf{y} + \mathbf{c}_3$ are invertible affine in K^{28} and K^{20} respectively, and $\phi_2 : \mathbf{x} = (x_0, x_1, \dots, x_{27}) \mapsto \mathbf{y} = (y_8, y_9, \dots, y_{27})$ is given below:

$$\begin{aligned} y_8 &= x_8 + a_8 x_0 x_7 + b_8 x_1 x_6 + c_8 x_2 x_5 + d_8 x_3 x_4; \\ &\vdots \quad \vdots \quad \vdots \\ y_k &= x_k + a_k x_{k-8} x_{k-1} + b_k x_{k-7} x_{k-2} + c_k x_{k-6} x_{k-3} + d_k x_{k-5} x_{k-4}; \\ &\vdots \quad \vdots \quad \vdots \\ y_{27} &= x_{27} + a_{27} x_{19} x_{26} + b_{27} x_{20} x_{25} + c_{27} x_{21} x_{24} + d_{27} x_{22} x_{23}. \end{aligned}$$

To Generate Keys: Randomly choose $\mathbf{c}_1 \in K^{28}$, nonzero parameters a_i, b_i, c_i, d_i in K for $8 \leq i \leq 27$, invertible⁵ $\mathbf{M}_1 \in K^{28 \times 28}$ and $\mathbf{M}_3 \in K^{20 \times 20}$. Find \mathbf{M}_1^{-1} and \mathbf{M}_3^{-1} . Compute \mathbf{c}_3 so that constant terms of $V = \phi_3 \circ \phi_2 \circ \phi_1$ vanish. *The $20 \times 28 \times (28 + 3)/2 = 8680$ coefficients of V are the public key; ϕ_1^{-1}, ϕ_3^{-1} and parameters (a_i, b_i, c_i, d_i) form the private key (1312 bytes).*

To sign a message M : Compute the message digest $\mathbf{z} = H(M)$; compute $\mathbf{y} = (y_8, y_9, \dots, y_{27}) = \mathbf{M}_3^{-1}(\mathbf{z} - \mathbf{c}_3)$; pick x_0, \dots, x_7 randomly, then solve sequentially for each of the x_i for $i = 8 \dots 27$; the signature is $\mathbf{w} = \mathbf{M}_1^{-1}(\mathbf{x} - \mathbf{c}_1)$.

To verify (M, \mathbf{w}) : Compare $V(\mathbf{w}) = \phi_3 \circ \phi_2 \circ \phi_1(\mathbf{w})$ against the hash $\mathbf{z} = H(M)$.

This variant was proposed in [6]. We shall propose some changes and explain why.

⁵ usually by LU decomposition, even though it does not yield all nonsingular matrices.

2.4 The New: TTS/4, an Efficient and More Secure TTS

We use the new $\phi_2 : \mathbf{x} = (x_0, x_1, \dots, x_{27}) \mapsto \mathbf{y} = (y_8, y_9, \dots, y_{27})$ given by:

$$y_8 = x_8 + a_8 x_0 x_7 + b_8 x_1 x_4 + c_8 x_2 x_6 + d_8 x_3 x_5;$$

$$\vdots \quad \vdots \quad \vdots$$

$$y_k = x_k + a_k x_{k-8} x_{k-1} + b_k x_{k-7} x_{k-4} + c_k x_{k-6} x_{k-2} + d_k x_{k-5} x_{k-3};$$

$$\vdots \quad \vdots \quad \vdots$$

$$y_{23} = x_{23} + a_{23} x_{15} x_{22} + b_{23} x_{16} x_{19} + c_{23} x_{17} x_{21} + d_{23} x_{18} x_{20};$$

$$y_{24} = x_{24} + a_{24} x_{16} x_{23} + b_{24} x_{17} x_{20} + c_{24} x_{18} x_{22} + d_{24} x_4 x_{24};$$

$$y_{25} = x_{25} + a_{25} x_{17} x_{24} + b_{25} x_{18} x_{21} + c_{25} x_4 x_{23} + d_{25} x_5 x_{25};$$

$$y_{26} = x_{26} + a_{26} x_{18} x_{25} + b_{26} x_4 x_{22} + c_{26} x_5 x_{24} + d_{26} x_6 x_{26};$$

$$y_{27} = x_{27} + a_{27} x_4 x_{26} + b_{27} x_5 x_{23} + c_{27} x_6 x_{25} + d_{27} x_7 x_{27}.$$

Some particular design features (items 2, 4, 5 are new TTS security enhancements):

1. If we write $y_k = x_k + \mathbf{x}^T F_k \mathbf{x}$, there is no canonical (symmetric) form for F_k since $\text{char } K = 2$, but the matrix $F_k + F_k^T$ is unique. Here $F_k + F_k^T$ has rank 8 for every k because no x_i appear twice in a quadratic term of the same equation.
2. The final 4 equations deviate from the general form; where $x_{19}, x_{20}, x_{21}, x_{22}$ would be expected, the variables x_4, x_5, x_6, x_7 are substituted so that *at least one index in each quadratic term will be between 4 and 18 inclusive*. We will explain why in Secs. 4.2, and 4.3.
3. The rules are set up such that all eighty quadratic terms $x_i x_j$ are distinct.
4. The formula of y_k is different from the initial proposed form (TTS/2 of the previous section) in [6]. The indices in a quadratic term differ by 2, 3, 4, or 7 instead of 1, 3, 5, or 7. Principally, this is to avoid a separation of the x_i into even and odd indexed parts (see Sec. 4.5).
5. The last four equations has its corresponding x_i in a quadratic term. However, *the entire collection of relations is still a tame-like map*. The reason is

$$(1 + d_k x_{k-20}) x_k = y_k + (\text{function in } (x_0, \dots, x_{k-1})) \text{ for } k = 24 \dots 27.$$

Since x_4, x_5, x_6 , and x_7 are random numbers independent of the message digest, we pick them so that $1 + d_{24} x_4, 1 + d_{25} x_5, 1 + d_{26} x_6$, and $1 + d_{27} x_7$ are all non-zero⁶ which ensures that $x_{24} \dots x_{27}$ are easily solvable. See Sec. 2.5 and 4.4 for the reason for this change in design.

To generate a key pair and to verify a signed message (M, \mathbf{w}) : Unchanged from the above section.

To sign a message M : Compute digest $\mathbf{z} = H(M)$; compute $\mathbf{y} = (y_8, y_9, \dots, y_{27}) = \mathbf{M}_3^{-1}(\mathbf{z} - \mathbf{c}_3)$; pick x_0, \dots, x_7 randomly such that $x_k \neq d_{k+20}^{-1}$ for $k = 4 \dots 7$ (see item 5 above), then sequentially solve for x_i (for $i = 8 \dots 27$); the signature is $\mathbf{w} = \mathbf{M}_1^{-1}(\mathbf{x} - \mathbf{c}_1)$, release (M, \mathbf{w}) .

⁶ A good way to visualize the theme is to think of x_4, \dots, x_7 as *variable constants* and $1 + d_k x_{k-20}$ as *variable coefficient* of x_k for $k = 24 \dots 27$, that is made part of the signature.

Scheme	Signature	Pub. Key	Priv. Key	Keys Setup	Signing	Verifying
RSA-PSS	1024 bits	128 B	320 B	2.7 sec	84 ms	2.0 ms
ECDSA	326 bits	48 B	24 B	1.6 ms	1.9 ms	5.1 ms
ESIGN	1152 bits	145 B	96 B	0.21 sec	1.2 ms	0.74 ms
QUARTZ	128 bits	71.0 kB	3.9 kB	3.1 sec	11 sec	0.24 ms
SFLASH ^{v2}	259 bits	15.4 kB	2.4 kB	1.5 sec	2.8 ms	0.39 ms
TTS/2	224 bits	8.6 kB	1.3 kB	5.3 ms	35 μ s	0.13 ms
TTS/4	224 bits	8.6 kB	1.3 kB	5.3 ms	36 μ s	0.13 ms

Table 1. TTS and NESSIE round 2 candidates signature schemes on a 500MHz Pentium III

Since our programs were mainly proof-of-concept with clarity and ease of maintenance as the goal, we did not optimize to the hilt and compiled with the old `gcc-2.96`. With the newer `gcc3` or Intel’s `icc`, better programming and aggressive optimizations, we estimate that TTS/4 can be at least $1.5\times$ to $2\times$ faster. Still, TTS⁸ performed credibly against the NESSIE candidates, and we note that:

- Multivariate schemes are fundamentally identical during verification — just substituting into polynomials. Here, TTS is faster than its multivariate relatives SFLASH^{v2} ([42]) and QUARTZ ([41]) on due to its smaller dimensions. Of course, it can also be somewhat inconvenient if the dimensions are not divisible by 4, as in the FLASH family’s $K^{26} \rightarrow K^{37}$.

Signing is determined by the kernel (middle quadratic portion) action. QUARTZ is slow since it has to solve high-order equations. A FLASH-like scheme is faster. Any TTS variant runs especially fast because inverting a tame-like map is simple and straightforward.

- Comparison with RSA-PSS, ECDSA or ESIGN is a different kettle of fish altogether. Multivariate schemes⁹ is a lot faster than the more traditional contenders. Here, we have the modular exponentiation in RSA, a well-understood problem painstakingly optimized by over a quarter-century of computer science. Yet SFLASH^{v2} is faster and TTS/4 even more pronouncedly so. Clearly, there are intrinsic advantages to multivariate PKC. Due to concerns of security (see [34]), NESSIE recommends RSA-1536 with higher exponents as opposed to RSA-1024 and $e = 3$, which surely would further cut down the speed by at least 50% or more without special hardware. This decision can be considered at least somewhat vindicated with news of recent advances (e.g. [44]) on factoring large numbers.
- While TTS/4 (and in general TTS) does very well speedwise, unfortunately (although predictably) it is not the best in every category. All variants of TTS suffer from that common bugbear of multivariate cryptography: large keys. ECDSA is the undisputed king in small key sizes, although it uses discrete logarithms and is also slower than the multivariate.

Thankfully, now smart cards can have on-card storage upward of 32kB. The TTS public keys, while not so trim, is smaller than that of its more robust French cousins and tolerable at 8.6kB. It was mentioned by the authors of SFLASH ([42]) that another choice is to have the private key on card and be able to spit out the public

⁸ TTS/2 and TTS/4 are almost identical with TTS/4 needing four extra lookups.

⁹ With the exception of the slow signing action of QUARTZ.

key when needed; the same holds true for TTS/4. *Using this technique, we are able to fit TTS key generation and signing functions on a very low-end 8051 device with only 8kB of ROM and no extra RAM. The signing time is 133 ms and total key set up time around one minute. This is rather better than SFLASH^{v2}.*

A note about future extensibility. Eventually the world will move to 256-bit hashes, and a rough estimate is that an analog to TTS/4 will use about 2.5 times as many CPU cycles; in contrast, a RSA-based scheme gets to use longer hashes for free; ECDSA will be hurt grievously when forced up to a longer hash; SFLASH will probably scale slightly worse than a TTS-based scheme. All told, we expect TTS type schemes to be speed-competitive 1-2 generations of hashes down the road. We experimented with various constructions which showed that we can make security scale as needed (Sec. 4.2).

4 Cryptanalysis

Solving generic quadratic equation systems is NP-hard ([18]), so barring an inherent flaw in TTS, there should be no sub-exponential time algorithms. But we should still take note of essentially brute-force solutions because some practical-sized systems may be solved that way, and in reasonable time. Other ideas, some ingenious, are used to cryptanalyze other multivariate schemes. We examine all ideas known to us but each seems to fail against TTS/4 without substantial modification.

First we deal with the more general approaches, designed to be applicable against *all multivariate signature schemes*, and describe in order the state of the art methods of both brute force searches (Sec. 4.1) and the more circumspect techniques of linearization (Sec. 4.2) and Gröbner bases (Sec. 4.3), and how each functions against TTS/4. Then we complement the above content by discussing attacks applicable to particular multivariate signature schemes.

4.1 Search Methods

The most advanced search methods against multivariate quadratic signature schemes ([9]) known to us so far were presented at PKC 2002. We summarize and test each given method in [9] against the current TTS/4. In each method, the aim is to solve m quadratic equations in $(w_i)_{i=1\dots n}$ over $\text{GF}(q)$.

Algorithm A The general idea is as follows: Pick $2k$ equations and divide the variables into groups of k and $n - k$. Separate the quadratic equations into crossterms involving variables of both groups and quadratics dependent in each group of variables only, i.e. without loss of generality:

$$z_i = g_i(w_1, \dots, w_k) + \sum_{j=1}^k w_j \cdot \left(\sum_{\ell=k+1}^n \beta_{ij\ell} w_\ell \right) + h_i(w_{k+1}, \dots, w_n).$$

We impose $2k^2$ linear relations $\sum_{\ell=k+1}^n \beta_{ij\ell} w_\ell = \gamma_{ij}$ on the variables w_{k+1}, \dots, w_n . If $n \geq 2k(k+1)$ and $m \geq 2k$, then we can find $\bar{k} = (n - k) - 2k^2 \geq k$ independent variables $\bar{w}_1, \dots, \bar{w}_{\bar{k}}$ such that $h_i(w_{k+1}, \dots, w_n) = h'_i(\bar{w}_1, \dots, \bar{w}_{\bar{k}})$. The

equations become

$$g_i(w_1, \dots, w_k) + \sum_{j=1}^k \gamma_{ij} w_j = z_i - h'_i(\bar{w}_1, \dots, \bar{w}_k).$$

By evaluating the left side for all possible q^k combinations and storing the results, then evaluating the right side for all $q^{\bar{k}}$ combinations, i.e. using a birthday attack and trading space for time, this can be solved in $\approx 2q^{\bar{k}} k \bar{k}^2$ time instead of $q^{k+\bar{k}} k \bar{k}^2$. The extra factor is of course the number of multiplications needed to evaluate $2k$ polynomials in \bar{k} variables.

The upshot is the search will take $\approx q^{-k}$ as long as it was originally. [9] gives the complexity as $C_A \approx q^{m-k}$ where $k = \min(m/2, \lfloor \sqrt{n/2} - \sqrt{n/2} \rfloor)$. If we count operations per search unit, it really should be $\approx mn^2 q^{m-k}$. For TTS/4 with $(q, n, m) = (2^8, 28, 20)$, $k = 3$, and $C_A \approx 2^{151}$.

Algorithm B The general thrust is that k variables are eliminated before embarking on a brute-force search: Treat all quadratic terms $w_i w_j$ with $i, j \leq k$ as variables and eliminated first, leaving a linear system of equations in $w_1 \dots w_k$ to be solved in terms of the other variables. To give an illustrative example, in the case of TTS/4, there are 28 variables and 20 equations, and we can use 15 of the equations as a linear system to solve for the 15 quadratic terms $w_0^2, w_0 w_1, \dots, w_3 w_4, w_4^2$, and eliminate them from the remaining 5. Now we run a brute-force search on the variables w_5, \dots, w_{27} , substituting each set of values into the five equations left above and solve the resulting system for w_0, w_1, w_2, w_3, w_4 before making the consistency check. Not only have we saved the number of variables we have to guess by $k (= 5)$, instead of doing mn^2 finite field multiplications (lookups) per guess we now only have $\approx k(m-k)^2 + k^3/3$.

[9] gives the complexity as $C_B \approx K \cdot q^{m-k}$, where $k = \lfloor \sqrt{2m+2} - 3/2 \rfloor$. The coefficient $K = \max(C_2, C_3)$ where C_3 is the number of operations needed to solve a system of k linear equations ($k^3/3$ multiplications) and C_2 is given as $\approx (k(m-k)^2)$. It appears that the authors of [9] were slightly careless in their presentation, because the requirement for k , the larger the better, is really $m - k(k+1)/2 \geq k$ so for $m = 20$ as for TTS/4, instead of $k = 4$ and $q^{m-k} = 2^{128}$ we should have $q^{m-k} = 2^{120}$. They also probably should have written $C_2 + C_3$ instead of $\max(C_2, C_3)$. Anyway, $C_B \approx 2^{130}$.

Algorithm C The general approach of this attack is to reduce into XL/FXL, but it is as inapplicable to TTS/4 just as to TTS/2 ([6]), because it requires $n \geq 2m$ ([9]).

4.2 Linearization-Type Attacks: XL and FXL

Kipnis and Shamir first introduced ([27]) *relinearization*, refining the linearization techniques usually used to solve systems of high-order polynomial equations by using relationships between monomials. The simplest variant, “degree-4 relinearization”, is based on the simple fact that $(ab)(cd) = (ac)(bd) = (ad)(bc)$, in any field. Relinearization ameliorates somewhat the problem of too many extraneous solutions and is used

against HFE. There are more complex higher-degree improvements, but the relinearization technique can be considered superseded by XL below (Sec. 4.2), because XL (or FXL) is expected to work in whenever relinearization does ([11]).

XL and FXL XL (and its variant FXL) can be viewed as refinements of relinearization ([11]), although both normally work with more equations than variables. The procedure at degree- D on quadratic equations (l_j) is:

1. Generate all products of arbitrary monomials of degree $D - 2$ or less with each l_j ; linearize by considering every monomial as an independent variable.
2. Performing Gaussian elimination on the set of equations, ordering the set of variable such that monomials in a given variable (say the first variable w_0) will always be the last to go. The remaining equation should only have terms in w_0 left.
3. Solve *a la* Berlekamp for w_0 ; repeat if any independent variable remains.

The FXL variant takes every possible guess at a number of variables then uses XL on the remainder. Normally, we want the number of equations to be at least one and usually 2 more than the number of variables. Another XL derivative, XL', reduce down to a system of equations in more than one variable. A recent report has it XL' can break FLASH^{v2} ([10]). Simulations supposedly show XL/FXL as effective on randomly selected quadratics — which also points to its undoing, as we shall see below.

Hilbert-Serre Theory, Solution Sets at Infinity and Why XL/FXL Fails Linearization type attacks such as XL and relinearization have a fatal weakness. They are only effective under certain circumstances, one where in a sense the set of equations is generic. In more mathematical terms, if XL can guarantee a solution, *the solution set at ∞ must be at most zero-dimensional*. This came from a venerable theory of Hilbert and Serre ([32]). Of course, a cryptographer will by definition be using non-generic quadratics *the infinity solution set thereof he or she can ensure to be positive-dimensional*. An extension of the argument implies that XL' is ineffective for cryptanalyzing TTS.

Let $V = \phi_3 \circ \phi_2 \circ \phi_1$ be a verification map of some TTS variant. Given a message digest \mathbf{z} , forging a digital signature is equivalent to finding a solution to the system of equations $\mathbf{z} = V(\mathbf{w})$. We homogenize $\mathbf{z} = V(\mathbf{w})$ in the *projective space* and let H_∞ be its *solution set at infinity*. It so happens that $\dim H_\infty$ is an important parameter for multivariate schemes because it relates directly to security under XL/FXL and Gröbner bases attacks. We claim that $\dim H_\infty \geq 12$ for TTS/4.

Since both ϕ_1 and ϕ_3 are affine and invertible, we need to consider how ϕ_2 behaves at ∞ only. We can ignore linear terms in ϕ_2 , because to every non-highest-degree term is multiplied a positive power of an extra variable x_∞ during homogenization, and “at ∞ ” means precisely $x_\infty = 0$. Since all quadratic terms in ϕ_2 vanish when we set $x_4 = \dots = x_{18} = 0$, there are at least the 13 free variables $x_0, \dots, x_3, x_{19}, \dots, x_{27}$ in this solution set, hence $\dim H_\infty \geq 13 - 1 = 12$. The claim is proved.

If the attacker successfully guess at 8 variables, the number of variables n will reduce to 20 and $\dim H_\infty$ to 4. Of course, this is not guaranteed to work! Take the example in Sec. 2.1, for all \mathbf{w} in the solution set, $w_0 = w_4 = 1$. These are the *inessential*

variables, and a randomly guessed assignment or restrictions on such a variable would most likely leading to a contradiction.

Part of Heisuke Hironaka’s Fields Medal work was an algorithm to determine the essential variables thereof ([23]) over characteristic zero fields. Unfortunately we are working with characteristic two fields, so Hironaka’s methods (using many partial derivatives) fail in this case. Absent an oracle, *the attacker now must guess which variables to fix*, adding considerably to the running time.

Assuming that 8 variables *are* guessed correctly. Now $\dim H_\infty = 4$. Over $\text{GF}(2)$, XL/FXL can always work by including $w_i^2 = w_i$ for every i ([12]). Here, for each extra dimension in $H_\infty = 4$, the attacker needs to augment the set of equations by a number of quadratics equivalent to a Fröbenius relation $w_i^{2^{56}} = w_i$ for an essential variable w_i — maybe $w_i = p_1^2, p_1 = p_2^2, \dots, p_6 = p_7^2, p_7 = w_i^2$. In fact the XL/FXL attacker would need to add 32 *extraneous equations and 28 more variables*.

[11] claims a running time of $Aq^\mu n^{c\sqrt{n}}$ for XL/FXL, where A is the time needed to evaluate a set of polynomials, or about $mn^2/2$ multiplications; μ is the number of variables in which must be assigned by exhaustive search (“F” is for to fix); and c “the order of the Gaussian reduction algorithm”, which was claimed to be $\log_2 7 \approx 2.8$. We have $n = 48, m = 52$ now, and at least 2^{108} complexity (times A , the amount of effort to evaluate one set of polynomials). In practice it should be a lot more.

Giving the best of all worlds to the attacker, he guess again at 4 correct variables, and succeeds in reducing $\dim H_\infty$ to zero. Even with that much luck, $n = 16$. Since $A \approx mn^2/2$, we have $A \cdot (256)^4 \cdot (16)^{2.8 \times \sqrt{16}} \approx 2^{88}$. Thus TTS/4 need not worry about XL/FXL (and hence relinearization). The same kind of construction shows that if XL represents the most effective family of attacks, we can scale the TTS schemes so that they go up in complexity by 2^{16} when m goes up by 2 and n by 3.

4.3 Gröbner Bases

Gröbner Bases is a well-known way of solving polynomial equations. The classic algorithm for computing Gröbner bases, Buchberger’s algorithm, involves ordering all monomials (usually lexicographically) and take some appropriate algebraic combinations of the equations to eliminate the top monomial serially, until only one variable remains and then solve for that variable (*a la* Berlekamp). This method has been extended into more powerful variants by J. Faugère, called \mathbf{F}_4 and \mathbf{F}_5 ([14, 15]). $\mathbf{F}_5/2$, an adaptation of \mathbf{F}_5 , was used to break an HFE challenge in April 2002 ([16]).

The linearization methods of Sec. 4.2 can be considered simplified versions of Gröbner bases, and the latter are also affected by the underdeterminedness of the system. The attacker must guess at enough variables to make the program run faster. So there is the problem as is described in Sec. 4.2.

Similar to XL/FXL, Gröbner bases method is also affected by $\dim H_\infty$. But there is a difference: Since dependencies are located and collated at runtime, Gröbner bases method does not become non-functional if there is a non-zero $\dim H_\infty$. Luckily for us, it does add immensely to its time complexity.

Computing the Gröbner basis of a system of m polynomial equations of maximal degree d in n variables has a worst-case time complexity $m^3 d^{O(n^3)}$ ([4]); when the

solution set is of dimension ≤ 0 , this bound can be cut down to $d^{O(n^2)}$ ([5]). There is one significant theoretical exception ([28], theorem 3): *if (and essentially only if) $\dim H_\infty \leq 0$, we can find a Gröbner basis of degree $\leq (d-1)n+2$ and hence finish in at most time $O(d^n)$.* As a practical matter, with a suitable algorithm the exponent can be made smaller by a factor $L(q)$, where $L(2) \doteq 11.11$ and $L(3) \doteq 6.455$, but decreases quickly to 1 for larger values of q . So over *small* base fields — and this was the case for the HFE challenge 1 mentioned above — we can also finish computing a Gröbner basis a lot faster. *In general, computing a Gröbner basis takes time square-exponential in n when $\dim H_\infty > 0$. Its actual running time is very hard to estimate, but we have expert opinions to the effect that ([2]) for large fields and practical dimensions, Gröbner Bases are currently unlikely to be effective as an attack.* Therefore, for large base fields such as $\text{GF}(2^8)$, current Gröbner-based methods cannot be used to cryptanalyze well-designed multivariate schemes, e.g., the current TTS variants, effectively.

4.4 The Coppersmith Attack vs Shamir’s Birational Permutations Schemes

Shamir proposed a family of birational permutation signature schemes in [43], but soon afterwards Coppersmith *et al* found a successful attack ([7]). One specific case¹⁰ attacked by Coppersmith *et al*, “sequentially linearized birational permutations”, has $y_1 = x_1$, and $y_k = \ell_k(x_1, \dots, x_{k-1})x_k + q_k(x_1, \dots, x_{k-1})$ for $k = 2 \dots n$ with ℓ ’s linear and q ’s homogeneously quadratic. Take two invertible \mathbb{Z}_N square matrices ($N = pp'$ with p, p' prime), and transform \mathbf{x} to \mathbf{w} , (y_2, \dots, y_n) to \mathbf{z} . The private key is the ℓ ’s, the q ’s, and the two invertible matrices; the user lets \mathbf{z} be the message digest and finds (y_2, \dots, y_n) , assigns a random x_1 , solves sequentially for the rest of \mathbf{x} , then finds the signature \mathbf{w} . The public key is the quadratic forms giving \mathbf{z} in \mathbf{w} .

Sketch of attack: take the symmetric matrices M_j of y_j considered as quadratic forms of \mathbf{x} . These have a decreasing sequence of kernels in \mathbf{x} -space (ditto their images in \mathbf{w} -space) that we will try to find. Take λ_i such that the characteristic polynomial for $\bar{z}_i = z_i - \lambda_i z_n$ has a double root. Run recursively on the \bar{z}_i , which all have at least the kernel of M_{n-1} . We will have found a set of quadratic forms that are essentially equal to the original ones and enables us to forge signatures.

One can only admire the ingenuity of Coppersmith, Stern, and Vaudenay in looking for common kernel spaces. Theobald took pains to issue a similar warning ([45]) that “varying ranks of quadratic forms” may cause security concerns. Thankfully a TTS designer can arrange for a kernel without an onion-like sequence of decreasing kernels. Still, we consider TTS/4 to be partly inspired by their work.

4.5 Separation of Oil and Vinegar

In a simplified illustrative example of Patarin’s *Oil and Vinegar* signature scheme, the private key is an invertible matrix $A \in K^{2n \times 2n}$ over a finite field K and n matrices $F_j \in K^{2n \times 2n}$ with zeroes in all of the upper left quarter $n \times n$ entries. The signer releases as the public key the matrices $G_j \equiv A^T F_j A$. To sign, take the message digest

¹⁰ We invert Shamir’s notations to be parallel to ours.

to be $(m_1, \dots, m_n) \in K^n$. Assign random variables to the last n components (“vinegar”) of \mathbf{y} , and solve the equations $\mathbf{y}^T F_j \mathbf{y} = m_j$ for the first n components (“oil”) of $\mathbf{y} \in K^{2n}$. Since each F_j has its upper left quarter zero, the equations are linear in the oil variables, and $\mathbf{x} = A^{-1} \mathbf{y}$ is the signature, verifiable via $\mathbf{x}^T G_j \mathbf{x} = m_j$.

Here each F_j maps the subspace with $y_{n+1} = y_{n+2} = \dots = y_{2n} = 0$ (“oil” subspace) to the subspace (“vinegar”) $y_1 = \dots = y_n = 0$. The cryptanalysis by Kipnis and Shamir builds on the corollary that each $F_j^{-1} F_i$ maps the “oil” subspace to itself, and each $G_j^{-1} G_i$ shares an eigenspace (the image of the oil subspace under A) for suitable (i, j) . This eigenspace can be determined, enabling forged signatures. See ([26]) for details on how K-S attacked Patarin’s original, more complete scheme.

“Unbalanced” Oil and Vinegar ([25]) aims to patch this by using more vinegar variables, but must tread a fine line: Too few vinegar variables, and there is still a successful reduction; too many, and brute force attacks of [9] (see Sec. 4.1) works. No such concerns exist in TTS/4. In TTS/2 we can see a separation of the variables into even and odd portions, and in TTS/4 we can also apportion the variables into x_4, x_5, \dots, x_{18} and $x_0, \dots, x_3, x_{19}, \dots, x_{27}$. But there are fundamental differences to [25, 26]:

1. In TTS, the vinegar (freely assigned) variables are x_0, \dots, x_7 . Suppose an attacker finds the common eigenspaces for the TTS/2. Where in OV or UOV he would have decomposed the signature space into dependent “Oil” and independent “Vinegar” components, here he finds himself with two identical and mutually codependent Creamy Italian portions of 4 vinegar to 10 oil (the x_i with even and odd indices respectively). The same successful determination for TTS/4 will result in dependent Vinaigrette (x_0, \dots, x_3 plus x_{19}, \dots, x_{27}) and independent Ranch (x_4, x_6, \dots, x_{18}) portions, neither of which seems particularly useful to him.
2. According to the analysis in [25], with more dependent “oil” variables than independent “vinegar” variables the attack of [26] carries over without modification¹¹, but not with more vinegar than oil. The latter is the case for TTS/4 with more variables in Ranch than in Vinaigrette.

4.6 Other Special Attacks

This includes MinRank, attacks on 2R and SFLASH, and IP-based attacks:

The MinRank attack: The MinRank attack ([22]) is a type of “intelligent brute-force approach”. Unfortunately the authors’ presentation contained some errors ([33]), so while the idea may be valuable it would not function against TTS/4 without extensive revision. As stated in ([6]), TTS/2 was written so that every y_i has the same high rank of 8 and can be easily adjusted upwards just in case MinRank can be made to run. TTS/4 retains this property. *We have constructed variants of TTS with just a few bytes more in private keys, and 10% more of running time that has rank of 10 or 12. This is sufficient to settle any possible problems with MinRank.*

Attacks against SFLASH: The FLASH family of public-key digital signature schemes ([42]) are based on C^{*-} ([40]), Patarin’s adaptation of C^* to avoid his own ([37])

¹¹ We find that hard to put into practice since the G_j ’s taken as square matrices are not invertible.

attack. The original SFLASH scheme used a subfield of $GF(2)$ for all coefficients in its private and public keys. Gilbert and Minier observed ([21]) this to be a vulnerability and attacked the original SFLASH successfully, but their attack affects neither the current SFLASH^{v2} nor TTS/4.

Geiselmann *et al* observed ([19, 20]) that the middle portion of any FLASH variant is *homogeneous of degree two*, and showed that to be a vulnerability. Their attack gives constant parts of both affine mappings cheaply. However, it does not entirely break SFLASH and is inapplicable to a Tame Transformation type method like TTS/4 since the tame portion has linear terms.

Attacks against 2R schemes: The recent “2-round schemes” proposed by Patarin drew lots of fire ([3, 46, 47]), but the general structure of 2R schemes were so different from TTS that it is improbable for any of the suggested attacks to function against TTS/4 without substantial modification.

Patarin’s IP approach: If all the parameters in the tame portion of TTS were to be fixed, then the security of TTS/4 will depend on the difficulty of the IP problem [38, 39]. But TTS/4 should not have to fear from an IP-based attack. The central portion of Tame Transformation based methods contain lots of parameters (in the private key). This approach therefore will be hard to patch to be working against any TTS variant.

5 Conclusions

Multivariate Public-Key Cryptography is clearly a burgeoning research area rich in surprises and new discovery. For example, we saw that trivial changes to the structure in the previous TTS formulations can make an attacker’s life harder and the scheme more secure, and made adjustments accordingly. We do not doubt that there shall be further attacks against multivariate schemes, attacks against the TTS genre and even specific attacks tailored against TTS/4, but we are confident that the myriad variations possible in the structure of tame and tame-like maps means that TTS will adapt and survive in the wilderness as a family of secure and fast signature schemes. In summary:

The just-proposed TTS/4 seems efficient and impervious to known attacks. Tame Transformations, literally the centerpiece of TTS, seem to have many good properties required of a low-degree birational permutation without its drawbacks. A principal advantage is that the central quadratic portion of the scheme — a tame-like map — is easily mutable, variable with many parameters, nonhomogeneous, and very fast.

We feel justified in stating that the TTS family merits further attention.

Acknowledgements

The authors are indebted to an anonymous referee whose incisive comments provided the necessary insight to correct an oversight that would have been very costly indeed.

The first author would like to thank Professor T. Moh for his encouragement and assistance. The second author would like to thank his faithful Ping for her support and tolerance of his eccentricities during the work leading up to this manuscript.

References

1. M. Akkar, N. Courtois, R. Duteuil, and L. Goubin, *A Fast and Secure Implementation of SFLASH*, PKC 2003, LNCS v. 2567, pp. 267–278.
2. M. Bardet, J.-C. Faugère, and B. Salvy, *Complexity of Gröbner Basis Computations for Regular Overdetermined Systems*, preprint and private communication.
3. E. Biham, *Cryptanalysis of Patarin's 2-Round Public Key System with S Boxes (2R)*, EUROCRYPT 2000, LNCS v. 1807, pp. 408–416.
4. L. Caniglia, A. Galligo, and J. Heintz, *Some New Effectivity Bounds in Computational Geometry*, AAEECC-6, 1988, LNCS v. 357, pp. 131–151.
5. L. Caniglia, A. Galligo, and J. Heintz, *Equations for the Projective Closure and Effective Nullstellensatz*, Discrete Applied Mathematics, 33 (1991), pp. 11–23.
6. J.-M. Chen and B.-Y. Yang, *Tame Transformation Signatures with Topsy-Turvy Hashes*, proc. IWAP '02, Taipei.
7. D. Coppersmith, J. Stern, and S. Vaudenay, *Attacks on the Birational Permutation Signature Schemes*, CRYPTO'93, LNCS v. 773, pp. 435–443.
8. D. Coppersmith, J. Stern, and S. Vaudenay, *The Security of the Birational Permutation Signature Schemes*, Journal of Cryptology, 10(3), 1997, pp. 207–221.
9. N. Courtois, L. Goubin, W. Meier, and J. Tacier, *Solving Underdefined Systems of Multivariate Quadratic Equations*, PKC 2002, LNCS v. 2274, pp. 211–227.
10. N. Courtois, L. Goubin, and J. Patarin, *SFLASH^{v3}, a Fast Asymmetric Signature Scheme*, preprint available at <http://eprint.iacr.org/2003/211>.
11. 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.
12. N. Courtois and J. Patarin, *About the XL Algorithm over GF(2)*, CT-RSA 2003, LNCS v. 2612, pp. 141–157.
13. W. Diffie and M. Hellman, *New Directions in Cryptography*, IEEE Trans. Info. Theory, vol. IT-22, no. 6, pp. 644–654.
14. J.-C. Faugère, *A New Efficient Algorithm for Computing Gröbner Bases (F4)*, Journal of Pure and Applied Algebra, 139 (1999), pp. 61–88.
15. J.-C. Faugère, *A New Efficient Algorithm for Computing Gröbner Bases without Reduction to Zero (F5)*, Proceedings of ISSAC, ACM Press, 2002.
16. 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.
17. H. Fell and W. Diffie, *Analysis of a Public Key Approach Based on Polynomial Substitution*, CRYPTO'85, LNCS v. 218, pp. 340–349.
18. M. Garey and D. Johnson, *Computers and Intractability, A Guide to the Theory of NP-completeness*, Freeman and Co., 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 441 Key Bits of SFLASH^{v2}*, Third NESSIE Workshop, 2002.
21. H. Gilbert and M. Minier, *Cryptanalysis of SFLASH*, EUROCRYPT 2002, LNCS v. 2332, pp. 288–298.
22. L. Goubin and N. Courtois, *Cryptanalysis of the TTM Cryptosystem*, ASIACRYPT 2000, LNCS v. 1976, pp. 44–57.
23. 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.

24. H. Imai and T. Matsumoto, *Algebraic Methods for Constructing Asymmetric Cryptosystems*, AAIECC-3, LNCS v. 229, pp. 108–119.
25. A. Kipnis, J. Patarin, and L. Goubin, *Unbalanced Oil and Vinegar Signature Schemes*, CRYPTO'99, LNCS v. 1592, pp. 206–222.
26. A. Kipnis and A. Shamir, *Cryptanalysis of the Oil and Vinegar Signature Scheme*, CRYPTO'98, LNCS v. 1462, pp. 257–266.
27. A. Kipnis and A. Shamir, *Cryptanalysis of the HFE Public Key Cryptosystem by Relinearization*, CRYPTO'99, LNCS v. 1666, pp. 19–30.
28. D. Lazard, *Gröbner Bases, Gaussian Elimination and Resolution of Systems of Algebraic Equations*, EUROCAL '83, LNCS v. 162, pp. 146–156.
29. B. Lucier, *Cryptography, Finite Fields, and AltiVec*, http://www.simdtech.org/apps/group_public/download.php/22/Cryptography.pdf
30. T. Matsumoto and H. Imai, *Public Quadratic Polynomial-Tuples for Efficient Signature-Verification and Message-Encryption*, EUROCRYPT'88, LNCS v. 330, pp. 419–453.
31. T. Moh, *A Public Key System with Signature and Master Key Functions*, *Communications in Algebra*, 27 (1999), pp. 2207–2222.
32. T. Moh, *On The Method of XL and Its Inefficiency Against TTM*, available at <http://eprint.iacr.org/2001/047>
33. T. Moh and J.-M. Chen, *On the Goubin-Courtois Attack on TTM*, available at <http://eprint.iacr.org/2001/072>
34. *NESSIE Security Report, V2.0*, available at <http://www.cryptonessie.org>
35. *Performance of Optimized Implementations of the NESSIE Primitives, V2.0*, available at <http://www.cryptonessie.org>
36. H. Ong, C. Schnorr, and A. Shamir, *A Fast Signature Scheme Based on Quadratic Equations*, 16th ACM Symposium on Theory of Computations, 1984, pp. 208–216.
37. J. Patarin, *Cryptanalysis of the Matsumoto and Imai Public Key Scheme of Eurocrypt'88*, CRYPTO'95, LNCS v. 963, pp. 248–261.
38. 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.
39. J. Patarin, L. Goubin, and N. Courtois, *Improved Algorithms for Isomorphism of Polynomials*, EUROCRYPT'98, LNCS v. 1403, pp. 184–200.
40. J. Patarin, L. Goubin, and N. Courtois, *C_{-+}^* and HM: Variations Around Two Schemes of T. Matsumoto and H. Imai*, ASIACRYPT'98, LNCS v. 1514, pp. 35–49.
41. J. Patarin, N. Courtois, and L. Goubin, *QUARTZ, 128-Bit Long Digital Signatures*, CT-RSA 2001, LNCS v. 2020, pp. 282–297. Available at <http://www.cryptonessie.org>.
42. J. Patarin, N. Courtois, and L. Goubin, *FLASH, a Fast Multivariate Signature Algorithm*, CT-RSA 2001, LNCS v. 2020, pp. 298–307. Also <http://www.cryptonessie.org>.
43. A. Shamir, *Efficient Signature Schemes Based on Birational Permutations*, CRYPTO'93, LNCS v. 773, pp. 1–12.
44. A. Shamir and E. Tromer, *Factoring Large Numbers with the TWIRL Device*, CRYPTO 2003, LNCS v. 2729, pp. 1–26.
45. T. Theobald, *How to Break Shamir's Asymmetric Basis*, CRYPTO'95, LNCS v. 963, pp. 136–147.
46. D. Ye, Z. Dai, and K. Lam, *Decomposing Attacks on Asymmetric Cryptography Based on Mapping Compositions*, *Journal of Cryptology*, 14(2), 2001, pp. 137–150.
47. D. Ye, K. Lam, and Z. Dai, *Cryptanalysis of "2R" Schemes*, CRYPTO'99, LNCS v. 1666, pp. 315–325.

Note: Full version @ IACR e-Print archive <http://eprint.iacr.org/2003/160>.