C^*_{-+} and HM: Variations around two schemes of T. Matsumoto and H. Imai

- Extended Version -

Jacques Patarin, Louis Goubin
BULL Smartcards and Terminals
68 route de Versailles - BP 45
78431 Louveciennes Cedex - France
e-mail : {J.Patarin,L.Goubin}@frlv.bull.fr

Nicolas Courtois
Modélisation et Signal
Université de Toulon et du Var - BP 132
83957 La Garde Cedex - France
e-mail : courtois@univ-tln.fr

Abstract

In [4], H. Imai and T. Matsumoto presented some new candidate trapdoor one-way permutations with a public key given as multivariate polynomials over a finite field. One of these schemes was later presented in [8] under the name C^*, and was based on the idea of hiding a monomial field equation. This scheme was broken in [9] by Jacques Patarin, due to unexpected algebraic properties. J. Patarin and L. Goubin then suggested ([10], [11], [12], [13]) some schemes to repair C^*, but this was done at the cost of slightly more complex public key or secret key computations. In part I of this paper, we will study some very simple variations of the C^* scheme, where the attack of [9] is avoided, and where the very simple secret key and public key computations are almost kept. The C^*_{-+} scheme will be one of these variations. We will design some new cryptanalysis that are efficient against some of – but not all – these variations.

Another scheme of [4], very different from C^* (despite the name), was called [C] and was based on the idea of hiding a monomial matrix equation. No cryptanalysis has been published so far for this scheme. In part II of this paper, we will show how to attack this scheme [C]. We will then study more general schemes, still using the idea of hiding matrix equations. The HM scheme will be one of these variations.

Note: This paper is the extended version of the paper with the same title published at ASIACRYPT'98.

1 Introduction

What is – at the present – the asymmetric signature algorithm with the most simple smartcard implementation (in terms of speed and RAM needed), and not broken ?
We think that it is one simple variation of the Matsumoto-Imai C^* algorithm that we will present in the part I of this paper. The C^* algorithm was presented in [4] and [8], and was broken in [9], due to unexpected algebraic properties. However, it is possible to imagine many ways of avoiding the cryptanalysis of [9].
In [10], J. Patarin suggested to use a "hidden polynomial" instead of a "hidden monomial". These "HFE" algorithms are still unbroken. However, the secret key computations in HFE schemes are sensibly more complex than in the original C^* scheme. In [11], [12] and [13], J. Patarin and L. Goubin also studied some variations, where the public equations are given in different forms (some of these schemes are also presented in [6]), but here again, in order to avoid the attacks, the secret key computations or the public key computations are generally slightly more complex than in the original C^* scheme.

We will keep a quadratic public key and the main secret key operations will still be the computation of a monomial function $f: x \mapsto x^h$ in a finite field. (The length of the elements of this finite field is much shorter than what we have in RSA, and this explains why the implementations are much more efficient.)
Some of the new variations will be broken in this paper. However, as we will see, there are still some very simple and efficient variations that we do not know how to break. These schemes are related to some problems of orthogonal polynomials (how to complete a set of orthogonal polynomials, how to eliminate some random polynomials linearly mixed with orthogonal polynomials, etc).
It can be noticed that all the very simple transformations that we will study in the case of the C^* scheme, can also be applied to the more general HFE scheme of [10] or to Dragon schemes of [11], or to the *HM* scheme. We concentrate on C^* because the secret computations of C^* are particularly efficient, and because we wanted to see if these simple ideas could be sufficient or not to enforce the security (in HFE, the analysis is more difficult since no efficient attacks are known at the present).
In part II of this paper, we will study a very different (despite the name) algorithm of [4], called $[C]$. It is based on the idea of hiding (with secret affine transformations) a monomial matrix equation. Since the multiplication of matrices is a non-commutative operation, it creates a scheme with very special features. However, as in C^* or HFE, the public key is still given as a set of multivariate polynomials on a finite field, and some of the ideas used in [9] will also be useful.
We will show how to break the original $[C]$ sheme (no cryptanalysis of this scheme was published before). We will then study some more general schemes, based on the same idea of hiding matrix equations.
Since all the unbroken algorithms presented in this paper are new and very similar to broken algorithms, we certainly do not recommend to use them for very sensible applications. However, we believe that it is nice to study them because they have very efficient implementations and because they provide a better understanding of the subtle links between the concept of asymmetric cryptosystem and the computations required for security.

Part I
Variations around C^*

2 A short description of HFE and C^*

We present a short description of the HFE and C^* schemes. See [8] (for C^*), or [10] (for HFE) for more details.

The quadratic function f

Let $K = \mathbf{F}_q$ be a finite field of cardinality q. Let $\mathbf{F}_{q^n}$ be an extension of degree n over $\mathbf{F}_q$. Let

$$f(a) = \sum_{i,j} \beta_{i,j} a^{q^{\theta_{ij}} + q^{\varphi_{ij}}} + \sum_k \alpha_k a^{q^{\xi_k}} + \mu \in \mathbf{F}_{q^n}[a]$$

be a polynomial in a over $\mathbf{F}_{q^n}$, of degree d, for integers θ_{ij}, φ_{ij} and $\xi_k \geq 0$.
Since $\mathbf{F}_{q^n}$ is isomorphic to $\mathbf{F}[x]/(g(x))$, if $g(x) \in \mathbf{F}_q[x]$ is irreducible of degre n, elements of $\mathbf{F}_{q^n}$ may be represented as n-uples over $\mathbf{F}_q$, and f may be represented by n polynomials in n variables a_1, ..., a_n over $\mathbf{F}_q$:

$$f(a_1, ..., a_n) = (f_1(a_1, ..., a_n), ..., f_n(a_1, ..., a_n)).$$

The f_i are quadratic polynomials, due to the choice of f and the fact that $a \mapsto a^q$ is a linear transformation of $\mathbf{F}_{q^n}$.

Secret affine transformation of f

Let s and t be two secret affine bijections $(\mathbf{F}_q)^n \to (\mathbf{F}_q)^n$, where $(\mathbf{F}_q)^n$ is regarded as an n-dimensional vector space over $\mathbf{F}_q$.

assigns $t(f(s(x)))$ to $x \in (\mathbf{F}_q)^n$ can be written as

$$t(f(s(x_1, ..., x_n))) = (p_1(x_1, ..., x_n), ..., p_n(x_1, ..., x_n)),$$

where the p_i are quadratic polynomials due to the choice of s, t and f.

The "basic" HFE (cf [10])

Public key: The polynomials P_i, for $i = 1, 2, ..., n$, as above.

Secret key: The function f and the two affine bijections s and t as above.

Encryption: To encrypt the n-uple $x = (x_1, ..., x_n)$, compute the ciphertext $y = (P_1(x_1, ..., x_n), ..., P_n(x_1, ..., x_n))$ (x should have redundancy, or a hash of x should also be sent).

Decryption: To decrypt y, first find all the solutions z to the equation $f(z) = t^{-1}(y)$ by solving a **monovariate** polynomial equation of degree d. This is always feasible when d is not too large (say $d \leq 1000$ for example) or when f has a special shape (as we will see below in the case of the C^* scheme). Next, compute all the $s^{-1}(z)$, and use the redundancy (or the hash of x) to find M from these.

The "basic" HFE in signature

HFE can also be used in signature, as explained in [10] (essentially, the idea is that now x will be the signature and y the hash of the message to be signed. If the equation $f(z) = t^{-1}(y)$ has no solution z, we compute another hash).

The C^* algorithm (cf [8])

C^* can be seen as a special case of the more general HFE scheme, where the function f is $f(a) = a^{1+q^\theta}$. Such a function f has some practical advantages: if K is of characteristic 2 and if $1 + q^\theta$ is coprime to $q^n - 1$, then f is a bijection, and the computation of $f^{-1}(b)$ is easy since $f^{-1}(b) = b^{h'}$, where h' is the inverse of $1 + q^\theta$ modulo $q^n - 1$.
However, the C^* scheme was broken in [9], essentially because – in the case of a C^* scheme – there always exist equations such as

$$\sum_{i,j} \gamma_{ij} x_i y_j + \sum_i \alpha_i x_i + \sum_j \beta_j y_j + \mu_0 = 0 \qquad (1)$$

from which it is possible to break the scheme (see [9]). (Here x is the cleartext (or the signature), y is the ciphertext (or the hash of the message), and γ_{ij}, α_i, β_i and μ_0 are elements of K.) Throughout this paper, we will call "equation of type (1)" any equation like (1).
In the case of HFE, no cryptanalysis has yet been found (when f is well chosen), but the secret key computations are more complex.

3 Four simple variations of C^* (and HFE)

3.1 Less public polynomials: C^*_-, HFE$_-$

The polynomials $(P_1, \ldots, P_n)$ of the "basic" HFE algorithm give y from x. However, it is possible to keep some of these polynomials secret. Let k be the number of these polynomials P_i that we do not give in the public key, so that only P_1, P_2, ..., P_{n-k} are public.

- In an encryption scheme, k must be small, because in order to recover x from y, we will compute the q^k possibilities for y, compute all the corresponding possible x, and find the good x thanks to the redundancy.

  When q is not too large, and when k is very small, for example with $k = 1$ or 2, this is clearly feasible.

P_i in order that the problem of finding a value x, whose images by P_1, ..., P_{n-k} are given values, is still intractable. A value $k = 1$, 2, or $k = \frac{n}{2}$ for example may be practical and efficient.

Note: This idea to keep some polynomials P_i secret may increase, or not, the security of some schemes. In this paper, we will study the cryptanalytic effects of this idea on the original C^* scheme. We will call C^*_- the obtained scheme ($-$ means that we have *less* public equations).

3.2 Introducing some random polynomials: C^*_+, HFE$^+$

Let P_i be the public polynomials in x_1, x_2, ..., x_n, of a "basic" HFE scheme.
We can imagine to introduce some random extra quadratic polynomials Q_i in x_1, ..., x_n, and to mix the polynomials Q_i and P_i with a secret affine bijection in the given public key. Let k be the number of these Q_i polynomials.

- In a signature scheme, k must be small, because for a given x, the probability to satisfy these extra Q_i equations is $\frac{1}{q^k}$. When m and k are small, the scheme is efficient: after about q^k tries, we will obtain a signature.
- In an encryption scheme, k may be much larger. However, the total number $k + n$ of quadratic public equations must be such that the problem of finding x from a given y is still intractable (hence $k + n$ must be $< \frac{n(n+1)}{2}$, because with $\frac{n(n+1)}{2}$ equations, the values $x_i x_j$ will be found by Gaussian reductions, and then the values x_i will be found). A value $k = 1$, 2 or $k = \frac{n}{2}$ for example may be practical and efficient.

Note 1: This idea of introducing some random polynomials may increase, or not, the security of some schemes. In this paper, we will study the cryptanalytic effects of this idea on the original C^* scheme. We will call C^*_+ the obtained scheme ($+$ means that we mixed the public equations with *additional* random equations).

Note 2: Of course, it is possible to combine the variations of sections 3.1 and 3.2. For example, it is possible to design a signature or an encryption scheme from a "basic" HFE with polynomials P_1, ..., P_n, by keeping P_n secret, introducing a random polynomial Q_n instead of P_n, and computing the public key as a secret affine transformation of P_1, ..., P_{n-1}, Q_n. In the case of a C^* scheme, we will call C^*_{-+} such algorithms.

Note 3: In this paper, we will study the cryptanalytic effects of these ideas on the original C^* scheme, but potentially, these general ideas (adding or/and eliminating some equations) can also be used in many other algorithms such as HFE of [10], or Dragon schemes of [11].

3.3 Introducing more x_i variables: C^*V, HFEV

In signature, it is easy to introduce more x_i variables. In a "basic" HFE scheme, we have $b = f(a)$, where:

$$f(a) = \sum_{i,j} \beta_{ij} a^{q^{\theta_{ij}} + q^{\varphi_{ij}}} + \sum_i \alpha_i a^{q^{\xi_i}} + \mu_0, \qquad (1)$$

where β_{ij}, α_i and μ_0 are elements of L_n.
Let $a' = (a'_1, ..., a'_k)$ be a k-uple of variables of K.
In (1), let now α_i be an element of L_n such that each of the n components of α_i in a basis is a secret random linear function of the variables a'_1, ..., a'_k.
And in (1), let now μ_0 be an element of L_n such that each one of the n components of μ_0 in a basis is a secret random quadratic function of the variables a'_1, ..., a'_k.
Then, the $n + k$ variables a_1, ..., a_n, a'_1, ..., a'_k, will be mixed in the secret affine bijection s in order to obtain the variables x_1, ..., x_{n+k}.
And, as before, $t(b_1, ..., b_n) = (y_1, ..., y_n)$, where t is a secret affine bijection.

To compute a signature, the values a'_1, ..., a'_k will simply be chosen at random. Then, the values μ_0 and α_i will be computed. Then, the monovariate equation (1) will be solved (in a) in L_n.

Note 1: The idea of this section 3.3, as before, may or may not increase the security of some schemes.

Note 2: This idea is easily applicable in a "basic" HFE scheme, because the complexity of computing f^{-1} depends mainly on the degree of f and not very much on the linear part. This idea is also applicable in a C^* scheme, but the "mixing" of the variables is less efficient in this case: it gives.a scheme that we call C^*V. These schemes with more variables are studied in the paper [5] (and not in the present paper).

3.4 Fixing some x_i variables: C^*F, HFEF

In the original public key of a C^* scheme, we can randomly fix a few values x_i. This can be done on a very small number of values x_i in signature, and it can be done on many variables in encryption.

3.5 Combining all these four ideas

All these four ideas can be combined: it gives the schemes C^*V^-, C^*F^+, C^*VF^{+-}, etc. However, the ideas 3.1 and 3.3 can be done many times in signature and only a few times in encryption. And ideas 3.2 and 3.4 can be done many times in encryption and only a few times in signature.

4 First remarks about C^*_-

First example of attack (with the birthday paradox)

For example, let us assume that we have a Matsumoto-Imai algorithm with $K = \mathbf{F}_2, n = 64, f(x) = x^{1+2^\theta}$.
So we will have 64 public polynomials $P_1, P_2, \ldots, P_{64}$.
This algorithm can be attacked as shown in [9].
Now let us assume that P_{64} is kept secret. Decryption of a message (with the secret key) is still possible as follows: we will try the two possibilities for P_{64}, so we will find exactly two solutions for the cleartext x, and thanks to redundancy in x, the right x will be found.
Is this scheme secure ? No.
To attack this scheme, we can proceed like this:

1. Find two values x and x', $x \neq x'$ such that $C(x) = C(x')$. By $C(x)$ we mean the encryption of x with the algorithm, *i.e.* the output of the polynomials $P_1, P_2, \ldots P_{63}$.

2. Let $P_{64} = \sum_{i=1}^{n}\sum_{j=1}^{n} \gamma_{ij} x_i x_j + \sum_{i=1}^{n} \mu_i x_i + \delta_0$.
   Since a Matsumoto-Imai algorithm is a permutation, we know that $P_{64}(x) \oplus P_{64}(x') = 1$. It gives us a linear relation in the coefficients γ_{ij}, μ_i.

3. Go back to 1 until we have found sufficiently many linear relations in γ_{ij}, μ_i in order to find a candidate P_{64} so that $P_1, P_2, \ldots, P_{64}$ are the components of a permutation of $\mathbf{F}_2^{64}$.

4. Attack $P_1, P_2, \ldots P_{64}$ as for a usual Matsumoto-Imai algorithm with the attack described in [9].

For Step 1, we will proceed like this:
1a. We will compute and store in a file F, say 2^{32} pairs $(x, C(x))$. (If this storage capacity is not available then some time/memory trade off are possible: we will need less storage but more time). The storage is done in a way to have easy access to x when $C(x)$ is given.
1b. Now we generate and compute 2^k new pairs $(x', C(x'))$, where k is an integer that we will fix afterwards. For each of these pairs, the probability that $\exists x \in F$ such that $C(x) = C(x')$ is about $1/2^{32}$. (This is because since the Matsumoto-Imai algorithm is a permutation, there is exactly one x,

the probability that this x is in F is about $2^{32}/2^{64} = 1/2^{32}$ because there are about 2^{32} values in F and 2^{64} possible values for x).
So the number of $x, x', x \neq x'$, so that $C(x) = C(x')$ that we will found is about $2^k/2^{32} = 2^{k-32}$.
So if we only need one such pair (x, x'), we can have $k \simeq 32$.
Here we need a few of these pairs and k will be slightly bigger than 32, but the attack will be very efficient.

An attack that does not work

In order to avoid the attack given above, one can suggest to have messages of at least 128 bits (*i.e.* $n \geq 128$ if $K = \mathbf{F}_2$), and/or to keep secret more than one bit of the output (for example to keep secret at least two polynomials if $K = \mathbf{F}_2$).
However, even with these transformations, we do not recommend to use such a scheme, since we will see in section 5 how to attack such a scheme. But, before this, we will show here an idea of attack that does not work.
In the cryptanalysis of Matsumoto-Imai scheme given in [9], the key idea is to compute all the "equations of type (1)", *i.e.* such as:

$$\sum \gamma_{ij} x_i y_j + \sum \alpha_i x_i + \sum \beta_i y_i + \delta_0 = 0 \qquad (1).$$

However, as we will see in the simulations below, we may not find enough such equations for a cryptanalysis if some public polynomials of Matsumoto and Imai are kept secret.
Nevertheless one can suggest, instead of computing these equations (1) to compute all the equations such as:

$$\sum \mu_{ijk} x_i x_j x_k + \sum \nu_{ij} x_i x_j + \sum \alpha_i x_i + \sum \beta_i y_i + \sum \gamma_{ij} x_i y_j + \delta_0 = 0. \qquad (1').$$

These equations (1') will be computed as usual, since each pair (cleartext/ciphertext) gives linear equations in the coefficients $\mu_{ijk}, \nu_{ij}, \alpha_i, \beta_i, \gamma_{ij}, \delta_0$.
So after some Gaussian reductions all the valid equations (1') will be found.
Moreover all the now secret expressions in y_i are polynomials of degree two in the x_i variables, so for each equation (1) of the original Matsumoto-Imai scheme there is one equation (1') of the scheme with less public polynomials.
So a lot of these equation (1') always exist. From these equations (1') one can think that, maybe, a cryptanalyst will be able to recover the expression of the secret (originaly public) polynomials linear in y_i.

However, this attack does not work, and the existence of these equations $(1')$ is not a problem for the scheme, as we will see below.
The vector space of the equations $(1')$ is of dimension at least $n^2 + n$, since all y_i, $1 \leq i \leq n$, and all $x_i y_j$, $1 \leq i \leq n$, $1 \leq j \leq n$, can be written as polynomials of degree 2 or 3 in the x_k variables. Moreover, the dimension is **exactly** $n^2 + n$, since if it was more than $n^2 + n$, by Gaussian reduction, we would obtain an equation $(1')$, say P, with only terms in the x_k variables. Since $P(x_1, ..., x_n) = 0$ for any $(x_1, ..., x_n)$, we would thus have $P = 0$.
It is therefore easy to see what the equations $(1')$ are: they can be seen as the expression as polynomials of degree 3 in x_k of the y_i and $x_i y_j$. As a result, they can be obtained immediately from the public key (*i.e.* from the y_j expressions). There is thus a great difference between equations (1) and $(1')$: equations (1) are true only for some very simple and very specific quadratic functions y (as in the C^* scheme), whereas equations $(1')$ always exist, even for random quadratic functions y. Therefore, the existence of these equations $(1')$ (unlike equations (1)) is not a dangerous threat for the schemes.

5 Toy simulations of C^*_{-+} with $n = 17$

We have made some toy simulations with $K = \mathbf{F}_2$ and $n = 17$ of the C^*_{-+} algorithm. (These are just "toy" simulations because here the value of n is very small. In real examples, n must be ≥ 64 if

In all these simulations, we have computed the exact number of independent equations between the 16 bits of the input x_1, ..., x_{16}, and the 16 bits of the output y_1, ..., y_{16} of the following form:

$$\sum \gamma_{ij} x_i y_j + \sum \alpha_i x_i + \sum \beta_i y_i + \delta_0 = 0 \qquad (1)$$

or

$$\sum \gamma_{ijk} x_i y_j y_k + \sum \mu_{ij} x_i y_j + \sum \nu_{ij} y_i y_j + \sum \alpha_i x_i + \sum \beta_i y_i + \delta_0 = 0 \qquad (2)$$

or

$$\sum \gamma_{ijk} x_i x_j y_k + \sum \mu_{ij} x_i y_j + \sum \nu_{ij} x_i x_j + \sum \alpha_i x_i + \sum \beta_i y_i + \delta_0 = 0 \qquad (3)$$

The obtained results are given in the tables below.
Throughout this paper, we call "equations of type (1)" (resp. 2, 3) any equation like (1), (resp. (2), (3)).

Note 1: In these tables, we have subtracted the number of independent "trivial" equations, such as $x_i^2 = x_i$, or y_i·"y_j"="y_i"·y_j, where "y_i" and "y_j" are written with their expression in the x_k variables.

Note 2: The notation $[\alpha]$ means that, when the y_k variables are given explicit values, we obtain in average α independent equations in the x_k variables.

(In the tables below, we always have $K = \mathbf{F}_2$ and $n = 17$.)

$C^*: \ f(x)$	x^3	x^5	x^9	x^{17}	x^{33}	x^{65}	x^{129}
Equations (1)	34 [16]	17 [16]	17 [16]	17 [16]	17 [16]	17 [16]	17 [16]
Equations (2)	612 [16]	340 [16]	323 [16]	340 [17]	323 [16]	374 [16]	323 [16]
Equations (3)	578 [153]	442 [153]	476 [153]	493 [153]	476 [153]	459 [153]	493 [153]

Table 1

$C^*_{-+1}: \ f(x)$	x^3	x^5	x^9	x^{17}	x^{33}	x^{65}	x^{129}
Equations (1)	17 [15]	1 [1]	1 [1]	1 [1]	1 [1]	1 [1]	1 [1]
Equations (2)	340 [15]	52 [15]	36 [15]	36 [15]	36 [15]	87 [15]	36 [15]
Equations (3)	443 [153]	307 [152]	341 [153]	358 [153]	341 [152]	324 [152]	358 [153]

Table 2

$C^*_{-+2}: \ f(x)$	x^3	x^5	x^9	x^{17}	x^{33}	x^{65}	x^{129}
Equations (1)	1 [1]	0 [0]	0 [0]	0 [0]	0 [0]	0 [0]	0 [0]
Equations (2)	54 [13]	0 [0]	0 [0]	0 [0]	0 [0]	0 [0]	0 [0]
Equations (3)	309 [151]	173 [135]	207 [151]	224 [152]	207 [150]	190 [152]	224 [153]

Table 3

$C^*_{-+3}: \ f(x)$	x^3	x^5	x^9	x^{17}	x^{33}	x^{65}	x^{129}
Equations (1)	0 [0]	0 [0]	0 [0]	0 [0]	0 [0]	0 [0]	0 [0]
Equations (2)	0 [0]	0 [0]	0 [0]	0 [0]	0 [0]	0 [0]	0 [0]
Equations (3)	176 [153]	51 [68]	74 [91]	91 [108]	74 [91]	57 [74]	91 [108]

Table 4

$C^*_{-+4}: \ f(x)$	x^3	x^5	x^9	x^{17}	x^{33}	x^{65}	x^{129}
Equations (1)	0 [0]	0 [0]	0 [0]	0 [0]	0 [0]	0 [0]	0 [0]
Equations (2)	0 [0]	0 [0]	0 [0]	0 [0]	0 [0]	0 [0]	0 [0]
Equations (3)	44 [61]	0 [17]	0 [17]	0 [17]	0 [17]	0 [17]	0 [17]

Table 5

As shown in these tables, the attacks of [9] do not work directly against C^*_{-+} if we have less public polynomials, at least if $f(x) = x^3$ is avoided and if two or more polynomials are kept secret.

Principle of the attack

We denote by P the complete public form of C^*. We suppose that the first r public equations have been removed. Let $P_{(r+1)...n}$ be the remaining part of P.
The aim of the attack is to recover the public equations $P_{1...r}$ and then to use the classical attack of [9]. Obviously, those equations can be found only modulo the vector space generated by all the public equations. In this section, we will describe an algorithm to recover these equations, with a complexity about $\mathcal{O}(q^n)$. (So this algorithm is expected to succeed when $q^r \leq 2^{40}$ for example.)

Description of the algorithm

Let Q be the *polar* form of P, defined by:

$$Q(x,t) := P(x+t) - P(x) - P(t).$$

1. We randomly choose $t \neq 0$ and $x^{(0)}$.
2. We compute $z_{(r+1)...n} := Q_{(r+1)...n}(x^{(0)}, t)$.
3. We solve the equation:
$$Q_{(r+1)...n}(x,t) = z_{(r+1)...n}$$
where x is the indeterminate. There are at least two solutions ($x^{(0)}$ and $x^{(0)}+t$) and at most 2.2^r solutions. This comes from the fact that – for a given value $z_{1...r}$ – (among 2^r possible), the equation $Q(x,t) = z$ has 0 or 2 solutions.

Proof: Suppose that x and x' are two *distinct* solutions.

$$Q(x,t) = Q(x',t) \Rightarrow P(x+t) - P(x) = P(x'+t) - P(x').$$

By definition of P, this yields:

$$t(s(x+t)^{1+2^\theta} - s(x)^{1+2^\theta}) = t(s(x'+t)^{1+2^\theta} - s(x')^{1+2^\theta}).$$

Since t is bijective, we also have:

$$s(x+t)^{1+2^\theta} - s(x)^{1+2^\theta} = s(x'+t)^{1+2^\theta} - s(x')^{1+2^\theta}.$$

If we let $\lambda = s(0)$, we can write:

$$(s(x)+s(t)+\lambda)(s(x)^{2^\theta} + s(t)^{2^\theta} + \lambda^{2^\theta}) - s(x).s(x)^{2^\theta}$$

$$= (s(x')+s(t)+\lambda)(s(x')^{2^\theta} + s(t)^{2^\theta} + \lambda^{2^\theta}) - s(x').s(x')^{2^\theta}.$$

After a few computations, we obtain:

$$(s(x'-x)+\lambda).(s(t)+\lambda)^{2^\theta} = (s(x'-x)+\lambda)^{2^\theta}.(s(t)+\lambda).$$

Since $x \neq x'$, this implies:

$$(s(x'-x)+\lambda)^{2^\theta-1} = (s(t)+\lambda)^{2^\theta-1}.$$

Finally, $a \mapsto a^{2^\theta-1}$ is bijective, because $gcd(2^\theta - 1, 2^n - 1) = 1$, so that

$$x' = x + t.$$

As a result, if x is a solution, there exists *exactly one* other solution: $x' = x + t$.

time a different choice for $t \neq 0$ and $x^{(0)}$). The average number of necessary tries is estimated to be about 2^r.

4. Suppose we have found $t \neq 0$ and $x^{(0)}$ such the equation

$$Q_{(r+1)...n}(x,t) = z_{(r+1)...n}$$

has exactly 2.2^r solutions:

$$\Big\{x^{(0)}, x^{(0)}+t, x^{(1)}, x^{(1)}+t, ..., x^{(2^r-1)}, x^{(2^r-1)}+t\Big\}.$$

Let k be an integer such that $1 \leq k \leq r$. For half of the solutions, we have $Q_k(x,t) = 0$, and for the other half, we have $Q_k(x,t) = 1$, and this remains true if we consider only the subset

$$\Big\{x^{(0)}, ..., x^{(2^r-1)}\Big\}$$

of the set of solutions. Therefore, a summation gives:

$$\sum_{\nu=0}^{2^r-1} Q_k(x^{(\nu)}, t) = 2^{r-1}.$$

This gives an equation of degree one on the $\frac{n(n-1)}{2}+1$ coefficients of Q_k (this equation is the same for all the values k, $1 \leq k \leq r$).

5. By repeating steps 1-4 $\mathcal{O}(n^2)$ times, with different choices of $(x^{(0)}, t)$, we expect to find $\frac{n(n-1)}{2} + 1 - n$ equations on the coefficients of the Q_k $(1 \leq k \leq r)$. This will give Q_1, ..., Q_r modulo the vector space generated by all the public equations.

6. The public polynomials P_k $(1 \leq k \leq r)$ are not yet completely determined: Q_k only contains terms of the form $x_i t_j + x_j t_i$ $(i \neq j)$ (which come from terms $x_i x_j$ $(i \neq j)$ of P_k), and a constant term (which is the same as the one of P_k). We can suppose that $K = \mathbf{F}_2$, and thus we can write:

$$P_k = \tilde{P}_k + \sum_i \nu_{ik} x_i \qquad (1 \leq k \leq r)$$

where $\tilde{P}_k$ is now known, but where the coefficients ν_{ik} are still to be found. We also note $P_k = \tilde{P}_k$ when $r+1 \leq k \leq n$, and $\tilde{y}_k = \tilde{P}_k(x_1, ..., x_n)$ for $1 \leq k \leq n$.

From the cryptanalysis of C^*, we know that there exist equations of the form

$$\sum_{i,j} \gamma_{ij} x_i y_j + \sum_i \alpha_i x_i + \sum_i \beta_i y_i + \delta_0 = 0.$$

i.e. (with the notations above):

$$\sum_{i,j} \gamma_{ij} x_i \tilde{y}_j + \sum_i \alpha_i x_i + \sum_i \beta_i \tilde{y}_i + \delta_0 + \sum_i \sum_{j=1}^{r} \gamma_{ij} x_i \Big(\sum_k \nu_{kj} x_k\Big) + \sum_{i=1}^{r} \beta_i \Big(\sum_k \nu_{ki} x_k\Big) = 0.$$

If we replace $\tilde{y}_j$ by $\tilde{P}_j(x_1, ..., x_n)$ and if we consider the terms of total degree 3 in the x_i, we obtain $\frac{n(n-1)(n-2)}{6}$ equations on the n^2 indeterminates γ_{ij}, which can thus be determined by gaussian reductions.

We then consider the terms of total degree 2 in x_i, and that gives $\frac{n(n-1)}{2}$ equations on the $rn+n$ indeterminates ν_{kj} and β_i.

7. Once P_1, ..., P_n are completely known, the classical attack on C^* can be applied, so that C^*_- (when r is small) is also broken. The complexity of this cryptanalysis is in $\mathcal{O}(q^r)$ plus the complexity of the cryptanalysis of the original C^* scheme.

theory about permutation polynomials, and the related notion of orthogonal systems of equations, can be found in [7], chapter 7.

7 The C^*_{--} algorithm

When $q^r \geq 2^{64}$, then the cryptanalysis given in section 6 is not efficient. The scheme is then called C^*_{--}. The C^*_{--} scheme cannot be used for encryptions any more, but this scheme is still a very efficient scheme for signatures, and its security is an open problem.

8 Cryptanalysis of C^*_+

The cryptanalysis of C^*_+ is very simple: it just works exactly as the original cryptanalysis of C^*. We will first generate all the equations

$$\sum_{i,j} \gamma_{ij} x_i y_j + \sum_i \alpha_i x_i + \sum_j \beta_j y_j + \delta_0 = 0. \qquad (1)$$

Since in C^*_+, we just have **added** some equations (and eliminated none), we will find at least as much equations (1) as in the original C^*.
Then, as explained in [9], from such equations (1) we will be able to find x from y (and thus to break the system).

Remark 1: Moreover, we will be able to eliminate the random added equations and to recover an original C^*, because an equation (1) generally comes from only the y_i of C^* (and not from the added equations). Therefore, by generating an equation (1), by writing it as $x_1(P_1(y)) + x_2(P_2(y)) + \ldots + x_n(P_n(y))$ (where P_1, ..., P_n are polynomials of degree one in y_1, ..., y_{n+k}), and by making the change of variables $y'_1 = P_1(y)$, ..., $y'_n = P_n(y)$, the variables y'_1, ..., y'_n are the outputs of an original C^* scheme.

Remark 2: However, this idea of adding an equation may be much more efficient in a scheme where no equation (1) exist (as in some HFE schemes) (or when we add **and** eliminate some equations, as we will see in C^*_{-+}).

9 Cryptanalysis of C^*_{-+}, second cryptanalysis of C^*_-

The idea

Let us consider – as an example – (A) and (B) the two following equations of type (1) on $K = \mathbf{F}_2$:

$$x_1y_1 + x_3y_4 + x_4y_4 + x_5y_2 = 0, \qquad (A)$$

$$x_3y_1 + x_4y_2 + x_5y_2 = 1. \qquad (B)$$

Then $x_3 \cdot (A) + x_1 \cdot (B)$ gives:

$$x_3y_4 + x_3x_4y_4 + x_3x_5y_2 + x_1x_4y_2 + x_1x_5y_2 = x_1. \qquad (C)$$

This equation is an equation "of type (3)", and it has no term in y_1.

Cryptanalysis of C^*_{-+1}

We will first use this idea for the cryptanalysis of C^*_{-+} when the number r of removed equations is $r = 1$.
We know that from the variables of the original C^* we have at least n independent equations of type (1).
So by multiplying these equations by one x_k, $1 \leq k \leq n$, we generate n^2 independent equations of type (3).

terms in y_1 (because we have at most $\frac{n(n+1)}{2}$ terms in $y_1 x_i x_j$ or $y_1 x_i$).
Now, when we give explicit values for y, we obtain (by Gaussian reductions on the $X_{ij} = x_i \cdot x_j$ variables) the x_i values. As a result, with the equations (3) we will be able to break C^*_{-+1}: *i.e.* to recover an x from a given y.

Remark: This attack works because (as shown in our simulations, see the tables of section 5) the number of independent equations does not decrease significantly when the y_k variables are given explicit values. Moreover, our simulations also show that in this attack (based on equations (3)), we will generally have more than $\frac{n(n-1)}{2}$ equations (3) for $r = 1$, so that the attack will work even better than expected.

Cryptanalysis of C^*_{-+r}, for $r = 2, 3$

As shown in the tables, we generally have more than $\frac{n(n-1)}{2}$ equations of type (3), so that the attack also works very well when $r = 2$ or $r = 3$, since we have more equations (3) than expected. Of course, when – after Gaussian reductions – we still have a few variables to guess, we can guess them by exhaustive search (if this number is very small).

Cryptanalysis of C^*_{-+r}, for $r \geq 4$

When $r \geq 4$, the attack given above may not work, so that we may need to generalize this attack by generating more general equations such as equations of total degree $d \geq 4$ (instead of three), and of degree one in the y_i variables.
We know that from the variables of the original C^* we have at least n independent equations of type (1). So by multiplying these equations by $d-2$ variables x_k, $1 \leq k \leq n$, we generate about $n \cdot \frac{n^{d-2}}{(d-2)!}$ independent equations of the following type:

$$\sum \gamma_{i_1 i_2 \ldots i_d} x_{i_1} x_{i_2} \ldots x_{i_{d-1}} y_d + \ldots = 0. \qquad (*)$$

By Gaussian reductions, we will obtain at least $n \cdot \frac{n^{d-2}}{(d-2)!} - r \cdot \frac{n^{d-1}}{(d-1)!}$ equations $(*)$ with no terms in y_1, y_2, ..., y_r (because we have at most $\frac{n^{d-1}}{(d-1)!}$ terms in $y_\mu x_{i_1} x_{i_2} \ldots x_{i_{d-1}}$, and r values μ such that $1 \leq \mu \leq r$).
Now, when we give explicit values for y, we obtain (by Gaussian reductions on the $X_{i_1 \ldots i_{d-1}} = x_{i_1} \ldots x_{i_{d-1}}$ variables) the x_i values if

$$n \cdot \frac{n^{d-2}}{(d-2)!} - r \cdot \frac{n^{d-1}}{(d-1)!} \geq \frac{n^{d-1}}{(d-1)!}$$

(because as shown in our simulations the number of independent equations do not dramatically decrease when we give explicit values for y), *i.e.* when $r \leq d-2$.

Complexity: The complexity of this attack is essentially the complexity of Gaussian reductions on $\mathcal{O}(n^d)$ terms. This complexity is in $\mathcal{O}(n^{\omega d})$, with $\omega = 3$ in the usual Gaussian reduction algorithms, or $\omega = 2.3755$ in the best known general purpose Gaussian reduction algorithm (see [1]). As a result, this complexity increases in $\mathcal{O}(n^{\omega r})$, *i.e.* exponentially in r.

Since our simulations show that this attack works sensibly better than described above (because we have a few more equations $(*)$), we expect that – with this attack – it is feasible to attack C^*_{-+r} when $r \leq 10$ approximately. Therefore, we think that any $r \leq 10$ is insecure. However, the complexity of the attack increases a lot when r increases. Hence, at the present, for practical applications, it is an open problem to find efficient cryptanalysis of C^*_{-+r} when $r > 10$.

Can we recover the corresponding C^*_- from C^*_{-+} ?

This is sometime feasible. For example, when we have equations of type (2) (this is generally the case only when r is very small: see the tables), then these equations generally come from y_k variables of the original C^*_-, and not from the added random quadratic equations. Therefore, by looking at the terms in factor of a monomial $x_i x_j$ in those equations (2), we will find the vector space generated by the public equations of the original C^*_- equations. (Then in such a case the C^*_{-+} algorithm can be attacked as a C^*_- algorithm.)

However, there is a technical problem with the equations of type (3): these equations are "mixed" with "trivial" equations $y_i \cdot "y_j" = "y_i" \cdot y_j$, where "$y_i$" and "$y_j$" are written with their quadratic expression in the x_k variables. And it is not clear how these "trivial" equations can be removed. This is why we have used equations (2) instead.

Part II
Schemes with a hidden matrix

10 The $[C]$ scheme

In this section, we recall the description of the $[C]$ scheme, presented by H. Imai and T. Matsumoto in [4].
Let $K = GF(2^m)$ be a *public* finite field of cardinality $q = 2^m$. The basic idea is to use the transformation $A \mapsto A^2$ of the set $\mathcal{M}_2(K)$ of the 2×2 matrices over the field K.
This transformation is not one-to-one, but it can be made bijective if we restrict ourselves to matrices with a *non-zero trace*:

Lemma 1 *Let $\mathcal{E} = \{M \in \mathcal{M}_2(K),\ tr(M) \neq 0\}$.*
Then $\Phi : \begin{cases} \mathcal{E} \to \mathcal{E} \\ A \mapsto A^2 \end{cases}$ is bijective. Moreover, for any $B \in \mathcal{E}$, we have:

$$\Phi^{-1}(B) = \frac{1}{\sqrt{tr(B)}} \cdot \Big(B + \sqrt{det(B)} \cdot I\Big),$$

where $\sqrt{\ }$ denotes the inverse of the bijective *function $\begin{cases} GF(2^m) \to GF(2^m) \\ \lambda \mapsto \lambda^2 \end{cases}$.*

Proof: Let $B \in \mathcal{E}$.

- From Cayley-Hamilton theorem (applied to B), we have:

  $$B^2 - (\mathrm{tr}\ B) \cdot B + (\det\ B) \cdot I = 0$$

  and thus $(B + \sqrt{\det(B)} \cdot I)^2 = (\mathrm{tr}\ B) \cdot B$. Since $\mathrm{tr}(B) \neq 0$, we obtain $B + \sqrt{\det(B)} \cdot I \neq 0$.

- Suppose that a matrix A exists such that $A^2 = B$. By applying Cayley-Hamilton theorem to A, we have:

  $$A^2 - (\mathrm{tr}\ A) \cdot A + (\det\ A) \cdot I = 0$$

  i.e.

  $$(\mathrm{tr}\ A) \cdot A = B + \sqrt{\det(B)} \cdot I.$$

  As a result, we have $\mathrm{tr}(A) \neq 0$ and $A = \lambda \cdot (B + \sqrt{\det(B)} \cdot I)$, which gives easily:

  $$A = \frac{1}{\sqrt{\mathrm{tr}(B)}} \cdot \Big(B + \sqrt{\det(B)} \cdot I\Big).$$

  Reciprocally, this A is a satisfies $A^2 = B$ and $\mathrm{tr}(A) \neq 0$.

- In conclusion, Φ is a bijective transformation of $\mathcal{E}$, and:

  $$\Phi^{-1}(B) = \frac{1}{\sqrt{\mathrm{tr}(B)}} \cdot \Big(B + \sqrt{\det(B)} \cdot I\Big).$$

We now describe the $[C]$ scheme used in encryption mode.

The set $\mathcal{M}_2(K)$ can be considered as a vector space of dimension 4 over K. Therefore, we can choose $s : K^4 \to \mathcal{M}_2(K)$ and $t : \mathcal{M}_2(K) \to K^4$ two *secret* linear bijections such that:

- s maps the hyperplane $\{x_1 = 0\}$ of K^4 onto the hyperplane $\{\mathrm{tr}(M) = 0\}$ of $\mathcal{M}_2(K)$;
- t maps the hyperplane $\{\mathrm{tr}(M) = 0\}$ of $\mathcal{M}_2(K)$ onto the hyperplane $\{x_1 = 0\}$ of K^4.

Representation of the messages

Each message M is represented by a 4-uple $(x_1, x_2, x_3, x_4) \in K^4$ such that $x_1 \neq 0$. The message space is $\mathcal{M} = \{(x_1, x_2, x_3, x_4) \in K^4,\ x_1 \neq 0\}$.

The quadratic function f

We then define the following quadratic function on the message space:

$$f : \begin{cases} \mathcal{M} \to \mathcal{M} \\ x \mapsto t(s(x)^2) \end{cases}.$$

The hypotheses made on s and t, together with lemma 1, show that the function f is a bijection.

Public key: The 4-uple (p_1, p_2, p_3, p_4) of 4-variate quadratic polynomials over K that represent f. They are defined by:

$$f(x_1, x_2, x_3, x_4) = (p_1(x_1, x_2, x_3, x_4), p_2(x_1, x_2, x_3, x_4), p_3(x_1, x_2, x_3, x_4), p_4(x_1, x_2, x_3, x_4)).$$

Secret key: The two linear bijections s and t.

Note: For s (respectively t), there are $|\mathrm{GL}_3(K)| \cdot |K^3| \cdot |K^*| = (q^3-1)(q^3-q)(q^3-q^2)q^3(q-1) \simeq q^{13}$ possibilities, instead of $|\mathrm{GL}_4(K)| = (q^4-1)(q^4-q)(q^4-q^2)(q^4-q^3) \simeq q^{16}$ if we remove the condition "s maps $\mathcal{M}$ onto $\mathcal{E}$" (respectively "t maps $\mathcal{E}$ onto $\mathcal{M}$").

Encryption

To encrypt the message M represented by $x = (x_1, x_2, x_3, x_4) \in \mathcal{M}$, compute the ciphertext $y = (y_1, y_2, y_3, y_4)$ with the following formulas:

$$\begin{cases} y_1 = p_1(x_1, x_2, x_3, x_4) \\ y_2 = p_2(x_1, x_2, x_3, x_4) \\ y_3 = p_3(x_1, x_2, x_3, x_4) \\ y_4 = p_4(x_1, x_2, x_3, x_4) \end{cases}$$

Decryption

To decrypt the ciphertext $y \in \mathcal{M}$, compute:

$$x = s^{-1}\left(\frac{1}{\sqrt{\mathrm{tr}(t^{-1}(y))}} \cdot \left(t^{-1}(y) + \sqrt{\det(t^{-1}(y))} \cdot I\right)\right).$$

11 First cryptanalysis of $[C]$

The security of the cryptosystem is based on the difficulty of solving the following system of 4 quadratic equations in 4 variables over $K = GF(2^m)$:

$$\begin{cases} y_1 = p_1(x_1, x_2, x_3, x_4) \\ y_2 = p_2(x_1, x_2, x_3, x_4) \\ y_3 = p_3(x_1, x_2, x_3, x_4) \\ y_4 = p_4(x_1, x_2, x_3, x_4) \end{cases}$$

bases. At the present, the best implementations of Gröbner bases can solve any set of n quadratic equations with n variables over any reasonable field K, when $n \leq 16$ approximately (cf [2]). Therefore, the original $[C]$ is not secure.
This first cryptanalysis shows that the parameter n must not be too small if we want to avoid attacks based on algebraic methods for solving systems of multivariate polynomial equations. That is why we are going to describe a generalization of the scheme to higher dimensions (for which Gröbner bases algorithms will be unefficient) in the next section.

12 The more general $[C_n]$ scheme

We present here a generalization of the $[C]$ scheme of H. Imai and T. Matsumoto, which involves $n \times n$ matrices over the field K, instead of 2×2 matrices. This cryptosystem will be called $[C_n]$.
As in the case of $[C]$, we take a public finite field $K = GF(2^m)$ of cardinality $q = 2^m$.
The basic idea is still to use the transformation $A \mapsto A^2$ of the set $\mathcal{M}_n(K)$ of the $n \times n$ matrices over the field K.

The set $\mathcal{M}_n(K)$ can be considered as a vector space of dimension n^2 over K, so that we can choose $s : K^{n^2} \to \mathcal{M}_n(K)$ and $t : \mathcal{M}_n(K) \to K^{n^2}$ two *secret* affine bijections.

We now describe the $[C]$ scheme used in encryption mode.

Representation of the messages
Each message M is represented by a n^2-uple $(x_1, ..., x_{n^2}) \in K^{n^2}$. The message space is $\mathcal{M} = K^{n^2}$.

The quadratic function f
We then define the following quadratic function on the message space:

$$f : \begin{cases} \mathcal{M} \to \mathcal{M} \\ x \mapsto t(s(x)^2) \end{cases}.$$

Public key: The n^2-uple $(p_1, ..., p_{n^2})$ of n^2-variate quadratic polynomials over K that represent f. They are defined by:

$$f(x_1, ..., x_{n^2}) = (p_1(x_1, ..., x_{n^2}), ..., p_{n^2}(x_1, ..., x_{n^2})).$$

Secret key: The two affine bijections s and t.

Encryption
To encrypt the message M represented by $x = (x_1, ..., x_{n^2}) \in \mathcal{M}$, compute the ciphertext $y = (y_1, ..., y_{n^2})$ with the following formulas:

$$\begin{cases} y_1 = p_1(x_1, ..., x_{n^2}) \\ \quad \vdots \\ y_{n^2} = p_{n^2}(x_1, ..., x_{n^2}) \end{cases}$$

Decryption
To decrypt the ciphertext $y \in \mathcal{M}$, one has to solve the equations $A^2 = B$, where $B = t^{-1}(y)$, and then to compute the cleartext $x = s^{-1}(A)$.
It is important to notice that $A \mapsto A^2$ is not a bijection any longer (contrary to the original $[C]$ scheme described in section 10). As a result, there may be several possible cleartexts for a given ciphertext. One solution to avoid this ambiguousness is to put some redundancy in the representation of the messages, by making use of an error correcting code or a hash function (for details, see [10] p. 34, where a similar idea is used in a different scheme).
The feasibility of choosing the right cleartext among the possible ones is due to the fact that – for an average B – the number of solutions A of the equations $A^2 = B$ remains reasonable, as shown in table 6 below:

0	6	252	34440	13-15	0	0	0
1	8	160	22272	16	0	0	672
2	0	42	5040	17-21	0	0	0
3	0	0	0	22	0	2	240
4	2	56	2240	23-315	0	0	0
5-11	0	0	0	316	0	0	2
12	0	0	630	> 316	0	0	0

Table 6: Repartition of the number of pre-images for $[C_n]$ over $K = GF(2)$

Note 1: In the case $n = 4$, the two matrices having 316 pre-images are 0 and I.

Note 2: These results are just "toy simulations" of $[C_n]$, because if $K = GF(2)$, n must be such that $n^2 \geq 64$ in real examples.

To solve the equation $A^2 = B$ when B is a given average matrix of $\mathcal{M}_n(K)$, two methods can be used:

- The first one is based on the Jordan reduction of matrices, and provides a polynomial time algorithm to compute the square roots of a given matrix. For details, see [3] (chapter VIII, p. 231).
- The second one is based on the Cayley-Hamilton theorem. Let us denote by

  $$\chi_M(\lambda) = \lambda^n + \alpha_{n-1}(M)\lambda^{n-1} + ... + \alpha_1(M)\lambda + \alpha_0(M)$$

  the characteristic polynomial of a matrix $M \in \mathcal{M}_n(K)$.

  Since K is a field of characteristic 2, it is easy to prove that $(\chi_M(\lambda))^2 = \chi_{M^2}(\lambda^2)$, and thus $\alpha_i(M^2) = (\alpha_i(M))^2$ $(0 \leq i \leq n-1)$.

  Suppose now that A satisfies $A^2 = B$ for a given B. Then, from the Cayley-Hamilton theorem:

  $$\chi_A(A) = A^n + \alpha_{n-1}(A) \cdot A + ... + \alpha_1(A) \cdot A + \alpha_0(A) \cdot I = 0.$$

  Hence:

  $$A\Big(\sqrt{\alpha_1(B)} \cdot I + \sqrt{\alpha_3(B)} \cdot B + \sqrt{\alpha_5(B)} \cdot B^2 + \ldots\Big) = \sqrt{\alpha_0(B)} \cdot I + \sqrt{\alpha_2(B)} \cdot B + \sqrt{\alpha_4(B)} \cdot B^2 + \ldots$$

  If we make the assumption that $\sqrt{\alpha_1(B)} \cdot I + \sqrt{\alpha_3(B)} \cdot B + \ldots$ is invertible, we obtain the following formula to compute A:

  $$A = \Big(\sqrt{\alpha_0(B)} \cdot I + \sqrt{\alpha_2(B)} \cdot B + \ldots\Big)\Big(\sqrt{\alpha_1(B)} \cdot I + \sqrt{\alpha_3(B)} \cdot B + \ldots\Big)^{-1}$$

  and thus

  $$x = s^{-1}\Big(\Big(\sqrt{\alpha_0(t^{-1}(y))} \cdot I + \sqrt{\alpha_2(t^{-1}(y))} \cdot t^{-1}(y) + \ldots\Big) \Big(\sqrt{\alpha_1(t^{-1}(y))} \cdot I + \sqrt{\alpha_3(t^{-1}(y))} \cdot t^{-1}(y) + \ldots\Big)^{-1}\Big)$$

Note 1: The first method always works, whereas the second one can be used only for ciphertexts y such that $B = t^{-1}(y)$ satisfies $\alpha_1(B) \cdot I + \alpha_3(B) \cdot B + \ldots$ invertible.

Note 2: The scheme can also be used in signature. To sign a message M, the basic idea is to compute x from $y = h(R||M)$ (as if we were deciphering a message), where h is a hash function and R is a small pad. If we succeed, (x, R) will be the signature of M. If we do not succeed (because the function is not a bijection), we try another pad R (for variants and details, see [10], where a similar idea is used).

In this section, we describe a polynomial attack against the $[C_n]$ algorithm, which proves that this scheme is insecure.
The key idea is to use the fact that $B = A^2$ implies $AB = BA$ (whereas two random matrices A and B do not commute in general).

- We begin by computing (by Gaussian reductions) all the equations of the following type (which we called "type (1)" in section 5):

$$\sum \gamma_{ij} x_i y_j + \sum \alpha_i x_i + \sum \beta_i y_i + \delta_0 = 0 \qquad (1)$$

The relation $AB = BA$ gives *a priori* n equations of this type. In fact, when we give explicit values to the y_i variables, we cannot obtain n independent linear equations on the x_i variables, since $AB = BA$ is also true when $B = P(A)$, where P is *any* polynomial in $K[X]$.

The exact number of independent linear equations coming from $AB = BA$ is given by the following result of [3]:

Theorem 13.1 *The number N of linearly independent matrices that commute with the matrix B is given by the formula*

$$N = n_1 + 3n_2 + ... + (2t-1)n_t$$

where n_1, n_2, ..., n_t are the degrees of the non constant invariant polynomials *$i_1(\lambda)$, ..., $i_t(\lambda)$ of B.*

(See [3], chapter VI, for the definition of the invariant polynomials, and chapter VIII for a proof of the theorem.)

In particular, we have $n \leq N \leq n^2$, with $N \simeq n$ in most of the cases.

- It remains – *a priori* – to perform an exhaustive search on $\simeq n$ variables to end the attack. In fact, we have made some simulations (see table 7 below) that suggest that there also exist many equations of type (2) (defined in section 5), and type (4) defined by:

$$\sum \gamma_{ij} x_i y_j + \sum \mu_{ij} y_i y_j + \sum \alpha_i x_i + \sum \beta_i y_i + \delta_0 = 0 \qquad (4)$$

	$n=2$	$n=3$	$n=4$
$p=2$	$^{10\ \ 16}_{39}$	$^{10\ \ 18}_{153}$	$^{17\ \ 32}_{292}$
$p=3$	$^{4\ \ 4}_{28}$	$^{9\ \ 9}_{89}$	$^{16\ \ 16}_{271}$
$p=31$	$^{3\ \ 3}_{14}$	$^{8\ \ 8}_{79}$	$^{15\ \ 15}_{254}$
$p=127$	$^{3\ \ 3}_{14}$	$^{8\ \ 8}_{79}$	$^{15\ \ 15}_{254}$

Table 7: Number of equations of $^{\text{type 1}\ \ \text{type 4}}_{\text{type 2}}$ for $[C_n]$ over $K = GF(p)$

Note: For $p = 2$, on these examples, we obtain $(n+1)$ (formally) linearly independent equations of type (1). This can be explained by the fact that – on the field $K = GF(2)$ – the equations $B = A^2$ implies $\mathrm{tr}(B) = \mathrm{tr}(A)$.

These equations of type (1), (2) and (4) can be found by Gaussian reductions on a polynomial number of cleartext/ciphertext paris. Therefore, the $[C_n]$ scheme is unlikely to be secure: by using all the found equations of type (1), (2) and (3), a cleartext will be easily found by Gaussian reductions.

The cryptanalysis of $[C_n]$ described in section 13 uses the fact that A and B commute when $B = A^2$. In order to avoid that very special algebraix property, we suggest to replace the transformation $B = A^2$ by the equation $B = A^2 + MA$, where M is a *secret* matrix randomly chosen in $\mathcal{M}_n(K)$.
The description of the obtained scheme – called *HM* – is exactly the same as for $[C_n]$. As in section 12, the transformation is generally not one-to-one, but the scheme can be used in a practical way because – as in the case of $[C_n]$ –, the number of pre-images of a given average matrix B remains under a reasonable limit. Table 8 below illustrates this fact (for a randomly chosen matrix M):

# pre-images	$n=2$	$n=3$	$n=4$	# pre-images	$n=2$	$n=3$	$n=4$
0	6	284	39552	16	0	0	72
1	8	112	12024	17	0	0	0
2	0	42	6576	18	0	0	12
3	0	32	2256	19	0	0	24
4	2	34	1868	20	0	0	24
5	0	0	960	21	0	0	0
6	0	0	972	22	0	0	24
7	0	0	168	23-25	0	0	0
8	0	2	324	26	0	0	36
9	0	0	48	27	0	0	0
10	0	2	144	28	0	0	6
11	0	0	96	29-33	0	0	0
12	0	4	162	34	0	0	4
13	0	0	56	35-39	0	0	0
14	0	0	72	40	0	0	8
15	0	0	48	> 40	0	0	0

Table 8: Repartition of the number of pre-images for *HM* over $K = GF(2)$

Note: These results are just "toy simulations" of $[C_n]$, because if $K = GF(2)$, n must be such that $n^2 \geq 64$ in real examples.
In order to obtain a practical scheme, one has to be able to solve the equation

$$A^2 + MA = B$$

for a given matrix $B \in \mathcal{M}_n(K)$.
There indeed exist a polynomial time algorithm to perform this computation (see [3], chapter VIII). The basic idea of this algorithm is to use the fact that:

$$B = A^2 + MA \Rightarrow g(A) = 0,$$

where $g(\lambda) = \det(\lambda^2 + \lambda \cdot M - B)$ is a polynomial with *scalar* coefficients (notice that this property is a generalization of the Cayley-Hamilton theorem). The equation $g(A) = 0$ can be solved by using the Jordan reduction of matrices.

The *HM* scheme seems to be less vulnerable to attacks based on affine multiple (*i.e.* on equations such as those of type (1), (2) or (4)), as shown in table 9 below:

	$n=2$	$n=3$	$n=4$
$p=2$	10 16 39	3 11 133	3 18 49
$p=3$	1 1 14	1 1 11	1 1 18
$p=31$	0 0 1	0 0 1	0 0 1
$p=127$	0 0 0	0 0 0	0 0 0

Table 9: Number of equations of type 2 for HM over $K = GF(p)$

However, we have made computations (see table 10 below) showing that equations of type (3) (defined in section 5) still exist, and also equations of "type (5)", defined by:

$$\sum \mu_{ij} x_i y_j + \sum \nu_{ij} x_i x_j + \sum \alpha_i x_i + \sum \beta_i y_i + \delta_0 = 0 \qquad (5)$$

HM	$n = 2$	$n = 3$	$n = 4$
Equations (5)	7	17	31
Equations (3)	9	30	58

Table 10: Number of linearly independent equations (after replacement of the y_i variables by explicit values)

Note: It may be noticed that $B = A^2$ implies the two following identities:

$$\begin{cases} AB - BA = AMA - MA^2 & \text{(type (5))} \\ A^2B - BA^2 = BMA - MAB & \text{(type (3))} \end{cases}$$

This explains – in part – the existence of such equations of type (5) and type (3).

The fact that such equations do exist threatens the HM scheme. In fact they make the cryptanalyst able to distinguish between a random quadratic transformation of K^{n^2} and a quadratic transformation corresponding to the HM scheme.
This fact explains that we do not recommend the HM scheme. However, at the present, the existence of equations of type (5) and type (3) does not seem sufficient to break the scheme. Therefore, the question of the security of HM remains open...

15 Conclusion

Among cryptologists that have studied the problem, two main opinions arise as concerns public key schemes built with multivariate polynomials. Some of them think that most of these schemes should be vulnerable to attacks based on general principles, still to be found. According to others, the status of those many schemes can be compared to the one of most secret key algorithms: no relative proof of security is known, but the great flexibility for the choice among the possible variants of the schemes, together with the relative easiness for building efficient schemes that avoid known attacks, may support a certain confidence in the security of the schemes, at least – *a priori* – for those which do not seem too close to known cryptanalytic techniques.
The present article does not settle the question once and for all. Nevertheless, it gives arguments for both opinions. On the one hand, we have shown how to break some schemes for which no cryptanalysis had been given before. On the other hand, we have studied some simple and general ideas (removing equations, adding ones, introducing new variables...) that might – *a priori* – sensibly enforce the security of some asymmetric schemes. Interesting mathematical questions naturally arise: better understanding and detecting orthogonal polynomials, using a non commutative ring of matrices to generate multivariate equations on a (commutative) field, etc. If we had to take a strong line as concerns the unbroken schemes, our current opinion is that the most provocative schemes (C^*_{--}, C^*_{-+}, HM) may be too close to known cryptanalysis to be recommended, but more complex schemes (such as HFE_{-+}) may be really secure... However, it is still too soon to have a definitive opinion, and we think that – above all – the important point is to go further into the understanding of the mysterious links between mathematics and the concepts of asymmetric cryptography and cryptanalysis.

References

[1] D. Coppersmith, S. Winograd, *Matrix Multiplication via Arithmetic Progressions*, J. Symbolic Computation, 1990, vol. 9, pp. 251-280.

[3] F.R. Gantmacher, *The Theory of Matrices*, volume 1, Chelsae Publishing Company, New-York.

[4] H. Imai, T. Matsumoto, *Algebraic Methods for Constructing Asymmetric Cryptosystems*, Algebraic Algorithms and Error Correcting Codes (AAECC-3), Grenoble, 1985, Springer-Verlag, Lectures Notes in Computer Science n$^{\text{O}}$ 229.

[5] A. Kipnis, J. Patarin, L. Goubin, *Unbalanced Oil and Vinegar Signature Schemes*, Proceedings of EUROCRYPT'99, Springer, pp. 206-222.

[6] N. Koblitz, *Algebraic Aspects of Cryptography*, Algorithms and Computation in Mathematics, Volume 3, Springer, 1998.

[7] R. Lidl, H. Niederreiter, *Finite Fields*, Encyclopedia of Mathematics and its applications, Volume 20, Cambridge University Press.

[8] T. Matsumoto, H. Imai, *Public Quadratic Polynomial-Tuples for Efficient Signature-Verification and Message-Encryption*, Advances in Cryptology, Proceedings of EUROCRYPT'88, Springer-Verlag, pp. 419-453.

[9] J. Patarin, *Cryptanalysis of the Matsumoto and Imai Public Key Scheme of Eurocrypt'88*, Advances in Cryptology, Proceedings of CRYPTO'95, Springer, pp. 248-261.

[10] J. Patarin, *Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP) : Two New Families of Asymmetric Algorithms*, Advances in Cryptology, Proceedings of EUROCRYPT'96, Springer, pp. 33-48.

[11] J. Patarin, *Asymmetric Cryptography with a Hidden Monomial*, Advances in Cryptology, Proceedings of CRYPTO'96, Springer, pp. 45-60.

[12] J. Patarin, L. Goubin, *Trapdoor One-way Permutations and Multivariate Polynomials*, Proceedings of ICICS'97, Springer, LNCS n$^{\text{o}}$1334, pp. 356-368.

[13] J. Patarin, L. Goubin, *Asymmetric Cryptography with S-Boxes*, Proceedings of ICICS'97, Springer, LNCS n$^{\text{o}}$1334, pp. 369-380.