Practical-Sized Instances of Multivariate PKCs: Rainbow, TTS, and ℓIC-Derivatives

Anna Inn-Tung Chen[1], Chia-Hsin Owen Chen[2], Ming-Shing Chen[2], Chen-Mou Cheng[1], and Bo-Yin Yang[2]

[1] Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan
{anna1110,doug}@crypto.tw
[2] Institute of Information Science, Academia Sinica, Taipei, Taiwan
{mschen,owenhsin,by}@crypto.tw, by@moscito.org

Abstract. We present instances of MPKCs (multivariate public key cryptosystems) with design, given the best attacks we know, and implement them on commodity PC hardware. We also show that they can hold their own compared to traditional alternatives. In fact, they can be up to an order of magnitude faster.

Keywords: Gröbner basis, multivariate public key cryptosystem.

1 Introduction

MPKCs (multivariate public key cryptosystems) [14, 31] are PKCs whose public keys are multivariate polynomials in many small variables. It has two properties that are often touted: Firstly, it is considered a significant possibility for Post-Quantum Cryptography, with potential to resist future attacks with quantum computers. Secondly, it is often considered to be faster than the competition.

Extant MPKCs almost always hide the private map $\mathcal{Q}$ via composition with two affine maps S, T. So, $\mathcal{P} = (p_1, \ldots, p_m) = T \circ \mathcal{Q} \circ S : \mathbb{K}^n \to \mathbb{K}^m$, or

$$\mathcal{P} : \mathbf{w} = (w_1, \ldots, w_n) \overset{S}{\mapsto} \mathbf{x} = \mathrm{M}_S\mathbf{w} + \mathbf{c}_S \overset{\mathcal{Q}}{\mapsto} \mathbf{y} \overset{T}{\mapsto} \mathbf{z} = \mathrm{M}_T\mathbf{y} + \mathbf{c}_T = (z_1, \ldots, z_m) \tag{1}$$

The public key consists of the polynomials in $\mathcal{P}$. $\mathcal{P}(0)$ is always taken to be zero.

In any given scheme, the *central map* $\mathcal{Q}$ belongs to a certain class of quadratic maps whose inverse can be computed relatively easily. The maps S, T are affine (sometimes linear) and full-rank. The x_j are called the central variables. The polynomials giving y_i in $\mathbf{x}$ are called the central polynomials; when necessary to distinguish between the variable and the value, we will write $y_i = q_i(\mathbf{x})$. The key of a MPKC is the design of the central map because, solving a generic multivariate quadratic system is hard, so the best solution for finding $\mathbf{w}$ given $\mathbf{z}$ invariably turns to other means, which depend on the structure of $\mathcal{Q}$.

J. Buchmann and J. Ding (Eds.): PQCrypto 2008, LNCS 5299, pp. 95–108, 2008.

1.1 Questions

Four or five years ago, it was shown that instances of TTS and C^{*-}, specifically TTS/4 and SFLASH, are faster signature schemes than traditional competition using RSA and ECC [1, 10, 33]. These two instances both been broken in the meantime [18, 20]. Now that the width of a typical ALU is 64 bits, commodity PC hardware has never been more friendly to RSA and ECC. While multivariates still represent a future-proofing effort, can we still say that MPKCs are efficient on commodity hardware?

1.2 Our Answers

Currently the fastest multivariate PKCs seems to be from the Rainbow and ℓIC families [16,17]. We run comparisons using Pentium III (P3) machines (on which NESSIE contestants are tested) and modern Core 2 and Opteron (hereafter C2 an K8) machines. On these test runs, we can say that compared to implementations using standard PKCs (DSA, RSA, ECDSA), present instances of MPKCs with design security levels of around 2^{80} can hold their own in terms of efficiency.

In this paper, we describe how we select our Rainbow and ℓIC-derived instances sketch our implementation. We also suggest the new approach of using bit-slicing when evaluating in GF(16) or other small fields during the construction of the private map.

In the comparison here, we use D. J. Bernstein's eBATs system to do benchmarking. We can conclude that

1. 3IC$^-$p is comparable to SFLASH, but not as fast as Rainbow.
2. Rainbow is fast and TTS faster, although the security is not as well studied.
3. 2IC$^+$i is a very fast way to build an encryption scheme.

Table 1. Current Multivariate PKCs Compared on a Pentium III 500

Scheme	result	SecrKey	PublKey	KeyGen	SecrMap	PublMap
RSA-1024	1024b	128 B	320 B	2.7 sec	84 ms	2.00 ms
ECC-GF(2^{163})	320b	48 B	24 B	1.6 ms	1.9 ms	5.10 ms
PMI+$(136, 6, 18, 8)$	144b	5.5 kB	165 kB	1.1 sec	1.23 ms	0.18 ms
rainbow $(2^8, 18, 12, 12)$	336b	24.8 kB	22.5 kB	0.3 sec	0.43 ms	0.40 ms
rainbow $(2^4, 24, 20, 20)$	256b	91.5 kB	83 kB	1.6 sec	0.93 ms	0.73 ms
TTS $(2^8, 18, 12, 12)$	336b	3.5kB	22.5kB	0.04 sec	0.11 ms	0.40 ms
TTS $(2^4, 24, 20, 20)$	256b	5.6kB	83kB	0.43 sec	0.22 ms	0.74 ms
2IC$^+$i (128,6,16)	144b	5 kB	165 kB	1 sec	0.03 ms	0.17 ms
2IC$^+$i (256,12,32)	288b	18.5 kB	1184 kB	14.9 sec	0.24 ms	2.60 ms
QUARTZ	128b	71.0 kB	3.9 kB	3.1 sec	11 sec	0.24 ms
3IC-p$(2^4, 32, 1)$	380b	9 kB	148 kB	0.6 sec	2.00 ms	1.90 ms
pFLASH	292b	5.5 kB	72 kB	0.3 sec	5.7 ms	1.70 ms

Table 2. Comparison on One core of an Intel Core 2 (C2)

Scheme	result	SecrKey	PublKey	KeyGen	SecrMap	PublMap
PMI+$(136, 6, 18, 8)$	144b	5.5 kB	165 kB	350.8 Mclk	335.4 kclk	51.4 kclk
PMI+$(136, 6, 18, 8)$64b	144b	5.5 kB	165 kB	350.4 Mclk	333.9 kclk	46.5 kclk
rainbow $(2^8, 18, 12, 12)$	336b	24.8 kB	22.5 kB	110.7 Mclk	143.9 kclk	121.4 kclk
rainbow $(2^4, 24, 20, 20)$	256b	91.5 kB	83 kB	454.0 Mclk	210.2 kclk	153.8 kclk
rainbow $(2^4, 24, 20, 20)$64b	256b	91.5 kB	83 kB	343.8 Mclk	136.8 kclk	79.3 kclk
TTS $(2^8, 18, 12, 12)$	336b	3.5kB	22.5kB	11.5 Mclk	35.9 kclk	121.4 kclk
TTS $(2^4, 24, 20, 20)$	256b	5.6kB	83kB	175.7 Mclk	64.8 kclk	78.9 kclk
2IC$^+$i (128,6,16)	144b	5 kB	165 kB	324.7 Mclk	8.3 kclk	52.0 kclk
2IC$^+$i (128,6,16)64b	144b	5 kB	165 kB	324.9 Mclk	6.7 kclk	46.9 kclk
2IC$^+$i (256,12,32)	288b	18.5 kB	1184 kB	4119.7 Mclk	26.7 kclk	385.6 kclk
2IC$^+$i (256,12,32)64b	288b	18.5 kB	1184 kB	4418.2 Mclk	23.0 kclk	266.9 kclk
3IC-p$(2^4, 32, 1)$	380b	9 kB	148 kB	173.6 Mclk	503 kclk	699 kclk
pFLASH	292b	5.5 kB	72 kB	86.6 Mclk	2410 kclk	879 kclk
DSA/ElGamal 1024b		148B	128B	1.08 Mclk	1046 kclk	1244 kclk
RSA 1024b		148B	128B	108 Mclk	2950 kclk	121 kclk
ECC 256b		96B	64B	2.7 Mclk	2850 kclk	3464 kclk

Table 3. Comparison on One Core of an Opteron/Athlon64 (K8)

Scheme	result	SecrKey	PublKey	KeyGen	SecrMap	PublMap
PMI+$(136, 6, 18, 8)$	144b	5.5 kB	165 kB	425.4 Mclk	388.8 kclk	63.9 kclk
PMI+$(136, 6, 18, 8)$64b	144b	5.5 kB	165 kB	424.7 Mclk	393.3 kclk	60.4 kclk
rainbow $(2^8, 18, 12, 12)$	336b	24.8 kB	22.5 kB	234.6 Mclk	297.0 kclk	224.4 kclk
rainbow $(2^4, 24, 20, 20)$	256b	91.5 kB	83 kB	544.6 Mclk	224.4 kclk	164.0 kclk
rainbow $(2^4, 24, 20, 20)$64b	256b	91.5 kB	83 kB	396.2 Mclk	138.7 kclk	83.9 kclk
TTS $(2^8, 18, 12, 12)$	336b	3.5kB	22.5kB	20.4 Mclk	69.1 kclk	224.4 kclk
TTS $(2^4, 24, 20, 20)$	256b	5.6kB	83kB	225.2 Mclk	103.8 kclk	84.8 kclk
2IC$^+$i (128,6,16)	144b	5 kB	165 kB	382.6 Mclk	8.7 kclk	64.2 kclk
2IC$^+$i (128,6,16)64b	144b	5 kB	165 kB	382.1 Mclk	7.5 kclk	60.1 kclk
2IC$^+$i (256,12,32)	288b	18.5 kB	1184 kB	5155.5 Mclk	31.1 kclk	537.0 kclk
2IC$^+$i (256,12,32)64b	288b	18.5 kB	1184 kB	5156.1 Mclk	26.6 kclk	573.9 kclk
3IC-p$(2^4, 32, 1)$	380b	9 kB	148 kB	200.7 Mclk	645 kclk	756 kclk
pFLASH	292b	5.5 kB	72 kB	126.9 Mclk	5036 kclk	872 kclk
DSA/ElGamal 148B		148B	128B	0.864 Mclk	862 kclk	1018 kclk
RSA 1024b		148B	128B	150 Mclk	2647 kclk	117 kclk
ECC 256b		96B	64B	2.8 Mclk	3205 kclk	3837 kclk

1.3 Previous Work

In [4], Berbain, Billet and Gilbert describe several ways to compute the *public maps* of MPKCs and compare their efficiency. However, they do not describe the evaluation of the *private maps*.

[18] summarizes the state of the art against generalized Rainbow/TTS schemes. The school of Stern *et al* developed differential attacks that breaks minus variants

[24, 20] and internal perturbation [23]. Ways to circumvent these attacks are proposed in [13, 19].

The above attacks the cryptosystem as an EIP or "structural" problem. To solve the system of equations, we have this

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

Best known methods for generic $\mathcal{MQ}$ are $\mathbf{F_4}$-$\mathbf{F_5}$ or XL whose complexities [11, 21,22,32] are very hard to evaluate; asymptotic formulas can be found in [2,3,32].

1.4 Summary and Future Work

Our programs are not very polished; it merely serves to show that MPKCs can still be fairly fast compared to the state-of-the-art traditional PKCs even on the most modern and advanced microprocessors. There are some recent advances in algorithms also, such as computations based on the inverted twisted Edwards curves [5, 6, 7], which shows that when tuned for the platform, the traditional cryptosystems can get quite a bit faster. It still remains to us to optimize more for specific architectures including embedded platforms. Further, it is an open question on whether the TTS schemes, with some randomness in the central maps, can be made with comparable security as equally sized Rainbow schemes. So far we do not have a conclusive answer.

2 Rainbow and TTS Families

We characterize a Rainbow [16] type PKC with u stages:

- The segment structure is given by a sequence $0 < v_1 < v_2 < \cdots < v_{u+1} = n$.
- For $l = 1, \ldots, u+1$, set $S_l := \{1, 2, \ldots, v_l\}$ so that $|S_l| = v_l$ and $S_0 \subset S_1 \subset \cdots \subset S_{u+1} = S$. Denote by $o_l := v_{l+1} - v_l$ and $O_l := S_{l+1} \setminus S_l$ for $l = 1 \cdots u$.
- The central map $\mathcal{Q}$ has component polynomials $y_{v_1+1} = q_{v_1+1}(\mathbf{x})$, $y_{v_1+2} = q_{v_1+2}(\mathbf{x}), \ldots, y_n = q_n(\mathbf{x})$ — *notice unusual indexing* — of the following form

$$y_k = q_k(\mathbf{x}) = \sum_{i=1}^{v_l} \sum_{j=i}^{n} \alpha_{ij}^{(k)} x_i x_j + \sum_{i<v_{l+1}} \beta_i^{(k)} x_i, \text{ if } k \in O_l := \{v_l + 1 \cdots v_{l+1}\}.$$

  In every q_k, where $k \in O_l$, there is no cross-term $x_i x_j$ where both i and j are in O_l at all. So given all the y_i with $v_l < i \le v_{l+1}$, and all the x_j with $j \le v_l$, we can compute $x_{v_l+1}, \ldots, x_{v_{l+1}}$.

  S_i is the i-th vinegar set and O_i the corresponding i-th oil set.
- To expedite computations, some coefficients ($\alpha_{ij}^{(k)}$) may be fixed (e.g., set to zero), chosen at random (and included in the private key), or be interrelated in a predetermined manner.

– To invert $\mathcal{Q}$, determine (usu. at random) $x_1, \ldots x_{v_1}$, i.e., all x_k, $k \in S_1$. From the components of $\mathbf{y}$ that corresponds to the polynomials $p'_{v_1+1}, \ldots p'_{v_2}$, we obtain a set of o_1 equations in the variables x_k, $(k \in O_1)$. We may repeat the process to find all remaining variables.

For historical reasons, a Rainbow type signature scheme is said to be a TTS [33] scheme if the coefficients of $\mathcal{Q}$ are sparse.

2.1 Known Attacks and Security Criteria

1. Rank (or Low Rank, MinRank) attack to find a central equation with least rank [33].
$$C_{\text{low rank}} \approx \left[q^{v_1+1} m(n^2/2 - m^2/6)/\right] \mathfrak{m}.$$
Here as below, the unit $\mathfrak{m}$ is a multiplications in $\mathbb{K}$, and v_1 the number of vinegars in layer 1. This is the "MinRank" attack of [25]. as improved by [8, 33].
2. Dual Rank (or High Rank) attack [9, 25], which finds a variable appearing the fewest number of times in a central equation cross-term [18, 33]:
$$C_{\text{high rank}} \approx \left[q^{o_n - v'} n^3/6\right] \mathfrak{m},$$
where v' counts the vinegar variables that never appears until the final segment.
3. Trying for a direct solution. The complexity is roughly as $\mathcal{MQ}(q; m, m)$.
4. Using the Reconciliation Attack [18], the complexity is as $\mathcal{MQ}(q; v_u, m)$.
5. Using the Rainbow Band Separation from [18], the complexity is determined by that of $\mathcal{MQ}(q; n, m+n)$.
6. Against TTS, there is Oil-and-Vinegar Separation [30, 26, 27], which finds an Oil subspace that is sufficiently large (estimates as corrected in [33]).
$$C_{\text{UOV}} \approx \left[q^{n-2o-1} o^4 + (\text{some residual term bounded by } o^3 q^{m-o}/3)\right] \mathfrak{m}.$$
o is the max. *oil set* size, i.e., there is a set of o central variables which are never multiplied together in the central equations, and no more.

2.2 Choosing Rainbow Instances

First suppose that we wish to use SHA-1, which has 160 bits. It is established by [18] that using GF(2^8) there is no way to get to 2^{80} security using roughly that length hash, unpadded.

Specifically, to get the complexity of $\mathcal{MQ}(2^8, m, m)$, to above 2^{80} (the direct attack) we need about $m = 24$. Then we need $\mathcal{MQ}(2^8, n, n+m)$ to get above 2^{80} (the Rainbow Band Separation), which requires at least $n = 42$. This requires an 192-bit hash digest plus padding and a signature length of 336 bits with the vinegar sequence $(18, 12, 12)$.

If we look at smaller fields, that's a different story. If we use GF(2^4), we need 20 oil variables each in the last segment and at least 20 vinegar variables in the

first segment to get by the minrank and high rank attacks. To be comparable to the sizes of $3IC$-p, we choose the vinegar (structural) sequence $(24, 20, 20)$. The digest is 160 bits and the signature 192. We use random parameters under this framework and don't do TTS. The implementations are described below. In each of the two instances, the central map is inverted by setting up and solving two identically-sized linear systems.

2.3 Choosing TTS Instances

TTS of the same size over GF(2^8) or GF(2^4) are $2\times$ or more the speed of than a Rainbow instance. They also tend to have instances also have much lower memory requirement. But we don't really know about their security.

The following are TTS instances built with exactly the same rainbow structural parameters and called henceforth TTS/7. They have exactly the same size input and output as the corresponding Rainbow instances:

TTS $(2^8, 18, 12, 12)$ $\mathbb{K}$ =GF(2^8), $n = 42$, $m = 24$. $\mathcal{Q}$ is structured as follows:

$$y_i = x_i + a_{i1}x_{\sigma_i} + a_{i2}x_{\sigma'_i} + \sum_{j=0}^{11} p_{ij}x_{j+18}x_{\pi_i(j)} + p_{i,12}x_{\pi_i(12)}x_{\pi_i(15)} + p_{i,13}x_{\pi_i(13)}x_{\pi_i(16)} + p_{i,14}x_{\pi_i(14)}x_{\pi_i(17)},\ i = 18 \cdots 29$$

[indices $0 \cdots 17$ appears exactly once in each random permutation π_i, and exactly once among the σ, σ' (where six σ'_i slots are empty)];

$$y_i = x_i + a_{i1}x_{\sigma_i} + a_{i2}x_{\sigma'_i} + a_{i3}x_{\sigma''_i} + \sum_{j=0}^{11} x_{j+29}(p_{ij}x_{\pi_i(j)} + p_{i,j+12}x_{\pi_i(j+12)}) + p_{i,24}x_{\pi_i(24)}x_{\pi_i(27)} + p_{i,25}x_{\pi_i(25)}x_{\pi_i(28)} + p_{i,26}x_{\pi_i(26)}x_{\pi_i(29)},\ i = 30 \cdots 41$$

[indices $0 \cdots 29$ appears exactly once in each random permutation π_i, and exactly once among the σ, σ', σ'' (where six σ''_i slots are empty)].

TTS $(2^4, 24, 20, 20)$ $\mathbb{K}$ =GF(2^4), $n = 64$, $m = 40$.

$$y_i = x_i + a_{i1}x_{\sigma_i} + a_{i2}x_{\sigma'_i} + \sum_{j=0}^{19} p_{ij}x_{j+23}x_{\pi_i(j)} + p_{i,20}x_{\pi_i(20)}x_{\pi_i(22)} + p_{i,21}x_{\pi_i(21)}x_{\pi_i(23)},\ i = 24 \cdots 43$$

[indices $0 \cdots 23$ appears exactly once in each random permutation π_i, and exactly once among the σ, σ' (there are only four σ'_i)];

$$y_i = x_i + a_{i1}x_{\sigma_i} + a_{i2}x_{\sigma'_i} + a_{i3}x_{\sigma''_i} + \sum_{j=0}^{19} x_{j+44}(p_{ij}x_{\pi_i(j)} + p_{i,j+20}x_{\pi_i(j+20)}) + p_{i,40}x_{\pi_i(40)}x_{\pi_i(42)} + p_{i,41}x_{\pi_i(41)}x_{\pi_i(43)},\ i = 44 \cdots 63$$

[indices $0 \cdots 43$ appears exactly once in each random permutation π_i, and exactly once among the σ, σ', σ'' (there are only four σ''_i)].

3 The ℓ-Invertible Cycle (ℓIC) and Derivatives

The ℓ-invertible cycle [17] can be best considered an improved version or extension of Matsumoto-Imai, otherwise known as C^* [28]. Let's review first the latter.

Triangular (and Oil-and-Vinegar, and variants thereof) systems are sometimes called "single-field" or "small-field" approaches to MPKC design, in contrast to the approach taken by Matsumoto and Imai in 1988. In such "big-field" variants, the central map is really a map in a larger field $\mathbb{L}$, a degree n extension of a finite field $\mathbb{K}$. To be quite precise, we have a map $\overline{\mathcal{Q}} : \mathbb{L} \to \mathbb{L}$ that we can invert, and pick a $\mathbb{K}$-linear bijection $\phi : \mathbb{L} \to \mathbb{K}^n$. Then we have the following multivariate polynomial map, which is presumably quadratic (for efficiency):

$$\mathcal{Q} = \phi \circ \overline{\mathcal{Q}} \circ \phi^{-1}. \tag{2}$$

then, one "hide" this map $\mathcal{Q}$ by composing from both sides by two invertible affine linear maps S and T in $\mathbb{K}^n$, as in Eq. 1.

Matsumoto and Imai suggest that we pick a $\mathbb{K}$ of characteristic 2 and this map $\overline{\mathcal{Q}}$

$$\overline{\mathcal{Q}} : \mathbf{x} \longmapsto \mathbf{y} = \mathbf{x}^{1+q^\alpha}, \tag{3}$$

where $\mathbf{x}$ is an element in $\mathbb{L}$, and such that $\gcd(1 + q^\alpha, q^n - 1) = 1$. The last condition ensures that the map $\overline{\mathcal{Q}}$ has an inverse, which is given by

$$\overline{\mathcal{Q}}^{-1}(\mathbf{x}) = \mathbf{x}^h, \tag{4}$$

where $h(1 + q^\alpha) = 1 \bmod (q^n - 1)$. This ensures that we can decrypt any secret message easily by this inverse. *Hereafter we will simply identify a vector space $\mathbb{K}^k$ with larger field $\mathbb{L}$, and $\mathcal{Q}$ with $\overline{\mathcal{Q}}$, totally omitting the isomorphism ϕ from formulas.*

ℓIC also uses an intermediate field $\mathbb{L} = \mathbb{K}^k$ and extends C^* by using the following central map from $(\mathbb{L}^*)^\ell$ to itself:

$$\begin{aligned} \mathcal{Q} : (X_1, \ldots, X_\ell) &\mapsto (Y_1, \ldots, Y_\ell) \\ &:= (X_1X_2,\ X_2X_3, \ldots,\ X_{\ell-1}X_\ell,\ \underline{X_\ell X_1^{q^\alpha}}). \end{aligned} \tag{5}$$

For "standard 3IC", $\ell = 3$, $\alpha = 0$. Invertion in $(\mathbb{L}^*)^3$ is then easy.

$$\mathcal{Q}^{-1} : (Y_1,\ Y_2,\ Y_3) \in (\mathbb{L}^*)^3 \mapsto (\sqrt{Y_1Y_3/Y_2},\ \sqrt{Y_1Y_2/Y_3},\ \sqrt{Y_2Y_3/Y_1},). \tag{6}$$

Most of the analysis of the properties of the 3IC map can be found in [17] — the 3IC and C^* maps has a lot in common. Typically, we take out 1/3 of the variables with a minus variation (3IC$^-$).

For encryption schemes, "2IC" or $\ell = 2$, $q = 2$, $\alpha = 1$ is suggested.

$$\mathcal{Q}_{\text{2IC}} : (X_1, X_2) \mapsto (X_1X_2, X_1X_2^2), \quad \mathcal{Q}_{\text{2IC}}^{-1} : (Y_1, Y_2) \mapsto (Y_1/Y_2^2, Y_2/Y_1). \tag{7}$$

We construct 2ICi like we do PMI [12]: Take $\mathbf{v} = (v_1, \ldots, v_r)$ to be an r-tuple of random affine forms in the variables $\mathbf{x}$. Let $\mathbf{f} = (f_1, \ldots, f_n)$ be a random r-tuple of quadratic functions in $\mathbf{v}$. Let our new $\mathcal{Q}$ be defined by

$$\mathbf{x} \mapsto \mathbf{y} = \mathcal{Q}_{2\mathrm{IC}}(\mathbf{x}) + \mathbf{f}(\mathbf{v}(\mathbf{x}))$$

where the power operation assumes the vector space to represent a field. *The number of Patarin relations decrease quickly down to 0* as r increases. For every $\mathbf{y}$, we may find $\mathcal{Q}^{-1}(\mathbf{y})$ by guessing at $\mathbf{v}(\mathbf{x}) = \mathbf{b}$, finding a candidate $\mathbf{x} = \mathcal{Q}_{2\mathrm{IC}}^{-1}(\mathbf{y} + \mathbf{b})$ and checking the initial assumption that $\mathbf{v}(\mathbf{x}) = \mathbf{b}$. Since we repeat the high going-to-the-h-th-power procedure q^r times, we are almost forced to let $q = 2$ and make r as low as possible.

3.1 Known Attacks to Internal Perturbation and Defenses

ℓIC has so much in common with C^* that we need the same variations. In other words, we need to do 3IC$^-$p (with minus and projection) and 2IC$^+$i (with internal perturbation and plus), paralleling C^{*-}p and C^{*+}i (a.k.a. PMI+).

The cryptanalysis of PMI and hence 2ICi depends on the idea that for a randomly chosen $\mathbf{b}$, the probability is q^{-r} that it lies in the kernel $\mathcal{K}$ of the linear part of $\mathbf{v}$. When that happens, $\mathbf{v}(\mathbf{x} + \mathbf{b}) = \mathbf{v}(\mathbf{x})$ for any $\mathbf{x}$. Since q^{-r} is not too small, if we can distinguish between a vector $\mathbf{b} \in T^{-1}\mathcal{K}$ (back-mapped into $\mathbf{x}$-space) and $\mathbf{b} \notin T^{-1}\mathcal{K}$, we can bypass the protection of the perturbation, find our bilinear relations and accomplish the cryptanalysis.

In [23], Fouque, Granboulan and Stern built a *one-sided distinguisher* using a test on the kernel of the *polar form* or *symmetric difference* $D\mathcal{P}(\mathbf{w}, \mathbf{b}) = \mathcal{P}(\mathbf{b} + \mathbf{w}) - \mathcal{P}(\mathbf{b}) - \mathcal{P}(\mathbf{w})$. We say that $t(\mathbf{b}) = 1$ if $\dim \ker_{\mathbf{w}} D\mathcal{P}(\mathbf{b}, \mathbf{w}) = 2^{\gcd(n,\alpha)} - 1$, and $t(\mathbf{b}) = 0$ otherwise. If $\mathbf{b} \in \mathcal{K}$, then $t(\mathbf{b}) = 1$ with probability one, otherwise it is less than one. In fact if $\gcd(n, \alpha) > 1$, it is is an almost perfect distinguisher. We omit the gory details and refer the reader to [23] for the complete differential cryptanalysis.

Typically, to defeat this attack, we need to add a random equations to the central map. For 2ICi as for PMI, both a and r are roughly proportional to n creating 2IC$^+$i like we did PMI+ [13]. PMI+(n, r, a, α) refers to a map from GF(2^n) with r perturbations, a extra variables, and a central map of $\mathbf{x} \to \mathbf{x}^{2^\alpha+1}$. Similarly, 2IC$^+i(n, r, a)$ refers to 2IC with r perturbations dimensions and a added equations.

3.2 Known Attacks to Minus Variants and Defenses

The attack found by Stern etc. can be explained by considering the case of C^* cryptosystem. We recollect that the symmetric differential of any function G, defined formally:

$$DG(\mathbf{a}, \mathbf{x}) := G(\mathbf{x} + \mathbf{a}) - G(\mathbf{x}) - G(\mathbf{a}) + G(0).$$

is bilinear and symmetric in its variables $\mathbf{a}$ and $\mathbf{x}$. Let ζ be an element in the big field $\mathbb{L}$. Then we have

$$D\mathcal{Q}(\zeta \cdot a, x) + D\mathcal{Q}(a, \zeta \cdot x) = (\zeta^{q^\alpha} + \zeta) D\mathcal{Q}(a, x).$$

Clearly the public key of C^{*-} inherits some of that symmetry. Now not every skew-symmetric action by a matrix M_ζ that corresponds to an $\mathbb{L}$-multiplication that result in $\mathrm{M}_\zeta^T H_i + H_i \mathrm{M}_\zeta$ being in the span of the public-key differential matrices, because $S := \mathrm{span}\{H_i : i = 1 \cdots n - r\}$ as compared to $\mathrm{span}\{H_i : i = 1 \cdots n\}$ is missing r of the basis matrices. However, as the authors of [20] argued heuristically and backed up with empirical evidence, if we just pick the first three $\mathrm{M}_\zeta^T H_i + H_i \mathrm{M}_\zeta$ matrices, or any three random linear combinations of the form $\sum_{i=1}^{n-r} b_i (\mathrm{M}_\zeta^T H_i + H_i \mathrm{M}_\zeta)$ and demand that they fall in S, then

1. There is a good chance to find a nontrivial M_ζ satisfying that requirement;
2. This matrix really correspond to a multiplication by ζ in $\mathbb{L}$;
3. Applying the skew-symmetric action of this M_ζ to the public-key matrices leads to other matrices in $\mathrm{span}\{H_i : i = 1 \cdots n\}$ that is not in S.

Why *three*? There are $n(n-1)/2$ degrees of freedom in the H_i, so to form a span of $n-r$ matrices takes $n(n-3)/2+r$ linear relations among its components ($n-r$ and not n because if we are attacking C^{*-}, we are missing r components of the public key). There are n^2 degrees of freedom in an $n \times n$ matrix U. So, if we take a random public key, it is always possible to find a U such that

$$U^T H_1 + H_1 U,\, U^T H_2 + H_2 U \in S = \mathrm{span}\{H_i : i = 1 \cdots n - r\},$$

provided that $3n > 2r$. However, if we ask that

$$U^T H_1 + H_1 U,\, U^T H_2 + H_2 U,\, U^T H_3 + H_3 U \in S,$$

there are many more conditions than degrees of freedom, hence it is unlikely to find a nontrivial solution for truly random H_i. Conversely, for a set of public keys from C^*, tests [20] shows that it almost surely eventually recovers the missing r equations and break the scheme.

Similarly, [24] and the related [29] shows a similar attack (with a more complex backend) almost surely breaks $3\mathrm{IC}^-$ and any other $\ell\mathrm{IC}^-$. For the ℓIC case, the point is the differential expose the symmetry for a linear map $(X_1, X_2, X_3) \mapsto (\xi_1 X_1,\, \xi_2 X_2,\, \xi_3 X_3)$. Exactly the same symmetric property is found enabling the same kind of attacks.

It was pointed out [15] that *Internal Perturbation* is almost exactly equal to *both Vinegar* variables *and Projection*, or fixing the input to an affine subspace. Let s be one, two or more. We basically set s variables of the public key to be zero to create the new public key. However, in the case of signature schemes, each projected dimension will slow down the signing process by a factor of q. A

differential attack looks for an invariant or a symmetry. Restricting to a subspace of the original **w**-space breaks a symmetry. Something like the *Minus* variant destroys an invariant. Hence the use of projection by itself prevents some attacks.

In [19], it was checked experimentally, for various C^* parameters n and θ, the effect of restricting the internal function to a randomly chosen subspace H of various dimensions s. This is a projected C^{*-} instance of parameters (q, n, r, s). We repeated this check for 3IC$^-$ and discover that again the attacks from [24,29] are prevented. We call this setup 3IC$^-$p(q, k, s).

3.3 Choosing Instances

For signature schemes, we choose $C^{*-}\mathrm{p}(2^4, 74, 22, 1)$, which uses 208-bit hashes and is related to the original FLASH by the fact that it uses half as wide variables and project one. We also choose 3IC$^-$p$(2^4, 32, 1)$, which acts on 256-bit hashes.

To invert the public map of projected minus signature schemes:

1. Put in random numbers to the "minus" coordinates.
2. Invert the linear transformation T to get **y**.
3. Invert the central map C^* or 3IC to get **x**.
4. Invert the final linear transformation S to get **w**.
5. If the last component (nybble) of **w** is zero, return the rest, else go to step 1 and repeat.

For the encryptions schemes, we choose PMI+$(136, 6, 18, 8)$ and 2IC (128,6,16) and (256,12,32).

To invert the public map of internally perturbed plus encryption schemes:

1. Invert the linear transformation T to get **y**.
2. Guess the vector $\mathbf{b} = \mathbf{v}(\mathbf{x})$.
3. Invert the central map C^* or 3IC on $\mathbf{y} - \mathbf{b}$ to get **x**.
4. Verify $\mathbf{b} = \mathbf{v}(\mathbf{x})$ and the extra a central equations; if they don't hold, then return to step 2 and repeat.
5. Invert the final linear S to get **w**.

4 Implementation Techniques

Most of the techniques here are not new, just implemented here. However, we do suggest that the bit-sliced Gaussian Elimination idea is new.

4.1 Evaluation of Public Polynomials

We pretty much follow the suggestions of [4] for evaluation of the public polynomials. I.e., over GF(2^8) we use traditional methods, i.e., logarithmic and exponential tables (full 64kB multiplication is faster for long streaming work but has a much higher up-front time cost for one-time use). Over GF(2^4) we use

bit-slicing and build lookup tables of all the cross-terms. Over GF(2) we evaluate only the non-zero polynomials.

4.2 Operating on Tower Fields

During working with the inversion of the central map, we operate the big-field systems using as much of tower fields as we can. We note that firstly, $\mathrm{GF}(2) = \{(0)_2, (1)_2\}$, where $(\cdot)_2$ means the binary representation. Then $t^2 + t + (1)_2$ is irreducible over GF(2). We can implement $\mathrm{GF}(2^{2^i})$ recursively. With a proper choice of α_i, we let $\mathrm{GF}(2^{2^i}) = \mathrm{GF}(2^{2^{i-1}})[t_i]/(t_i^2 + t_i + \alpha_i)$.. One can also verify that $\alpha_{i+1} := \alpha_i t_i$ will lead to a good series of extensions.

For $a, b, c, d \in \mathrm{GF}(2^{2^{i-1}})$, we can do Karatsuba-style

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

where the addition is the bitwise XOR and the multiplication of expressions of a, b, c, d and α_i are done in $\mathrm{GF}(2^{2^{i-1}})$. Division can be effected via $(at_i + b)^{-1} = (at_i + a + b)(ab + b^2 + a^2\alpha_i)^{-1}$.

While most of the instances we work with only looks at tower fields going up powers of two, a degree-three extension is similar with the extension being quotiented against t^3+t+1 and similar polynomials, and a three-way Karatsuba is relatively easy. We can do a similar thing for raising to a power of five.

4.3 Bit-Sliced GF(16) Rainbow Implementations

It is noted in [4] that GF(4) and GF(16) can be bitsliced for good effect. Actually, any $\mathrm{GF}(2^k)$ for small k can be bitsliced this way. In particular, it is possible to exploit the bitslicing to evaluate the private map.

1. Invert the linear transformation T to get $\mathbf{y}$ from $\mathbf{z}$. We can use bitslicing here to multiply each z_i to one columne of the matrix M_T^{-1}.
2. Guess at the initial block of vinegar variables
3. Compute the first system to be solved.
4. Solve the first system via Gauss-Jordan elimination with bitslice.
5. Compute the second system to be solved.
6. Solve the second system via Gauss-Jordan elimination with bitslice. We have computed all of $\mathbf{x}$.
7. Invert the linear transformation S to get $\mathbf{w}$ from $\mathbf{x}$.

Note that during the bitslice solving, every equation can be stored as four bit-vectors (here 32-bit or double words suffices), which stores every coefficient along with the constant term. In doing Gauss-Jordan elimination, we use a sequence of bit test choices to multiply the pivot equation so that the pivot coefficient becomes 1, and then use bit-slicing SIMD multiplication to add the correct multiple to every other equation. Bit-Sliced GF(16) is not used for TTS since the set-up takes too much time.

4.4 TTS Implementations

There are a few things to note:

1. Due to the sparsity of the central maps, setting up the Gaussian elimination to run using bitslice takes too much time. Hence, for TTS in GF(16) we complete the entire computation of the private map expressing each GF(16) element as a nybble (4 bits or half a byte) and start the evaluation of the public map by converting the nybble vector packed two to a byte, to the bitslice form.
2. Again for GF(16), we maintain two 4kByte multiplication tables that allows us to lookup either abc or ab and ac at the same time.
3. We use the special form of key generation mentioned in [33, 34]. That is, following Imai and Matsumoto [28], we divide the coefficients involved in each public key polynomial into linear, square, and crossterm portions thus:

$$z_k = \sum_i P_{ik} w_i + \sum_i Q_{ik} w_i^2 + \sum_{i<j} R_{ijk} w_i w_j = \sum_i w_i \left[P_{ik} + Q_{ik} w_i + \sum_{i<j} R_{ijk} w_j \right].$$

R_{ijk}, which comprise most of the public key, may be computed as in [34]:

$$R_{ijk} = \sum_{\ell=n-m}^{n-1} \left[(\mathrm{M}_T)_{k,(\ell-n+m)} \left(\sum_{p\, x_\alpha x_\beta \text{ in } y_\ell} p \left((\mathrm{M}_S)_{\alpha i} (\mathrm{M}_S)_{\beta j} + (\mathrm{M}_S)_{\alpha j} (\mathrm{M}_S)_{\beta i} \right) \right) \right]$$

The second sum is over all cross-terms $p\ x_\alpha x_\beta$ in the central equation for y_ℓ. For every pair $i < j$, we can compute at once R_{ijk} for every k in $O(n^2)$ totalling $O(n^4)$. Similar computations for P_{ik} and Q_{ik} take even less time.

The instances that we chose are tested not to suffer the same kind of attacks that fell previous TTS schemes, but we still don't have any conclusive evidence one way or the other of how likely this type of system can stand in the long run.

Acknowledgements

The authors thank Prof. Jintai Ding and Pei-Yuan Wu for invaluable comments and discussions, and also to National Science Council for sponsorship under Grant 96-2221-E-001-031-MY3.

References

1. Akkar, M.-L., Courtois, N.T., Duteuil, R., Goubin, L.: A fast and secure implementation of SFLASH. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 267–278. Springer, Heidelberg (2002)
2. Bardet, M., Faugère, J.-C., Salvy, B.: On the complexity of Gröbner basis computation of semi-regular overdetermined algebraic equations. In: Proceedings of the International Conference on Polynomial System Solving, pp. 71–74, Previously INRIA report RR-5049 (2004)

3. Bardet, M., Faugère, J.-C., Salvy, B., Yang, B.-Y.: Asymptotic expansion of the degree of regularity for semi-regular systems of equations. In: Gianni, P. (ed.) MEGA 2005 Sardinia (Italy) (2005)
4. Berbain, C., Billet, O., Gilbert, H.: Efficient implementations of multivariate quadratic systems. In: Biham, E., Youssef, A.M. (eds.) SAC 2006. LNCS, vol. 4356, pp. 174–187. Springer, Heidelberg (2007)
5. Bernstein, D.J., Birkner, P., Joye, M., Lange, T., Peters, C.: Twisted edwards curves. In: Vaudenay, S. (ed.) AFRICACRYPT 2008. LNCS, vol. 5023, pp. 389–405. Springer, Heidelberg (2008)
6. Bernstein, D.J., Lange, T.: Faster addition and doubling on elliptic curves. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 29–50. Springer, Heidelberg (2007)
7. Bernstein, D.J., Lange, T.: Inverted edwards coordinates. In: Boztaş, S., Lu, H.-F. (eds.) AAECC 2007. LNCS, vol. 4851, pp. 20–27. Springer, Heidelberg (2007)
8. Billet, O., Gilbert, H.: Cryptanalysis of rainbow. In: De Prisco, R., Yung, M. (eds.) SCN 2006. LNCS, vol. 4116, pp. 336–347. Springer, Heidelberg (2006)
9. Coppersmith, D., Stern, J., Vaudenay, S.: The security of the birational permutation signature schemes. Journal of Cryptology 10, 207–221 (1997)
10. Courtois, N., Goubin, L., Patarin, J.: SFLASH: Primitive specification (second revised version), Submissions, Sflash, 11 pages (2002), `https://www.cosic.esat.kuleuven.be/nessie`
11. Courtois, N.T., Klimov, A., Patarin, J., Shamir, A.: Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 392–407. Springer, Heidelberg (2000), `http://www.minrank.org/xlfull.pdf`
12. Ding, J.: A new variant of the Matsumoto-Imai cryptosystem through perturbation. In: Bao, F., Deng, R., Zhou, J. (eds.) PKC 2004. LNCS, vol. 2947, pp. 305–318. Springer, Heidelberg (2004)
13. Ding, J., Gower, J.: Inoculating multivariate schemes against differential attacks. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T. (eds.) PKC 2006. LNCS, vol. 3958. Springer, Heidelberg (2006), `http://eprint.iacr.org/2005/255`
14. Ding, J., Gower, J., Schmidt, D.: Multivariate Public-Key Cryptosystems. In: Advances in Information Security. Springer, Heidelberg (2006)
15. Ding, J., Schmidt, D.: Cryptanalysis of HFEv and internal perturbation of HFE. In: Vaudenay, S. (ed.) PKC 2005. LNCS, vol. 3386, pp. 288–301. Springer, Heidelberg (2005)
16. Ding, J., Schmidt, D.: Rainbow, a new multivariable polynomial signature scheme. In: Ioannidis, J., Keromytis, A., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 164–175. Springer, Heidelberg (2005)
17. Ding, J., Wolf, C., Yang, B.-Y.: ℓ-invertible cycles for multivariate quadratic public key cryptography. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 266–281. Springer, Heidelberg (2007)
18. Ding, J., Yang, B.-Y., Chen, C.-H.O., Chen, M.-S., Cheng, C.-M.: New differential-algebraic attacks and reparametrization of rainbow. In: Bellovin, S.M., Gennaro, R., Keromytis, A., Yung, M. (eds.) ACNS 2008. LNCS, vol. 5037, pp. 242–257. Springer, Heidelberg (2008), `http://eprint.iacr.org/2008/108`
19. Ding, J., Yang, B.-Y., Dubois, V., Cheng, C.-M., Chen, O.C.-H.: Breaking the symmetry: a way to resist the new differential attack. In: ICALP 2008. LNCS. Springer, Heidelberg (2008), `http://eprint.iacr.org/2007/366`

20. Dubois, V., Fouque, P.-A., Shamir, A., Stern, J.: Practical cryptanalysis of SFLASH. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 1–12. Springer, Heidelberg (2007)
21. Faugère, J.-C.: A new efficient algorithm for computing Gröbner bases (F_4). Journal of Pure and Applied Algebra 139, 61–88 (1999)
22. Faugère, J.-C.: A new efficient algorithm for computing Gröbner bases without reduction to zero (F_5). In: International Symposium on Symbolic and Algebraic Computation — ISSAC 2002, pp. 75–83. ACM Press, New York (2002)
23. Fouque, P.-A., Granboulan, L., Stern, J.: Differential cryptanalysis for multivariate schemes. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 341–353. Springer, Heidelberg (2005)
24. Fouque, P.-A., Macario-Rat, G., Perret, L., Stern, J.: Total break of the ℓIC- signature scheme. In: Public Key Cryptography, pp. 1–17 (2008)
25. Goubin, L., Courtois, N.T.: Cryptanalysis of the TTM cryptosystem. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 44–57. Springer, Heidelberg (2000)
26. Kipnis, A., Patarin, J., Goubin, L.: Unbalanced Oil and Vinegar signature schemes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 206–222. Springer, Heidelberg (1999)
27. Kipnis, A., Shamir, A.: Cryptanalysis of the oil and vinegar signature scheme. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 257–266. Springer, Heidelberg (1998)
28. Matsumoto, T., Imai, H.: Public quadratic polynomial-tuples for efficient signature verification and message-encryption. In: Günther, C.G. (ed.) EUROCRYPT 1988. LNCS, vol. 330, pp. 419–545. Springer, Heidelberg (1988)
29. Ogura, N., Uchiyama, S.: Remarks on the attack of fouque et al. against the ℓic scheme. Cryptology ePrint Archive, Report 2008/208 (2008), http://eprint.iacr.org/
30. Wolf, C., Braeken, A., Preneel, B.: Efficient cryptanalysis of RSE(2)PKC and RSSE(2)PKC. In: Blundo, C., Cimato, S. (eds.) SCN 2004. LNCS, vol. 3352, pp. 294–309. Springer, Heidelberg (2005), http://eprint.iacr.org/2004/237
31. Wolf, C., Preneel, B.: Taxonomy of public key schemes based on the problem of multivariate quadratic equations. Cryptology ePrint Archive, Report 2005/077, 64 pages, May 12 (2005), http://eprint.iacr.org/2005/077/
32. Yang, B.-Y., Chen, J.-M.: All in the XL family: Theory and practice. In: Park, C.-s., Chee, S. (eds.) ICISC 2004. LNCS, vol. 3506, pp. 67–86. Springer, Heidelberg (2005)
33. Yang, B.-Y., Chen, J.-M.: Building secure tame-like multivariate public-key cryptosystems: The new TTS. In: Boyd, C., González Nieto, J.M. (eds.) ACISP 2005. LNCS, vol. 3574, pp. 518–531. Springer, Heidelberg (2005)
34. Yang, B.-Y., Chen, J.-M., Chen, Y.-H.: TTS: High-speed signatures on a low-cost smart card. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 371–385. Springer, Heidelberg (2004)