

The reason of using an embedding ρ in TTM is partly due to fact that, to have a secure and efficient cryptosystem, TTM needs to have carefully designed Q_k -modules in φ_3 to make the composite $\varphi_3 \circ \varphi_2$ a collection of polynomials of low degrees. For the construction of such modules, please refer to [1, 3, 4]. In short, ρ indicates that, for $\varphi_3 \circ \varphi_2$ to work properly, the domain of TTM is a subspace \mathbb{F}^n within a larger space \mathbb{F}^{n+k} while the range of TTM is \mathbb{F}^{n+k} . ρ acts like padding k zeros for any given vector in \mathbb{F}^n , thus get a vector in \mathbb{F}^{n+k} . After that, $\varphi_1, \varphi_2, \varphi_3, \varphi_4$ are all bijections. Note that, TTM enjoys error-detection property, since, given a C in \mathbb{F}^{n+k} , we must have $\varphi_1^{-1}\varphi_2^{-1}\varphi_3^{-1}\varphi_4^{-1}(C)$ be of the special form as k zeros in the last k positions for C to be a ciphertext of TTM.

As Moh pointed in [6],

“The map $\hat{\pi}$ is not an onto map. However, we may restrict the map to a suitable subspace. Let $V = \{(d_1, \dots, d_j, 0, \dots, 0) : d_i \in \mathbf{K}\} \subset \mathbf{K}^{64}$ where j is a fixed integer less than or equal to 62. Let $\bar{V} = \phi_1^{-1}(V)$. We will require that ϕ_4 induces a linear transformation on $W = \{(e_1, \dots, e_j, 0, \dots, 0) : e_i \in \mathbf{K}\} \subset \mathbf{K}^{100}$. Let $\tau : (c_1, \dots, c_j, \dots, c_{100}) \mapsto (c_1, \dots, c_j)$ be a projection. Clearly $\tau\hat{\pi}$ is a one to one and onto map from \bar{V} to the j -dimensional affine space. Moreover, the map is *tame*, and its inverse at a point (y'_1, \dots, y'_j) can be found. The inverse at (y'_1, \dots, y'_j) forms a *signature*.”

To a mathematician, the above statement suffices for a signature. The scheme of TTS (cf. [2]) is without the mathematical cloud, and is clear for applications.

3. Brief Review of TTS

In [2], J. Chen and B. Yang proposed TTS-0 and TTS-r.

3.1 TTS-0

TTS-0 has been proposed with default values: $(n+k, n) = (24, 20)$. However, these values can be modified.

Let P, S denote plaintext and signature respectively. Without loss of generality, the signing and verifying processes of TTS-0 are as the following diagrams:

$$\begin{array}{ccccccc} \text{signing:} & \mathbb{F}^{24} & \xleftarrow{\varphi_1^{-1}} & \mathbb{F}^{24} & \xleftarrow{\varphi_2'} & \mathbb{F}^{20} & \xleftarrow{\varphi_3^{-1}} & \mathbb{F}^{20} \\ & S & \leftarrow & X & \leftarrow & Y & \leftarrow & P \\ \\ \text{verifying:} & \mathbb{F}^{24} & \xrightarrow{\varphi_1} & \mathbb{F}^{24} & \xrightarrow{\varphi_2} & \mathbb{F}^{20} & \xrightarrow{\varphi_3} & \mathbb{F}^{20} \\ & S & \rightarrow & X & \rightarrow & Y & \rightarrow & P \end{array}$$

where φ_1 and φ_3 are invertible linear transformation and φ_2', φ_2 are as follows. Given any $x = (x_0, x_1, x_2, \dots, x_{23}) \in \mathbb{F}^{24}$, we have $\varphi_2(x) = y = (y_0, y_1, y_2, \dots, y_{19}) \in \mathbb{F}^{20}$ by

$$\begin{array}{l} y_0 = x_0 + a_0x_4x_{20} + b_0x_6x_{21} \\ \quad + c_0x_8x_{22} + d_0x_{10}x_{23} \\ y_1 = x_1 + a_1x_5x_{20} + b_1x_7x_{21} \\ \quad + c_1x_9x_{22} + d_1x_{11}x_{23} \\ y_2 = x_2 + a_2x_0x_1 + b_2x_{20}x_{21} \\ \quad + c_2x_{12}x_{22} + d_2x_{14}x_{23} \\ y_3 = x_3 + a_3x_1x_2 + b_3x_{22}x_{23} \\ \quad + c_3x_{13}x_{20} + d_3x_{15}x_{21} \\ y_4 = x_4 + a_4x_0x_2 + b_4x_1x_3 \\ \quad + c_4x_{16}x_{20} + d_4x_{18}x_{22} \\ y_5 = x_5 + a_5x_0x_3 + b_5x_1x_4 \\ \quad + c_5x_{17}x_{21} + d_5x_{19}x_{23} \\ y_6 = x_6 + a_6x_0x_4 + b_6x_1x_5 \\ \quad + c_6x_2x_3 + d_6x_{20}x_{22} \\ y_7 = x_7 + a_7x_0x_5 + b_7x_1x_6 \\ \quad + c_7x_2x_4 + d_7x_{21}x_{23} \\ y_8 = x_8 + a_8x_0x_6 + b_8x_1x_7 \\ \quad + c_8x_2x_5 + d_8x_3x_4 \\ y_9 = x_9 + a_9x_0x_7 + b_9x_1x_8 \\ \quad + c_9x_2x_6 + d_9x_3x_5 \\ y_{10} = x_{10} + a_{10}x_2x_9 + b_{10}x_3x_8 \\ \quad + c_{10}x_4x_7 + d_{10}x_5x_6 \\ y_{11} = x_{11} + a_{11}x_3x_{10} + b_{11}x_4x_9 \\ \quad + c_{11}x_5x_8 + d_{11}x_6x_7 \\ \vdots \\ y_k = x_k + a_kx_{k-8}x_{k-1} + b_kx_{k-7}x_{k-2} \\ \quad + c_kx_{k-6}x_{k-3} + d_kx_{k-5}x_{k-4} \\ \vdots \\ y_{19} = x_{19} + a_{19}x_{11}x_{18} + b_{19}x_{12}x_{17} \\ \quad + c_{19}x_{13}x_{16} + d_{19}x_{14}x_{15} \end{array}$$

and, for $\varphi_2' : \mathbb{F}^{20} \rightarrow \mathbb{F}^{24}$, given any $y = (y_0, y_1, y_2, \dots, y_{19})$ in \mathbb{F}^{20} , we have $\varphi_2'(y) = x = (x_0, x_1, x_2, \dots, x_{19}, x_{20}, x_{21}, x_{22}, x_{23})$ by, first letting $x_{20} = x_{21} = x_{22} = x_{23} = 0$, then using the formulas in φ_2 to solve $x_0, x_1, x_2, \dots, x_{19}$ in sequential order. Note that, this actually means that, by padding 4 zeros at the last, φ_2 becomes a tame transformation. Therefore it is called a “tame-like transformation” in [2].

3.2 TTS-r

TTS-r has been proposed with default values: $(n+k, n) = (32, 24)$. However, these values can be modified.

Let P, S denote plaintext and signature respectively. Without loss of generality, the signing and verifying processes of TTS-0 are as the following diagrams:

$$\begin{array}{ccccccc} \text{signing:} & \mathbb{F}^{32} & \xleftarrow{\varphi_1^{-1}} & \mathbb{F}^{32} & \xleftarrow{\varphi_2'} & \mathbb{F}^{24} & \xleftarrow{\varphi_3^{-1}} & \mathbb{F}^{24} \\ & S & \leftarrow & X & \leftarrow & Y & \leftarrow & P \end{array}$$

$$\text{verifying: } \mathbb{F}^{32} \xrightarrow{\varphi_1} \mathbb{F}^{32} \xrightarrow{\varphi_2} \mathbb{F}^{24} \xrightarrow{\varphi_3} \mathbb{F}^{24}$$

$$S \rightsquigarrow X \rightsquigarrow Y \rightsquigarrow P$$

where φ_1 and φ_3 are invertible linear transformation and φ'_2, φ_2 are as follows. Given any $x = (x_0, x_1, x_2, \dots, x_{32}) \in \mathbb{F}^{32}$, we have $\varphi_2(x) = y = (y_8, y_9, y_{10}, \dots, y_{31}) \in \mathbb{F}^{24}$ by

$$\begin{aligned} y_8 &= x_8 + a_8 x_0 x_7 + b_8 x_1 x_6 \\ &\quad + c_8 x_2 x_5 + d_8 x_3 x_4 \\ y_9 &= x_9 + a_9 x_1 x_8 + b_9 x_2 x_7 \\ &\quad + c_9 x_3 x_6 + d_9 x_4 x_5 \\ &\quad \vdots \\ y_k &= x_k + a_k x_{k-8} x_{k-1} + b_k x_{k-7} x_{k-2} \\ &\quad + c_k x_{k-6} x_{k-3} + d_k x_{k-5} x_{k-4} \\ &\quad \vdots \\ y_{31} &= x_{31} + a_{31} x_{23} x_{30} + b_{31} x_{24} x_{29} \\ &\quad + c_{31} x_{25} x_{28} + d_{31} x_{26} x_{27} \end{aligned}$$

and, for $\varphi'_2 : \mathbb{F}^{24} \rightarrow \mathbb{F}^{32}$, given any $y = (y_8, y_9, y_{10}, \dots, y_{31})$ in \mathbb{F}^{24} , we have $\varphi'_2(y) = x = (x_0, x_1, x_2, \dots, x_7, x_8, x_9, x_{10}, \dots, x_{31})$ by, first picking 8 random variable $r_0, r_1, r_2, \dots, r_7 \in \mathbb{F}$ and letting $x_i = r_i, i = 0, 1, 2, \dots, 7$, then using the formulas in φ_2 to solve $x_8, x_9, x_{10}, \dots, x_{31}$ in sequential order. Note that, this actually means that, by padding 8 random variables as $y_i = x_i = r_i, i = 0, 1, 2, \dots, 7$ at the first, φ_2 becomes a tame transformation. Therefore it is also called a ‘‘tame-like transformation’’ in [2].

4. Implementation Discussion and Analysis

All following discussions are based on TTS from \mathbb{F}^n to \mathbb{F}^r , with $n > r$.

5. TTS-0

5.1 Signing time complexity

The signing key is a triple $(\varphi_1, \varphi_2, \varphi_3)$ where φ_1 is a linear transformation represented as a $n \times n$ matrix with a vector, φ_2 is a nonlinear tame-like transformations represented as r polynomials in n variables, and φ_3 is a linear transformation represented as a $r \times r$ matrix with a vector. Since both φ_1, φ_3 are both invertible linear transformations, with a technique proposed in [5], the time needed to compute φ_1^{-1} and φ_3^{-1} are $\mathcal{O}(n)$ and $\mathcal{O}(r)$, respectively. As to the time needed to compute φ_2^{-1} is $\mathcal{O}(r)$ due to the substitution property enjoyed by tame and tame-like transformations (cf. [5]). Thus the overall time to sign a block is $\mathcal{O}(n)$, and the time needed to sign a byte is $\mathcal{O}(n) \frac{1}{r} = \mathcal{O}(\frac{n}{r})$. Therefore, in this case, the signing speed is $\mathcal{O}(\frac{r}{n})$.

5.2 Verifying time complexity

The verifying key $v = \varphi_1 \circ \varphi_2 \circ \varphi_3$ consists of r polynomials in n variables. Since the time to evaluate a quadratic polynomial is $\mathcal{O}(n^2)$, and there are r such polynomials, the time to verify a block is $\mathcal{O}(rn^2)$. Thus the time to verify a byte is $\mathcal{O}(rn^2) \frac{1}{r} = \mathcal{O}(n^2)$.

5.3 Discussion

Although in key generation, the verifying key is represented as r polynomials in n variables. If the surplus $n - r$ polynomials are also contained in it, then the signer can easily perform error detection by checking whether the last $n - r$ components of the verified vector are zero or not.

6. TTS-r

6.1 Signing time complexity

Following the exact same lines as in the case for TTS-0, we have, for TTS-r, the overall time to sign a block is $\mathcal{O}(n)$, and the time needed to sign a byte is $\mathcal{O}(\frac{n}{r})$. The signing speed is $\mathcal{O}(\frac{r}{n})$.

6.2 Verifying time complexity

Following the exact same lines as in the case for TTS-0, we have, for TTS-r, the time to verify a block is $\mathcal{O}(rn^2)$. Thus the time to verify a byte is $\mathcal{O}(rn^2) \frac{1}{r} = \mathcal{O}(n^2)$.

6.3 Discussion

Note that while verifying, there are only r polynomials needed to be computed (indexed from $n - r$ to $n - 1$), for that the first $n - r$ random numbers are padded while signing and have no relations with the original plaintext. If the signer kept these random numbers and the verifying key is represented as n polynomials instead of r polynomials, the signer can easily perform error detection.

7. Performance Report

We used the following for our implementation. CPU: AMD Athlon 1.4 GHz; RAM: 512 MB; Compiler: Visual Studio 6.0; OS: Windows 2000 Professional.

We first did actual implementations for TTS-0 (24,20) and TTS-r (32,24), respectively. And then we did further implementations for TTS-0 (28,24), TTS-0 (32,28), TTS-r (40,32) and TTS-r (48,40), respectively.

In practice, TTS is applied on digest whose size is at most several hundred bytes. However, due to the overwhelming speed of it, we chose large files to evaluate its performance. The unit for our first measurement is Mega Bytes Per Second (MBPS). Our second measure is number of Digests Per Second (DPS). The digest size is 128 bit. In the following tables, “G” means key generation which is measured in milliseconds (ms), “S” means signing and “V” means verifying.

TTS-0 (24,20)	G	260 ms
	MBPS	DPS
S of a 128K file	3.1453	196581
V of a 128K file	0.3941	24631
S of a 256K file	3.4163	213519
V of a 256K file	0.3345	20906
S of a 512K file	3.2759	204744
V of a 512K file	0.3262	20388
S of a 1024K file	3.2882	205513
V of a 1024K file	0.3251	20319

TTS-0 (28,24)	G	490 ms
	MBPS	DPS
S of a 128K file	2.8280	176750
V of a 128K file	0.2769	17306
S of a 256K file	2.7071	169194
V of a 256K file	0.2895	18094
S of a 512K file	2.9065	181656
V of a 512K file	0.2912	18200
S of a 1024K file	3.0292	189325
V of a 1024K file	0.2903	18144

TTS-0 (32,28)	G	821 ms
	MBPS	DPS
S of a 128K file	3.2631	203944
V of a 128K file	0.217	13563
S of a 256K file	3.1218	195113
V of a 256K file	0.2175	13594
S of a 512K file	2.8193	176206
V of a 512K file	0.218	13625
S of a 1024K file	2.7841	174006
V of a 1024K file	0.2186	13663

TTS-r (32,24)	G	680 ms
	MBPS	DPS
S of a 128K file	2.5778	161113
V of a 128K file	0.2544	15900
S of a 256K file	2.4685	154281
V of a 256K file	0.2569	16056
S of a 512K file	2.5935	162094
V of a 512K file	0.2574	16088
S of a 1024K file	2.64598	165374
V of a 1024K file	0.2571	16069

TTS-r (40,32)	G	1962 ms
	MBPS	DPS
S of a 128K file	2.4467	152919
V of a 128K file	0.1473	9206
S of a 256K file	2.5743	160894
V of a 256K file	0.1417	8856
S of a 512K file	2.6404	165025
V of a 512K file	0.1442	9013
S of a 1024K file	2.6304	164400
V of a 1024K file	0.1444	9025
TTS-r (48,40)	G	4456 ms
	MBPS	DPS
S of a 128K file	2.3200	145000
V of a 128K file	0.0447	2794
S of a 256K file	2.2413	140081
V of a 256K file	0.0447	2794
S of a 512K file	2.3390	146188
V of a 512K file	0.0449	2806
S of a 1024K file	2.4705	154406
V of a 1024K file	0.0450	2813

8. Acknowledgement

We sincerely thank Prof. T. Moh, Prof. L. Wang and Mr. J. Chen for their many invaluable advises and suggestions on TTS. And a special thank goes to Mr. J. Chen for kindly granting us to first implement TTS before anyone else.

References

- [1] Jiun-Ming Chen, *Square-free Component Q_8 in TTM Cryptosystem*, preprint
- [2] Jiun-Ming Chen and Bo-Yin Yang, *Tame Transformation Signatures With Topsy-Turvy Hashes*, preprint
- [3] Chun-Yen Chou, D. J. Guan, and Jiun-Ming Chen, *A Systematic Construction of a Q_{2^k} -module in TTM*, Communications in Algebra, 30(2), pages 551-562 (2002).
- [4] Chun-Yen Chou and D. J. Guan, *Square-free Q_4, Q_6 and Q_8 modules in TTM*, preprint
- [5] Yuh-Hua Hu, Lih-Chung Wang, Jiun-ming Chen, Feipei Lai, and Chun-Yen Chou, *A Performance Report and Security Analysis of a fast TTM implementation*, preprint.
- [6] T. Moh, *A Public Key System With Signature And Master Key Functions*, Communications in Algebra, 27(5), 2207-2222 (1999).