Tame Transformation Signatures

Jiun-Ming Chen[1] Bo-Yin Yang[2]

[1] Purdue University, W. Lafayette, Indiana, USA. <jmchen@math.purdue.edu>

[2] Tamkang University, Tamsui, Taiwan. <by@moscito.org>

Submitted June 15, 2002,
revised for IWAP proceedings September 20, 2002,
exerpted for IACR e-print archive August 7, 2003

Note from the authors

This paper was presented at IWAP '02; after heavy editing it appeared in its proceedings. An excerpted version is hereby submitted to e-print archives for easy reference.

Abstract

We introduce the new $\mathrm{GF}(2^8)$-based digital signature scheme TTS (Tame Transformation Signatures). TTS is a consequence of the public-key cryptosystem TTM (Tame Transformation Method) and shares many of its superior properties, resulting in low signature delays, fast verification and high complexity. The commercial applications of TTS is protected under the patent of TTM. TTS can be used with any hash function (such as MD5 or SHA-1). We describe the principles and implementation of TTS and analyze their properties – both in absolute and comparatively to alternative schemes.

KEY WORDS
Finite Field, Tame Transformation, Digital Signature, TTM, TTS

1 Introduction

Secure authorization and authentication of information have been important and imminent problems in this age of the Internet. Identity fraud and sometimes outright theft runs rampant and many solutions have been proposed to rein in these beasts. Most involve some form of digital signatures and hash functions, hence faster and more secure hashes and digital signature schemes will be of great service in many ways.

In the course of this article, we will use the principles behind TTM (Tame Transformation Method, [14]) to derive a new digital signature scheme TTS (Tame Transformation Signatures). TTM and TTS both work on a finite field and have very similar designs. Due to their common ancestry, they share many properties including high complexity (security), ease of implementation and good execution speed. The following is a summary of the remaining sections:

Sec. 2 A brief recap of how Tame Transformations are used to construct the current TTM cryptosystem and the basic properties of TTM.

Sec. 3 Describing the basic ideas behind TTS (Tame Transformation Signatures).

Secs. 4 A practical TTS implementation.

Secs. 5–8 Qualitatitve and relative analysis of TTS.

2 Tame Transformations to TTM

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

$$\begin{aligned} y_1 &= x_1; \\ y_2 &= x_2 + q_2(x_1); \\ y_3 &= x_3 + q_3(x_1, x_2); \\ &\vdots \quad \vdots \qquad \vdots \\ y_i &= x_i + q_i(x_1, x_2, \ldots, x_{i-1}); \\ &\vdots \quad \vdots \qquad \vdots \\ y_n &= x_n + q_n(x_1, x_2, \ldots, x_{n-1}). \end{aligned}$$

Tame Transformations had been first researched in algebraic geometry, but its use was first proposed by T. Moh for public-key cryptography infrastructure ([14]). They possess the desirable property that

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

$$\begin{aligned} x_1 &= y_1; \\ x_2 &= y_2 - q_2(x_1); \\ x_3 &= y_3 - q_3(x_1,\, x_2); \\ &= y_3 - q_3(y_1,\, y_2 - q_2(y_1)); \\ x_4 &= y_4 - q_4(x_1,\, x_2,\, x_3) \\ &= y_4 - q_4(y_1,\, y_2 - q_2(y_1), \\ &\qquad\qquad y_3 - q_3(y_1,\, y_2 - q_2(y_1))); \\ &\vdots \quad \vdots \qquad \vdots \\ x_n &= y_n - q_n(x_1,\, x_2, \ldots,\, x_{n-1}); \\ &= y_n - q_n(y_1, y_2 - q_2(y_1) \ldots y_{n-1} - q_{n-1}(\cdots)). \end{aligned}$$

When TTM was first proposed it had an LTL (linear-tame-linear) form, with $K = \mathrm{GF}(2^8)$, $\mathbf{y} = L_2 \circ T \circ L_1(\mathbf{x})$ (where $\circ$ denotes composition, i.e. substitution). L_1 and L_2 are affine (linear) and T is tame and homogeneous quadratics for each q_i, and an expansion rate of 1 during encryption. This is susceptible to attacks by P. Montgomery and A. Sathaye (both unpublished) due to the fact that the first coordinate in the tame portion is fixed (as is practically the second coordinate, since q_2 is essentially constricted to $x \mapsto x^2$).

To ameliorate this flaw, current implementations of TTM ([14]) use an LTTL (linear-tame-tame-linear) form[1] $\mathbf{y} = \phi_4 \circ \phi_3 \circ \phi_2 \circ \phi_1(\mathbf{x})$. The components ϕ_1 and ϕ_4 are affine (linear), **but ϕ_2 is from K^n to K^r with $n < r$. It is really a Tame Transformation in K^r applied after the canonical embedding $K^n \to K^r$ (i.e. pad $x_{n+1}, \ldots, x_r$ with zero's).** Again all displacements q_i are homogeneous and quadratic. The major deviation concerns $\phi_3 : \mathbf{x} \mapsto \mathbf{y}$, which is a specially constructed Tame Transformation $K^r \to K^r$ with this form (where $1 < s < r$):

$$\begin{aligned} y_1 &= x_1 + p_1(x_2, x_3, \ldots, x_r); \\ y_2 &= x_2 + p_2(x_3, x_4, \ldots, x_r); \\ &\vdots \quad = \quad \vdots \end{aligned}$$

[1] In principle, the TTM encryption map can be $LT \cdots TL$ with one or more T; more T's can be added as security dictates; in practice this is seldom necessary.

$$\begin{aligned} y_s &= x_s + p_s(x_{s+1}, \ldots, x_r); \\ y_{s+1} &= x_{s+1}; \\ \vdots &= \vdots \\ y_r &= x_r; \end{aligned}$$

such that the degrees of the polynomials p_j's are suitably large[2] but the composition $\phi_3 \circ \phi_2 : K^n \to K^r$ are quadratic in each component of the image. Thus, $\phi = \phi_4 \circ \phi_3 \circ \phi_2 \circ \phi_1$ looks like a generic quadratic with r degree-2 equations of n variables each, and is given in the composite form as the public key. To get this desirable form, we need r to be substantially larger than n. Trade-offs must be made between encryption time ($\propto (nr)$, or proportional to the square of encryption block-size and the expansion rate) and safety. E.g. a current implementations of TTM uses $n = 20$ and $r = 48$, multiples of 4 being easier on the programmer (on current computer architectures). The private key is the collection of all information built into each of the ϕ_i modified in such a way to maximize decryption speed.

Improved as above, TTM is secure and speedy to the point where it can be used on its own and not in conjunction with a symmetric cipher. In more traditional PKI, a public key cryptosystem is only used to exchange the session key, and such is the case for the well-known SSH (Secure Shell) and PGP (Pretty Good Privacy) protocols. In its current form, TTM remains a "strong cryptosystem" under all known attacks (see Sec. 8 and [15]-[18] for more details).

In retrospect, one might marvel at some of the similarities in the design of AES(Rijndael) and TTM/TTS. Both work on $\mathrm{GF}(2^8)$ and include both linear and non-linear operations. The linear operations have a diffusive effect, quickly mixing all components and masking underlying algebraic relations; but the non-linear operations are necessary to disrupt the structure of linear mappings (otherwise the result of a mapping will be determined by the result on a basis of the space). AES uses the action of taking a multiplicative inverse as the non-linear operation and in TTM/TTS we use quadratic polynomial substitutions.

Further information on TTM can be found at `http://www.usdsi.com`. We stress that *computations pertaining to Tame Transformations (as well as other tame-like mappings) can be greatly accelerated using SIMD (Single Instruction Multiple Data) operations*, available (say) via Motorola's AltiVec technology. See [13] for one such implementation for TTM.

3 Basic Idea behind TTS

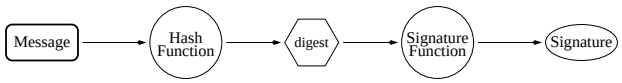
To sign a message digitally, one takes its digest (hash) value according to some agreed hash function (in the well-known PGP protocol, this is SHA-1), then run that hash value through a signature function (for PGP, this is RSA-1024 or -2048). The result is the digital signature. During authentication, the recipient substitutes the signature into a "verification trap-door function" and compares the result with the message digest (using the same hash function). If they are the same, the message is presumed "clean".

T. Moh's seminal paper [14] sketched how to derive digital signature schemes using the same principles behind TTM (i.e. Tame Automorphisms) known to yield strong cryptosystems with good properties, but our article is the first time that the details have been fleshed out and Tame Transformation proposed as the centerpiece of digital signature infrastructure without any mathematical cloud.

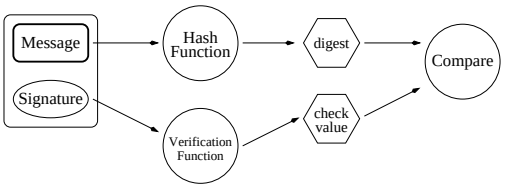
In a public-key cryptosystem one releases an injective function (the encryption map) whose trap-door inverse (the decryption map) is hard to find. In a digital signature scheme, the verification map released is an inverse to a hard-to-find injective trap-door (the signing map). Just fine for a symmetric cryptosystem like RSA, but the LTTL-form TTM encryption mapping is no longer a bijection (not surjective!) and cannot be used directly for digital signatures.

In TTS, we follow the theory in [14] and switch back to a LTL format for the mapping. The general idea is as follows: the message or hash value is padded out in a certain way before a tame transformation is applied. Again before and after the tame transformation there is an affine mapping portion.

[2] $\deg p_1 = \deg p_2 = 6$, the other p_j's are a handful of cubics.



(a) Signing a message (with private key)



(b) Verifying a signature (with public key)

Figure 1: Digital Signature and Authentication

Padding is a necessary evil because a simple LTL scheme will be relatively unsafe. In the current version we pad. Via padding we maintain the kernel as a homogeneous quadratic with the same nice properties as Tame Transformations.

4 Current TTS: TTS/2

TTS is really a genre of related schemes, but "TTS" below will principally refer to the illustrative variant, which may be denoted "TTS/2" for future reference.

This implementation utilizes 8 random elements from $K = \mathrm{GF}(2^8)$ and the signature will be 8 bytes longer than the hash value. 64 bits from a sufficiently random bitstream is needed; without dedicated hardware, the best source is Operating System level entropy collection based on I/O latency, as in the pseudodevice files `/dev/random` or `/dev/urandom` in various free Unices. TTS is slightly unusual in that the same message may not result in the same signature. In practice this is not a problem.

4.1 Signing a Message

Without loss of generality, take $\mathbf{y}$ to represent a 20-byte (160 bits) hash value. We obtain the signature $\mathbf{x} = \phi_1^{-1} \circ \phi_2^{-1} \circ \phi_3^{-1}(\mathbf{y})$, an 28-byte string, with the following component maps:

$$\begin{aligned}
\phi_1 &: K^{28} \to K^{28}, \quad \mathbf{x} \mapsto M_1\mathbf{x} + \mathbf{c_1};\\
\phi_2 &: K^{28} \to K^{20}, \quad \mathbf{x} \mapsto \text{Tame-Transform of } (\mathbf{x});\\
\phi_3 &: K^{28} \to K^{20}, \quad \mathbf{x} \mapsto M_3\mathbf{x} + \mathbf{c_3}.
\end{aligned}$$

The Tame Transform portion ϕ_2 is given by:

$$\begin{aligned}
y_0 &= x_0 = \text{Uniform Random Variable in } K;\\
&\vdots \quad \vdots \quad \vdots\\
y_7 &= x_7 = \text{Uniform Random Variable in } K;\\
y_8 &= x_8 + a_8x_0x_7 + b_8x_1x_6 + c_8x_2x_5 + d_8x_3x_4;\\
y_9 &= x_9 + a_9x_1x_8 + b_9x_2x_7 + c_9x_3x_6 + d_9x_4x_5;\\
&\vdots \quad \vdots \quad \vdots\\
y_k &= x_k + a_kx_{k-8}x_{k-1} + b_kx_{k-7}x_{k-2} + c_kx_{k-6}x_{k-3} + d_kx_{k-5}x_{k-4},\\
&\vdots \quad \vdots \ \vdots\\
y_{27} &= x_{27} + a_{27}x_{19}x_{26} + b_{27}x_{20}x_{25} + c_{27}x_{21}x_{24} + d_{27}x_{22}x_{23};
\end{aligned}$$

where $\mathbf{y} = (y_8, y_9, \ldots, y_{27}) \in K^{20}$ (note subscripts), and $\mathbf{x} = (x_0, x_1, \ldots, x_{27}) \in \phi_2^{-1}(\mathbf{y}) \subset K^{28}$ will be constructed during the signing process by solving the above equations. The signing portion or private key is given by the information in each part of the composite $\phi_1^{-1} \circ \phi_2^{-1} \circ \phi_3^{-1}$, and during the process each step is evaluated separately. **As shown below (see Sec. 7), it's possible to take certain shortcuts in the signing process.**

4.2 Verifying a Message

The verification mapping or public key is given by the composite of $\phi_3 \circ \phi_2 \circ \phi_1$. This also evaluates to a generic set of quadratic relations

$$y_i = \mathbf{x}^T A_i \, \mathbf{x} + \mathbf{b}_i^T \mathbf{x}, \quad i = 8, \ldots 31.$$

The matrices A_i and the vectors $\mathbf{b}_i$ therein are then recorded as the public key. We should mention *again* that normally there would be a constant term for each y_i except that we adjust $\mathbf{c}_3$ above to zero out them all. Key sizes can be found in Table 1.

	TTS		
equations	16	20	24
variables	24	28	32
message (bits)	128	160	192
signature (bits)	192	224	256
linear terms	24	28	32
quad. terms	300	406	528
terms per eq.	324	434	560
public key #par	5184	8680	13440
linear-1 #coeff	600	812	1056
TAME #coeff	64	80	96
linear-3 #coeff	272	420	600
private key #par	936	1312	1752

Table 1: Basic data for TTS/2 (the current TTS)

5 Flexibility

TTS can be easily adapted to any size of hash value. For example, in TTS, a hash of 192 bits or 24 bytes can be accommodated by a $K^{32} \to K^{24}$ Tame Transformation with a 256-bit (32 bytes) signature. If necessary, "trampolines" (data created to be programs) corresponding to other hash sizes can be created on the fly in response to the user's needs.

TTS can be adapted for higher level of security by adding more square-free quadratic terms. This will not affect the speed of verification, just increase the size of the private key and slightly slow the signing speed.

6 A Comparative Study

TTS like current implementations of TTM work with the affine spaces over the finite field $\mathrm{GF}(2^8)$. We venture our humble opinion that small working sizes associated with finite fields can be easier implemented effectively than alternatives with very large working sizes, such as the astronomically big groups used by RSA, ECC, ElGamal and their relatives. AES (see [1] and [7]), favorite symmetric cryptosystem *du jour* based on the Rijndael block cipher, is also based on computations over the very same finite field $\mathrm{GF}(2^8)$. $\mathrm{GF}(2^8)$ is a natural choice for being convenient computationwise and allowing compatible subfields of $\mathrm{GF}(2^4)$ and $\mathrm{GF}(2^2)$ (thus making for easier practical implementation with reasonably-sized keys).

6.1 Comparison to traditional PKI

Most extant PKI (Public Key Infrastructure) are based on RSA. RSA signatures are much larger (usu. 1024 or 2048 bits, much larger than in finite-field-based schemes). Signing a message under RSA is also significantly slower than under TTS, as is generating a key pair. All in all, there are distinctive advantages to use TTS rather than RSA. The superiority is more pronounced against (say) the even slower DSA/DSS (based on discrete logarithms). **The length of keys is a common bugbear of multivariate quadratic cryptography. In Sec. 7 we see how this problem can be mostly alleviated in practice.**

6.2 Brief Comparison against other multivariate Schemes

TTS belongs to the multivariate quadratic finite-field-based family of signature schemes as do two of the five NESSIE ([19]) digital signature scheme candidates, QUARTZ ([4]) and SFLASH ([5]). In any scheme of this type, to verify a message means substitutions into a set of quadratic polynomials much like encrypting a message in any such Public Key Cryptosystem, and hence they all use comparable time and resources. Indeed both QUARTZ and SFLASH performs similarly here.

However, signing a message is really like decrypting a message under an analogous cryptosystem, and here we see substantial performance differences (Table 2). TTS is structured so that signing is also substitutions, hence it is easy to code and quick to run. SFLASH (based on C^{*--}) is more troublesome in this aspect. The signing procedure for QUARTZ, equivalent to the decrypting process of HFE from which it is derived, is based on solving equations in a largish finite field, and as such is hard to code for and snail-paced in comparison. One can see from Table 2 that TTS shines against (at least) these rival schemes for digital signatures.

	QUARTZ	SFLASH	TTS/2
Signature (bits)	128	259	224
Public key (kB)	71	15.4	8.6
Private key (kB)	3	2.4	1.3
Generate keys (s)	≈ 4	≈ 1	$\approx 10^{-2}$
Signing speed (s)	≈ 10	2.7×10^{-3}	$\approx 10^{-4}$
Verifying speed (s)	$\approx 10^{-3}$	8×10^{-4}	$\approx 10^{-3}$

Table 2: Comparison of signature schemes (Pentium III/500, non-optimized gcc code).

As mentioned in Sec. 2, the underlying algebraic structure is why TTS signs and generates keys faster than SFLASH or QUARTZ. Certainly the use of $\mathrm{GF}(2^7)$ in SFLASH (instead of $\mathrm{GF}(2^8)$) precludes exploiting the existence of intermediate field extensions. Embarassingly, QUARTZ can fail to sign a message (very low probability), and the culprit is again the underlying structure. Variable parameters and non-homogeneity in the central non-affine map as TTM/TTS have (as opposed to SFLASH, with a fixed, quadratic, central portion) is a plus, because SFLASH can thus be subjected to the kind of attack tailored by Steinwandt *et al* ([23]-[25]).

7 Key Size Reduction

As can be seen from Sec. 6, TTS keys (while still shorter than SFLASH keys) are longer than RSA keys. This is not significant for modern general-purpose hardware, but can definitely be a problem for embedded applications. Since an embedded cryptosystem application, such as a smart card, principally restricts the size of **private** keys, we will show how to make keys (mostly private keys) significantly smaller by restricting the mappings involved. **The following applies just fine to TTM and most multivariate quadratic finite-field-based schemes.** While complexity of cracking TTS would also decrease, but analysis (Sec. 8) shows that it is still a "strong cryptosystem" and beyond the reach of crackers in the foreseeable future.

7.1 Public Keys

We fix a 4-bit subfield $K' = \mathrm{GF}(2^4)$ that is compatible (has a compatible multiplication table) with our $\mathrm{GF}(2^8)$, and use elements from K' for all the quadratic terms in the Tame Transformation and all matrix elements in the affine mappings. This effectively halves the public key size and can **nearly double the execution speed with suitable programming exercises**.

7.2 Private Keys

While public TTS keys are just coefficients for quadratic polynomials, a private TTS key is the sum of its three parts, and there are more tricks available:

1. Using a compatible $K' = \mathrm{GF}(2^4)$ cuts the corresponding private key size.
2. Using 1-bit entries for the linear-part square matrices. For TTS with 20-byte hashes and 28-byte signature, we would instead of $28^2 + 20^2$ bytes in the private keys have 1/8 times that many bytes for a total of 148.

   This trick lets the programmer avoid costly table lookups and use a variety of SIMD techniques to do several operations in parallel. In some cases it is not even required that the CPU has SIMD instructions!
3. In the linear maps, instead of generic invertible matrices use products of a number of component matrices in some given form such that both the number of parameters and the number of operations needed in the linear transformation is linear in the dimension as opposed to being proportion to the square of the dimension.

Preliminary tests show these techniques can result in a net increase of the signing speed by a factor of 4–7. It will take further case-by-case study to determine the ones that do not result in significant decrease of cryptanalyzing complexity.

8 Brief Cryptanalysis

Solving quadratic systems is an NP-complete problem ([8], [12]), so no attack can do more damage than brute force unless there is an inherent flaw in TTS. Given the kinship between TTM and TTS, most attacks against TTM may be surmised to apply to TTS, but there are signatures-specific attacks, too.

8.1 Recap of TTM attacks

Some notable attempts against TTM are as follows:

1. Jacques Patarin in [20] and later [21] gave and used algorithms for solving IP (Isomorphism of Polynomials) Problem: given sets of two quadratic relations (mappings $\mathbf{x} \in K^m \mapsto \mathbf{y} \in K^n$ with $y_j = f_j(x_1, \ldots, x_m)$, $j = 1 \cdots n$.) P and Q, find affine (linear) mappings L_1 and L_2 such that $Q = L_1 \circ P \circ L_2$. (Reduction to) IP is ineffective against TTM because (see [17]) Patarin's algorithm requires explicit knowledge of P and Q, but all current production-quality Tame Transformation based methods include many user-determined terms in the kernel mappings.
2. A. Kipnis and A. Shamir invented the **relinearization** improvement to the usual linearization approach solving sets of high-order equations, based on the simple fact that

   $$(ab)(cd) = (ac)(bd) = (ad)(bc),$$

   in any field. This attack is considered superseded ([6]) by the next one.
3. In [6] we see that experiments have been performed by N. Courtois., A. Klimov, A. Shamir, and J. Patarin with new methods "XL" and "FXL". One might be justified in

concluding that someone had (some version of) TTM in his sights. However, it is illustrated in [16] that such attempts are futile according to the theory established by Hilbert and Serre, because for XL to have any efficacy the solution set at ∞ must be either empty or 0-dimensional. *For TTS/2, each quadratic term has one x_i with even i and one with odd i, so the solution set at infinity has at least dimension thirteen – we can let $x_1 = x_3 = \cdots = x_{27} = 0$ and make all quadratics vanish.*

4. In [4] L. Goubin and N. Courtois asserted that the **MinRank attack** is effective against TTM. Unfortunately the paper was so full of inaccuracies as pointed out in [18], that it is hard to see how the attack can actually operate.

In each polynomial of the kernel Tame Transformation for any TTS scheme is four distinct square-free quadratic terms, none of which ever being replicated in another polynomial. Any linear combinations will not result in elimination of any terms. 4 square-free terms are considered plenty by current authorities; should there be new attacks that can exploit the smallness of number of terms in each polynomial, a few new terms can be added without substantial penalty.

We conclude that an analog of an attack on TTM (see [14] for details) on TTS will not do much better than a brute force search and all reasonable implementations of TTS has an effective complexity well above 2^{80}.

8.2 Signature-specific attacks

One kind of attacks is tailored for a given scheme. The aforementioned IP type of attack depends specifically on the central non-affine mapping having a simple, fixed form; the attack on SFLASH by Steinwandt *et al* ([23]-[25]) depends on the central quadratic portion being homogeneous. These kind of attacks are not applicable to TTS.

There may also be effective attack on generic schemes. Three attacks ([3]) are designed against "underdefined" system with number of variables $n \geq m$. We evaluate each proposed attack on the currently testing TTS with $q = 2^8$, $n = 28$ and $m = 20$ using formulas in [3]).

A The complexity is $O(q^{m-k}) \approx 2^{136}$ where the constant k is $\min(m/2, \left\lfloor \sqrt{n/2 - \sqrt{n/2}} \right\rfloor)$, equal to 3 in this instance.

B The complexity is $K \cdot q^{m-k}$. Here $k \equiv \lfloor \sqrt{2m+2} - 1.5 \rfloor = 5$ and $q^{m-k} = 2^{128}$. $K = \max(C_2, C_3)$ where C_3 is the time needed to solve a system of 4 equations (36 multiplications) and C_2 is given as $O(k\,(m-k)^2) \approx 2^{10}$, making for a complexity of 2^{130}.

C Not applicable to TTS since it requires $n \geq 2m$.

We conclude that TTS appears to be reasonably safe, even in its most basic form.

References

[1] `http://csrc.nist.gov/encryption/aes` the AES homepage

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

[3] N. Courtois, L. Goubin, W. Meier, and J.-D. Tacier, *Solving Underdefined Systems of Quadratic Equations*, pp. 211–227 in *Public Key Cryptography – PKC 2002*, LNCS V. 2274, Springer-Verlag, 2002.

[4] N. Courtois, L. Goubin, and J. Patarin, *Quartz, 128-bit long digital signatures.* in *Cryptographers' Track RSA Conference 2001*, LNCS V. 2020, Springer-Verlag, 2001.

[5] N. Courtois, L. Goubin, and J. Patarin, *FLASH, a fast multivariate signature algorithm.* in *Cryptographers' Track RSA Conference 2001*, LNCS V. *2020*, Springer-Verlag, 2001; updated version of [4] and [5] can be found at `http://www.cosic.esat.kuleuven.ac.be/nessie/tweaks.html`

[6] N. Courtois., A. Klimov, J. Patarin and A. Shamir. *Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations* pp. 392–407 in EUROCRYPT *2000*, LNCS V. *1807*, Springer-Verlag, 2000.

[7] Joan Daemen and Vincent Rijmen, *The Design of Rijndael, AES - The Advanced Encryption Standard.* Springer-Verlag, 2002.

[8] Michael Garey and David Johnson, *Computers and Intractability, a guide to the theory of NP-completeness.* Freeman, 1979.

[9] L. Goubin and N. Courtois, *Cryptanalysis of the TTM Cryptosystem.* pp. 44–57 in ASIACRYPT *2000*, LNCS V. *1976*, Springer-Verlag, 2000.

[10] A. Kipnis, J. Patarin, and L. Goubin, *Unbalanced Oil and Vinegar Signature Schemes*, pp. 206-222 in EUROCRYPT*'99*, LNCS V. *1592*, Springer-Verlag, 1999.

[11] A. Kipnis and A. Shamir, *Crypanalysis of the HFE public key cryptosystem*, pp. 19–30 in CRYPTO *1999*, LNCS V. *1666*, Springer-Verlag, 1999.

[12] K. Manders and L. Adleman, *NP-complete decision problems for quadratic polynomials*, pp. 23–29 in *Conference record of the eighth annual ACM Symposium on Theory of Computing: papers presented at the Symposium, Hershey, Pennsylvania, May 3–5, 1976*, ACM Press, 1976.

[13] B. Lucier, *Cryptography, Finite Fields, and AltiVec*, preprint at `http://www.altivec.org/articles/`.

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

[15] T. Moh, *Relinearization and TTM*, Preprint 1999.

[16] T. Moh, *On The Method of XL and Its Inefficiency Against TTM* in Cryptology ePrint Archive (2001/47).

[17] T. Moh, *A Cryptanalysis of TTM* in *Multivariate Cryptography*, International Press, 2003.

[18] T. Moh and J.-M. Chen, *On the Goubin-Courtois Attack on TTM* in Cryptology ePrint Archive (2001/72). Moh's papers also available at `http://www.usdsi.com`, the homepage of U.S. Digitial Security, Inc. and `http://www.chnds.com.tw` (Chinese Digital Security, Inc.).

[19] `http://www.cosic.esat.kuleuven.ac.be/nessie/`, the NESSIE (New European Schemes for Signatures, Integrity, and Encryption) selection project homepage.

[20] Jacques Patarin, *Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): two new families of Asymmetric Algorithms*, pp. 33–48 in EUROCRYPT *1996*, LNCS V. 1070, Springer-Verlag, 1996.

[21] J. Patarin, L. Goubin and N. Courtois, *Improved Algorithms for Isomorphisms of Polynomials*, pp. 184–200 in EUROCRYPT *1998*, LNCS V. 1403, Springer-Verlag, 1998.

[22] W. Stallings, *Crytography and Network Security: Principles and Practice*, 2nd ed. Prentice Hall, 1998.

[23] R. Steinwandt, W. Geiselmann, and Th. Beth, *Revealing the Affine Parts of* SFLASH^{v1} SFLASH^{v2}*, and* FLASH, to appear in *VII Reunion Espanola sobre Criptologia y Seguridad de la Informacion, VII RECSI Proceedings.*

[24] R. Steinwandt, W. Geiselmann, and Th. Beth, *Attacking the Affine Parts of* SFLASH, pp. 355-359, in *Cryptography and Coding, 8th IMA International Conference Proceedings*, B. Honary, ed., LNCS V. 2260, Springer-Verlag, 2001.

[25] R. Steinwandt, W. Geiselmann, and Th. Beth, *A Theoretical DPA-Based Cryptanalysis of the NESSIE Candidates* FLASH *and* SFLASH, pp. 280-293 in *Information Security, 4th International Conference, ISC 2001 Proceedings*, G. I. Davida, Y. Frankel, eds., LNCS V. 2200, Springer-Verlag, 2001. The last two papers are also presented at the 2nd NESSIE workshop.