Multivariate Public Key Cryptography

Jintai Ding[1] and Bo-Yin Yang[2]

[1] University of Cincinnati and Technical University of Darmstadt, ding@math.uc.edu
[2] Academia Sinica and Taiwan InfoSecurity Center, Taipei, Taiwan, by@moscito.org

Abstract. A multivariate public key cryptosystem (MPKCs for short) have a set of (usually) quadratic polynomials over a finite field as its public map. Its main security assumption is backed by the NP-hardness of the problem to solve nonlinear equations over a finite field. This family is considered as one of the major families of PKCs that could resist potentially even the powerful quantum computers of the future. There has been fast and intensive development in Multivariate Public Key Cryptography in the last two decades. Some constructions are not as secure as was claimed initially, but others are still viable. The paper gives an overview of multivariate public key cryptography and discusses the current status of the research in this area.

Keywords: Gröbner basis, multivariate public key cryptosystem, linear algebra, differential attack

1 Introduction

As envisioned by Diffie and Hellman, a public key cryptosystem (hereafter PKC for short) depends on the existence of class of "trapdoor one-way functions". This class and the mathematical structure behind it will determine all the essential characteristics of the PKC. So for example behind elliptic cryptography is the elliptic curve group, and behind NTRU stands the structure of an integral lattice.

Multivariate (Public-Key) Cryptography is the study of PKCs where the trapdoor one-way function takes the form of a multivariate quadratic polynomial map over a finite field. Namely the public key is in general given by a set of quadratic polynomials:

$$\mathcal{P} = (p_1(w_1, \ldots, w_n), \ldots, p_m(w_1, \ldots, w_n)),$$

where each p_i is a (usu. quadratic) nonlinear polynomial in $\mathbf{w} = (w_1, \ldots, w_n)$:

$$z_k = p_k(\mathbf{w}) := \sum_i P_{ik} w_i + \sum_i Q_{ik} w_i^2 + \sum_{i>j} R_{ijk} w_i w_j \tag{1}$$

with all coefficients and variables in $\mathbb{K} = \mathbb{F}_q$, the field with q elements. The evaluation of these polynomials at any given value corresponds to either the encryption procedure or the verification procedure. Such PKCs are called multivariate public key cryptosystems (hereafter MPKCs). Inverting a multivariate quadratic map is equivalent to solving a set of quadratic equations over a finite field, or the following problem:

Problem $\mathcal{MQ}$: 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 $\mathbb{K} = \mathbb{F}_q$, the field with q elements.

$\mathcal{MQ}$ is in general an NP-hard problem. Such problems are believed to be hard unless the class P is equal to NP. Of course, a random set of quadratic equations would not have a trapdoor and hence not be usable in an MPKC. The corresponding mathematical structure to a system of polynomial equations, not necessarily generic, is the ideal generated by those polynomials. So, philosophically speaking, multivariate cryptography relate to mathematics that handles polynomial ideals, namely algebraic geometry.

In contrast, the security of RSA-type cryptosystems relies on the complexity of integer factorization and is based on results in number theory developed in the 17^{th} and 18^{th} centuries. Elliptic curve cryptosystems employ the use of mathematics from the 19^{th} century. This quote is actually from Whitfield Diffie at the RSA Europe conference in Paris in 2002. At least Algebraic Geometry, the mathematics that MPKCs use, is developed in the 20^{th} century.

Since we are no longer dealing with "random" or "generic" systems, but systems where specific trapdoors exist, the security MPKCs is then not guaranteed by the NP-hardness of $\mathcal{MQ}$, and effective attacks may exist for any chosen trapdoor. The history of MPKCs therefore evolves as we understand more and more about how to design secure multivariate trapdoors.

Sec. 2 is a sketch of how MPKCs work in general. Sec. 3 gives examples of current MPKCs. Sec. 4 describes the known trapdoor constructions in somewhat more detail. Sec. 5 describes the most important mode of attacks. The last section will be a short discussion about future development.

2 The Basics of Multivariate PKCs

After Diffie-Hellman [28], cryptolographers proposed many trapdoor functions. Most of these were forgotten and RSA became dominant. The earliest published proposals of MPKCs scheme by Shigeo Tsujii and Hideki Imai, seemed to have arisen around this time. They are independently known to have worked on this topic in the early 1980s. Certainly lectures are given on this topic no later than 1983. However, for several years, their work were not published in anything other than Japanese, and remained largely unknown outside Japan.

As far as we know, the first article written in English describing a PKC with more than one independent variable may be the one from Ong *et al* [78], and the first use of more than one equation is by Fell and Diffie [52]. The earliest attempt bearing some resemblance to today's MPKCs (with 4 variables) seems to be [71]. In 1988, the first MPKC in the modern form appears [70]. It seems as if basic construction described below (cf. Sec. 2.1) has not changed for 20 years.

2.1 The Standard (Bipolar) Construction and Notations

Even if we restrict ourselves to cryptosystems for which the public key is a set of polynomials $\mathcal{P} = (p_1, \ldots, p_m)$ in variables $\mathbf{w} = (w_1, \ldots, w_n)$ where all variables and coefficients are in $\mathbb{K} = \mathbb{F}_q$, the way to hide the trapdoor is not unique.

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

$$\mathcal{P} : \mathbf{w} = (w_1, \ldots, w_n) \overset{S}{\mapsto} \mathbf{x} = \mathrm{M}_S\mathbf{w} + \mathbf{c}_S \overset{\mathcal{Q}}{\mapsto} \mathbf{y} \overset{T}{\mapsto} \mathbf{z} = \mathrm{M}_T\mathbf{y} + \mathbf{c}_T = (z_1, \ldots, z_m) \tag{2}$$

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.

The public key consists of the polynomials in $\mathcal{P}$. In practice, this is always the collection of the coefficients of the p_i's, compiled in some order conducive to easy computation. Since we are doing public-key cryptography, $\mathcal{P}(0)$ is always taken to be zero, hence public polynomials do not have constant terms.

The secret key consists of the informations in S, T, and $\mathcal{Q}$. That is, we collect $(\mathrm{M}_S^{-1}, \mathbf{c}_S)$, $(\mathrm{M}_T^{-1}, \mathbf{c}_T)$ and whatever parameters there exist in $\mathcal{Q}$. In theory one of $\mathbf{c}_S$ and $\mathbf{c}_T$ is extraneous but we keep it anyway.

To verify a signature or to encrypt a block, one simply computes $\mathbf{z} = \mathcal{P}(\mathbf{w})$. To sign or to decrypt a block, one computes $\mathbf{y} = T^{-1}(\mathbf{z})$, $\mathbf{x} = \mathcal{Q}^{-1}(\mathbf{y})$ and $\mathbf{w} = S^{-1}(\mathbf{x})$ in turn. Notice that these may be only one of the many pre-images, not necessarily an inverse function in the strict sense of the word.

We summarize the notations used in Table 1 and will henceforth use it consistently to make our exposition easier to understand. And we summarize operating details below so that the reader will have some basic sense of about how these schemes can be applied practically.

Cipher block or Message digest Size: m elements of $\mathbb{F}_q$
Plaintext block or Signature Size: n elements of $\mathbb{F}_q$
Public Key Size: $mn(n+3)/2$ $\mathbb{F}_q$-elements, often stored in log-form
Secret Key Size: Usually $\left(n^2 + m^2 + [\#\text{ parameters in } \mathcal{Q}]\right)$ $\mathbb{F}_q$-elements, often stored in log-form
Secret Map Time Complexity: $(n^2 + m^2)$ $\mathbb{F}_q$-multiplications, plus whatever time it is needed to invert $\mathcal{Q}$
Public Map Time Complexity: About $mn^2/2$ $\mathbb{F}_q$-multiplications
Key Generation Time Complexity: n^2 times the invocation cost of $\mathcal{P}$; between $O(n^4)$ and $O(n^5)$

We immediately see the major disadvantage with MPKCs: Their keys are very large compared to traditional systems like RSA or ECC. For example, the public key size of RSA-2048 is not much more than 2048 bits, but a current version of the Rainbow signature scheme has $n = 42$, $m = 24$, $q = 256$, i.e.,

α	the power in a C^* construction
$\mathbf{a}, \mathbf{b}, \mathbf{c}$	constant vectors
$\mathbf{c}_S, \mathbf{c}_T$	constant parts of linear maps S, T
$C^* = (c_1^*, c_2^*, \ldots, c_n^*)$	the Matsumoto-Imai map $C^*_{q,n,\alpha} : \mathbf{x} \mapsto \mathbf{y} = \mathbf{x}^{q^\alpha+1}$ in $\mathbb{F}_{q^n}$
DF	(symmetric) differential of the function/map F
D, D_{reg}, and D_{XL}	degree in system-solving degree, operating degree of $\mathbf{F_4}/\mathbf{F_5}$ and XL
$\mathbb{F}_q$	finite (Galois) field of q elements, any representation of
g	sometimes, a generator of $\mathbb{K} = \mathbb{F}_q$
H_i	symmetric matrices for quadratic part of p_i (or z_i) in w_i
h, i, j, k, l	index variables, k often $:= [\mathbb{L} : \mathbb{K}]$, dimension of $\mathbb{L}$ over $\mathbb{K}$
$\mathcal{K}$	denoting a kernel
$\ker_{\mathbf{v}} f$	kernel of the symmetric matrix denoting quadratic part of f as function of $\mathbf{v}$.
$\mathbb{K}$	the base field, usually $= \mathbb{F}_q$
$\mathbb{L}$	$\mathbb{F}_{q^k}$, a field that is larger than $\mathbb{K}$
M_i	symmetric matrices for the quadratic part of y_i in x_j
M_S, M_T	matrices of linear maps S, T.
m	number of equations
$\mathfrak{m}$	a multiplication, as a unit of time
n	number of variables
$O()$, $o()$, $\Omega()$	standard big-O, small-o, Omega notations
o	number of oil variables
P_{ik}	Matsumoto-Imai notation for coefficient of w_i in z_k
$\mathcal{P} = (p_1, \ldots, p_m)$	public map
Q_{ik}	Matsumoto-Imai notation for coefficient of w_i^2 in z_k
$\mathcal{Q} = (q_1, \ldots, q_m)$	central map
q	the size of the base field
R_{ijk}	Matsumoto-Imai notation for coefficient of $w_i w_j$ in z_k
R	$\lvert\mathcal{R}\rvert$, the number of relations (equations) in XL or $\mathbf{F_4}$
$\mathcal{R}^{(D)}$ or $\mathcal{R}$	Set of equations in XL or $\mathbf{F_4}$
r	usu. the minimum rank or # of removed (minus) equations
S	the initial linear map, $S(\mathbf{w}) = \mathbf{x} = \mathrm{M}_S\mathbf{w} + \mathbf{c}_S$
T	the final linear map, $T(\mathbf{y}) = \mathbf{z} = \mathrm{M}_T\mathbf{y} + \mathbf{c}_T$, or #terms in XL ($\lvert\mathcal{T}\rvert$ below)
$\mathcal{T}^{(D)}$ or $\mathcal{T}$	set of terms (monomials) in XL or $\mathbf{F_4}$
u	often the high rank parameter or # of Rainbow stages
v	number of vinegar variables
$v_1 < v_2 < \cdots < v_{u+1} = n$	structure of Rainbow $(v_1, o_1, \ldots, o_u)$, $o_i := v_{i+1} - v_i$
$\mathbf{w} = (w_1, \ldots, w_n)$	signature or plaintext block
X_i, Y_j	elements in intermediate fields
$\mathbf{x} = (x_1, \ldots, x_n)$	central variables, input to central map $\mathcal{Q}$
$\mathbf{y} = (y_1, \ldots, y_m)$	output of central map $\mathcal{Q}$, central polynomials
$\mathbf{z} = (z_1, \ldots, z_m)$	digest or ciphertext block

Table 1. Notations and Terminology

the size of the public key is 22680 bytes, above the 16kB of flash memory that some small smartcards have. Private keys are smaller, but still formidable for small embedded devices which has memory constraints. However operating on units hundreds of bits long (for Elliptive Curve groups and especially RSA) is prohibitively expensive for embedded devices without a co-processor. So MPKCs have some compensating advantages and still has potential on those devices.

2.2 Other Constructions

It should be noted that MPKCs are also sometimes called trapdoor $\mathcal{MQ}$ schemes for a reason, all the construction currently used do quadratic public keys for speed reasons – with higher order terms, the explosion in number of coefficients offset any possible gain in efficiency. Furthermore, in the bipolar form, higher-order terms may in fact hurt the security.

Here we cover two alternatives in which multivariate polynomials can be used for PKCs. These are called the Implicit Form and Isomorphisms of Polynomials.

Implicit Form MPKCs The public key is a system of l equations

$$\mathcal{P}(\mathbf{w}, \mathbf{z}) = \mathcal{P}(w_1, \ldots, w_n, z_1, \ldots, z_m) = (p_1(\mathbf{w}, \mathbf{z}), \ldots, p_l(\mathbf{w}, \mathbf{z})) = (0, \ldots, 0) \quad (3)$$

where each p_i is a polynomial in $\mathbf{w} = (w_1, \ldots, w_n)$ and $\mathbf{z} = (z_1, \ldots, z_m)$. This $\mathcal{P}$ is built from the secret $\mathcal{Q}$

$$\mathcal{Q}(\mathbf{x}, \mathbf{y}) = q(x_1, \ldots, x_n, y_1, \ldots, y_m) = (q_1(\mathbf{x}, \mathbf{y}), \ldots, q_l(\mathbf{x}, \mathbf{y})) = (0, \ldots, 0),$$

where $q_i(\mathbf{x}, \mathbf{y})$ is polynomial in $\mathbf{x} = (x_1, \ldots, x_n)$, $\mathbf{y} = (y_1, \ldots, y_m)$ such that

- For any given specific element $\mathbf{x}'$, we can easily solve the equation

$$\mathcal{Q}(\mathbf{x}', \mathbf{y}) = (0, \ldots, 0); \quad (4)$$

- for any given specific element $\mathbf{y}'$, we can easily solve the equation

$$\mathcal{Q}(\mathbf{x}, \mathbf{y}') = (0, \ldots, 0), \quad (5)$$

- (usu.) Eq. 4 is linear and Eq. 5 is nonlinear but specialized to be solvable.

Now, we can build

$$\mathcal{P} = L \circ h(S(\mathbf{w}), T^{-1}(\mathbf{z})) = (0, \ldots, 0),$$

where S, T are invertible affine maps and L is linear. To verify a signature $\mathbf{w}$ with the digest $\mathbf{z}$, one checks that $\mathcal{P}(\mathbf{w}, \mathbf{z}) = 0$. If we want to use $\mathcal{P}$ to encrypt the plaintext $\mathbf{w}$, we would solve $\mathcal{P}(\mathbf{w}, \mathbf{z}) = (0, \ldots, 0)$, and find the ciphertext $\mathbf{z}$. To invert (i.e., to decrypt or more likely to sign) $\mathbf{z}$, one first calculates $\mathbf{y}' = T^{-1}(\mathbf{z})$, then plugs $\mathbf{y}'$ into the equation (5) and solve for $\mathbf{x}$. The result plaintext or signature is given by $\mathbf{w} = S^{-1}(\mathbf{x})$.

To recap, in an *implicit-form MPKC*, the public key consists of the l polynomial components of $\mathcal{P}$ and the field structure of k. The secret key mainly consists of L, S and T. Depending on the case the equation $\mathcal{Q}(X, Y) = (0, \ldots, 0)$ is either known or has parameters that is a part of the secret key. Again the basic idea is that S, T, L serve the purpose to "hide" the equation $\mathcal{Q}(\mathbf{x}, \mathbf{y}) = 0$, which otherwise could be easily solved for any $\mathbf{y}$. Mixed schemes are relatively rare, one example being Patarin's Dragon [82].

Isomorphism of Polynomials The IP problem originated by trying to attack MPKCs by finding the secret keys. Let $\bar{F}_1, \bar{F}_2$ with

$$\bar{F}_i(x_1, \ldots, x_n) = (\bar{f}_{i1}, \ldots, \bar{f}_{im}), \tag{6}$$

be two polynomial maps from K^n to K^m. The IP problem is to look for two invertible affine linear transformations S on K^n and T over K^m (if they exist) such that

$$\bar{F}_1(x_1, \ldots, x_n) = T \circ \bar{F}_2 \circ S(x_1, \ldots, x_n). \tag{7}$$

It is clear that this problem is closely related to the attack of finding private keys for a MPKC, for example the Matsumoto-Imai cryptosystems, and was first proposed by Patarin [83], where the verification process is performed through showing the equivalence (or isomorphism) of two different maps. A simplified version is called the isomorphism of polynomials with one secret (IP1s) problem, where we only need to find the map S (if it exists), while the map T is known to be the identity map. More later in this direction are [51, 57, 68, 86, 87].

3 Examples of Multivariate PKCs

In this section, we bring to you three curent MPKCs; each with special properties, advantages and disadvantages. We don't try to discuss their security in this section — that will be left until the next section.

Scheme	result	SecrKey	PublKey	KeyGen	SecrMap	PublMap
RSA-1024	1024b	128 B	320 B	2.7 sec	84 ms	2.0 ms
ECDSA-$\mathbb{F}_{2^{163}}$	320b	48 B	24 B	1.6 ms	1.9 ms	5.1 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	0.74 ms
QUARTZ	128b	71.0 kB	3.9 kB	3.1 sec	11 sec	0.24 ms

Table 2. Current Multivariate PKCs Compared on a Pentium III 500

3.1 The Rainbow $(2^8, 18, 12, 12)$ Signature Scheme

We characterize a Rainbow [40] 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 $v_l < i \leq v_{l+1}$, and all the x_j with $j \leq v_l$, we can compute $x_{v_l+1}, \ldots, x_{v_{l+1}}$.
- 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 [108] scheme if the coefficients of $\mathcal{Q}$ are sparse. *We suggest a reference Rainbow design with the following concrete parameters: $q = 256$, $n = 42$, $m = 24$, structure sequence $(18, 12, 12)$ with no omitted central terms, expected security 2^{80} multiplications in $\mathbb{F}_{2^8}$. The size of the public key is 22680 bytes, the private key is 17748 bytes.* It's called Rainbow $(2^8, 18, 12, 12)$ for obvious reasons. A smaller version with $\mathbb{F}_{2^4}$ as the base field is also given in the table.

3.2 PMI+(136, 6, 18, 8), a Perturbed Matsumoto-Imai Plus

We may always represent the field $\mathbb{F}_{q^n}$ as an n-dimensional vector space over $\mathbb{F}_q$ via $\mathbb{F}_{q^n} \cong \mathbb{F}_q[X]/P(X)$, where P is any irreducible polynomial of degree n. Once we select P, we will then hereafter identify $\mathbb{F}_{q^n}$ with $(\mathbb{F}_q)^n$. The map induced by $\mathbf{x} \in \mathbb{F}_{q^n} \mapsto \mathbf{x}^q$ is then a linear transformation. We thus know that a map $g : \mathbf{x} \mapsto \mathbf{y} = \mathbf{x}^{q^\alpha+1}$ is homogeneously quadratic in $\mathbf{x}$. Furthermore, if and only if $\gcd(q^\alpha + 1, q^n - 1) = 1$, then this map is invertible. In fact we can find an h such that $g^{-1}(\mathbf{y}) = (\mathbf{y})^h$. This g will be termed $C^*_{q,n,\alpha}$, where the parameters may be omitted if context permits. We also write its components as $C^* = (c_1^*, c_2^*, \ldots, c_n^*)$. That is the central map of C^* or Matsumoto-Imai itself.

For Perturbed Matsumoto-Imai Plus we both perturb and add polynomials. That is, we set $q = 2$ (to make guessing easier later) and choose $\mathbf{v} = (v_1, \ldots, v_r)$, a collection of r linear forms in $\mathbf{x}$, and $\mathbf{f} = (f_1, \ldots, f_n)$, a random n-tuple of

quadratic functions in $\mathbf{v}$. Further take $\mathbf{g} = (g_1, \ldots, g_a)$ be an a-tuple of random quadratic functions of $\mathbf{x}$. We define $\mathcal{Q} := (C^* + \mathbf{f}(\mathbf{v})) \parallel \mathbf{g}$. That is, $\mathcal{Q}$ is a map from $\mathbb{F}_{2^n}$ to $\mathbb{F}_{2^{n+a}}$ whose components are given by

$$q_i(\mathbf{x}) := \begin{cases} c_i^*(\mathbf{x}) + f_i\left(\mathbf{v}(\mathbf{x})\right), & i = 1 \cdots n; \\ g_{i-n}(\mathbf{x}), & i = n+1 \cdots n+a. \end{cases}$$

How do invert $\mathcal{Q}$? That is, if $\mathbf{y} = \mathcal{Q}(\mathbf{x})$, how would we then find $\mathbf{x}$? First, we toss out the last a components, and randomly guess at the perturbation term $\mathbf{v}(\mathbf{x})$. That is, let h is the exponent that can be used to invert C^*. If $\mathbf{y}'$ is the first n components of $\mathbf{y}$, for all possible $\mathbf{b} \in \mathbb{F}_{2^r}$ we compute $\mathbf{x} = (\mathbf{y}' - \mathbf{b})^h$ and check to see whether $\mathbf{v}(\mathbf{x}) = \mathbf{b}$. Since inverting C^* is relatively slow, we can say that the perturbation made it 2^r times slower to decrypt than the corresponding C^*. *The last a components can also ensure the correctness of the ciphertext.*

It remains to give the system some concrete parameters. *At the moment, our choices are as in [33]:* $(n, r, a, \alpha) = (136, 6, 18, 8)$. *The public key size is* $n(n+1)(n+a)/2$ *bits or* 167688 *bytes; the secret key is* $(n+a)^2 + n^2 + nr(r+3)/2 + an(n+1)/2$ *bits or 26324 bytes. Design security is* 2^{83}.

3.3 The Quartz or HFEv-(2, 129, 103, 3, 4) Signature Scheme

An immediate extension of the C^* concept is Hidden Field Equations, introduced by Patarin [83]. In place of the C^* polynomial, we would substitute this :

$$\mathcal{Q} : \mathbf{x} \in (\mathbb{F}_q)^n \longmapsto \mathbf{y} = \sum_{0 \le i,j < n} a_{ij} \mathbf{x}^{q^i + q^j} + \sum_{0 \le i < n} b_i \mathbf{x}^{q^i} + c,$$

Where the coefficients are chosen more or less at random. It is also quadratic in the components of $\mathbf{x}$. Computing $\mathcal{P}^{-1}(\mathbf{y})$ by the Berlekamp Algorithm has time complexity $O(nd^2 \log d + d^3)$ where d is the maximum degree ($= 129$ in Quartz). Quartz uses the vinegar modification suggested by Kipnis and Patarin [64], with an auxiliary variable $\bar{\mathbf{x}}$ which occupies a subspace of small rank in $\mathbb{F}_q^n$ as follows:

$$\mathcal{Q}(\mathbf{x}, \bar{\mathbf{x}}) := \sum_{i,j} a_{ij} \mathbf{x}^{q^i + q^j} + \sum_{i,j} b_{ij} \mathbf{x}^{q^i} \bar{\mathbf{x}}^{q^j} + \sum_{i,j} \alpha_{ij} \bar{\mathbf{x}}^{q^i + q^j} + \sum_i b_i \mathbf{x}^{q^i} + \sum_i \beta_i' \bar{\mathbf{x}}^{q^i} + c. \tag{8}$$

The public key of the Quartz signature scheme uses $q = 2$, $n = 103$, dimension 4 for the $\bar{\mathbf{x}}$ subspace, and furthermore uses the *minus variant* by *removing three polynomials* from the public key. So there are 107 variables and 100 equations. The actual verification procedure in Quartz is even more complex [21], involving using the public map *four times* to avoid birthday attacks, since the design goal is a short signature (here 128 bits) and not speed. Despite this detail, the ability to solve such system still enables one to forge a signature.

The secret key of Quartz is 3kB, the public key size is $(100 \times 107 \times 108/2)$ *bits = 71kB. Design security is* 2^{80}.

3.4 Some Computational Aspects of MPKCs

Many computations of MPKCs will be conducted in $\mathbb{K} = \mathbb{F}_q$. Often q is a small power of 2 so that each element in $\mathbb{K}$ can be stored in a byte and addition represented by bitwise exclusive-or. To multiply, normally one choose a generator g in $\mathbb{K}$ such that all non-zero x can be written as $x = g^i$ (this i is also denoted $\log_g x$). We build logarithm and exponential tables and evaluate multiplications between non-zero x and y as $g^{(\log_g x + \log_g y)}$. Doing each multiplication from scratch via this method takes three table lookups and two conditional jumps and is comparatively time-consuming. This is why time-complexities are often counted in $\mathbb{K}$-multiplications. *To save time, elements of $\mathbb{F}_{2^7}$ or $\mathbb{F}_{2^8}$ that will only be used for multiplication are always stored as logarithms, for example coefficients.*

For today's highly pipelined CPUs, accessing memory is particularly expensive, and buffer memory for the most often used data (known as *L1 cache memory*) is limited to between 32kB and 256kB. Therefore complete multiplication tables of size q^2 are almost never used (except maybe when $q = 16$).

Things change when working with small microcontrollers. For example, the table exponentials is usually $2q$ $\mathbb{K}$-elements long. But for 8-bit microcontrollers, one can't have indices larger than a byte and hence evaluate $(\log_g x + \log_g y)$ mod 255 using a single extra `ADC` (add with carry) instruction instead.

SIMD (single instruction, multiple data) is an important factor. It is very important to pack data so that one can make use of the 64- and 128-bit-wide `XOR` instructions, especially if $q = 2$ or 4 (it's called "bit-slicing", cf. [7]).

Operations in an extension $\mathbb{L} = \mathbb{F}_q^n$ as vectors over $\mathbb{K} = \mathbb{F}_q$ is frequent (e.g., in big-field MPKCs). A product in $\mathbb{L}$ is like multiplying two degree $< n$ polynomials over $\mathbb{F}_q$. Using schoolbook multiplication and then reducing the terms with degree $\geq n$ takes at most $2n^2$ multiplications. A more advanced method like Karatsuba takes less time. A division is a little slower than a multiplication.

It is also not a trivial issue to build keys for an MPKC. The classical way to compute the keys is interpolation [70]. In general, one select M_S, $\mathbf{c}_S$, M_T plus whatever parameters in $\mathcal{Q}$, if any. We can set

$$\mathbf{c}_T := \mathrm{M}_T \mathcal{Q}(\mathbf{c}_S),$$

which makes all the constant terms zero. Now we can evaluate $\mathcal{P}(\mathbf{w}) = T \circ \mathcal{Q} \circ S(\mathbf{w})$ for any $\mathbf{w}$. Write the Matsumoto-Imai form public key (Eq. 1) as:

$$z_k = \sum_i w_i \left[P_{ik} + Q_{ik} w_i + \sum_{j<i} R_{ijk} w_j \right]. \tag{9}$$

In $\mathbb{F}_2$, $x^2 = x$ for any x, so there is no Q_{ik} term. One also note that to evaluate the public key one need to do one $\mathbb{F}_q$ multiplication per element of the public key. Let $\mathbf{b}_i \in \mathbb{F}_q^n$ be the unit vector in the i-th axis, and for $q > 2$, we choose

any $a \neq 0, 1$ and get

$$\begin{aligned} Q_{ik} &:= (a(a-1))^{-1} \left(p_k(a\mathbf{b}_i) - a p_k(\mathbf{b}_i)\right) \\ P_{ik} &:= p_k(\mathbf{b}_i) - Q_{ik} \\ R_{ijk} &:= p_k(\mathbf{b}_i + \mathbf{b}_j) - Q_{ik} - Q_{jk} - P_{ik} - P_{jk} \end{aligned} \tag{10}$$

For $\mathbb{F}_2$, it becomes

$$\begin{aligned} P_{ik} &:= p_k(\mathbf{b}_i) \\ R_{ijk} &:= p_k(\mathbf{b}_i + \mathbf{b}_j) - P_{ik} - P_{jk} \end{aligned}$$

So key generation means invoke n^2 times the combination $T \circ \mathcal{Q} \circ S$. We can see that both S and T takes about n^2 time. If we write $\mathcal{Q}$ coefficientwise, it would take $n^3/2$ multiplications. So we see that worst case key generations takes about $n^5/2$ multiplications in $\mathbb{K} = \mathbb{F}_q$. In certain situations, it is closer to $O(n^4)$.

Let's demonstrate this for a C^* based scheme where the rate-determining mechanism is the evaluation of $C^* : \mathbf{x} \in \mathbb{L} = \mathbb{F}_{q^n} \mapsto \mathbf{y} = \mathbf{x}^{q^\alpha+1}$. There is a linear map L in $(\mathbb{F}_q)^n$ that maps $\mathbf{x} \mapsto \mathbf{x}^{q^\alpha}$. This we precompute. Evaluating $L\mathbf{x}$ takes n^2 multiplications in $\mathbb{F}_q$. Then the product in $\mathbb{L}$ is $2n^2$ multiplications max.

Other big-field variants based on ℓIC and HFE have a similar property. For single-field MPKCs where key generations takes close to $O(n^4)$, see Sec. 4.4.

4 Basic Constructions and Variations

MPKCs are built in many ways. We aim to give you the major types of constructions, maybe accent some important associated algebraic characteristics (like this), and common variations thereof. A summary of variants is given in Table 3.

4.1 Historical Constructions

The first attempt to construct a multivariate signature [78, 79] is based on a quadratic equation

$$y = x_1^2 + \alpha x_2^2 \pmod{n}, \tag{11}$$

where $n = pq$ is an RSA modulus, the product of two large primes. To sign a message y, we need to find one of the many (about n) solutions (x_1, x_2) to Eq. 11, which is easy if we know the factorization of n. The public key is essentially the integer n and Eq. 11. Since the security relies on the factorization of n, this system is really a derivative of RSA, though it indeed initiated the idea of multivariate cryptosystems. This system was broken by Pollard and Schnorr in [89], where they found a probabilistic algorithm to solve Eq. 11 for any y without knowing the factors of n. Assuming the generalized Riemann hypothesis, a solution can be found with a time complexity of $O((\log n)^2 \log\log|k|)$ in $O(\log n)$-bit integer operations.

The idea of Diffie and Fell [52] was to build a cryptosystems using the composition of invertible linear maps and simple tame maps of the form $T(x_1, x_2) = (x_1 + g(x_2), x_2)$, where g is a polynomial.

Tame maps are easily invertible and hard to unscramble when composed with each other, however [52] used only two variables and equations; not surprisingly, the authors concluded that it appeared very difficult to build such a cryptosystem with practical value that is both secure and has a public key of practical size.

An attempt to build a true multivariate (with four variables) public key cryptosystem were also made by Matsumoto, Imai, Harashima and Miyagawa [71], where the public keys are given by quadratic polynomials. However it was soon defeated [77]. People soon realized that more than 4 variables are needed.

4.2 Triangular Constructions

Of course, the tame maps used in [52] are a special case of the "triangular" or *de Jonquières* maps from algebraic geometry, which are more generally defined by:

$$J(x_1, \ldots, x_n) = (x_1 + g_1(x_2, \ldots, x_n), \ldots, x_{n-1} + g_{n-1}(x_n), x_n), \qquad (12)$$

where the g_i are arbitrary polynomial functions. We note that J can be easily inverted assuming that the g_i are known. The invertible affine linear maps over k^n together with the de Jonquières maps belong to the family of so-called tame transformations from algebraic geometry, including all transformations that arise as a composition of elements of these two types of transformations. Tame transformations are elements of the group of automorphisms of the polynomial ring $k[x_1, \ldots, x_n]$. Elements in this automorphism group that are not tame are call wild. Given a polynomial map, it is in general very difficult to decide whether or not the map is tame, or even if there is indeed any wild map [75], a question closely related to the Jacobian conjecture. This problem was possibly solved in 2003 when [93] claims to prove that the Nagata map is wild.

The first attempt in the English literature with a clear triangular form is the Birational Permutations construction by Shamir [92]. However, triangular constructions were earlier pursued unsuccessfully in Japan under the name " sequential solution type systems" [61,96,97]. Their construction is actually even more general in the sense that they use rational functions instead of just polynomial. These works are not so well-known, partially because they were in Japanese.

Triangular maps are lightning fast to evaluate and to invert. However, they do have another definitive characteristic, an algebraic one, that must be accounted for. On the small end of a triangular system, so to speak, a variable is mapped to some simple function of itself. On the bigger end, one variable appears in a single equation only. The other equations involve successively more variables.

In other words, let us write the quadratic portion of the central polynomials $y_i = q_i(\mathbf{x})$ as bilinear forms, or take the symmetric matrix denoting the symmetric differential of the central polynomials as in

$$q_i(\mathbf{x} + \mathbf{b}) - q_i(\mathbf{x}) - q_i(\mathbf{b}) + q_i(0) := \mathbf{b}^T M_i \mathbf{x}, \qquad (13)$$

then $\operatorname{rank} M_i$ increases monotonically as i increases. In fact, if $q = 2^k$, the equation dealing with x_1 always has rank zero. Furthermore, $\ker M_1 \subset \ker M_2 \cdots$. This is the *chain of kernels* as pointed out by Coppersmith *et al* [18].

This *rank* and chain relation is invariant under invertible map S. That is, consider y_i as a function of $\mathbf{w}$, the corresponding differential is $\mathbf{b}^T \left(\mathrm{M}_S^T M_i \mathrm{M}_S\right) \mathbf{x}$. For the most part, M_S is full-rank, hence $\operatorname{rank}\left(\mathrm{M}_S^T M_i \mathrm{M}_S\right) = \operatorname{rank} M_i$.

This leads to what is known as *rank attacks* based on linear algebra [18, 58]. Therefore triangular/tame constructions can't be used alone. Some ways to design around this problem are *lock polynomials* (Sec. 4.6), *solvable segments* (Sec. 4.4) and *plus-minus* (Sec. 4.5).

4.3 Big-Field Families: Matsumoto-Imai (C^*) and HFE

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 [70]. 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):

$$\mathcal{Q} = \phi \circ \overline{\mathcal{Q}} \circ \phi^{-1}. \tag{14}$$

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. 2.

Now we briefly recap how C^* is defined earlier (cf. Sec. 3.2). Matsumoto and Imai suggest that we pick a $\mathbb{K}$ of characteristic 2 and this map $\overline{\mathcal{Q}}$

$$\overline{\mathcal{Q}} : \mathbf{x} \longmapsto \mathbf{y} = \mathbf{x}^{1+q^\alpha}, \tag{15}$$

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

$$\overline{\mathcal{Q}}^{-1}(\mathbf{x}) = \mathbf{x}^h, \tag{16}$$

where $h(1 + q^\alpha) = 1 \bmod (q^n - 1)$. This ensures that we can decrypt any secret message easily by this inverse. *For the rest of this chapter, 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. When necessary to distinguish the inner product in a vector space over $\mathbb{K}$ and the larger field $\mathbb{L}$, the former will be denoted by a dot ($\cdot$) and the latter an asterisk ($*$). One more important thing is that the map $\mathcal{Q}$ is always quadratic due to the linearity of the Frobenius map $\mathbf{x} \to \mathbf{x}^{q^\alpha}$.*

A significant algebraic implication of C^* and Eq. 15 is $\mathbf{y}^{q^\alpha - 1} = \mathbf{x}^{q^{2\alpha} - 1}$ or

$$\mathbf{x}\mathbf{y}^{q^\alpha} = \mathbf{x}^{q^{2\alpha}}\mathbf{y}. \tag{17}$$

This enabled Jacques Patarin [81] to cryptanalyze the original C^* with his bilinear relations (see Sec. 5.1). Though the original idea of C^* failed, it has inspired many new designs, mostly from Patarin and his collaborators (cf. Secs. 4.5 and 4.8).

The most significant of the C^* derivatives is likely HFE (Hidden Field Equations). As mentioned in Sec. 3.3, instead of using for $\mathcal{Q}$ the monomial used by C^*, we would substitute the *extended Dembowski-Ostrom polynomial map*:

$$\mathcal{Q} : \mathbf{x} \in \mathbb{L} = \mathbb{F}_q^n \longmapsto \mathbf{y} = \sum_{0 \le i \le j < r} a_{ij} \mathbf{x}^{q^i + q^j} + \sum_{0 \le i < r} b_i \mathbf{x}^{q^i} + c, \qquad (18)$$

This map is in general *not* one-to-one; some kind of checksum is required to identify the inverse from one of a number of possible candidates. Inverting $\mathcal{Q}$ is equivalent to solving a univariate equation of high degree in $\mathbb{L}$. It is fairly well-studied and straightforward to implement but *not very fast*, using some version of the Berlekamp (or say Cantor-Zassenhaus) algorithm [8,14,56]. Typically, the cost of this solution is $O(nd^2 \log d + d^3)$, where d is the maximum degree of $\mathcal{Q}$.

One might conclude, then, that we should have as low d as possible, or since usually $d = 2q^r$ or $q^r + 1$, as low r as possible. It turns out not to be like this. Just as Eq. 15 intrinsically meant that the C^* map has a rank of 2 and leads to Eq. 17 and all the known cryptanalysis of C^* related systems, Eq. 18 fundamentally is responsible for all the algebraic properties of HFE.

A key fact is that the intrinsic rank of the map is bounded by r, and usually achieves that value for randomly chosen parameters. This rank is very closely related to the complexity of current attacks [23, 50]. For example, the HFE Challenge 1 solved by Faugère and Joux [50] has an intrinsic rank of 4.

HFE with a high d is unbroken, although it can be really slow to decrypt/invert. Quartz probably set a record for the slowest cryptographical algorithm when submitted to NESSIE — on a Pentium III 500MHz, it took half a minute to do a signature [since improved to 10s with better programming].

Finally, C^* and HFE each can be modified by techniques mentioned elsewhere (Plus-Minus, vinegar variables, and internal perturbation). Also related are the ℓIC system (Sec. 4.7) and probabilistic big-field based MPKCs [59]. One can safely say that all in all, C^* really spawned a lot of useful research.

4.4 Unbalanced Oil and Vinegar and Derivatives

The Oil and Vinegar and later derived unbalance Oil and Vinegar schemes [64,80] are suitable for signatures. This construction is inspired by the idea of linearization equations (cf. Sec. 4.3). Suppose $v < n$ is an integer and $m = o = n - v$. The variables $x_1, \ldots, x_v$ are termed *vinegar* variables and $x_{v+1}, \ldots, x_n$ *oil* variables.

Take a map $\mathcal{Q} : \mathbb{K}^n \to \mathbb{K}^m$ with form $\mathbf{y} = \mathcal{Q}(\mathbf{x}) = (q_1(\mathbf{x}), \ldots, q_o(\mathbf{x}))$, where

$$q_l(\mathbf{x}) = \sum_{i=1}^{v} \sum_{j=i}^{n} \alpha_{ij}^{(l)} x_i x_j, \; l = 1 \cdots o$$

and all coefficients are randomly chosen from the base field $\mathbb{K}$. Here we notice that there are no quadratic terms of oil variables, which means the oil variables and vinegar variables are not fully mixed (like oil and vinegar in a salad dressing) and this explains the name of this scheme.

The public map $\mathcal{P}$ is constructed as $\mathcal{P} = \mathcal{Q} \circ S$, where S is an invertible linear map. Here the change of basis is a process to "mix" fully oil and vinegar, so one can not see what is oil and what is vinegar. Note that with the pure OV and UOV constructions, we need not use a T, and it is in fact usually omitted.

The original Oil and Vinegar signature scheme has $m = o = v = n/2$. When $o < v$, it becomes the unbalance Oil and Vinegar signature scheme. The public key are $\mathcal{P} = (p_1, \ldots, p_o)$, the polynomial components of $\mathcal{P}$. The secret key consists of the linear map S and the map $\mathcal{Q}$.

Given a message $\mathbf{y} = (y_1, \ldots, y_o)$, in order to sign it, one needs to try to find a vector $\mathbf{w} = (w_1, \ldots, w_n)$ such that $\mathcal{P}(\mathbf{w}) = \mathbf{y}$. With the secret key it can be done easily. First, one guesses values for each vinegar variable $x_1, \ldots, x_v$, and obtains a set of o linear equations with the o oil variables $x_{v+1}, \ldots, x_n$. With high probability it has a solution. If it does not have a solution, one guesses at the vinegar variables again until one finds a pre-image of a given element in $\mathbb{K}^o$. Then one applies S^{-1}. To check if $\mathbf{w}$ is indeed a legitimate signature for $\mathbf{y}$, one only needs to get the public map $\mathcal{P}$ and check if indeed $\mathcal{P}(\mathbf{w}) = \mathbf{y}$.

What algebraic property is most significant in an unbalanced Oil-and-Vinegar system? No doubt the lack of pure oil cross-terms. Equivalently, if we have an UOV structure, then the quadratic part of each component q_i in the central map from $\mathbf{x}$ to $\mathbf{y}$, when expressed as a symmetric matrix (cf. Eq. 13), looks like

$$M_i := \left[\begin{array}{ccc|ccc} \alpha^{(i)}_{11} & \cdots & \alpha^{(i)}_{1v} & \alpha^{(i)}_{1,v+1,} & \cdots & \alpha^{(i)}_{1n} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \alpha^{(i)}_{v1} & \cdots & \alpha^{(i)}_{vv} & \alpha^{(i)}_{v,v+1,} & \cdots & \alpha^{(i)}_{vn} \\ \hline \alpha^{(i)}_{v+1,1,} & \cdots & \alpha^{(i)}_{v+1,v,} & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \alpha^{(i)}_{n1} & \cdots & \alpha^{(i)}_{nv} & 0 & \cdots & 0 \end{array}\right] \quad \left(\text{or for short, } \left[\begin{array}{c|c} * & * \\ \hline * & 0 \end{array}\right]\right). \tag{19}$$

We should mention the fact that there are many *equivalent keys* [103]. Computing the essential part of secret keys is part of the attack of Sec. 5.5.

UOV as a Booster Stage At some point one would be bound to ask: Suppose we have an MPKC with m equations in n variables, which is a size too small for our needs. How could we reasonably make it $m+v$ equations in $n+v$ variables? Or even $m+v'$ equations in $n+v$ variables, where $v' > v$? How can we build these "booster stages"? *The answer today is: What you can do seems limited to:*

- *Do not add extra variables, that is "Plus" (Sec. 4.5), with limited use.*
- *Solve linear equations for extra variables. That is a UOV stage, like rainbow.*
- *Solve higher-degree equations. The cost is prohibitive.*
- *Use brute-force guessing. Proposed [62, 63] and promptly broken [102].*

By stacking several layers of Unbalanced Oil-Vinegar together for an easily invertible central map, we arrive at the Rainbow-type constructions [40]. To

recap (Sec. 4.4), for a u-stage Rainbow $0 < v_1 < v_2 < \cdots < v_{u+1} = n$ and

$$y_k = q_k(\mathbf{x}) = \sum_{i=1}^{v_l} \sum_{j=i}^{n} \alpha_{ij}^{(k)} x_i x_j + \sum_{i<v_{l+1}} \beta_i^{(k)} x_i, \quad \text{if } v_l < k \leq v_{l+1}. \tag{20}$$

Starting from a random choice of initial vinegar variables $x_1, \ldots, x_{v_1}$, one solve for more x_i's in sets of equations until we have all the x_i's. Note that the components of $\mathbf{y}$ in a Rainbow-type construction is typically written to have indices $v_1 + 1, \ldots, n$. In the pure Rainbow scheme, S and T and the coefficients α and β are totally randomly chosen. The essential structure of the Rainbow instance is determined by $0 < v_1 < v_2 < \cdots < v_{u+1} = n$ or more often the "Rainbow Structure Sequence" $(v_1, o_1, o_2, \ldots, o_u)$, where $o_i := v_{i+1} - v_i$.

What is the main algebraic property of an UOV stage? First and foremost is that it is of course, a special case of UOV; however, the form of

$$o_u \text{ equations of form } \left[\begin{array}{c|c} * & * \\ \hline * & 0 \end{array}\right] \text{ following } m - o_u \text{ equations of form } \left[\begin{array}{c|c} * & 0 \\ \hline 0 & 0 \end{array}\right]$$

leads to a different attack of which the reader will be appraised later in Sec. 5.5.

Aside from attacks peculiar to UOV and Rainbow, the Rainbow-type constructions also share enough characteristics of triangular schemes, that there is the need to account for rank-based attacks (Sec. 5.4), such as the two improved attacks in [9, 45]. At the moment, none of these attacks are considered essential.

Sparsity and Speed: TTS We want the central map and its inverse be fast. However, if a booster stage can only solve linear systems for $x_{v_i+1}, \ldots, x_{v_{i+1}}$ with coefficients determined by $x_1, \ldots, x_{v_i}$, i.e., be like UOV (with $o_i = v_{i+1} - v_i$) in essence, then our hands are tied. What can we do to speed this up?

1. Setting up the system to be solved takes $o_i v_i v_{i+1}$ $\mathbb{K}$-multiplications. If we make the central map sparse, one can make this a small multiple of o_i^2.
2. Solving an $o_i \times o_i$ system in $\mathbb{K}$ takes $\sim o_i^3/3$ $\mathbb{K}$-multiplications via Gaussian elimination. For small o_i, this does not get much faster. It might be faster as an inversion in an extension field of $\mathbb{K}$. A side effect is also to make a segment sparse (with any reasonable representation of $\mathbb{F}_{q^{o_i}} \equiv \mathbb{K}^{o_i}$).

TTS (Tame Transformation Signatures) are categorically Rainbow schemes with a sparse central map, even though the term TTS came first [108].

TTS instances differ widely. The earlier ones known by that name, such as [16] are close to Triangular-minus. Later they became [108, 109] much more like Rainbow with few terms. The TRMS [100] of Wang *et al* are of course also a TTS instance, although they use the larger field structure as above. Having sparse terms helps a lot: We have less to store in the private key, the private map becomes a lot quicker to execute, and even key generation is faster, since

when the central map is $\mathbb{K}^n \to \mathbb{K}^m$ with sparse terms, then we can do [108]:

$$P_{ik} = \sum_{h=0}^{m-1} \left[(\mathrm{M}_T)_{kh} \left((\mathrm{M}_S)_{hi} + \sum_{p\, x_\alpha x_\beta \text{ in } q_h} p\, ((\mathrm{M}_S)_{\alpha i}(\mathbf{c}_S)_\beta + (\mathbf{c}_S)_\alpha (\mathrm{M}_S)_{\beta i}) \right) \right]$$

$$Q_{ik} = \sum_{h=0}^{m-1} \left[(\mathrm{M}_T)_{kh} \left(\sum_{p\, x_\alpha x_\beta \text{ in } q_h} p\, (\mathrm{M}_S)_{\alpha i}(\mathrm{M}_S)_{\beta i} \right) \right]$$

$$R_{ijk} = \sum_{h=0}^{m-1} \left[(\mathrm{M}_T)_{kh} \left(\sum_{p\, x_\alpha x_\beta \text{ in } q_h} p\, ((\mathrm{M}_S)_{\alpha i}(\mathrm{M}_S)_{\beta j} + (\mathrm{M}_S)_{\alpha j}(\mathrm{M}_S)_{\beta i}) \right) \right]$$

What are the drawbacks of TTS? Since TTS (TRMS) can also be viewed as Rainbow type of signature schemes, they have all the vulnerabilities of Rainbow structures. Due to their sparsity, there also exist certain extra possibilities of linear algebra and related vulnerabilities, principally UOV-type vulnerabilities such as [42].

4.5 Plus-Minus Variations

Minus and *Plus* are simple but useful ideas, earliest mentioned by Matsumoto, Patarin and Shamir (probably found independently [85, 92]).

Minus for Big-Field Schemes: SFLASH *et al* Initially [85], several (r) equations are removed from the public keys in big-field multivariates. When inverting the public map, the legitimate users take random values for the missing variables. Minus is very suitable for signature schemes without even a performance loss, because a document need not have a unique signature.

However, for encryption this is a significant slowdown, since the missing coordinates must be guessed. To clarify a little, in theory the public map of an encryption method should injective or nearly so. If we have to guess r variables in $\mathbb{F}_q$, we effectively have q^{n+r} results, only q^n of which should represent valid ciphertexts, hence the expected number of guesses taken per decryption is q^n. Hence, decryption is slowed by that same factor of q^n.

Minus or removing some public equations makes a C^*-based system much harder to solve. SFLASH [1, 22, 84], a C^{*-} instance with $(q, n, r) = (2^7, 37, 11)$, was accepted as an European security standard for low-cost smart cards by the New European Schemes for Signatures, Integrity and Encryption [76].

However, in 2007, a method was discoverd to defeat the SFLASH family of cryptosystems [46, 47]. The key of the attack is to look at the symmetry and the invariants of the differential of the public map $\mathcal{P}$ (Sec. 5.3). If C^*-based signature schemes, it will probably need the new variant called *Projection* (Sec. 4.8).

Plus-Minus for Single-Field Schemes In the case of *Minus* as applied to triangular constructions, one need to remove instead central equations — here, the lowest-ranked ones. Actually, removing central equations in C^* works too.

Just as *Minus* can remove the equations with smallest ranks from view and remove the problem at one end of the triangle, *Plus* is the the obverse: add random central equations to the original $\mathcal{Q}$; this masks from view the high-end of the triangle. For encryption methods, this again does not affect performance much [except for a slightly larger key]; for digital signatures there is a slowdown as the extra variables again needs to be guessed. Regardless, *Plus-Minus* variations defend against attacks that are predicated on the rank of equations.

As one might well guess, *Plus-Minus* alone does not make triangular constructions safe. Indeed, [58] discuss this in detail and concludes exactly the opposite: Triangle-Plus-Minus constructions can be broken by very straightforward attacks using simple linear algebra. Some more elaborate possibilities [9, 45, 108] are discussed in the following sections.

4.6 TTM and Related Schemes: "Lock" or Repeated Triangular

PKC's based on just triangular constructions were not pursued again until a much more complex defense against rank attacks was proposed, with the tame transformation method (TTM) of Tsong-Tsieng Moh [72].

One can see that de Jonquières maps can be upper triangular as well as lower triangular. In fact, you can arrange the indices any which way you want. Moh [72] suggested a construction where the central map $\mathcal{Q}$ is given by

$$\mathcal{Q} = J_u \circ J_l \circ I(x_1, \ldots, x_n). \tag{21}$$

Here J_u is a $\mathbb{K}^m$ upper triangular de Jonquières map and J_l is a $\mathbb{K}^m$ lower triangular de Jonquières map and the linear map I is the embedding of k^n into k^m: $I(x_1, \ldots, x_n) = (x_1, \ldots, x_n, 0, 0, \ldots, 0)$. The main achievement of such a construction is that any non-trivial linear combinations of the components of $\mathcal{Q}$ is quadratic. Moh's real trick is actually in using map I. One can see that

$$J_l \circ I(x_1, \ldots, x_n) = (x_1, x_2 + g_1(x_1), \ldots, x_n + g_{n-1}(x_1, \ldots, x_{n-1}), \\ g_n(x_1, \ldots, x_n), \ldots, g_{m-1}(x_1, \ldots, x_n)),$$

which gives us the freedom to choose any g_i, $i = n, \ldots, m-1$. When decrypting, one evaluates the de Jonquières maps backwards.

The multiplitude of central polynomials of low rank present in published TTM instances [15, 72, 74] is the main source of known attacks. [74, Appendix II] gives you an idea of the polynomials of a TTM instance can look like.

A few examples of such constructions were given and a family of challenges with monetary award was set up by the US Data Security, Inc. (`www.usdsi.com`). Shortly afterwards Courtois and Goubin [58] used the MinRank method (cf. Sec. 5.4) to attack this system. MinRank is to look for non-zero matries with minimum rank in a space of matrices; it is NP-hard in general but can be easy for special cases. Despite the inventor's claim that TTM systems are very secure from all standard attacks, Goubin-Courtois did decrypt a `www.usdsi.com` TTM challenge. To maintain fairness in reporting, the author claimed this to be non-conformant to his conditions of contest. He posted a new implementation of his

scheme [15] soon thereafter. As mentioned above, other TTM instances had been published [73, 74] since, more complex but mostly resembling the earlier ones.

The idea of sequentially solvable equations (or stages) can also be used in conjunction with other ideas. Some of the more notable attempts are from L.-C. Wang, who had written about a series of schemes called "Tractable Rational Map Cryptosystems" (TRMC) versions 1–4. Names not withstanding, the versions of TRMC are actually quite distinct. We believe that TRMC v1 is essentially no different from early TTM [15] except for some "gratuitous incompatibility" in the bijection $\mathbf{x} \mapsto \mathbf{x}^2$. The central map of TRMCv2 [98] has a small random overdetermined block on one end (something like 7 variables and 11 equations) and the rest of the variables are determined in the triangular (tame) style. Versions 3 and 4 [99, 101] use a similar trick as 3IC (cf. Sec. 4.7).

Although the TTM construction is original and very intriguing, so far existing constructions of the TTM cryptosystem and related schemes do not work for public-key encryption. In fact, most of the schemes proposed are not presented in any systematic way, and no explanation is given why and how they work. We can tell you a little about why some of these fail, however, in Secs. 5.1 and 5.4.

More sophistication is needed and we suspect that to create a successful TTM-like scheme may require deep insight from algebraic geometry.

4.7 Intermediate Fields: MFE and ℓIC

In C^* and HFE, we use a big field $\mathbb{L} = \mathbb{K}^n$, or at least the number of components in the big field is close to the number of variables. In Rainbow/TTS or similar schemes, each component is as small as the base field. It stands to reason that we can use something in between, as seen below in MFE (Medium Field Encryption) and ℓIC (ℓ-Invertible Cycles) we describe below. Both these schemes also happen to share a characteristic: the use a standard Cremona transform in algebraic geometry, where $\mathbb{L}^* := \mathbb{L}\backslash\{0\}$ for some field $\mathbb{L}$:

$$(X_1,\, X_2,\, X_3) \in (\mathbb{L}^*)^3 \longmapsto (Y_1,\, Y_2,\, Y_3) := (X_1X_2,\, X_1X_3,\, X_2X_3) \in (\mathbb{L}^*)^3 \quad (22)$$

This is a bijection for any field $\mathbb{L}$, and inverts via $X_1 := \sqrt{Y_1Y_2/Y_3}$, etc.

Medium Field Encryption Let $\mathbb{L} = \mathbb{K}^k$ and define $\mathcal{Q} : \mathbb{L}^{12} \to \mathbb{L}^{15}$ as follows:

$$\begin{cases}
Y_1 = X_1 + X_5X_8 + X_6X_7 + Q_1; & \\
Y_2 = X_2 + X_9X_{12} + X_{10}X_{11} + Q_2; & \\
Y_3 = X_3 + X_1X_4 + X_2X_3 + Q_3; & \\
Y_4 = X_1X_5 + X_2X_7; & Y_5 = X_1X_6 + X_2X_8; \\
Y_6 = X_3X_5 + X_4X_7; & Y_7 = X_3X_6 + X_4X_8; \\
Y_8 = X_1X_9 + X_2X_{11}; & Y_9 = X_1X_{10} + X_2X_{12}; \\
Y_{10} = X_3X_9 + X_4X_{11}; & Y_{11} = X_3X_{10} + X_4X_{12}; \\
Y_{12} = X_5X_7 + X_2X_{11}; & Y_{13} = X_5X_{10} + X_7X_{12}; \\
Y_{14} = X_6X_9 + X_8X_{11}; & Y_{15} = X_6X_{10} + X_8X_{12}.
\end{cases} \quad (23)$$

Here each X_i and Y_i is in $\mathbb{L} = \mathbb{K}^k$. Usu. $\mathbb{K} = \mathbb{F}_{256}$. Split $X_1, X_2, X_3, Q_1, Q_2, Q_3$ into components in $\mathbb{K}^k$, such that $q'_1 = 0$, $q'_2 = (x_1)^2$ and for $i = 3 \cdots 3k$, q'_i is a more or less a random quadratic in variables $(x_1, \ldots, x_{i-1})$.

$$X_1 = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_k \end{bmatrix}, X_2 = \begin{bmatrix} x_{k+1} \\ x_{k+2} \\ \vdots \\ x_{2k} \end{bmatrix}, X_3 = \begin{bmatrix} x_{2k+1} \\ x_{2k+2} \\ \vdots \\ x_{3k} \end{bmatrix};$$

$$Q_1 = \begin{bmatrix} q'_1 \\ q'_2 \\ \vdots \\ q'_k \end{bmatrix}, Q_2 = \begin{bmatrix} q'_{k+1} \\ q'_{k+2} \\ \vdots \\ q'_{2k} \end{bmatrix}, Q_3 = \begin{bmatrix} q'_{2k+1} \\ q'_{2k+2} \\ \vdots \\ q'_{3k} \end{bmatrix}.$$

Decrypting MFE: Arrange $X_{1,\ldots,12}$ and $Y_{4,\ldots,15}$, into $\mathbb{L}^{2\times 2}$ matrices:

$$A_1 = \begin{bmatrix} X_1 & X_2 \\ X_3 & X_4 \end{bmatrix}, \quad A_2 = \begin{bmatrix} X_5 & X_6 \\ X_7 & X_8 \end{bmatrix}, \quad A_3 = \begin{bmatrix} X_9 & X_{10} \\ X_{11} & X_{12} \end{bmatrix};$$

$$A_1 A_2 = \begin{bmatrix} Y_4 & Y_5 \\ Y_6 & Y_7 \end{bmatrix}, \; A_1 A_3 = \begin{bmatrix} Y_8 & Y_9 \\ Y_{10} & Y_{11} \end{bmatrix}, A_2^T A_3 = \begin{bmatrix} Y_{12} & Y_{13} \\ Y_{14} & Y_{15} \end{bmatrix}. \tag{24}$$

The first step to inverting $\mathcal{Q}$ comes from Eq. 24 via simple linear algebra:

$$Y_4 Y_7 - Y_5 Y_6 = \det(A_1 A_2) = \det A_1 \det A_2;$$

and similarly,

$$Y_8 Y_{11} - Y_9 Y_{10} = \det A_1 \det A_3; \; Y_{12} Y_{15} - Y_{13} Y_{14} = \det A_2 \det A_3.$$

Thus, knowing $Y_4, \ldots, Y_{15}$, we can find $\det A_1$, $\det A_2$, and $\det A_3$, provided that none of them is zero (we will need to take square roots in $\mathbb{L}$). Furthermore,

$$Y_1 = X_1 + \det A_2 + Q_1, \; Y_2 = X_2 + \det A_1 + Q_2, \; Y_3 = X_3 + \det A_3 + Q_3.$$

Therefore, having found $\det A_1, \det A_2, \det A_3$, we reduce the components of Y_1, Y_2, Y_3 to a triangular form in the x_i:

$$\begin{aligned} X_1 + Q_1 &= Y_1 + \sqrt{(Y_4 Y_7 + Y_5 Y_6)(Y_8 Y_{11} + Y_9 Y_{10})(Y_{12} Y_{15} + Y_{13} Y_{14})^{-1}} \\ X_2 + Q_2 &= Y_2 + \sqrt{(Y_4 Y_7 + Y_5 Y_6)(Y_8 Y_{11} + Y_9 Y_{10})^{-1}(Y_{12} Y_{15} + Y_{13} Y_{14})} \\ X_3 + Q_3 &= Y_3 + \sqrt{(Y_4 Y_7 + Y_5 Y_6)^{-1}(Y_8 Y_{11} + Y_9 Y_{10})(Y_{12} Y_{15} + Y_{13} Y_{14})} \end{aligned}$$

then we apply a second triangular step to compute X_1, X_2, and X_3 component by component. If $X_1 \neq 0$, from $\det A_1$ we can also find X_4 and complete the inversion. [101] has details on how to handle $X_1 = 0$. Of course, cases where one of the $\det A_i$ is 0 result in a decryption failure.

The main algebraic property of MFE is the central round of three matrix products. Today, everyone knows to defend against linearization relations, and MFE did in fact achieve this when they put $A_2^T A_3$ instead of $A_2 A_3$ in the center. But it does not avoid all the problems, as you can see in Sec. 5.1.

The ℓ-invertible cycle The ℓ-invertible cycle 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:

$$\begin{aligned} \mathcal{Q} : (X_1, \ldots, X_\ell) &\mapsto (Y_1, \ldots, Y_\ell) \\ &:= (X_1X_2,\ X_2X_3, \ldots,\ X_{\ell-1}X_\ell,\ \underline{X_\ell X_1^{q^\alpha}}). \end{aligned} \tag{25}$$

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

$$\mathcal{Q}^{-1} : (Y_1,\ Y_2,\ Y_3) \in (\mathbb{L}^*)^3 \mapsto (\sqrt{Y_1Y_3/Y_2},\ \sqrt{Y_1Y_2/Y_3},\ \sqrt{Y_2Y_3/Y_1},). \tag{26}$$

This is $10\times$ faster computationally than the inverse of C^*. Aside from that, analysis of the properties of the 3IC map can be found in [43] — the 3IC and C^* maps has so much in common that the former can almost be viewed as a turbocharged version of the latter especially when looking at signature schemes.

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

$$\mathcal{Q} : (X_1, X_2) \mapsto (X_1X_2, X_1X_2^2), \quad \mathcal{Q}^{-1} : (Y_1, Y_2) \mapsto (Y_1/Y_2^2, Y_2/Y_1). \tag{27}$$

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

4.8 More on Variations and a Summary

Var.		Meaning	Slows
Plus	+	extra polynomials in the central map	Slows Signatures
Minus	-	remove central or public polynomials	Slows Encryption
Perturb	i	internal perturbation	Slows All
Project	p	Fix a central variable to be 0	Slows Signatures
Vinegar	v	extra variables that can be set arbitrarily	Slows Encryption
Sparse	s	make single-field central map sparse	*General Speedup*

Table 3. A Summary of Major Modifications in MPKCs, cf. [104]

Internally Perturbed Matsumoto-Imai can produce this variation [29]: 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

$$\mathbf{x} \mapsto \mathbf{y} = (\mathbf{x})^{q^\alpha+1} + \mathbf{f}(\mathbf{v}(\mathbf{x}))$$

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 $\mathbf{x} = (\mathbf{y}+\mathbf{b})^h$

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.

We observe that there are extraneous solutions just as in HFE. Therefore, we must manufacture some redundancy in the form of a hash segment or checksum. PMI (or MIAi as classified by [104]) looked very promising, especially since there are no unbroken $\mathcal{MQ}$-encryption-schemes with any speed at that time. However, this was broken [54] via a surprising *differential cryptanalysis* (cf. Sec. 5.3). Thus, internal perturbation is usually coupled with the *plus* variation (Sec. 3.2).

Vinegar and Projection The idea of *Vinegar* variables had been introduced earlier with UOV, and used as a defense in Quartz. The idea is to use an auxilliary variable that occupies only a small subspace of the input space (cf. Sec. 3.3). It was pointed out [39] that *Internal Perturbation* is almost exactly equal to *both Vinegar* variables *and Projection*, or fixing the input to an affine subspace. We basically set one, two or more 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.

We need to tell the reader why is *Projection* useful for us. Since (Sec. 5.3) a structural attack is always by looking for an invariant or a symmetry, we should break both. Restricting to a subspace of the original $\mathbf{w}$-space breaks a symmetry. Something like the *Minus* variant destroys an invariant. Hence the use of projection by itself prevents some attacks, such as [46,47,55]. The differential attack against C^* (and ℓIC) derivatives uses the structure of the big field $\mathbb{L}$. Hence *projection* is expected to prevent such an attack [32].

5 Standard Attacks

Solving an MPKC directly as an MQ problem instance is usually futile; the cryptanalyst usually try to attack it as an extended IP problem, or to exploit the algebraic structures to find extra relations to make the solution easier. We hope to present enough on every approach but avoid too much detail.

5.1 Linearization Equations

A Linearization Equation is a relation between the components $\mathbf{w}$ and $\mathbf{z}$ that always holds for a given set of public keys, such that when substituted with the actual values of $\mathbf{z}$ we get an affine (linear) relation between the w_i's. Each one effectively eliminates one variable from the system.

The prime example is the direct attack against C^* found by Jacques Patarin. As mentioned in Sec. 4.3, a principal algebraic property of the C^* central map (cf. Eq. 15) is Eq. 17. Given Eq. 17. and that we know

1. $L : \mathbf{x} \mapsto \mathbf{x}^{q^{2\alpha}}$ and $L' : \mathbf{y} \mapsto \mathbf{y}^{q^{\alpha}}$ are linear maps in $\mathbb{K}^n$, and

2. $\mathbf{x} * \mathbf{y}$ in $\mathbb{L} = \mathbb{K}^n$ is bilinear, i.e., there are n matrices $\bar{\mathrm{M}}_1, \ldots, \bar{\mathrm{M}}_n$ satisfying

$$\mathbf{x} *_{\mathbb{L}} \mathbf{y} = \left(\mathbf{x}^T \cdot \bar{\mathrm{M}}_1 \cdot \mathbf{y},\, \mathbf{x}^T \cdot \bar{\mathrm{M}}_2 \cdot \mathbf{y}, \ldots, \mathbf{x}^T \cdot \bar{\mathrm{M}}_n \cdot \mathbf{y}\right).$$

We find the following bilinear relations

$$\mathbf{x}^T \cdot \mathrm{M}'_i \cdot \mathbf{y} := \mathbf{x}^T \cdot \left(L^T \bar{\mathrm{M}}_i - \bar{\mathrm{M}}_i L'\right) \cdot \mathbf{y} = 0,\ \forall i = 1 \cdots n. \tag{28}$$

After we substitute $\mathbf{w} = \mathrm{M}_S^{-1}(\mathbf{x} - \mathbf{c}_S)$ and $\mathbf{z} = \mathrm{M}_T \mathbf{y} + \mathbf{c}_T$ we get (as found by Patarin [81]) for this family of cryptosystem, due to the properties of the map $\mathcal{Q}$, the cipher satisfies n equations of the following form:

$$\sum a_{ij} z_i w_j + \sum b_i z_i + \sum c_j w_j + d = 0, \tag{29}$$

which are called *Patarin relations* or *bilinear relations*. For any C^* public key, we can compute $\mathbf{z}$ from $\mathbf{w}$, and substitute enough $(\mathbf{w}, \mathbf{z})$ pairs and solve for a_{ij}, b_i, c_j, and d. A basis for the solution space gives us all the linearization relations. If we given the ciphertext, i.e., the values of z_i, these n bilinear relations will produce linear equations satisfied by components of the the plaintext $\mathbf{w}$.

In similar systems like 3IC (Sec. 4.8), for example, Linearization Equations are also present in large numbers as in $X_1 Y_2 = X_2 Y_3 = X_3 Y_1$.

In most cases including 3IC and C^*, either there are not enough linearization relations or some relations will become redundant after the substitution of the z_j, linearizations equations does not actually find all the w_i, but it narrows down the search space by enough that we are able to find w_i easily.

Unlocking via Bilinear Relations and Others Normally, the number of linearization equations has to be high enough such that the remaining variables can be guessed by brute force. It is shown in [37,38] that even when the number of linearization equations is not so large, their existence can lead to defeat.

Ding and Schmidt noted that the low-rank central polynomials — often rank 2 — in currently existing implementation schemes for the TTM cryptosystem makes it possible to extend the linearization method by Patarin [81] to attack all current TTM implementation schemes (cf. Sec. 5.1). *For the Ding-Schmidt attack, the number of linearization equations is not that high, but the "lock polynomial" that defends a TTM instance against a simple rank attack is eliminated.*

HOLEs (Higher-Order Linearization Equations) The discerning reader can figure out immediately that the linearization relation does not actually need to be linear in $\mathbf{z}$, only in $\mathbf{w}$. A Higher-Order Linearization Equation (HOLE) is a linearization relation that is higher degree in the components of $\mathbf{z}$. In particular, a SOLE (second order linearization equation) would look like

$$\sum_{i<j} a_{ijk}\, z_i z_j w_k + \sum_{i \le j} b_{ij}\, z_i z_j + \sum c_{ij}\, z_i w_j + \sum d_i\, z_i + \sum e_j\, w_j + f = 0$$

It is natural for the reader to think that this shouldn't happen very often, and it doesn't. However, the possibility that we can use such relations restricts our options when designing systems, as witness the trap that befell MFE.

Let the associated matrix of a square matrix M (replace each entry with the cofactor of that position) be M^*. Hence $(\det M)M^{-1} = M^*$, $MM^* = (\det M)1_x$, where 1_x is the identity matrix. With the same notations as Sec. 4.7, we set

$$B_1 = A_1A_2 := \begin{bmatrix} Y_4 & Y_5 \\ Y_6 & Y_7 \end{bmatrix}, \; B_2 = A_1A_3 := \begin{bmatrix} Y_8 & Y_9 \\ Y_{10} & Y_{11} \end{bmatrix}.$$

Hence $(\det B_2B_2^*)\, B_1 = A_3^{-1}A_2$, or $A_3B_2^*B_1 = (\det B_2)A_2$, or (cf. [36])

$$\begin{pmatrix} X_9 & X_{10} \\ X_{11} & X_{12} \end{pmatrix} \begin{pmatrix} Y_{11} & -Y_9 \\ -Y_{10} & Y_8 \end{pmatrix} \begin{pmatrix} Y_4 & Y_5 \\ Y_6 & Y_7 \end{pmatrix} = (Y_8Y_{11} - Y_9Y_{10}) \begin{pmatrix} X_5 & X_6 \\ X_7 & X_8 \end{pmatrix}. \tag{30}$$

There are many ways to write down other equations that are homogeneous of degree two in the Y_i's and linear in the X_i's, but [101] showed some will lead to redundant equations. A set sure to lead to independent linear relations is

$$\begin{pmatrix} X_5 & X_6 \\ X_7 & X_8 \end{pmatrix} \begin{pmatrix} Y_{15} & -Y_{14} \\ -Y_{13} & Y_{12} \end{pmatrix} \begin{pmatrix} Y_8 & Y_{10} \\ Y_9 & Y_{11} \end{pmatrix} = (Y_{12}Y_{15} - Y_{13}Y_{14}) \begin{pmatrix} X_1 & X_2 \\ X_3 & X_4 \end{pmatrix} \tag{31}$$

That's at least $8k$ linear dependencies out of $12k$ variables. A cryptanalyst's task has gotten much easier. [101] used another trick – the fact that *squaring is linear in a char-2 field* – to get it down to $2k$ remaining variables at most and concluded that solving for the remainder is easy. *The existence of linearization relations at a higher degree when the designers certainly were trying their best to avoid such shows multivariate encryption schemes design in the triangular style to be full of potholes and very difficult without a higher algebraic breakthrough.*

5.2 Lazard-Faugère System Solvers

To mount a direct attack, we try to solve the m equations $\mathcal{P}(\mathbf{w}) = \mathbf{z}$ in the n variables $w_1, \ldots w_n$. If $m \geq n$, we are (over-)determined, which is good. If $m < n$, we are underdetermined. For most cases we can't do much more than to guess at $m - n$ variables randomly and continue with $m = n$ [20].

Today, the difficulty of solving "generic" or randomly chosen systems of non-linear equations is generally conceded. However, it is hard to quantify exactly how non-generic a system is. Furthermore, many techniques of algebraic cryptanalysis requires system-solving methods at the end for more or less generic systems. So we must handle many instances of the $\mathcal{MQ}$ problem, where we want to solve the system $p_1 = p_2 = \cdots = p_m = 0$, where each p_i is a quadratic polynomial in $\mathbf{x} = (x_1, \ldots, x_n)$. Coefficients and variables are in the field $\mathbb{K} = \mathbb{F}_q$.

At the moment, the best known methods to solve equations are the descendants of Buchberger's algorithm [12] to compute a Gröbner basis, first investigated by Daniel Lazard's group [67]. Macaulay generalized Sylvester's matrix to multivariate polynomials [69]. The idea is to construct a matrix whose lines

contain the multiples of the polynomials in the original system, the columns representing a basis of monomials up to a given degree. It was observed by D. Lazard [67] that for a large enough degree, ordering the columns according to a monomial ordering and performing row reduction without column pivoting on the matrix is equivalent to Buchberger's algorithm. Reductions to 0 correspond to lines that are linearly dependent upon the previous ones and the leading term of a polynomial is given by the leftmost nonzero entry in the corresponding line.

Lazard's idea was rediscovered in 1999 by Courtois, Klimov, Patarin, and Shamir [24] as **XL**. Courtois *et al* proposed several adjuncts [19, 25, 26] to XL. One tweak called XL2 merits a mention as an easy to understand precursor to $\mathbf{F_4}$. Another of these proved to be a real improvement for $\mathbf{F_4}/\mathbf{F_5}$ as well as XL. This is FXL, where F means "fixing" (guessing at) variables.

Some time *prior* to this, J. -C. Faugère had proposed a much improved Gröbner bases algorithm called $\mathbf{F_4}$ [48]. A later version, $\mathbf{F_5}$ [49], made headlines [50] when it was used to solve HFE Challenge 1 in 2002. Commercially, $\mathbf{F_4}$ is only implemented in the computer algebra system MAGMA [17].

How to solve likely non-generic systems better is an important topic that we come back to in the last section. For the rest of this paper, we will denote the monomial $x_1^{b_1} x_2^{b_2} \cdots x_n^{b_n}$ by $\mathbf{x}^\mathbf{b}$, and its total degree $|\mathbf{b}| = b_1 + \cdots + b_n$. The set of degree-$D$-or-lower monomials is denoted $\mathcal{T} = \mathcal{T}^{(D)} = \{\mathbf{x}^\mathbf{b} : |\mathbf{b}| \le D\}$. $|\mathcal{T}|$ is the number of degree $\le D$ monomials and denoted $T^{(D)} = T$.

XL Multiply each equation p_i, $i = 1 \cdots m$ by all monomials $\mathbf{x}^\mathbf{b} \in \mathcal{T}^{(D-2)}$. Reduce as a linear system of the equations $\mathcal{R}^{(D)} = \{\mathbf{x}^\mathbf{b} p_j(\mathbf{x}) = 0 : 1 \le j \le m,\ |\mathbf{b}| \le D-2\}$, with the monomials $\mathbf{x}^\mathbf{b} \in \mathcal{T}^{(D)}$ as independent variables. Repeat with higher D until we have a solution, a contradiction, or reduce the system to a univariate equation in some variable. The number of equations and independent equations are denoted $R^{(D)} = R = |\mathcal{R}|$ and $I^{(D)} = I = \dim(\mathrm{span}\mathcal{R})$.

If we accept solutions in arbitrary extensions of $K = \mathbb{F}_q$, then $T = \binom{n+D}{D}$ regardless of q. However, most crypto applications require solutions in $\mathbb{F}_q$ only. The above expression for T then only holds for large q, since we may identify x_i^q with x_i and cut substantially the number of monomials we need to manage. This "Reduced XL" (cf. C. Diem [27]) can lead to extreme savings compared to "Original XL," e.g., if $q = 2$, then $T = \sum_{j=0}^{D} \binom{n}{j}$.

Proposition 1 ([5, 107]). *The number of monomials is* $T = [t^D] \dfrac{(1-t^q)^n}{(1-t)^{n+1}}$ *which reduces to* $\binom{n+D}{D}$ *when* q *is large. We can then find* $R = R^{(D)} = mT^{(D-2)}$.

We note that the XL of [24,25] terminates more or less reliably when $T - I \le \min(D, q-1)$, but sparse matrix computation is only possible when $T - I \le 1$ [106]. Further, Lazard-Faugère methods work for equations of any degree [6,107]. If $\deg(p_i) = d$, we will only multiply the equation p_i with monomials up to degree $D - d$ in generating $\mathcal{R}^{(D)}$. The principal result is:

Proposition 2 ([107, Theorem 7]). *If the equations p_i, with $\deg p_i := d_i$, and (*) relations $\mathcal{R}^{(D)}$ has no dependencies except the obvious ones generated by $p_i p_j = p_j p_i$ and $p_i^q = p_i$, then*

$$T - I = [t^D]\, G(t) = [t^D]\, \frac{(1-t^q)^n}{(1-t)^{n+1}} \prod_{j=1}^{m} \left(\frac{1-t^{d_j}}{1-t^{q\,d_j}} \right). \tag{32}$$

There is always a certain degree D_{XL} above which Eq. 32 and hence the underlined condition (*) above cannot continue hold if the system has a solution, because the right hand side of Eq. 32 goes nonpositive. This is $D_{XL} := \min\{D : [t^D]\ G(t) \leq 0\}$, called the degree of regularity for XL. If (*) holds for as long as possible (which means for degrees up to D_{XL}), we say that the system is **K-semi-regular** or **q-semi-regular** (cf. [5, 107]).

Diem proves [27] for char 0 fields, and conjectures for all K that (i) a generic system (no algebraic relationship betweem the coefficients) is K-semi-regular and (ii) if $(p_i)_{i=1\cdots m}$ are *not* K-semi-regular, I can only decrease from the Eq. 32 prediction. Most experts seem to believe the conjecture [27] that a *random* system behaves like a generic system with probability close to 1.

Corollary 1. *$T - I = [t^D] \left((1-t)^{-n-1} \prod_{j=1}^{m} \left(1 - t^{d_i}\right) \right)$ for generic equations if $D \leq \min(q, D_{XL}^{\infty})$, where D_{XL}^{∞} is the degree of the lowest term with a non-positive coefficient in $G(t) = \left((1-t)^{-n-1} \prod_{j=1}^{m} \left(1 - t^{d_i}\right) \right)$.*

We would note that (F)XL can only be a solver and not a true Gröbner basis method as are $\mathbf{F_4}/\mathbf{F_5}$. However, the analysis much parallels that of $\mathbf{F_4}/\mathbf{F_5}$ by Dr. Faugère et al, hence our categorical name "Lazard-Faugère" solvers.

Proposition 3 (XL with Wiedemann). *With a sparse matrix solver like the Wiedemann algorithm to solve the final matrix equation, XL has running time*

$$C_{XL} \gtrsim 3\, t\, T^2 \text{ multiplications}, \tag{33}$$

where t is the average number of terms in an equation.

Gröbner Bases and $\mathbf{F_4}/\mathbf{F_5}$ XL2 [25] is a tweak of XL as follows: Tag each equation with its maximal degree. Run an elimination on the system with monomials in degree-lex. In the remaining (row echelon form) system, multiply by each variable $x_1, x_2 \cdots$ *all remaining equations with the maximum tagged degree* and eliminate again. When we cannot eliminate all remaining monomials of the maximum degree, increment the operating degree and reallocate more memory.

XL+XL2 can be considered a primitive or inferior matrix form of $\mathbf{F_4}$ or $\mathbf{F_5}$ [3]. $\mathbf{F_4}$ inserts elimination between expansion stages, which compresses the number of rows that needs to be handled. $\mathbf{F_5}$ is a further refinement of $\mathbf{F_4}$. The set of equations is actually generated one by one (or the matrix row by row). In the process, an algebraic criterion is used to determine, ahead of an

elimination process, whether a row will be reduced to zero or not and only the meaningful rows are retained. A complication resulting from the tagging is that the elimination must be done in a strictly ordered way. This corresponds in the matrix form to no row exchanges in a Gaussian. There are two separate degrees in $\mathbf{F_4}/\mathbf{F_5}$, an apparent "operating degree" D_{F4} and a higher intrinsic degree equal to that of the equivalent XL system. For the full power of $\mathbf{F_4}$ or $\mathbf{F_5}$, auxillary algorithms such as FGLM are needed. See [48, 49] for complete details.

Proposition 4 ([5]). *If the eqs. p_i are q-semi-regular, at the operating degree*

$$D_{reg} := \min\left\{ D : [t^D]\ \frac{(1-t^q)^n}{(1-t)^n}\ \prod_{i=1}^{m}\left(\frac{1-t^{d_i}}{1-t^{qd_i}}\right) < 0 \right\}$$

both $\mathbf{F_4}$-$\mathbf{F_5}$ will terminate. Note that by specializing to a large field, we find

$$D^{\infty}_{reg} := \min\left\{ D : [t^D]\ (1-t)^{-n}\ \prod_{i=1}^{m}\left(1-t^{d_i}\right) < 0 \right\}. \tag{34}$$

If we compare this formula with Cor. 1, we see that the only difference is a substitution of n for $n+1$. In other words, we are effectively running with one fewer variable in the large field case. This explains why $\mathbf{F_4}$-$\mathbf{F_5}$ can be much faster than XL. However, the savings is smaller over small fields like $\mathbb{F}_2$, and even for large fields, removing one variable may not be enough of a savings, because the systems that we aim to solve will spawn millions of monomials (variables). Eliminating in the usual way means that we will run out of memory before time.

Proposition 5. $\mathbf{F_4}/\mathbf{F_5}$ *runs in ($\omega :=$ the "order of matrix multiplications")*

$$C_{XL} \propto c_\omega\, T^\omega \text{ multiplications.} \tag{35}$$

According to the description we received from the MAGMA project and Dr. Faugère, even though memory management is very critical, elimination is still relatively straightforward in current implementations of $\mathbf{F_4}$-$\mathbf{F_5}$, and in the process we see reasonably dense matrices, not extremely sparse ones. All said, $\mathbf{F_4}$-$\mathbf{F_5}$ *are still the most sophisticated general system-solving algorithms today. The famous complete solution of HFE challenge 1 is a run of* $\mathbf{F_5}$*, specialized and optimized for* $\mathbb{F}_2$*, which took 4 days on a 4-CPU Alpha workstation. While the HFE challenge 1 was an instance with a particularly low rank (4), it was usually argued that it should always break HFE for practical r [60].* Recently, it is disputed [41] for odd char $\mathbb{K}$. We await more developments.

5.3 Differential Attacks

Structural attack on MPKC are of two related types:

Invariants: invariants (mostly, subspaces) that can be guessed.

Symmetries: transformations that leave certain quantities unchanged and hence can be computed by a system of equations.

Of course, these two are related, given that invariants are defined according to symmetry. Previous designers sometimes neglected the importance of symmetry. In this section we present the symmetry or invariants used in the new differential attacks on the C^* family of cryptosystems as exemplified by the Differential Attacks, from the school of Stern at the Ècole Normale Superieur.

Attacking Internal Perturbations The cryptanalysis of PMI was a novelty for a technique usually associated with symmetric key cryptography, since PMI was a PKC. We use 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 [54], 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. If not, we can employ two other tricks. In the more important of the two, we observe $\mathcal{K}$ is a vector space, so $\Pr(t(\mathbf{b}+\mathbf{b}') = 0 | t(\mathbf{b}') = 0)$ will be relatively high if $\mathbf{b} \in \mathcal{K}$ and relatively low otherwise. We omit the gory details and refer the reader to [54] for the complete differential cryptanalysis.

This brilliantly executes a powerful attack. But there is apparently a surprisingly simple defense dating back to [85] (which introduced SFLASH). By using the "plus" (+) variant, i.e., *appending a random quadratics to* $\mathcal{P}$, enough false positives are generated to overwhelm the distinguishing test of [54]. The extra equations also serve as a distinguisher when there are extraneous solutions.

Again, we do not include all the details. Basically, the more "plus" equations, the less discriminating power of the abovementioned test. Based on empirical results of Ding and Gower [33], when $r = 6$, $a = 12$ should be sufficient, and $a = 14$ would be a rather conservative estimate for the amount of "plus" needed to mask the PMI structure.

The Skew Symmetric Transformation The symmetry 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 just like in Eq. 36:

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

is bilinear and symmetric in its variables $\mathbf{a}$ and $\mathbf{x}$. In the first version of this attack [47], we look at the the differential of the public map $\mathcal{P}$, and look for

so-called skew-symmetric maps with respect to this bilinear function, namely, the linear maps M such that

$$D\mathcal{P}(\mathbf{c}, M(\mathbf{w})) + D\mathcal{P}(M(\mathbf{c}), \mathbf{w}) = 0$$

The reason that this works is that the central map $\mathcal{Q}$ and the public key, which encapsulates the vital information in the central map, unfortunately has very strong symmetry in the sense that all the differentials from these maps share some common nontrivial skew-symmetric map M. Since $\mathcal{Q}(\mathbf{x}) = \mathbf{x}^{1+q^\alpha}$, its differential is

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

As pointed out in [47], the maps M skew-symmetric with respect to this $D\mathcal{Q}(\mathbf{a}, \mathbf{x})$ are precisely those induced from the multiplication by some element ζ satisfying the condition

$$\zeta^{q^\alpha} + \zeta = 0.$$

Clearly this skew-symmetry will hold if we translate it into $\mathbf{w}$-space. Further it can be seen that the skew-symmetry continues to hold even when we discard some components of $\mathcal{P}$. In terms of the public key, this means that if we write

$$D\mathcal{P}(\mathbf{c}, \mathbf{w}) := (\mathbf{c}^T H_1 \mathbf{w},\, \mathbf{c}^T H_2 \mathbf{w}, \ldots,\, \mathbf{c}^T H_m \mathbf{w})$$

and try to solve $M^T H_i + H_i M = 0$ for all $i = 1 \cdots m$ simultaneously, we should find just k-multiples of the identity if n and α are coprime, and a d-dimensional subspace in the space of linear maps if $d = \gcd(n, \alpha) > 1$.

For a randomly chosen map G, it should be expected that only trivial solutions $M = u1_n$, where $u \in \mathbb{K}$, will satisfy this condition. This means that there is a very strong condition on C^{*-} cryptosystems. This symmetry can be utilized to break C^{*-} systems for which $d = \gcd(n, \alpha) > 1$.

The Multiplicative Symmetry We call the second symmetry the multiplicative symmetry, which again comes from the differential $D\mathcal{P}(\mathbf{c}, \mathbf{w})$. Let ζ be an element in the big field $\mathbb{L}$. Then we have

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

This is also a very strong symmetry, namely it implies that if

$$\mathrm{M}_\zeta = \mathrm{M}_S^{-1} \circ (X \mapsto \zeta X) \circ \mathrm{M}_S$$

is the linear map in $\mathbb{K}^n$ corresponding to multiplication by ζ, then

$$\mathrm{span}\{\mathrm{M}_\zeta^T H_i + H_i \mathrm{M}_\zeta : i = 1 \cdots n\} = \mathrm{span}\{H_i : i = 1 \cdots n\}.$$

I.e., the space spanned by the quadratic polynomials from the central map is invariant under the skew-symmetric action as defined above.

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 [46] 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 [46] shows that it almost surely eventually recovers the missing r equations and break the scheme. The only known attempted defense is [32].

5.4 Rank Attacks

We can consider *Rank attacks* to cover the UOV attacks (next section). But here we only cover attacks that specifically targets high or low rank. Let H_i be the symmetric matrix corresponding to the quadratic part of $z_i(\mathbf{w})$. Without loss of generality, we may let the fewest number of appearances of all variables in the cross-terms of the central equations be the last variable x_n appearing s times.

High Rank Attacks Since *rank attack* often meant attacking *low rank*, some also call the High Rank attack the *Dual Rank Attack*. The High Rank Attack first appeared with [18] where Coppersmith *et al* defeated a Triangular construction.

Algorithm 1 *High Rank Attack of Goubin-Courtois and Yang-Chen [58, 108]:*

1. *Compute the differential* $\mathcal{P}(\mathbf{w}+\mathbf{c})-\mathcal{P}(\mathbf{w})-\mathcal{P}(\mathbf{c})$ *and take its* j*-th component (which is bilinear in* $\mathbf{w}$ *and* $\mathbf{c}$*) as* $\mathbf{c}^T H_j \mathbf{w}$*.* H_k *is representing the quadratic crossterms in the* k*-th polynomial of the public key. Note that the* H_i *are symmetric, so if* $\mathrm{char}\,\mathbb{K} = 2$*,* $\mathbf{x}^T H_i \mathbf{x} = 0$*. This was not made clear in [58].*

2. *Form an arbitrary linear combination* $H = \sum_i \alpha_i H_i$. *Find* $V = \ker H$.
3. *When* $\dim V \geq 1$, *set* $(\sum_j \lambda_j H_j)V = \{\mathbf{0}\}$ *and check if the solution set* $\hat{V}$ *of the* (λ_i) *form a subspace dimension* $m - s$. *Note: a matrix in* $K^{n\times n}$ *have at most* n *different eigenvalues, so at least* $1 - (n/q)$ *of the time it does.*
4. *With probability* q^{-s} *we have found a small subspace representing* x_n. *For an UOV construction, we have found* V *corresponding to constant* $x_1 \cdots x_{v_u}$.

As each trial run consists of running an elimination and some testing, we can realistically do this with $\sim \left(sn^2 + \frac{n^3}{6}\right) q^s$ *field multiplications, by taking linear combinations from only* $(s+1)$ *of the matrices* H_i *and hope not to get too unlucky. An upper bound is* $\left[mn^2 + \frac{n^3}{6} + \frac{n}{q}(m^3/3 + mn^2)\right] q^s$.

The above formulation of the high rank attack works for "plus"-modified Triangular systems; it is also easier to understand than the [18] formulation. Against UOV, we might possibly do even better on this attack with differentials [45].

MinRank Attack We first describe the Goubin-Courtois version.

Algorithm 2 *[58] Let* r *be the smallest rank in linear combinations of central equations, which without loss of generality we take to be the first central equation itself. Goubin and Courtois outline how to find the smallest ranked combination (and hence break Triangle-Plus-Minus) in expected time* $O(q^{\lceil \frac{m}{n} \rceil r} m^3)$*:*

1. *Take* $P = \sum_{i=1}^{m} \lambda_i H_i$, *an undetermined linear combination of the symmetric matrices representing the homogeneous quadratic portions of the public keys. A quadratic* $C_{ab}x_a x_b + C_{cd}x_c x_d + \cdots$ *with all indices distinct will have a corresponding symmetric matrix with kernel* $\{\mathbf{x} : 0 = x_a = x_b = x_c = x_d = \cdots\}$. *We will call this the kernel of the quadratic and use the shorthand* $\ker y_i$ *(or* $\ker_{\mathbf{x}} y_i$ *to specify what space). With* p *cross-terms with distinct indices, the rank of the matrix is* $2p$. *For example, in the scheme TTS/2′, the first equation is* $y_8 = x_8 + a_8 x_0 x_7 + b_8 x_1 x_6 + c_8 x_2 x_5 + d_8 x_3 x_4$. *Hence* $\ker_{\mathbf{x}} y_8 = \{\mathbf{x} : x_0 = \cdots = x_7 = 0\}$ *for TTS/2′.*
2. *Guess at a random* k*-tuple* $(\mathbf{w}_1, \ldots, \mathbf{w}_k)$ *of vectors in* $\mathbb{K}^n$, *where* $k = \lceil \frac{m}{n} \rceil$. *Set* $P\mathbf{w}_1 = \cdots = P\mathbf{w}_k = \mathbf{0}$ *and solve for* λ_i *via Gaussian elimination. If uniquely solvable* P *is likely the quadratic part of* y_1, *the first central equation.*
3. *Assume the matrix corresponding to* y_1 *has the minrank of* r, *then its kernel (the inverse image* $H_1^{-1}(\mathbf{0})$*) has dimension* $n - r$, *hence when we guess at* $(\mathbf{w}_1, \ldots, \mathbf{w}_k)$ *randomly, they have a probability of at least* q^{-kr} *to be all in* $H_1^{-1}(\mathbf{0})$. *This* P *is the quadratic portion of* y_1 *and the coefficients* λ_i *the row of* M_T^{-1} *(up to a factor).*

Yang and Chen have extended the effectiveness of this attack [108]. Such that if c mostly distinct kernels have the same r, we can accomplish our task in $1/c$ the time. In an exaggerated example, against UOV [9, 45], we can substitute r with $v_1 + 1$ if the latter is smaller.

MinRank Attacks on Big-Field Schemes The break of HFE challenge 1 by Faugère and Joux [50], a *direct solution* of the 80 equations in 80 variables, is not the first serious attempt on HFE.

That honor belongs rather to a rank-based attack. Kipnis and Shamir suggested [66] the idea first. The attack proceeds by moving the problem back to the extension field, where all the underlying structure can be seen. This is a very natural approach if we intend to exploit the design structure of HFE in the attack. To put it simply: the minimum rank of linear combinations of the H_i should be exactly r (as in Sec. 4.3). This is the MinRank problem [13] and is in general exponential, but can be easier if r is small.

Kipnis and Shamir later suggested to take an linear combination of the H_i and take all $(r+1)\times(r+1)$ submatrices to have determinant zero. This clearly leads to a huge assortment of equations. To solve this system, they introduce an idea which they call *relinearization*, which led to the well-known XL paper [24]. It has been argued that using a Lazard-Faugère solver on this system of equations is effective [23] and equally effective as the direct attack. Sec. 5.2 has more on equation-solving.

5.5 Distilling Oil from Vinegar and Other Attacks on UOV

To a forge a signature for a UOV scheme as in Sec. 4.4, one needs to solve the equation $\mathcal{P}(\mathbf{w}) = \mathbf{y}$. When $o = v$ as with the original Oil-and-Vinegar, this turned out to be fairly easy due to the attack by Kipnis and Shamir [65].

The basic idea here is that one treats each component $y_i = p_i(\mathbf{w})$ of the public key $\mathcal{P}$ as a bilinear form. Equivalently, take their associated symmetric matrices via the *symmetric differential* as follows:

$$Dp_i(\mathbf{w},\mathbf{c}) := p_i(\mathbf{w}+\mathbf{c}) - p_i(\mathbf{w}) - p_i(\mathbf{c}) + p_i(0) := \mathbf{c}^T H_i \mathbf{w}, \tag{36}$$

A basic fact of OV: each matrix M_i (cf. Eq. 13) is in the rough form form of $\begin{bmatrix} * & * \\ * & 0 \end{bmatrix}$ *but not the matrices H_i. This reduces a cryptanalysis to the algebraic problem of finding a basis change for a set of bilinear forms into a common form.*

The problem is interesting enough that we will sketch you one solution. Recall that $v = o = n/2 = m$. We will call the vectors $\mathbf{x}$ that have all vinegar coordinates $x_1, \ldots, x_v$ equal to zero, to be the Oil Space $\mathcal{O}$, i.e. the collection of $\mathbf{x}$-vectors looking like $\begin{bmatrix} 0 \\ * \end{bmatrix}$, and similarly a $\mathbf{x}$-vector in the Vinegar Space $\mathcal{V}$ has all oil coordinates $x_{v+1}, \ldots, x_n$ equal to zero and looks like $\begin{bmatrix} * \\ 0 \end{bmatrix}$. Clearly, if each M_i is nonsingular, we have

$$\begin{bmatrix} * & * \\ * & 0 \end{bmatrix} \begin{bmatrix} 0 \\ * \end{bmatrix} = \begin{bmatrix} * \\ 0 \end{bmatrix}, \text{ or } M_i\mathcal{O} = \mathcal{V}\ \forall i.$$

Hence, we have $\left(M_j^{-1}M_i\right)\mathcal{O} = \mathcal{O}$. It then follows that

$$\left(H_j^{-1}H_i\right)(S^{-1}\mathcal{O}) = (S^{-1}\mathcal{O}),$$

which in English states that any $H_j^{-1}H_i$ has the common invariant subspace (finding which is a known problem) of $S^{-1}\mathcal{O}$, or the oil subspace expressed in $\mathbf{w}$ coordinate form. Knowing $S^{-1}\mathcal{O}$ is sufficient to find an equivalent form for S. Later it was shown by Kipnis *et al* [64] that the same argument works if $v < o$; even if $v > o$ it can be done in time directly proportional to q^{v-o}, and hence $v-o$ cannot be too small. When there are two or three times more vinegar variables than oil variables the method appears to be secure, despite the claims of [11].

Reconciliation There is more than the Kipnis-Shamir attack to transform the public maps of an UOV scheme to the Eq. 13 Common form. We could instead [45] attempts to find a sequence of change of basis that let us invert the public map, as in an improved brute force attack.

First, no matter what M_T is, it won't change the basic shape, so we let T be the identity map for the moment. What can S be like? Suppose we pick M_S as totally random, most often (see below) it decompose to

$$\mathrm{M}_S := \begin{bmatrix} *_{v\times v} & *_{v\times o} \\ *_{o\times v} & *_{o\times o} \end{bmatrix} = \begin{bmatrix} 1_{v\times v} & *_{v\times o} \\ 0_{o\times v} & 1_{o\times o} \end{bmatrix} \begin{bmatrix} *_{v\times v} & 0_{v\times o} \\ *_{o\times v} & *_{o\times o} \end{bmatrix} \tag{37}$$

where 1 means identity matrix, 0 means just zeros and $*$ means random or anything. In fact, this decomposition always hold unless the lower-right $o \times o$ submatrix is singular. It should be clear that the $\begin{bmatrix} *_{v\times v} & 0_{v\times o} \\ *_{o\times v} & *_{o\times o} \end{bmatrix}$ portion of M_S, as a coordinate change leaves the M_i's with the same shape. I.e., if we can find the correct $\begin{bmatrix} 1_{v\times v} & *_{v\times o} \\ 0_{o\times v} & 1_{o\times o} \end{bmatrix}$ portion and perform the basis change in reverse, we will again make the resulting public map into the same form (all zeroes on the lower right) and be easily inverted. Hence, no more security at all. More about this phenomenon ("equivalent keys") in MPKCs can be found in, say, [103].

Let this essential part of M_S to be recreated be P. I.e., the linear transformation $\mathbf{w} \mapsto \mathbf{x} = P\mathbf{w}$ create all zeroes on the lower right. We can decompose this P into a product of $P := P_{v+1}P_{v+2}\cdots P_n$, where each matrix look like

$$P_n = 1_n + \left[\begin{array}{ccc|c} 0 & \cdots & 0 & a_1 \\ 0 & \cdots & 0 & a_2 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & a_v \\ \hline 0 & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & 0 \end{array}\right]; \quad P_{n-1} = 1_n + \left[\begin{array}{ccc|c|c} 0 & \cdots & 0 & a'_1 & 0 \\ 0 & \cdots & 0 & a'_2 & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & a'_v & 0 \\ \hline 0 & \cdots & 0 & 0 & 0 \\ \hline \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & 0 & 0 \end{array}\right]; \cdots$$

Indeed, the multiplication is actually commutative among the various P_i's. *Let us then start with the differential matrices H_i and simultaneously transform them to make their lower-right corner a square of 0's using exactly such P_i's.*

Algorithm 3 (UOV Reconciliation Attack) *The following is an attack on a UOV scheme with o oil and $v = n - o$ vinegar variables (which has the smaller indices):*

1. *Perform basis change $w_i := w'_i - \lambda_i w'_n$ for $i = 1 \cdots v$, $w_i = w'_i$ for $i = v+1 \cdots n$. Evaluate $\mathbf{z}$ in $\mathbf{w}'$.*
2. *Let all coefficients of $(w'_n)^2$ be zero and solve for the λ_i. We may use any method such as $\mathbf{F_4}/\mathbf{F_5}$ or FXL. There will be m equations in v unknowns.*
3. *Repeat the process to find P_{n-1}. Now we set $w'_i := w''_i - \lambda_i w''_{n-1}$ for $i = 1 \cdots v$, and set every $(w''_{n-1})^2$ and $w''_n w''_{n-1}$ term to zero (i.e., more equations in the system) after making the substitution. This time it should be faster since we solve $2m$ equations in v unknowns.*
4. *Continue in this fashion for $P_{n-2}, \ldots, P_{v+1}$ (easier, even more equations).*

In the state-of-the-art system-solving today, we can expect the complexity to be determined in solving the initial system. Hence, if $v < m$, solving m equations in v variables will be easier than m equations in n equations.

Proposition 6. *The Reconciliation Attack fails with probability $\approx \frac{1}{q-1}$.*

Proof (Sketch). Provided that lower-right $o \times o$ submatrix of M_S is non-singular, we can see that the construction of P_n will eliminate the quadratic term in the last variable. P_{n-1} will eliminate all quadratic terms in the last two variables, and so on, and each sequential construction will not disturb the structure built by the prior transformations. The number of nonsingular $k \times k$ matrices in over $\mathbb{F}_q$ is $(q^k-1)(q^k-q)(q^k-q^2)\cdots(q^k-q^{k-1})$, because the first row has 1 possibility to be zero, the second row q possibilities to be a multiple of the first, the third row q^2 possibilities to be dependent on the first two, etc., so the chance that the above attack works is roughly

$$\left(1-\frac{1}{q}\right)\left(1-\frac{1}{q^2}\right)\cdots\left(1-\frac{1}{q^k}\right) > 1-\left(\frac{1}{q}+\frac{1}{q^2}+\cdots+\frac{1}{q^k}\right) > 1-\frac{1}{q-1}.$$

Attacking Rainbow and TTS Alg. 3 is just a *unbalanced oil and vinegar attack*. Rainbow systems have multiple layers (cf. 4.4). So the symmetric matrix

M_i for the quadratic part of a Rainbow central polynomial q_i looks more like

$$M_i = \left[\begin{array}{ccc|ccc} \alpha_{11}^{(i)} & \cdots & \alpha_{1v}^{(i)} & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \alpha_{v1}^{(i)} & \cdots & \alpha_{vv}^{(i)} & 0 & \cdots & 0 \\ \hline 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & 0 \end{array}\right] \text{ if } i \le m-o; \tag{38}$$

$$= \left[\begin{array}{ccc|ccc} \alpha_{11}^{(i)} & \cdots & \alpha_{1v}^{(i)} & \alpha_{1,v+1,}^{(i)} & \cdots & \alpha_{1n}^{(i)} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \alpha_{v1}^{(i)} & \cdots & \alpha_{vv}^{(i)} & \alpha_{v,v+1,}^{(i)} & \cdots & \alpha_{vn}^{(i)} \\ \hline \alpha_{v+1,1,}^{(i)} & \cdots & \alpha_{v+1,v,}^{(i)} & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \alpha_{n1}^{(i)} & \cdots & \alpha_{nv}^{(i)} & 0 & \cdots & 0 \end{array}\right] \text{ if } i > m-o.$$

I.e., the last o equations looks like Eq. 19, but the initial $m-o$ equations only have non-zero entries in the upperleft submatrix. The attack below exploits this. Actually it applies to all final schemes with a final UOV booster stage, since we do not use in the attack the property that the first $m-o$ usually are UOV matrices themselves, i.e., has a block of zeros on the lower right.

At this point, we should no longer consider T as the identity. Let us think about what the matrix M_T does in Rainbow. At the moment that we distill the P_n portion out, $m-o$ of the new M_i's should show a zero last column. *However we don't;* M_T *mixes the* M_i*'s together so that they in fact don't – we will see most of the time only the lower right entry as zero. But if we take any* $o+1$ *of those last columns, there will be a non-trivial linear dependency. We can verify that by setting one of those columns as the linear combination as the other* o*, the resulting equations are still quadratic!*

[This idea was first mentioned by Y.-H. Hu in a private discussion.]

Algorithm 4 (Rainbow Band Separation) *The Reconciliation attack may be extended for a Rainbow scheme where the final stage has* o *oil and* $v = n-o$ *vinegar variables (which has the smaller indices):*

1. *Perform basis change* $w_i := w_i' - \lambda_i w_n'$ *for* $i = 1 \cdots v$, $w_i = w_i'$ *for* $i = v+1 \cdots n$. *Evaluate* $\mathbf{z}$ *in* $\mathbf{w}'$.
2. *Find* m *equations by setting all coefficients of* $(w_n')^2$ *to be zero; there are* v *variables in the* λ_i*'s.*
3. *Set all cross-terms involving* w_n' *in* $\mathbf{z}_1 - \sigma_1^{(1)}\mathbf{z}_{v+1} - \sigma_2^{(1)}\mathbf{z}_{v+2} - \cdots - \sigma_o^{(1)}\mathbf{z}_m$ *to be zero and find* $n-1$ *more equations. Note that* $(w_n')^2$ *terms are assumed gone already, so we can no longer get a useful equation.*
4. *Solve* $m+n-1$ *quadratic equations in* $o+v=n$ *unknowns. We may use any method (e.g.,* $\mathbf{F_4}$ *or XL).*

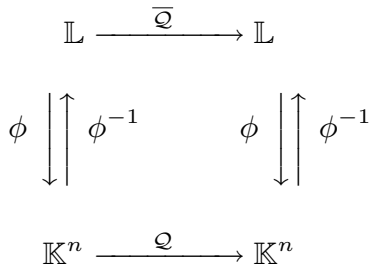

Fig. 1. Identifying maps on a $\mathbb{K}$-vector space with those on extension fields $\mathbb{L}/\mathbb{K}$.

5. *Repeat the process to find* P_{n-1}. *Now set* $w_i' := w_i'' - \lambda_i w_{n-1}''$ *for* $i = 1 \cdots v$, *and set every* $(w_{n-1}'')^2$ *and* $w_n'' w_{n-1}''$ *term to zero after making the substitution. Also set* $\mathbf{z}_2 - \sigma_1^{(2)}\mathbf{z}_{v+1} - \sigma_2^{(2)}\mathbf{z}_{v+2} - \cdots - \sigma_o^{(2)}\mathbf{z}_m$ *to have a zero second-to-last column. This time there are* $2m + n - 2$ *equations in* n *unknowns.*
6. *Continue similarly to find* $P_{n-2}, \ldots, P_{v+1}$ *(now easier with more equations).*

To repeat, **the Alg. 4 attack works for all constructions with a UOV final stage, including all Rainbow and TTS constructions.** That explains why the current proposed parameters of Rainbow [45] looks like those in Sec. 3.1.

6 The Future

In the last ten years, MPKCs have seen very active and fast developments, producing many interesting new ideas, tools and constructions in both theory and its applications. Due to the consideration of quantum computer threat and the potential of its applications in ubiquitous computing devices, we foresee that the research in MPKCs will move on to the next level in the next decade. Here, we would like present some of our thoughts on the future of the research in multivariate public key cryptography.

6.1 Construction of MPKCs

The real breakthrough of MPKCs should be attributed to the work by Matsumoto and Imai in 1988 [70], a fundamental catalyst. The new idea of Matsumoto and Imai should be called the "Big Field" construction, where we build first a map in a degree n extension field (Big Field) $\mathbb{L}$ over a small finite field $\mathbb{K}$, then move it down to a vector space over the small finite field with the identification map $\phi : \mathbb{L} \longrightarrow \mathbb{K}^n$, the standard $\mathbb{K}$-linear isomorphism between $\mathbb{L}$ and $\mathbb{K}^n$.

Great efforts are still being devoted to developing MPKCs using this idea [101], [43], [36] and [55]. This is also the idea behind the new Zhuang-Zi algorithm [34], where we lift the problem of solving a set of multivariate polynomial equations over a small finite field to solving a set of single variable equations over an extension field. Recently, a new idea of reviving HFE using field of odd characteristics was proposed [41].

What we have seen is that what really drives the development of the designs in MPKCs are indeed new mathematical ideas that bring new mathematical structures and insights in the construction of MPKCs. We believe the mathematical idea we have used are just some of the very basic ideas developed in mathematics and there is great potential in pushing this idea further using some of the more sophisticated mathematical constructions in algebraic geometry. Therefore, there is great potential to study and search for further mathematical ideas and structures that could be used to construct MPKCs. One particularly interesting problem would be to make the TTM cryptosystems work where a systematic approach should be established. This definitely demands some deep insights and the usage of some intrinsic combinatorial structures from algebraic geometry.

From the point view of practical applications, there are two critical problems that deserve more attention in designing new MPKCs. The first one is the problem of the public key size. For a MPKC with m polynomials and n variables, the public key size normally has $m(n+2)(n+1)/2$ terms, where m is at least 25 and n is at least 30. Compared with all other public key cryptosystems, for example RSA, one disadvantage is that in general a MPKC has a relatively large public key (tens of Kbytes). This is not a problem from the point view of modern computers, such as the PCs we use, but it could be a problem if we want to use it for small devices with limited memory resources. This would also be a problem if a device with limited communication abilities needs to send the public key for each transaction, for example in the case of authentication.

One idea is to do something like in [95], where a cryptosystem is built with a very small number of variables (5) but with a higher degree (4) over a much bigger base field (32 bits). In other words, we can try high degree constructions with fewer variables but over a much bigger field. In general, any new idea for how to reduce the public key size or in how to manage it in practical applications would be really appreciated.

A second idea is that of using sparse polynomials constructions. The first explicit usage of such constructions should be attributed to the works of Yang and Chen [16]. But some of the early such constructions were broken exactly because of the usage of sparse polynomials [42], which brought unexpected weakness to the system. However, we believe that the idea of using sparse polynomials is an excellent idea, especially from the point view of practical applications. From the theoretical point of view, one critical question that needs to be addressed carefully is that of whether or not the use of specific sparse polynomials has any substantial impact on the security of the given cryptosystem. The answer to this problem will help us to establish the principles for how we should choose sparse polynomials that do not affect the security of the given cryptosystem. An unexpected consequence of answering this problem is that it might also shed some light on the problem mentioned above about reducing the size of the public key.

6.2 Attack on MPKCs and Provable Security

Several major methods have been developed to attack the MPKCs. They can be roughly grouped into the following two categories.

- **Structure-based** – These attacks rely solely on the specific structures of the corresponding MPKC. Here, we may use several methods, for example, the rank attack, the invariant subspace attack, the differential attack, the extension field structure attack, the low degree inverse, and others.
- **General Attack** – This attack uses the general method of solving a set of multivariate polynomial equations, for example using the Gröbner basis method, including the Buchberger algorithm, its improvements (such as F_4 and F_5), the XL algorithm, and the new Zhuang-Zi algorithm.

Of course, we may also combine both methods to attack a specific MPKC.

It is clear that for a given multivariate cryptosystem, we should first try the general attack and then we may then look for methods that use the weaknesses of the underlying structure.

Though a lot of work has been done in analyzing the efficiency of different attacks, we still do not fully understand the full potential or the limitations of some of the attack algorithms, such as the MinRank algorithm, Gröbner basis algorithms, the XL algorithm, and the new Zhuang-Zi algorithm. For example, we still know very little about how these general attacks will work on the internal perturbation type systems such as PMI+ [33, 35], though we do have some experimental data to give us some ideas about how things work. Another interesting question is to find out exactly why and how the improved Gröbner basis algorithms like F_4 and F_5 work on HFE and its simple variants with low parameter D [49, 50]. The question is why the hidden structure of HFE can be discovered by these algorithms.

Much work is still needed to understand both the theory and practice of how efficiently general attack algorithms work and how to implement them efficiently. From the theoretical point of view, to answer these problems, the foundation again lies in modern algebraic geometry as in [27]. One critical step would be to prove the maximum rank conjecture pointed in [27], which is currently the theoretical basis used to estimate the complexity of the XL algorithm and the F_4 and F_5 algorithms for example. Another interesting problem is to mathematically prove some fo the commonly used complexity estimate formulas in [106].

One more important problem we would like to emphasize is the efficient implementation of general algorithms. Even for the same algorithm, the efficiency of various implementations can be substantially different. For example, one critical problem in implementing F_4 or F_5, or the XL type algorithms, is that the programs tend to use a large amount of memory for any nontrivial problem. Often the computation fails not because of time constraints but because the program runs out of memory. Therefore, efficient implementations of these algorithms with good memory management should be studied and tested carefully.

Chen, Yang, and Chen [110] developed a new XL implementation with a Wiedemann solver that is probably as close to optimal as might be possible.

They showed that in a few cases the simple FXL algorithm can even outperform the more sophisticated F_4 and F_5 algorithms. More new ideas of improving the algorithms, such as using the concept of mutant [30,31], are also being developed. In general, any new idea or technique in implementing these algorithms efficiently could have very serious practical implications.

In order to convince industry to actually use MPKCs in practical applications, the first and the most important problem is the concern of security. Industry must be convinced that MPKCs are indeed secure. A good answer to this problem is to prove that a given MPKC is indeed secure with some reasonable theoretical assumptions; that is, we need to solve the problem of provable security of MPKCs. From this point of view, the different approaches taken in attacking MPKCs present a very serious problem in terms of provable security. Many people have spent a considerable amount of time thinking about this problem, but there are still no substantial results in this area. One possible approach should be from the point view of algebraic geometry; that is, we need to study further all the different attacks and somehow put them into one theoretical framework using some (maybe new) abstract notion. This would allow us to formulate some reasonable theoretical assumptions, which is the foundation of any type of provable security. This is likely a very hard problem.

6.3 Practical Applications

Currently, a very popular notion in the computing world is the phrase "ubiquitous computing." This phrase describes a world where computing in some form is virtually everywhere, usually in the form of some small computing device such as RFID, wireless sensors, PDA, and others. Some of these devices often have very limited computing power, batteries, memory capacity, and communication capacity. Still, because of its ever growing importance in our daily lives, the security of such a system will become an increasingly important concern. It is clear that public key cryptosystems like RSA cannot be used in these settings due to the complexity of the computations.

In some way, MPKCs may provide an alternative in this area. In particular, there are many alternative multivariate signature schemes such as Rainbow, TTS and TRMC. Recently [4, 111] it is shown that systems like TTS and Rainbow have great potential for application in small computing devices. Due to its high efficiency, a very important direction in application of MPKCs is to seek new applications where the classical public key cryptosystems like RSA cannot work satisfactorily. This will also likely be the area where MPKCs will find a real impact in practical applications.

6.4 Broad Connections

As MPKCs develops, it starts to interact more and more with other topics, one example is the algebraic attacks. Algebraic attacks are a very popular research topic in attacking symmetric block ciphers like AES [26] and stream ciphers [2] and analyzing hash functions [94]. We would like to point out that the origin of

such an idea is actually from MPKCs, and in particular Patarin's linearization equation attack method. From recent developments we see that there is a trend that the research of MPKCs will interact very closely with that in symmetric ciphers and stream ciphers. We believe some of the new ideas we have seen in MPKCs will have much more broad applications in the area of algebraic attacks. The idea of multivariate construction was also applied to the symmetric constructions. Recently, new methods had been proposed to build secure hash functions using random quadratic maps [44] [10]. These constructions are very simple and therefore easy to study. They may also have very good property in terms of provable security. Similar ideas may have further applications in desigining stream ciphers and block ciphers. We foresee that the theory of functions on a space over a finite field (multivariate functions) will play an increasingly important role in the unification of the research in all these related areas.

It is evident that the research in MPKCs has already presented new mathematical challenges that demand new mathematical tools and ideas. In the future, we expect to see a mutually beneficial interaction between MPKCs and algebraic geometry to grow rapidly. We further believe that MPKCs will provide excellent motivation and critical problems in the development of the theory of functions over finite fields. There is no doubt that the area of MPKC will welcome the new mathematical tools and insights that will be critical for its future development.

References

1. M.-L. Akkar, N. T. Courtois, R. Duteuil, and L. Goubin. A fast and secure implementation of SFLASH. In PKC [105], pages 267–278.
2. F. Armknecht and M. Krause. Algrebraic attacks on combiners with memory. In *Crypto 2003, August 17-21, Santa Barbara, CA, USA*, volume 2729 of *LNCS*, pages 162–176. Springer, 2003.
3. G. Ars, J.-C. Faugère, H. Imai, M. Kawazoe, and M. Sugita. Comparison between XL and Gröbner Basis algorithms. In AsiaCrypt [88], pages 338–353.
4. S. Balasubramanian, A. Bogdanov, A. Rupp, J. Ding, and H. W. Carter. Fast multivariate signature generation in hardware: The case of rainbow. Poster Session, FCCM 2008.
5. M. Bardet, J.-C. Faugère, and B. Salvy. On the complexity of Gröbner basis computation of semi-regular overdetermined algebraic equations. In *Proceedings of the International Conference on Polynomial System Solving*, pages 71–74, 2004. Previously INRIA report RR-5049.
6. M. Bardet, J.-C. Faugère, B. Salvy, and B.-Y. Yang. Asymptotic expansion of the degree of regularity for semi-regular systems of equations. In P. Gianni, editor, *MEGA 2005 Sardinia (Italy)*, 2005.
7. C. Berbain, O. Billet, and H. Gilbert. Efficient implementations of multivariate quadratic systems. In E. Biham and A. M. Youssef, editors, *Selected Areas in Cryptography*, volume 4356 of *Lecture Notes in Computer Science*, pages 174–187. Springer, 2007.
8. E. R. Berlekamp. Factoring polynomials over finite fields. *Bell Systems Technical Journal*, 46:1853–1859, 1967. Republished in: Elwyn R. Berlekamp. "Algebraic Coding Theory". McGraw Hill, 1968.

9. O. Billet and H. Gilbert. Cryptanalysis of rainbow. In *Security and Cryptography for Networks*, volume 4116 of *LNCS*, pages 336–347. Springer, September 2006.
10. O. Billet, M. J. B. Robshaw, and T. Peyrin. On building hash functions from multivariate quadratic equations. In J. Pieprzyk, H. Ghodosi, and E. Dawson, editors, *ACISP*, volume 4586 of *Lecture Notes in Computer Science*, pages 82–95. Springer, 2007.
11. A. Braeken, C. Wolf, and B. Preneel. A study of the security of Unbalanced Oil and Vinegar signature schemes. In *The Cryptographer's Track at RSA Conference 2005*, volume 3376 of *Lecture Notes in Computer Science*, pages 29–43. Alfred J. Menezes, ed., Springer, 2005. also at `http://eprint.iacr.org/2004/222/`.
12. B. Buchberger. *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal.* PhD thesis, Innsbruck, 1965.
13. J. F. Buss, G. S. Frandsen, and J. O. Shallit. The computational complexity of some problems of linear algebra. Research Series RS-96-33, BRICS, Department of Computer Science, University of Aarhus, Sept. 1996. `http://www.brics.dk/RS/96/33/`, 39 pages.
14. D. G. Cantor and H. Zassenhaus. A new algorithm for factoring polynomials over finite fields. *Mathematics of Computation*, 36(587–592), 1981.
15. J.-M. Chen and T.-T. Moh. On the Goubin-Courtois attack on TTM. Cryptology ePrint Archive, 2001. http://eprint.iacr.org/2001/072.
16. J.-M. Chen and B.-Y. Yang. A more secure and efficacious TTS signature scheme. In J. I. Lim and D. H. Lee, editors, *ICISC*, volume 2971 of *LNCS*, pages 320–338. Springer, 2003.
17. Computational Algebra Group, University of Sydney. The MAGMA computational algebra system for algebra, number theory and geometry. http://magma. maths.usyd.edu.au/magma/, 2005.
18. D. Coppersmith, J. Stern, and S. Vaudenay. The security of the birational permutation signature schemes. *Journal of Cryptology*, 10:207–221, 1997.
19. N. Courtois. Algebraic attacks over $GF(2^k)$, application to HFE challenge 2 and SFLASH-v2. In PKC [53], pages 201–217. ISBN 3-540-21018-0.
20. N. Courtois, L. Goubin, W. Meier, and J.-D. Tacier. Solving underdefined systems of multivariate quadratic equations. In *Public Key Cryptography — PKC 2002*, volume 2274 of *Lecture Notes in Computer Science*, pages 211–227. David Naccache and Pascal Paillier, editors, Springer, 2002.
21. N. Courtois, L. Goubin, and J. Patarin. *Quartz: Primitive specification (second revised version)*, Oct. 2001. `https://www.cosic.esat.kuleuven.be/nessie` Submissions, Quartz, 18 pages.
22. N. Courtois, L. Goubin, and J. Patarin. *SFLASH: Primitive specification (second revised version)*, 2002. `https://www.cosic.esat.kuleuven.be/nessie`, Submissions, Sflash, 11 pages.
23. N. T. Courtois, M. Daum, and P. Felke. On the security of HFE, HFEv- and Quartz. In PKC [105], pages 337–350. `http://eprint.iacr.org/2002/138`.
24. N. T. Courtois, A. Klimov, J. Patarin, and A. Shamir. Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In *Advances in Cryptology — EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 392–407. Bart Preneel, ed., Springer, 2000. Extended Version: `http://www.minrank.org/xlfull.pdf`.

25. N. T. Courtois and J. Patarin. About the XL algorithm over gf(2). In *The Cryptographer's Track at RSA Conference 2003*, volume 2612 of *Lecture Notes in Computer Science*, pages 141–157. Springer, 2003.
26. N. T. Courtois and J. Pieprzyk. Cryptanalysis of block ciphers with overdefined systems of equations. In *Advances in Cryptology — ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 267–287. Yuliang Zheng, ed., Springer, 2002.
27. C. Diem. The XL-algorithm and a conjecture from commutative algebra. In AsiaCrypt [88], pages 323–337. ISBN 3-540-23975-8.
28. W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, Nov. 1976.
29. J. Ding. A new variant of the Matsumoto-Imai cryptosystem through perturbation. In PKC [53], pages 305–318.
30. J. Ding, J. Buchmann, M. S. E. Mohamed, W. S. A. E. Mohamed, and R.-P. Weinmann. Mutant xl. accepted for the First International Conference on Symbolic Computation and Cryptography, SCC 2008.
31. J. Ding, D. Carbarcas, D. Schmidt, J. Buchmann, and S. Tohaneanu. Mutant groebner basis algorithms. accepted for the First International Conference on Symbolic Computation and Cryptography, SCC 2008.
32. J. Ding, V. Dubois, B.-Y. Yang, C.-H. O. Chen, and C.-M. Cheng. Could SFLASH be repaired? In L. Aceto, I. Damgard, L. A. Goldberg, M. M. Halldórsson, A. Ingólfsdóttir, and I. Walukiewicz, editors, *ICALP (2)*, volume 5126 of *Lecture Notes in Computer Science*, pages 691–701. Springer, 2008. E-Print 2007/366.
33. J. Ding and J. Gower. Inoculating multivariate schemes against differential attacks. In *PKC*, volume 3958 of *LNCS*. Springer, April 2006. Also available at `http://eprint.iacr.org/2005/255`.
34. J. Ding, J. Gower, and D. Schmidt. Zhuang-Zi: A new algorithm for solving multivariate polynomial equations over a finite field. Cryptology ePrint Archive, Report 2006/038, 11th March 2006. `http://eprint.iacr.org/`, 6 pages.
35. J. Ding, J. E. Gower, D. Schmidt, C. Wolf, and Z. Yin. Complexity estimates for the F_4 attack on the perturbed Matsumoto-Imai cryptosystem. In *CCC*, volume 3796 of *LNCS*, pages 262–277. Springer, 2005.
36. J. Ding, L. Hu, X. Nie, J. Li, and J. Wagner. High order linearization equation (hole) attack on multivariate public key cryptosystems. In *PKC*, volume 4450 of *LNCS*, pages 230–247. Springer, April 2007.
37. J. Ding and D. Schmidt. A common defect of the TTM cryptosystem. In *Proceedings of the technical track of the ACNS'03, ICISA Press*, pages 68–78, 2003. http://eprint.iacr.org/2003/085.
38. J. Ding and D. Schmidt. The new TTM implementation is not secure. In K. Feng, H. Niederreiter, and C. Xing, editors, *Workshop on Coding Cryptography and Combinatorics, CCC2003 Huangshan (China)*, volume 23 of *Progress in Computer Science and Applied Logic*, pages 113–128. Birkhauser Verlag, 2004.
39. J. Ding and D. Schmidt. Cryptanalysis of HFEv and internal perturbation of HFE. In PKC [91], pages 288–301.
40. J. Ding and D. Schmidt. Rainbow, a new multivariable polynomial signature scheme. In *Conference on Applied Cryptography and Network Security — ACNS 2005*, volume 3531 of *Lecture Notes in Computer Science*, pages 164–175. Springer, 2005.

41. J. Ding, D. Schmidt, and F. Werner. Algebraic attack on hfe revisited. In *ISC 2008*, Lecture Notes in Computer Science. Springer. to appear.
42. J. Ding, D. Schmidt, and Z. Yin. Cryptanalysis of the new tts scheme in ches 2004. *Int. J. Inf. Sec.*, 5(4):231–240, 2006.
43. J. Ding, C. Wolf, and B.-Y. Yang. ℓ-invertible cycles for multivariate quadratic public key cryptography. In *PKC*, volume 4450 of *LNCS*, pages 266–281. Springer, April 2007.
44. J. Ding and B.-Y. Yang. Multivariate polynomials for hashing. In *Inscrypt*, Lecture Notes in Computer Science. Springer, 2007. to appear, cf. `http://eprint.iacr.org/2007/137`.
45. J. Ding, B.-Y. Yang, C.-H. O. Chen, M.-S. Chen, and C.-M. Cheng. New differential-algebraic attacks and reparametrization of rainbow. In *Applied Cryptography and Network Security*, volume 5037 of *Lecture Notes in Computer Science*, pages 242–257. Springer, 2008. cf. `http://eprint.iacr.org/2008/108`.
46. V. Dubois, P.-A. Fouque, A. Shamir, and J. Stern. Practical cryptanalysis of SFLASH. In *Advances in Cryptology — CRYPTO 2007*, volume 4622 of *Lecture Notes in Computer Science*, pages 1–12. Alfred Menezes, ed., Springer, 2007.
47. V. Dubois, P.-A. Fouque, and J. Stern. Cryptanalysis of SFLASH with slightly modified parameters. In M. Naor, editor, *EUROCRYPT*, volume 4515 of *Lecture Notes in Computer Science*, pages 264–275. Springer, 2007.
48. J.-C. Faugère. A new efficient algorithm for computing Gröbner bases (F_4). *Journal of Pure and Applied Algebra*, 139:61–88, June 1999.
49. J.-C. Faugère. A new efficient algorithm for computing Gröbner bases without reduction to zero (F_5). In *International Symposium on Symbolic and Algebraic Computation — ISSAC 2002*, pages 75–83. ACM Press, July 2002.
50. J.-C. Faugère and A. Joux. Algebraic cryptanalysis of Hidden Field Equations (HFE) using Gröbner bases. In *Advances in Cryptology — CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 44–60. Dan Boneh, ed., Springer, 2003.
51. J.-C. Faugère and L. Perret. Polynomial equivalence problems: Algorithmic and theoretical aspects. In S. Vaudenay, editor, *EUROCRYPT*, volume 4004 of *Lecture Notes in Computer Science*, pages 30–47. Springer, 2006.
52. H. Fell and W. Diffie. Analysis of public key approach based on polynomial substitution. In *Advances in Cryptology — CRYPTO 1985*, volume 218 of *Lecture Notes in Computer Science*, pages 340–349. Hugh C. Williams, ed., Springer, 1985.
53. Feng Bao, Robert H. Deng, and Jianying Zhou (editors). *Public Key Cryptography — PKC 2004*, volume 2947 of *Lecture Notes in Computer Science*. Springer, 2004. ISBN 3-540-21018-0.
54. P.-A. Fouque, L. Granboulan, and J. Stern. Differential cryptanalysis for multivariate schemes. In Eurocrypt [90]. 341–353.
55. P.-A. Fouque, G. Macario-Rat, L. Perret, and J. Stern. Total break of the ℓIC-signature scheme. In *Public Key Cryptography*, pages 1–17, 2008.
56. K. O. Geddes, S. R. Czapor, and G. Labahn. *Algorithms for Computer Algebra*. Amsterdam, Netherlands: Kluwer, 1992.
57. W. Geiselmann, W. Meier, and R. Steinwandt. An attack on the Isomorphisms of Polynomials problem with one secret. Cryptology ePrint Archive, Report 2002/143, 2002. `http://eprint.iacr.org/2002/143`, version from 2002-09-20, 12 pages.

58. L. Goubin and N. T. Courtois. Cryptanalysis of the TTM cryptosystem. In *Advances in Cryptology — ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 44–57. Tatsuaki Okamoto, ed., Springer, 2000.
59. A. Gouget and J. Patarin. Probabilistic multivariate cryptography. In P. Q. Nguyen, editor, *VIETCRYPT*, volume 4341 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2006.
60. L. Granboulan, A. Joux, and J. Stern. Inverting HFE is quasipolynomial. In C. Dwork, editor, *CRYPTO*, volume 4117 of *Lecture Notes in Computer Science*, pages 345–356. Springer, 2006.
61. S. Hasegawa and T. Kaneko. An attacking method for a public key cryptosystem based on the difficulty of solving a system of non-linear equations. In *Proc. 10th Symposium on Information Theory and Its applications*, pages JA5–3, 1987.
62. M. Kasahara and R. Sakai. A construction of public-key cryptosystem based on singular simultaneous equations. In *Symposium on Cryptography and Information Security — SCIS 2004*. The Institute of Electronics, Information and Communication Engineers, Jan. 27–30 2004. 6 pages.
63. M. Kasahara and R. Sakai. A construction of public key cryptosystem for realizing ciphtertext of size 100 bit and digital signature scheme. *IEICE Trans. Fundamentals*, E87-A(1):102–109, Jan. 2004. Electronic version: `http://search.ieice.org/2004/files/e000a01.htm\#e87-a,1,102`.
64. A. Kipnis, J. Patarin, and L. Goubin. Unbalanced Oil and Vinegar signature schemes. In *Advances in Cryptology — EUROCRYPT 1999*, volume 1592 of *Lecture Notes in Computer Science*, pages 206–222. Jacques Stern, ed., Springer, 1999.
65. A. Kipnis and A. Shamir. Cryptanalysis of the oil and vinegar signature scheme. In *Advances in Cryptology — CRYPTO 1998*, volume 1462 of *Lecture Notes in Computer Science*, pages 257–266. Hugo Krawczyk, ed., Springer, 1998.
66. A. Kipnis and A. Shamir. Cryptanalysis of the HFE public key cryptosystem. In *Advances in Cryptology — CRYPTO 1999*, volume 1666 of *Lecture Notes in Computer Science*, pages 19–30. Michael Wiener, ed., Springer, 1999. `http://www.minrank.org/hfesubreg.ps` or `http://citeseer.nj.nec.com/kipnis99cryptanalysis.html`.
67. D. Lazard. Gröbner-bases, Gaussian elimination and resolution of systems of algebraic equations. In *EUROCAL 83*, volume 162 of *Lecture Notes in Computer Science*, pages 146–156. Springer, March 1983.
68. F. Levy-dit-Vehel and L. Perret. Polynomial equivalence problems and applications to multivariate cryptosystems. In *Progress in Cryptology — INDOCRYPT 2003*, volume 2904 of *Lecture Notes in Computer Science*, pages 235–251. Thomas Johansson and Subhamoy Maitra, editors, Springer, 2003.
69. F. S. Macaulay. *The algebraic theory of modular systems*, volume xxxi of *Cambridge Mathematical Library*. Cambridge University Press, 1916.
70. T. Matsumoto and H. Imai. Public quadratic polynomial-tuples for efficient signature verification and message-encryption. In *Advances in Cryptology — EUROCRYPT 1988*, volume 330 of *Lecture Notes in Computer Science*, pages 419–545. Christoph G. Günther, ed., Springer, 1988.
71. T. Matsumoto, H. Imai, H. Harashima, and H. Miyagawa. High speed signature scheme using compact public key, 1985. National Conference of system and information of the Electronic Communication Association of year Sowa 60, S9-5.
72. T. Moh. A public key system with signature and master key function. *Communications in Algebra*, 27(5):2207–2222, 1999. Electronic version: `http://citeseer/moh99public.html`.

73. T.-T. Moh. The recent attack of Nie *et al* on TTM is faulty. http://eprint.iacr.org/2006/417.
74. T.-T. Moh. Two new examples of TTM. http://eprint.iacr.org/2007/144.
75. M. Nagata. *On Automorphism Group of $K[x, y]$*, volume 5 of *Lectures on Mathematics*. Kyoto University, Kinokuniya, Tokyo, 1972.
76. NESSIE: New European Schemes for Signatures, Integrity, and Encryption. Information Society Technologies programme of the European commission (IST-1999-12324). `http://www.cryptonessie.org/`.
77. E. Okamoto and K. Nakamura. Evaluation of public key cryptosystems proposed recently. In *Proc 1986's Symposium of cryptography and information security*, volume D1, 1986.
78. H. Ong, C. Schnorr, and A. Shamir. Signatures through approximate representations by quadratic forms. In *Advances in cryptology, Crypto '83*, pages 117–131. Plenum Publ., 1984.
79. H. Ong, C. Schnorr, and A. Shamir. Efficient signature schemes based on polynomial equations. In G. R. Blakley and D. Chaum, editors, *Advances in cryptology, Crypto '84*, volume 196 of *LNCS*, pages 37–46. Springer, 1985.
80. J. Patarin. The oil and vinegar signature scheme. Dagstuhl Workshop on Cryptography, September, 1997.
81. J. Patarin. Cryptanalysis of the Matsumoto and Imai public key scheme of Eurocrypt'88. In *Advances in Cryptology — CRYPTO 1995*, volume 963 of *Lecture Notes in Computer Science*, pages 248–261. Don Coppersmith, ed., Springer, 1995.
82. J. Patarin. Asymmetric cryptography with a hidden monomial. In *Advances in Cryptology — CRYPTO 1996*, volume 1109 of *Lecture Notes in Computer Science*, pages 45–60. Neal Koblitz, ed., Springer, 1996.
83. J. Patarin. Hidden Field Equations (HFE) and Isomorphisms of Polynomials (IP): two new families of asymmetric algorithms. In *Advances in Cryptology — EUROCRYPT 1996*, volume 1070 of *Lecture Notes in Computer Science*, pages 33–48. Ueli Maurer, ed., Springer, 1996. Extended Version: `http://www.minrank.org/hfe.pdf`.
84. J. Patarin, N. Courtois, and L. Goubin. Flash, a fast multivariate signature algorithm. In C. Naccache, editor, *Progress in cryptology, CT-RSA*, volume 2020 of *LNCS*, pages 298–307. Springer, 2001.
85. J. Patarin, L. Goubin, and N. Courtois. C^*_{-+} and HM: Variations around two schemes of T. Matsumoto and H. Imai. In *Advances in Cryptology — ASIACRYPT 1998*, volume 1514 of *Lecture Notes in Computer Science*, pages 35–49. Kazuo Ohta and Dingyi Pei, editors, Springer, 1998. Extended Version: `http://citeseer.nj.nec.com/patarin98plusmn.html`.
86. J. Patarin, L. Goubin, and N. Courtois. Improved algorithms for Isomorphisms of Polynomials. In *Advances in Cryptology — EUROCRYPT 1998*, volume 1403 of *Lecture Notes in Computer Science*, pages 184–200. Kaisa Nyberg, ed., Springer, 1998. Extended Version: `http://www.minrank.org/ip6long.ps`.
87. L. Perret. A fast cryptanalysis of the isomorphism of polynomials with one secret problem. In Eurocrypt [90]. 17 pages.
88. Pil Joong Lee, ed. *Advances in Cryptology — ASIACRYPT 2004*, volume 3329 of *Lecture Notes in Computer Science*. Springer, 2004. ISBN 3-540-23975-8.
89. J. M. Pollard and C. P. Schnorr. An efficient solution of the congruence $x^2 + ky^2 = m \pmod{n}$. *IEEE Trans. Inform. Theory*, 33(5):702–709, 1987.
90. Ronald Cramer, ed. *Advances in Cryptology — EUROCRYPT 2005*, Lecture Notes in Computer Science. Springer, 2005.

91. Serge Vaudenay, ed. *Public Key Cryptography — PKC 2005*, volume 3386 of *Lecture Notes in Computer Science*. Springer, 2005. ISBN 3-540-24454-9.
92. A. Shamir. Efficient signature schemes based on birational permutations. In *Advances in Cryptology — CRYPTO 1993*, volume 773 of *Lecture Notes in Computer Science*, pages 1–12. Douglas R. Stinson, ed., Springer, 1993.
93. I. P. Shestakov and U. U. Umirbaev. The Nagata automorphism is wild. *Proc. Natl. Acad. Sci. USA*, 100:12561–12563, 2003.
94. M. Sugita, M. Kawazoe, and H. Imai. Gröbner basis based cryptanalysis of sha-1. Cryptology ePrint Archive, Report 2006/098, 2006. `http://eprint.iacr.org/`.
95. S. Tsujii, A. Fujioka, and Y. Hirayama. Generalization of the public key cryptosystem based on the difficulty of solving a system of non-linear equations. In *ICICE Transactions (A) J72-A*, volume 2, pages 390–397, 1989. English version is appended at http://eprint.iacr.org/2004/336.
96. S. Tsujii, A. Fujioka, and T. Itoh. Generalization of the public key cryptosystem based on the difficulty of solving a system of non-linear equations. In *Proc. 10th Symposium on Information Theory and Its applications*, pages JA5–3, 1987.
97. S. Tsujii, K. Kurosawa, T. Itoh, A. Fujioka, and T. Matsumoto. A public key cryptosystem based on the difficulty of solving a system of nonlinear equations. *ICICE Transactions (D) J69-D*, 12:1963–1970, 1986.
98. L.-C. Wang and F.-H. Chang. Tractable rational map cryptosystem (version 2). `http://eprint.iacr.org/2004/046`, ver. 20040221:212731.
99. L.-C. Wang and F.-H. Chang. Tractable rational map cryptosystem (version 4). `http://eprint.iacr.org/2004/046`, ver. 20060203:065450.
100. L.-C. Wang, Y.-H. Hu, F. Lai, C.-Y. Chou, and B.-Y. Yang. Tractable rational map signature. In PKC [91], pages 244–257. ISBN 3-540-24454-9.
101. L.-C. Wang, B.-Y. Yang, Y.-H. Hu, and F. Lai. A "medium-field" multivariate public-key encryption scheme. In *CT-RSA 2006*, volume 3860 of *LNCS*, pages 132–149. David Pointcheval, ed., Springer, 2006. ISBN 3-540-31033-9.
102. C. Wolf, A. Braeken, and B. Preneel. Efficient cryptanalysis of RSE(2)PKC and RSSE(2)PKC. In *Conference on Security in Communication Networks — SCN 2004*, volume 3352 of *Lecture Notes in Computer Science*, pages 294–309. Springer, Sept. 8–10 2004. Extended version: `http://eprint.iacr.org/2004/237`.
103. C. Wolf and B. Preneel. Superfluous keys in $\mathcal{M}$ultivariate $\mathcal{Q}$uadratic asymmetric systems. In PKC [91], pages 275–287. Extended version `http://eprint.iacr.org/2004/361/`.
104. C. Wolf and B. Preneel. Taxonomy of public key schemes based on the problem of multivariate quadratic equations. Cryptology ePrint Archive, Report 2005/077, 12$^{\text{th}}$ of May 2005. `http://eprint.iacr.org/2005/077/`, 64 pages.
105. Y. Desmedt, ed. *Public Key Cryptography — PKC 2003*, volume 2567 of *Lecture Notes in Computer Science*. Springer, 2002. ISBN 3-540-00324-X.
106. B.-Y. Yang and J.-M. Chen. All in the XL family: Theory and practice. In *ICISC 2004*, volume 3506 of *Lecture Notes in Computer Science*, pages 67–86. Springer, 2004.
107. B.-Y. Yang and J.-M. Chen. Theoretical analysis of XL over small fields. In *ACISP 2004*, volume 3108 of *Lecture Notes in Computer Science*, pages 277–288. Springer, 2004.
108. B.-Y. Yang and J.-M. Chen. Building secure tame-like multivariate public-key cryptosystems: The new TTS. In *ACISP 2005*, volume 3574 of *Lecture Notes in Computer Science*, pages 518–531. Springer, July 2005.

109. B.-Y. Yang, J.-M. Chen, and Y.-H. Chen. TTS: High-speed signatures on a low-cost smart card. In *CHES 2004*, volume 3156 of *Lecture Notes in Computer Science*, pages 371–385. Springer, 2004.
110. B.-Y. Yang, O. C.-H. Chen, and J.-M. Chen. The limit of XL implemented with sparse matrices. Workshop record, PQCrypto workshop, Leuven 2006. http://postquantum.cr.yp.to/pqcrypto2006record.pdf.
111. B.-Y. Yang, D. C.-M. Cheng, B.-R. Chen, and J.-M. Chen. Implementing minimized multivariate public-key cryptosystems on low-resource embedded systems. In *SPC 2006*, volume 3934 of *Lecture Notes in Computer Science*, pages 73–88. Springer, 2006.

Table of Contents